Oracle Database 12c Release 2 Real Application Clusters Handbook

Concepts, Administration, Tuning, and Troubleshooting

Architect a Scalable, Highly Available Database Environment

K. Gopalakrishnan
Oracle ACE Alumni

Sam R. Alapati
Oracle ACE Alumni

https://t.me/learningnets

About the Authors

K. Gopalakrishnan (aka Gopal) is an award-winning author (*Oracle Magazine*'s "Oracle Author of the Year 2005") of the bestselling book *Oracle Wait Interface: A Practical Guide to Performance Diagnostics & Tuning* (Oracle Press/McGraw-Hill Education, 2004). Gopal has also authored *Oracle Database 11*g*Real Application Clusters Handbook* (Oracle Press/McGraw-Hill Education, 2012). Oracle Technology Network (OTN) recognizes him as an Oracle ACE.

Gopal is a recognized expert in Oracle RAC and database internals, and has used his extensive expertise in solving many vexing performance issues all across the globe for telecom giants, banks, financial institutions, and universities in over 30 countries across five continents.

Gopal is currently working with Cloud Performance Scalability and Reliability Engineering at Oracle Corporation, managing one of the largest SaaS fleet on Oracle Exadata Database Machine and Oracle Real Application Clusters.

Sam R. Alapati works as a data administrator for Solera, in Westlake, Texas, just outside Dallas. Sam has worked as an Oracle DBA for many years and has authored many books for Oracle DBAs, including *Expert Oracle Database 11*g *Administration* (Apress, 2008) and *OCP Upgrade to Oracle Database 12*c *Exam Guide (Exam 1Z0-060)* (Oracle Press/McGraw-Hill Education, 2014).

https://t.me/learningnets

ORACLE®

Oracle Press™

Oracle Database 12c Release 2 Real Application Clusters Handbook

Concepts, Administration, Tuning & Troubleshooting

K. Gopalakrishnan
Sam R. Alapati

Mc Graw Hill Education

New York Chicago San Francisco
Athens London Madrid Mexico City
Milan New Delhi Singapore Sydney Toronto

https://t.me/learningnets

https://t.me/learningnets

Copyright © 2018 by McGraw-Hill Education (Publisher). All rights reserved. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

ISBN: 978-0-07-183047-8
MHID: 0-07-183047-2

The material in this eBook also appears in the print version of this title: ISBN: 978-0-07-183048-5, MHID: 0-07-183048-0.

eBook conversion by codeMantra
Version 1.0

All trademarks are trademarks of their respective owners. Rather than put a trademark symbol after every occurrence of a trademarked name, we use names in an editorial fashion only, and to the benefit of the trademark owner, with no intention of infringement of the trademark. Where such designations appear in this book, they have been printed with initial caps.

McGraw-Hill Education eBooks are available at special quantity discounts to use as premiums and sales promotions or for use in corporate training programs. To contact a representative, please visit the Contact Us page at www.mhprofessional.com.

Information has been obtained by McGraw-Hill Education from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, McGraw-Hill Education, or others, McGraw-Hill Education does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from the use of such information.

Oracle Corporation does not make any representations or warranties as to the accuracy, adequacy, or completeness of any information contained in this Work, and is not responsible for any errors or omissions.

TERMS OF USE

This is a copyrighted work and McGraw-Hill Education and its licensors reserve all rights in and to the work. Use of this work is subject to these terms. Except as permitted under the Copyright Act of 1976 and the right to store and retrieve one copy of the work, you may not decompile, disassemble, reverse engineer, reproduce, modify, create derivative works based upon, transmit, distribute, disseminate, sell, publish or sublicense the work or any part of it without McGraw-Hill Education's prior consent. You may use the work for your own noncommercial and personal use; any

https://t.me/learningnets

other use of the work is strictly prohibited. Your right to use the work may be terminated if you fail to comply with these terms.

THE WORK IS PROVIDED “AS IS.” McGRAW-HILL EDUCATION AND ITS LICENSORS MAKE NO GUARANTEES OR WARRANTIES AS TO THE ACCURACY, ADEQUACY OR COMPLETENESS OF OR RESULTS TO BE OBTAINED FROM USING THE WORK, INCLUDING ANY INFORMATION THAT CAN BE ACCESSED THROUGH THE WORK VIA HYPERLINK OR OTHERWISE, AND EXPRESSLY DISCLAIM ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. McGraw-Hill Education and its licensors do not warrant or guarantee that the functions contained in the work will meet your requirements or that its operation will be uninterrupted or error free. Neither McGraw-Hill Education nor its licensors shall be liable to you or anyone else for any inaccuracy, error or omission, regardless of cause, in the work or for any damages resulting therefrom. McGraw-Hill Education has no responsibility for the content of any information accessed through the work. Under no circumstances shall McGraw-Hill Education and/or its licensors be liable for any indirect, incidental, special, punitive, consequential or similar damages that result from the use of or inability to use the work, even if any of them has been advised of the possibility of such damages. This limitation of liability shall apply to any claim or cause whatsoever whether such claim or cause arises in contract, tort or otherwise.

https://t.me/learningnets

Dedicated to my son, Sriharan, and my wife, Mallika
—K. Gopalakrishnan
I dedicate this book to Eddie (Eligio) Colon—an officer and a gentleman, and my good friend, whose debt I'll never be able to pay back
—Sam R. Alapati

https://t.me/learningnets

Contents at a Glance

PART I
High Availability Architecture and Clusters

1 Introduction to High Availability and Scalability

2 Oracle Database Clustering Basics and Its Evolution

3 Oracle RAC Architecture

PART II
Installation, Configuration, and Storage

4 Oracle Grid Infrastructure Installation

5 Installing Oracle RAC and Creating an Oracle RAC Database

6 Automatic Storage Management

PART III
Oracle RAC Administration and Management

7 Oracle RAC Basic Administration

8 Managing Oracle Clusterware

9 Oracle RAC Backup and Recovery

10 Oracle RAC Performance Management

PART IV
Advanced Oracle RAC Concepts

https://t.me/learningnets

11 Global Resource Directory

12 A Closer Look at Cache Fusion

13 Workload Management, Connection Management, and Application Continuity

14 Oracle RAC Troubleshooting

PART V

Deploying Oracle RAC

15 Extending Oracle RAC for Maximum Availability

16 Developing Applications for Oracle RAC

PART VI

Appendixes

A Oracle RAC Reference

B Adding and Removing Cluster Nodes

Index

https://t.me/learningnets

Contents

Acknowledgments
Introduction

PART I
High Availability Architecture and Clusters

1 Introduction to High Availability and Scalability
High Availability
HA Terminology
Planned and Unplanned Outages
An End-to-End Perspective
Cost of Downtime
Building Redundant Components
Common Solutions for HA
Cluster, Cold Failover, and Hot Failover
HA Option Pros and Cons
Scalability
The Oracle RAC Solution
Agility
The Oracle Database 12*c* RAC Solution
Summary

2 Oracle Database Clustering Basics and Its Evolution
Cloud Computing with Clusters
Shared Storage in Clustering

https://t.me/learningnets

Types of Clustering Architectures
Hadoop Clusters
Architecture of Hadoop
Historical Background of Oracle RAC
Oracle Parallel Storage Evaluation
Oracle Parallel Server Architecture
Components of an OPS Database
Cluster Group Services (CGS)
Distributed Lock Manager (DLM)
Locking Concepts in Oracle Parallel Server
Cache Fusion Stage 1, CR Server
The Oracle RAC Solution
Availability
Scalability
Affordability
Summary

3 Oracle RAC Architecture
Introduction to Oracle Real Application Clusters
Single-Instance Environment vs. Oracle RAC Environment
Oracle Flex Clusters
Oracle Extended Clusters
Oracle Multitenant and Oracle RAC
Administrator- vs. Policy-Managed Databases
Dynamic Services
Oracle Database Quality of Service Management
Oracle RAC Components
Shared Disk System
Oracle Clusterware
The Oracle High Availability Services Technology Stack
Oracle RAC Networking Concepts and Components
Key Networking Concepts
The Networking Stack Components
Oracle Kernel Components
Global Cache and Global Enqueue Services
Global Resource Directory
Oracle RAC Background Processes
Summary

PART II
Installation, Configuration, and Storage

https://t.me/learningnets

4 Oracle Grid Infrastructure Installation
An Overview of the Grid Infrastructure Installation Process
Preinstallation Tasks
Understanding the Installer, CVU, and ORAchk
Configuring the Operating System
Configuring the Network
Configuring NTP
Setting Up the Groups and Users
Creating the Required Linux Directories
Configuring Shared Storage
Secure Shell and User Limits Configuration
Setting User Limits
Configuring the Kernel Parameters
Running the Cluster Verification Utility
Installing Oracle Grid Infrastructure with OUI
Installing Oracle Grid Infrastructure
Choosing the Installation Options and Naming Your Cluster
Specifying the Cluster Nodes and Verifying SSH Connectivity
Specifying the Network Interfaces
Selecting the Storage Options
Specifying Management Options and Privileged OS System Groups
Performing the Prerequisite Checks
Running the root Scripts
The Product Installation
Verifying the Oracle Grid Infrastructure Installation
Summary

5 Installing Oracle RAC and Creating an Oracle RAC Database
Installing Oracle Real Application Clusters
Creating the Oracle RAC Database
Summary

6 Automatic Storage Management
Standard Oracle ASM and Oracle Flex ASM
Introduction to Automatic Storage Management
Physical Limits of ASM
ASM in Operation
ASM Building Blocks
Managing Oracle ASM Files and Directories
ASM Filenames

https://t.me/learningnets

Creating and Referencing ASM Files
Managing Disk Group Directories
ASM Administration and Management
Managing an ASM Instance
ASM Initialization Parameters
Managing ASM Disk Groups
Creating a Disk Group
Adding Disks to a Disk Group
Dropping, Undropping, Resizing, and Renaming Disks in a Disk Group
Administering ACFS
Setting Up ACFS
Creating an ACFS Snapshot
ASM Disk Rebalancing
Manually Rebalancing a Disk Group
Rebalancing Phase Options
Monitoring the Performance of Balancing Operations
Tuning Disk Rebalancing Operations
Backup and Recovery in ASM
ASM Flex Clusters
Configuration of Oracle ASM in Flex ASM
Setting Up Flex ASM
Managing ASM Flex Disk Groups
Understanding ASM File Groups and ASM Quota Groups
Oracle Extended Disk Groups
ASM Tools
ASMCA: The ASM Configuration Assistant
ASMCMD: The ASM Command-Line Utility
ASM FTP Utility
ASMLib
Installing ASMLib
Configuring ASMLib
Oracle ASM Filter Driver (Oracle ASMFD)
Summary

PART III
Oracle RAC Administration and Management

7 Oracle RAC Basic Administration
Oracle RAC Initialization Parameters
Parameters That Are Unique to an Instance
Identical Parameters

https://t.me/learningnets

Instance Parameters That “Should” Be the Same
Managing the Parameter File
Backing Up the Server Parameter File
Search Order for the Parameter Files in an Oracle RAC Database
Starting and Stopping Instances
Using SRVCTL to Start/Stop Databases and Instance(s)
Administering the Oracle ASM Instances
Using CRSCTL to Stop Databases and Instances
Using SQL*Plus to Start/Stop Instances
Common SRVCTL Management Commands
Database-Related SRVCTL Commands
Instance-Related SRVCTL Commands
Listener-Related SRVCTL Commands
Setting, Unsetting, and Displaying the Environment Variables
Changing the Configuration of Databases and Instances
Relocating Services
Removing the Configuration Information for Specific Targets
Predicting the Impact of Failures
Managing Pluggable Databases in a RAC Environment
Administering Undo in an Oracle RAC Database
Administering a Temporary Tablespace
Managing Traditional (Global) Tablespaces
Managing Local Temporary Tablespaces
Hierarchy of Temporary Tablespaces
Administering Online Redo Logs
Enabling Archive Logs in the Oracle RAC Environment
Enabling the Flashback Area
Managing Database Configuration with SRVCTL
Killing Sessions on Specific Instances of a Cluster
Managing Database Objects
Managing Sequences
Managing Tables
Managing Indexes
Scope of SQL Commands
Database Connections
Administering Server Pools
Configuring Server Pools
Consolidation of Databases
Creating a Server Pool
Converting an Administrator-Managed Database to a Policy-Managed Database

https://t.me/learningnets

Summary

8 Managing Oracle Clusterware

Configuring and Administering Oracle Clusterware
- Benefits of Server Pools
- Server Pools and Policy-Based Management
- Server Pools and Categorization
- How Server Pools Work
- Types of Server Pools
- Creating Server Pools
- Evaluating the Addition of a Server Pool
- Deleting a Server Pool

Role-Separated Management
- Managing Cluster Administrators
- Configuring Role Separation
- Using the crsctl setperm Command

Weight-Based Server Node Eviction

Administering SCAN
- Starting and Stopping SCAN
- Displaying the SCAN Status
- Administering the Grid Naming Service (GNS)

Using the CLUVFY Utility for Managing Oracle RAC

Clusterware Startup
- The Clusterware Startup Process
- Clusterware Starting Sequence
- Oracle Clusterware Auto-Startup
- Oracle Clusterware Manual Startup

Using CRSCTL to Manage the Clusterware
- Starting and Stopping CRS
- Clusterized (Cluster-Aware) CRSCTL Commands
- Verifying the Status of CRS
- Disabling and Enabling CRS
- The CRSCTL EVAL Commands

Using Other Utilities to Manage Oracle Clusterware
- Using the olsnodes Command
- The GPnP Tool
- The Cluster Health Monitor
- The OCLUMON Tool
- Oracle Interface Configuration: oifcfg
- Cluster Configuration Utility: clscfg
- The Cluster Name Check Utility: cemutlo

https://t.me/learningnets

Oracle Trace File Analyzer
Administering the OCR
Checking OCR Integrity
Dumping OCR Information
Managing the OCR with the OCRCONFIG Utility
Maintaining a Mirror OCR
Migrating the OCR to ASM
Administering the Oracle Local Registry
Administering the Voting Disks
Using ASM Storage for Voting Files
Backing Up Voting Disks
Restoring Voting Disks
Adding and Deleting Voting Disks
Migrating Voting Disks
Summary

9 Oracle RAC Backup and Recovery
Introduction to Backups
Oracle Backup Options
Oracle Backup Basics
Performing Backups in Oracle
Instance Recovery in Oracle RAC
Redo Threads and Streams
Redo Records and Change Vectors
Checkpoints
Crash Recovery
Steps in Crash Recovery (Single Instance)
Crash Recovery in Oracle RAC
Instance Recovery
Crash Recovery and Media Recovery
Bounded Recovery
Block-Written Record (BWR)
Past Image (PI)
Two-Pass Recovery
Cache Fusion Recovery
Dynamic Reconfiguration and Affinity Remastering
Fast Reconfiguration in Oracle RAC
Internals of Cache Fusion Recovery
Backup and Recovery of the Voting Disk and OCR
Backup and Recovery of Voting Disks
Backup and Recovery of OCR

https://t.me/learningnets

Validating OCR Backups
Summary

10 Oracle RAC Performance Management
Oracle RAC Design Considerations
Oracle Design Best Practices
Oracle RAC–Specific Design Best Practices
Partitioning the Workload
Scalability and Performance
Choosing the Block Size for an Oracle RAC Database
Introduction to the V$ and GV$ Views
Parallel Query Slaves
V$ Views Containing Cache Fusion Statistics
Oracle RAC Wait Events
Understanding Cluster Waits
Global Cache Statistics
Global Cache Statistics Summary
Global Cache Service Times
Global Cache Service Times Summary
Enqueue Tuning in Oracle RAC
Oracle AWR Report
Interpreting the AWR Report
ADDM
ASH Reports
Tuning the Cluster Interconnect
Verifying That Private Interconnect Is Used
Interconnect Latencies
Verifying That Network Interconnect Is Not Saturated
Summary

PART IV
Advanced Oracle RAC Concepts

11 Global Resource Directory
Resources and Enqueues
Grants and Conversions
Locks and Enqueues
Cache Coherency
Global Enqueue Services
Latches and Enqueues
Global Locks Database and Structure
Messaging in Oracle RAC

https://t.me/learningnets

Global Cache Services
Lock Modes and Lock Roles
Consistent Read Processing
GCS Resource Mastering
Read-Mostly Locking
Summary

12 A Closer Look at Cache Fusion
Key Components in Cache Fusion
Ping
Deferred Ping
Past Image (PI) Blocks
Lock Mastering
Types of Contention
Cache Fusion I, or Consistent Read Server
Cache Fusion II, or Write/Write Cache Fusion
Cache Fusion in Operation
Cache Fusion Walkthrough
Resource Mastering and Remastering
Background Processes and Cache Fusion
LMON: Lock Monitor Process
LMS: Lock Manager Server
LMD: Lock Manager Daemon Process (LMDn)
LCKn: Lock Process (LCK0)
DIAG: Diagnostic Daemon (DIAG)
Summary

13 Workload Management, Connection Management, and Application Continuity
Understanding Dynamic Database Services
Service Characteristics
Services and Policy-Managed Databases
Resource Management and Services
Using Services with Oracle Scheduler
Administering Services
Using Views to Get Service Information
Distributed Transaction Processing
AQ_HA_Notifications
Workload Distribution and Load Balancing
Hardware and Software Load Balancing
Client-Side Load Balancing

https://t.me/learningnets

Server-Side Load Balancing
Transparent Application Failover
TAF Considerations
Workload Balancing
Measuring Workloads by Service
Using Service-Level Thresholds
Oracle RAC High Availability Features
High Availability, Notifications, and FAN
Event-Based Notification
Application Failure Issues
Using Transaction Guard for Efficient Client Failover
Application Continuity for Masking Database Outages
Summary

14 Oracle RAC Troubleshooting
Installation Log Files
Log Directory Structure in the Oracle RDBMS
Log Directory Structure in Oracle Grid Infrastructure
Troubleshooting a Failed Oracle Grid Infrastructure Installation
Inside the Database Alert Log
RAC ON and OFF
Database Performance Issues
Hung Database
Hanganalyze Utility
Debugging Node Eviction Issues
Cluster Health Monitor
Instance Membership Recovery
Advanced Debugging for Oracle Clusterware Modules
Debugging Various Utilities in Oracle RAC
Using ORAchk to Troubleshoot RAC
Summary

PART V
Deploying Oracle RAC

15 Extending Oracle RAC for Maximum Availability
Benefits of Extended RAC Clusters
Full Utilization of Resources
Extremely Rapid Recovery
Design Considerations
Speed of Light
Network Connectivity

https://t.me/learningnets

Cache Fusion Performance
Data Storage
Common Techniques for Data Mirroring
Array-Based Mirroring
Host-Based Mirroring
ASM Preferred Read
Challenges in Extended Clusters
Extended Oracle RAC Limitations
Extended Oracle RAC vs. Oracle Data Guard
Summary

16 Developing Applications for Oracle RAC
Application Partitioning
Best Practice: Application Partitioning Schemes
Data Partitioning
Best Practice: Guidance Systems
Buffer Busy Waits/Block Contention
Index Partitioning
Buffer Busy Waits: Index Branch/Leaf Blocks Contention
Sorted Hash Clusters
Working with Sequences
CACHE and NOORDER
CACHE and ORDER
NOCACHE and ORDER
Best Practice: Use Different Sequences for Each Instance
Connection Management
Full Table Scans
Identifying Full Table Scans
Interconnect Protocol
Ethernet Frame Size
Library Cache Effect in the Parsing
Commit Frequency
Summary

PART VI
Appendixes

A Oracle RAC Reference
Global Cache Services and Cache Fusion Diagnostics
V$CACHE
V$CACHE_TRANSFER

https://t.me/learningnets

V$INSTANCE_CACHE_TRANSFER
V$CR_BLOCK_SERVER
V$CURRENT_BLOCK_SERVER
V$GC_ELEMENT
Global Enqueue Services Diagnostics
V$LOCK
V$GES_BLOCKING_ENQUEUE
V$ENQUEUE_STATISTICS
V$LOCKED_OBJECT
V$GES_STATISTICS
V$GES_ENQUEUE
V$GES_CONVERT_LOCAL
V$GES_CONVERT_REMOTE
V$GES_RESOURCE
Dynamic Resource Remastering Diagnostics
V$HVMASTER_INFO
V$GCSHVMASTER_INFO
V$GCSPFMASTER_INFO
Cluster Interconnect Diagnostics
V$CLUSTER_INTERCONNECTS
V$CONFIGURED_INTERCONNECTS

B Adding and Removing Cluster Nodes
Adding a Node
Performing Pre-Installation Checks
Executing the addNode.sh Script
Installing the Oracle Database Software
Creating a Database Instance
Removing a Node
Deleting the Instance on the Node to Be Deleted
Removing the Node from the Database
Removing the RAC4 Node from the Clusterware

Index

https://t.me/learningnets

Acknowledgments

’Tis never good to let the thought of good things done thee pass away.

Writing a book is not just simply about the author putting down thoughts or sharing knowledge via the printed material. It is like directing an orchestra. Many musicians have to come together and play their individual instruments in tune in an orchestra under the guidance of a good conductor in order to sound melodious. In the same way, a great editorial team has to work together to help the author create a book. When a stellar team comes together this way, the result is a book that is both pleasurable to read and simple to comprehend. As the author, I am greatly indebted to the members of this team who stood behind me and helped me create this book.

First and foremost, my sincere thanks to my long-term friend, philosopher, and guide John Kanagaraj for his support in every project I start. John, an Oracle author himself, has been an inspiration from the very beginning. I salute him for his patience and guidance. John kept my spirits up whenever I was running low.

I am thankful to everyone who contributed to this book, either directly or indirectly. During the past few years, I have learned a lot from many people, and my special thanks go to Steve Adams, Jonathan Lewis, Vijay Lunawat, Scott Gossett, Scott Heisey, Gaja Vaidyanatha, and James Morle. A favor conferred in the time of need, though it be small (in itself), is (in value) much larger than the world!

It is an honor for me to work with some of the smartest minds at Oracle Corporation, and I would like to show my sincere gratitude for their help. I am thankful to my colleagues Chandra Pabba, Sreekanth Krishnavajjala, and Yesh Jaishwal for their initial reviews. Chandra contributed immensely on the initial rewrite of the chapters. A special thanks to my coauthor, Sam Alapati, for revising this book for the latest version.

I would also like to extend my sincere thanks to my managers Dillip Praharaj and Harsha Gadagkar at Oracle Performance Scalability and Reliability Engineering, and

https://t.me/learningnets

Shankar Jayaganapathy at Oracle Enterprise Solutions Group, for supporting and encouraging me in this effort. Without their help and guidance, I am afraid that I would not have found time to work on this book. I also thank Vice President Vikram Kumar, from Oracle Corporation, for his help and support at various stages.

I am truly indebted to my esteemed clients. They not only posed challenges, but also were willing to implement my solutions in their systems. They played a big role in the knowledge acquisition and enhancement that I shared in this book.

It was great working with the awesome people at McGraw-Hill Education. Thank you all very much for being so patient with me, and for making sure that this book project stayed on track and on time.

Last but not the least, I wish to thank again my technical editors and reviewers John Kanagaraj, Arup Nanda, and Sandesh Rao, whose great efforts are reflected in the rich content and timely completion of the book.

—K. Gopalakrishnan

We were fortunate to again have the services of Arup Nanda, John Kanagaraj, and Sandesh Rao for reviewing the drafts of this book. This book is immensely better due to the painstaking, conscientious technical reviews performed by these three well-known Oracle experts. Of course, the authors are solely responsible for any remaining errors and omissions.

I'd like to take this opportunity to thank my coauthor, Gopal, who kindly invited me to join him in revising the previous edition of his popular book.

Much work was performed behind the scenes by the publishing team to keep this book on track. For the past two years, both Wendy Rinaldi and Claire Yee have hung in there with us, the authors, encouraging and supporting us while we revised the book, first for the Oracle Database 12.1 release, and again for the Oracle Database 12.2 release. I'd like to express my heartfelt thanks for the warm support provided to me, as well as for the superb professionalism exhibited by Wendy Rinaldi (editorial director) and Claire Yee (editorial coordinator) at McGraw-Hill Education. Thank you, Wendy and Claire, for never losing faith in us and for hanging in there with a lot of grace and patience until the end of a very long process! Claire, extra thanks for your persistence, and for keeping track of everything so well—without your hard work, we'd still be at it!

I'd like to express my thanks for the great copyediting job done by Bart Reed. His work went way beyond what most copy editors do. Bart not only focused on the language, style, and so on, but also on the context—he caught several potential errors by bringing them to my attention. I was fortunate to have someone with the high level of editing skills that Bart has brought to this project. The project editor of the book, Patty Mon, was also instrumental in keeping the flow of the book-making process going, but she also contributed in various ways to improve the quality of the writing in every chapter.

I have dedicated this book to Eligio (Eddie) Colon, a truly remarkable person who I'm proud to call my friend. I'll always owe Eddie, but this book is a small token to let him know that I still remember, and that I appreciate everything he has done for me,

https://t.me/learningnets

and for his kind ways. Thank you, Eddie!

I'd like to express my gratitude for the attention my dear father, Dr. Appa Rao, has bestowed on me. All my scholastic accomplishments are solely due to my father's kind interest in me and his enormous affection. I would also like to acknowledge the love and help of my family—my mother, Swarna Kumari, and my brothers, Hari Hara Prasad and Siva Sankara Prasad. I also don't want to forget Aruna, Vanaja, Teja, Ashwin, Aparna, and Soumya. My wife, Valerie, has provided the peaceful and helpful environment for working on the book. My children, Nina and Nicholas, have always encouraged me to do my very best by simply loving me the way they do!

—Sam R. Alapati

https://t.me/learningnets

Introduction

Whatever is heard from whosever's mouth,
Wisdom will rightly discern its true meaning.
(Kural 423: Thiruvalluvar)

Gopal, one of the two authors of this book, used to work as a consultant and visited many sites for Oracle RAC implementations around the globe. His work involved answering simple questions, such as what platform to choose for RAC, as well as working on very complex performance problems. Most of the customers Gopal met complained that there is no definite text available on the subject, and many of them still treat RAC as a "black box." This is true even for seasoned database administrators who have been working on Oracle databases for many years!

Although a number of other books on the market are devoted to Oracle RAC, we have not seen a single book that covers the complete spectrum of topics related to RAC. As a result, there is a significant gap in knowledge about RAC internals and nuances in the public domain. Gopal's failure in finding any such book was the impetus for writing the first edition of this book. Gopal was also emboldened as his other book (*Oracle Wait Interface: A Practical Guide to Performance Diagnostics & Tuning*) was a great success and even fetched him *Oracle Magazine*'s Editor's Choice "Oracle Author of the Year" Award in 2005.

As in the previous edition of this book, our goal is to explain how to implement and use RAC in the most efficient manner, rather than providing a theoretical overview about the grid. If you look at the contents, you will find that we do not discuss anything about the grid technologies or grid architecture. Similarly, you will not find any details about grid management, including the Enterprise Manager or Grid Control and any of the fancy technical concepts.

We believe in the proverb "Give a man a fish, and you feed him for a day. Teach a man to fish, and you feed him for a lifetime." Thus, the purpose of this book is to

https://t.me/learningnets

provide a solid background in the fundamentals of Oracle Real Application Clusters rather than to provide a heap of commands and syntax that are readily available in the standard documentation and other Oracle texts.

Although this book covers a spectrum of topics related to RAC, it is still by no means complete—it's just the start of a long and exciting journey. The expectation is that you can use this book as a reference and concepts guide (with a long shelf life) rather than using it for a specific release.

Some of the concepts discussed here are truly complex, and we recommend that you skip them during the first read. Once you are done reading the book, revisit these chapters a few times until you obtain a clear understanding of the concepts.

You may also notice that some of the technical topics are treated very lightly. As a DBA, you just need to have a basic but solid understanding about the architecture of Oracle and the way it works. We believe that discussions about some deep technical aspects of Oracle RAC are impractical and not really required everywhere. For example, the inner workings of Cache Fusion and distributed lock managers cannot be explained in a single chapter. In our opinion, each topic deserves an independent book of its own. Therefore, be aware that the discussions are modeled according to the context of this title.

The book is organized in five distinct parts. Part I deals with the history and architecture of the high availability clusters and compares the various clustering architectures. It also dives deep into Oracle clustering architecture, including Oracle Parallel Server and the evolution of the Oracle clustering technologies. We also discuss the architecture of Real Application Clusters and Oracle kernel components that make RAC a working solution.

Chapter 1 talks about high availability architecture and clusters. Here, you will learn the most common techniques used for availability and see the impact of planned and unplanned downtime for the business. We also discuss the most common solutions used to achieve high availability and scalability. Hardware clustering is the most commonly used method to achieve high availability and "on-demand" scalability.

Chapter 2 introduces users to the rich history of RAC with some details about the basics of clustering technologies and a discussion on the early days of Oracle Parallel Server. We also discuss the intrinsic limitations of Oracle Parallel Server and how RAC overcomes those limitations with the new techniques.

Chapter 3 introduces Oracle RAC architecture and the components that make it work. We explain why global coordination is required for RAC and also briefly discuss RAID, because shared storage is key in the RAC infrastructure. We also introduce Oracle's Automatic Storage Management (ASM) and other technologies such as Oracle Grid Infrastructure.

Part II of this book deals with the installation of Real Application Clusters software and ASM. It provides basic details of preparing the hardware for Real Application Clusters and installing RAC in generic Unix and Linux environments. The fundamentals of Automatic Storage Management are reviewed in this section.

Chapter 4 is about preparing the hardware for RAC installation. A proper and sound

https://t.me/learningnets

preparation of the hardware for the RAC installation tasks is the key to a successful deployment. Oracle Grid Infrastructure is the Oracle Clusterware, which logically binds the servers at the operating system level, and we discuss installing the Oracle Clusterware in this chapter.

Chapter 5 deals exclusively with the installation of RAC on the cluster. We show how to install Oracle RAC step-by-step, with the help of screenshots. We also use the new Cluster Verification Utility to validate the integrity of the installation.

Chapter 6 is on Automatic Storage Management. ASM is a database file system from Oracle, and you'll learn how to manage disk groups and administer disk groups in the ASM environment. You also learn the enhancements to ASM, such as ASM Cluster File System (ACFS) and various command-line tools. The Oracle-provided utility ASMLib is also briefly discussed in this chapter.

Part III covers the generic administration of the RAC databases. It has notes on the basic administration of a RAC database and lists the similarities and differences between single-instance management and Real Application Clusters database management. The RAC performance management chapter introduces the most common problems, issues, and wait scenarios in Oracle RAC, along with potential solutions where applicable. In addition to advanced administration, this chapter introduces service management in RAC.

Chapter 7 talks about administering an Oracle RAC database from a DBA's perspective. Administering RAC is similar to administering a single-instance database, with some changes. We look at the considerations for RAC database administration and also cover the administration topics for Oracle Cluster Ready Services and voting disks.

Chapter 8 deals with administering services in the RAC environment. "Services" is a relatively new concept in databases that simplifies resource management and workload distribution and provides high availability to the workload. Oracle Clusterware command-line interfaces are also discussed in detail in this chapter.

Chapter 9 talks about backup and recovery concepts for RAC as well as the concepts of instance and database recovery in Real Application Clusters. We do not discuss the commands and the syntax involved in the backup and recovery procedures, which are similar to those you use for a single-instance Oracle database. This chapter delves deep into the architecture of recovery in a single instance and provides good insight into the different types of recovery in an Oracle RAC database.

Chapter 10 deals with performance management in RAC. Managing and achieving good performance in any system is probably the primary objective, and even more so in RAC. Performance tuning of Real Application Clusters needs a few additional considerations when compared to single-instance tuning because of the additional instances accessing the same set of resources, and we discuss them in detail in this chapter. We discuss RAC-related Oracle Wait Interface events in fine detail. This chapter also offers advice on tuning those events.

Part IV of this book is intended for advanced topics. We get into the resource management aspect of a Real Application Clusters environment and discuss how the

https://t.me/learningnets

resources are shared and managed. We discuss the Global Cache Services and Global Enqueue Services and their inner workings. We also discuss the Cache Fusion topics in more detail. This includes a complete overview of how things worked in the past and how Cache Fusion has changed the dynamics of data sharing between the instances.

Chapter 11 provides a detailed discussion of the Global Resource Directory (GRD) and what it does. You learn about the different locking and serialization mechanisms as well as their importance and relevance in operating an Oracle RAC database. You should note that the discussions are quite deep in nature, so you should plan to read the chapter a few times to grasp its contents effectively. The GRD and resource-mastering issues are discussed in this chapter.

Chapter 12 provides most of the details you'll need to know about Cache Fusion. We look at how Cache Fusion *really* works, with examples and demonstrations. This is the most important chapter in the book, and understanding it will help you appreciate the intelligence built into this component. It will also help you significantly in designing a scalable RAC solution and debugging most complex performance problems. You learn some of the exciting internal techniques of how RAC works, and we provide a well-documented example to help you understand the facts easily.

Chapter 13 introduces workload management from Oracle's perspective. You also learn about Transparent Application Failover (TAF) and how to implement it. Oracle 12*c* has enhanced Fast Application Notification (FAN) and starting with this release, FAN is configured and enabled out of the box with Oracle Real Application Clusters. This chapter discusses both TAF and FAN and their usage.

Chapter 14 discusses RAC troubleshooting—one of the least-known RAC topics. This chapter provides you with some methods you can use to quickly troubleshoot a misbehaving RAC instance. This chapter talks about troubleshooting from an operational perspective and introduces performance diagnosis as well as troubleshooting instance recovery.

Part V covers deploying RAC, including extending RAC to a geo-cluster environment, and common application development best practices. The most common RAC application development techniques are discussed here.

Chapter 15 is on Extended RAC and discusses extending RAC in a WAN environment. Oracle RAC is most commonly used as a scalability and availability solution. However, in specific cases, it could also be used as a disaster recovery solution. This configuration is known as *extended RAC clustering*, and we discuss the most common issues related to this topic.

Chapter 16 introduces some of the best practices in application development for RAC. We cover some of the most commonly found issues in an Oracle RAC environment and provide some best-practice methods for overcoming those challenges.

The most useful and commonly used V$ views are explained in Appendix A. The dynamic performance views are grouped together based on their use. Appendix B discusses adding and removing nodes to and from the cluster.

https://t.me/learningnets

By no means have we attempted to cover all aspects of Oracle RAC. We believe this book is just the beginning.

As the authors, we would like to hear from you about how you liked the book and what could be done to improve it. Please feel free to contact us at kaygopal@yahoo.com and samalapati@yahoo.com with whatever comments you have on this book.

—K. Gopalakrishnan
—Sam R. Alapati

https://t.me/learningnets

PART I

High Availability Architecture and Clusters

https://t.me/learningnets

Technet24

CHAPTER 1

Introduction to High Availability and Scalability

In today's fast-paced world, data and application availability can make or break a business. With access to these businesses granted via the ubiquitous and "always-on" Internet, data availability is an extremely important component in any business function.

Database systems are growing at an enormous rate in terms of the number of simultaneously connected and active users as well as the volume of data they handle. Even though the servers that were used for hosting databases have also improved in performance and capacity, a *single* server, powerful though it may be, is frequently unable to handle the database load and capacity requirements for these active databases. This factor makes it necessary to scale the processing capacity or to scale the hardware or software to accommodate these requirements.

High Availability

When availability is crucial for a business, extremely high levels of disaster tolerance must allow the business to continue in the face of a calamity, without the end users or

https://t.me/learningnets

customers noticing any adverse consequences. The effects of global companies conducting business across time zones spanning “24/7/forever” operations, e-commerce, and the challenges associated with today’s “flat world” all drive businesses to maintain a level of disaster tolerance capable of ensuring continuous survival and profitability.

Different businesses require the adoption of varying levels of risk with regard to loss of data and potential downtime. You can use a range of technical solutions to provide varying levels of protection with respect to these business needs. The ideal solutions would have no downtime and allow no data to be lost. Although such solutions do exist, they are expensive, and hence their costs must be weighed against the potential impact of a disaster and its effects on the business.

Because computers are capable of working at faster and faster rates, the businesses that depend on them are placing more and more demands on them. As a result, the various interconnections and dependencies in the computing fabric, consisting of different components and technologies, are becoming more complex every day. The availability of worldwide access via the Internet is placing extremely high demands on businesses as well as the IT departments and administrators that run and maintain these computers in the background.

Adding to this complexity is the globalization of businesses, which ensures there is no “quiet time” or “out-of-office hours” that are so essential to the maintenance requirements of these computer systems. Hence, businesses’ computer systems—the lifeblood of an organization—must be available at all times: day or night, weekday or weekend, local holiday or workday. The term *24/7/forever* effectively describes business computer system availability. This term is so popular that it is being used in everyday language to describe non-computer-based entities such as 911 call centers and other emergency services.

The dictionary defines the word *available* as follows:

1. Present and ready for use; at hand; accessible.
2. Capable of being gotten; obtainable.
3. Qualified and willing to be of service or assistance.

When applied to computer systems, the word’s meaning is a combination of all these factors. Thus, access to an application means that the application should be present and ready for use, capable of being accessed, and qualified and willing to be of service. In other words, an application should be available easily for use at any time and should perform at a level that is both acceptable and useful. Although this is a broad, sweeping statement, a lot of complexity and different factors come into play before true high availability is achieved and sustained.

HA Terminology

The term *high availability* (*HA*), when applied to computer systems, means that the application or service in question is available all the time, regardless of time of day,

https://t.me/learningnets

Technet24

location, and other factors that can influence its availability. In general, high availability refers to the ability to continue a service for extremely long durations without any visible or impactful interruptions. All HA systems fail because the components used to build them fail, but they are designed and built so that failure of an individual component does not affect the overall availability. Thus, the right HA systems fail without externally visible or impactful interruptions.

Typical technologies for HA include redundant power supplies and fans for servers, RAID (Redundant Array of Inexpensive/Independent Disks) configuration for disks, clusters for servers, multiple network interface cards (NICs), redundant routers for networks, and even multiple datacenters within the same metro area to provide an extremely high level of availability and load balancing.

Fault Tolerance

A *fault-tolerant* computer system or component is designed so that, in the event of component failure, a backup component or procedure can immediately take its place with no loss of service. Fault tolerance can be provided with software, embedded in hardware, or provided by some combination of the two. It goes one step further than HA to provide the highest possible availability within a single datacenter and within a single application execution environment such as a database.

Disaster Recovery

Disaster recovery (DR) is the ability to resume operations after a disaster—including destruction of an entire datacenter site and everything in it. In a typical DR scenario, significant time elapses before a datacenter can resume IT functions, and some amount of data typically needs to be reentered to bring the system data back up to date.

Disaster Tolerance

Disaster tolerance (DT) is the art and science of preparing for disasters so that a business is able to continue operation after a disaster. The term is sometimes used incorrectly in the industry, particularly by vendors who can’t really achieve it. DT is much more difficult to achieve than DR because it involves designing systems that enable a business to continue in the face of a disaster, without the end users or customers noticing any adverse effects. The ideal DT solution would result in no downtime and no lost data, even during a disaster. Such solutions do exist, but they cost more than solutions that have some amount of downtime or data loss associated with a disaster.

In order to provide disaster tolerance, Oracle Database 12*c* provides several features such as Active Data Guard (available in Oracle Database 11*g* and above, and greatly improved in Oracle Database 12c), Far Sync to provide zero data loss at any distance, Fast-Start Failover to enable both HA and DR, Flex ASM, Global Data Services to efficiently route clients to the nearest and best copy, and Transaction Guard with Application Continuity to enable applications to recover quickly from a disaster without even loss of in-flight data.

https://t.me/learningnets

Planned and Unplanned Outages

So what happens when an application stops working or stops behaving as expected, due to the failure of even one of the crucial components? Such an application is deemed *down* and the event is called an *outage*. This outage can be planned for—for example, consider the outage that occurs when a component is being upgraded or worked on for maintenance reasons.

Whereas planned outages are a necessary evil, an unplanned outage can be a nightmare for a business. Depending on the business in question and the duration of the downtime, an unplanned outage can result in such overwhelming losses that the business is forced to close. Regardless of the nature, outages are something that businesses usually do not tolerate. There is always pressure on IT to eliminate unplanned downtime totally and to drastically reduce, if not eliminate, planned downtime. Later in this book, you'll learn how these two requirements can be effectively met, for at least the Oracle database component, through the use of Oracle Database RAC 12*c*.

Note that an application or computer system does not have to be totally down for an outage to occur. It is possible that the performance of an application degrades to such a degree that it is unusable. In this case, although the application is accessible, it does not meet the third and final qualification of being willing to serve in an adequately acceptable fashion. As far as the business or end user is concerned, this application is down, although it is available. You will see later in this book how Oracle Real Application Clusters (RAC) can provide the horizontal scalability that can significantly reduce the risk of an application not providing adequate performance. When combined with features such as Global Data Services and Active Data Guard, Oracle Database 12*c* can provide much-needed relief during both planned and unplanned downtime.

An End-to-End Perspective

From the start, you should be clear that high availability is not just dependent on the availability of physical components such as hardware, system software (operating system and database), environment, network, and application software. It is also dependent on other "soft" resources such as experienced and capable administrators (system, network, database, and application specialists), programmers, users, and even known, repeatable business processes.

It is entirely possible that a business installs and configures highly available "hard" components but does not employ competent administrators who are able to maintain these systems properly. Even if the administrators are competent, availability can be adversely affected when a business process, such as change control, is not followed properly, or when incorrect and untested changes are made that could bring such a system down. High availability therefore needs to be seen with an end-to-end perspective that covers all aspects.

Having said this, we should now define the term *single point of failure* (SPOF),

https://t.me/learningnets

Technet24

which refers to any single component that can bring down the entire system as a result of failure. For example, in a computer system that has a single controller interfacing with the disk subsystem, a hardware failure of this controller will bring the whole system down. Although the other components are working, this one *single* component has caused a failure. Identification of and protection against SPOFs are crucial tasks of providing HA.

It's not possible to cover all aspects of HA in an Oracle-specific book such as this. We focus mostly on how you set up and administer HA with an Oracle RDBMS, which is an important component of the overall HA picture for an organization. We'll also equip you—the database administrator, big data administrator, Oracle Database machine administrator, Oracle Exadata administrator, programmer, or architect—with techniques that enable you to achieve HA in this area.

HA is not something that can be achieved simply by installing HA-aware hardware and software components, employing competent administrators, creating proper procedures, and walking away from it all. The HA process requires continual adjustment in order to evolve and adapt to changing circumstances and environments. Also, this uphill battle occurs on a continual basis—so be prepared!

What's more, you should periodically review the HA architecture both from "soft" changes in the environment as well as the new features that later versions of the Oracle database provide. We have already listed some of these features, including Active Data Guard and Global Data Services.

Cost of Downtime

As hinted at earlier, there is a cost to downtime, just as there is a cost to ensuring that downtime is drastically reduced or even completely eliminated. The trick is to build your systems so that they never fail, even though you *know* that they *will* fail at some time or another. Making downtime the last option will ensure HA. Of course, most companies cannot continue to throw large sums of money at this issue. At some point in time, the additional money spent will return only marginal benefits. Therefore, it is essential to price out your downtime and then use that figure to determine how much you can afford to spend to protect against planned/unplanned downtime. With some effort and experience, this expense can be determined, and you might want to use this information while providing various options and scenarios to management.

The cost of failure of a system usually amounts to lost user productivity, and the actual cost is mostly dependent on what work the users perform when accessing the affected systems. For example, if your development server went down for one hour during prime office time, and 10 developers sat idle for that hour waiting for the server to come up, and each developer costs $100 per hour, then the downtime has effectively cost $100 × 10 × 1 = $1,000. However, if the server that went down served a major shopping site on the Internet during a holiday gift-buying season, you might count the losses in millions of dollars, even if the downtime was brief, because shoppers may move away to a competing site rather than wait for yours to become usable. Figure 1-1 shows a sample chart comparing downtime to cost.

https://t.me/learningnets

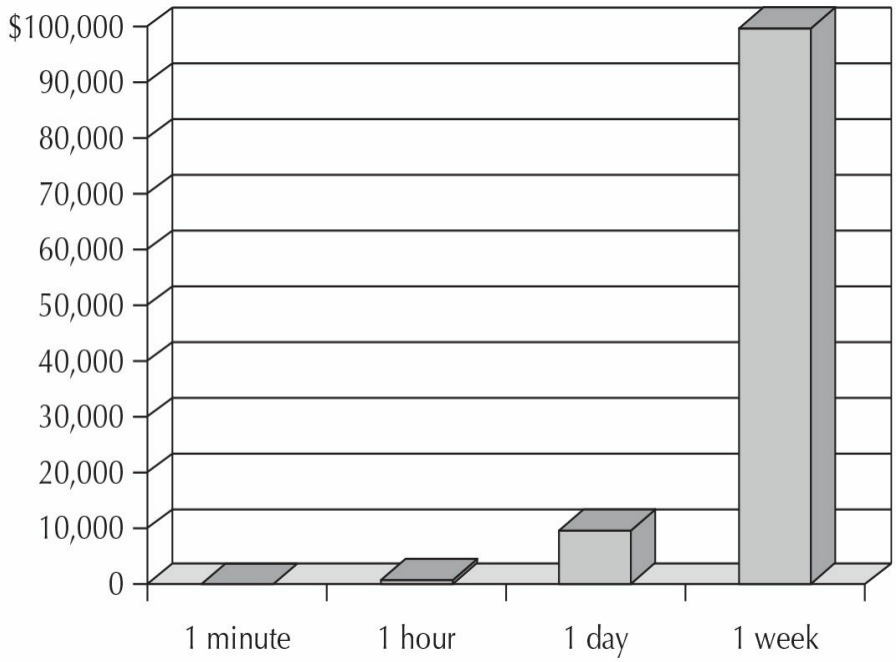


FIGURE 1-1. *Cost of downtime*

The potential cost of downtime is also dependent on various factors such as time of day and duration of the downtime. For example, an online stock brokerage firm cannot afford to be down even for seconds during business hours. On the other hand, it could go down for hours during nontrading hours without any consequences. Cost of downtime is not linearly dependent on the duration of the downtime. For example, a two-hour outage may not necessarily cost the same as two one-hour downtime periods.

One helpful trick used with balancing the cost of downtime versus the cost of ensuring against downtime is the "availability curve." The more you spend on HA components, the higher you move up the curve. However, the incremental costs of moving from one level to the next increase as you move up the curve.

Here are the four distinct levels of system availability components on the curve:

- **Basic systems** These are systems with no protection or those that employ no special measures to protect their data and accessibility. Normal tape backups occur at scheduled intervals, and administrators work to restore the system from the last-known good backup if and when it breaks. There is no extra cost for HA.
- **Redundant data** Some level of disk redundancy is built into the system to protect against loss of data due to disk failures. At the most basic level, this is provided by RAID 5 or RAID 1–based disk subsystems. At the other end of the scale, redundancy is provided by storage area networks (SANs) that have built-in disk protection mechanisms such as various RAID levels, hot-swappable disks, "phone home"–type maintenance, and multiple paths to the SAN. The cost of such protection includes procurement of the SAN, attendant SAN fabric

https://t.me/learningnets

Technet24

and controllers, as well as extra sets of disks to provide RAID protection.

- **System failover** In this case, two or more systems are employed to do the work of one. When the primary system fails, the other (usually called the "secondary" system) takes over and performs the work of the primary. A brief loss of service occurs, but everything quickly works as it did before the failure. The cost of this solution is more than double that of basic systems. Usually, a SAN needs to be employed to make sure that the disks are protected and to provide multiple paths to the disks from these servers.
- **Disaster recovery** In this case, in addition to the systems at the main site (which in themselves may incorporate the previous highest level of protection), all or part of these systems are duplicated at a backup site that is usually physically remote from the main site. You must develop ways of replicating the data and keeping it up to date. The costs are more than double that of the previous level, because you will also have to duplicate an entire site, including datacenters, real-estate facilities, and so on.

As you can easily see, higher and higher levels of availability equate to escalating costs. When faced with even a rough estimate of cost, business leaders (and especially accounting staff) are quick to adjust their levels of expectancy.

Underpinning every aspect, of course, is the fact that you are monitoring, measuring, and recording all this uptime (or downtime, as the case may be). It is a given that you cannot quantify what you do not measure. Nevertheless, many organizations that demand 100 percent uptime do not even have basic measurement tools in place.

Five Nines

The phrase "five nines" is usually thrown about during discussions of high availability, and you need to understand what this means before agreeing (as an administrator or system architect) to provide such a level of availability. A user or project leader will invariably say that 100 percent availability is a necessity, and barring that, at least "five nines" availability must be maintained—that is, 99.999 percent availability.

To make this concept a bit clearer, Table 1-1 compares uptime and downtime percentages to real-time figures. As you study this table, keep in mind that the cost of providing higher and higher levels of uptime becomes progressively (and sometimes prohibitively) expensive. As you work with management, understanding this can help you provide a clear explanation of these terms and what they mean when translated to actual downtime and attendant costs.

https://t.me/learningnets

Percent Uptime	Percent Downtime	Downtime per Year	Downtime per Week
98	2	7.3 days	3 hours, 22 minutes
99	1	3.65 days	1 hour, 41 minutes
99.8	0.2	17 hours, 30 minutes	20 minutes, 10 seconds
99.9	0.1	8 hours, 45 minutes	10 minutes, 5 seconds
99.99	0.01	52.5 minutes	1 minute
99.999	0.001	5.25 minutes	6 seconds

TABLE 1-1. *Uptime and Downtime Percentages with Real-Time Figures*

Building Redundant Components

High availability is made possible by providing availability in multiple layers of the technical stack. The inclusion of redundant components that reduce or eliminate SPOFs is the primary key in achieving high availability. For example, more than one host bus adaptor (HBA), a controller for communicating with remote disks, is usually present in each server that connects to a SAN. These HBAs, in turn, are able to connect into two or more network adaptor switches to which the SANs are themselves connected. This way, the failure of one HBA or even one network switch will not bring down the server and the application hosted on that server. *Multihosting* (the ability to attach multiple hosts to a single set of disks) and *multipathing* (the ability to attach a single host to its set of disks via more than one path) are common ways of introducing redundancy in such HA systems.

Redundant components exist in the software layers as well. For example, multiple web servers can be front-ended by a load balancer that directs all web requests to a bank of web servers. In this case, when one web server fails, existing connections migrate over to surviving web servers, and the load balancer connects new requests to these surviving web servers.

Redundancy is not restricted to hardware and software, however. Redundancy also includes building physical, environmental, and other elements into the framework. Most of the major Internet datacenters or Internet exchange points now have complete redundancy in terms of power, air conditioning, and other factors, so that the failure in any one of the provider's resources won't affect the operation.

In New York City, for example, two telecommunication systems were strategically placed in the erstwhile World Trade Center complex—one in each tower—with the assumption that the probability of both buildings collapsing was close to zero. However, unfortunately, that assumption was proved wrong. Now, companies are building redundant datacenters that are geographically separated across state or even country boundaries to avoid natural or other catastrophic events. Availability of dark

https://t.me/learningnets

Technet24

fibers and the improvements in technology such as dense wavelength division multiplexers (DWDMs) make this possible. The reason why dark fibers and DWDM technologies enable this geographic dispersion of datacenters is high throughput and low latency.

Redundancy in the network layer is achieved through the redundant hardware engines in a chassis, a redundant network through multiple chassis, or a combination of the two. Host protocols such as ICMP Route Discovery Protocol (IRDP), Cisco's Hot Standby Routing Protocol (HSRP), and Virtual Router Redundancy Protocol (VRRP) help choose the best next-hop router to reach if one of the routers is unavailable from the server's perspective. In the routing level, Non-Stop Forwarding (NSF) protocol suites combined with millisecond timers reduce the failure or switchover time in case of primary hardware switching engine failure.

In the transport level, physical layer redundancy can be achieved by SDH/SONET self-healing, which restores the traffic in an alternate path in case of fiber link failure. A major transport provider experienced a fiber cut in its long-haul, coast-to-coast transport network in the United States and rerouted the traffic through Europe without most of the end users knowing that the rerouting even took place.

Finally, it is now possible to provide redundant database services via Oracle RAC, and you will see this in detail in subsequent chapters. Suffice it to say at this time that redundancy in database services is an important part of providing HA in the organization, and Oracle RAC enables such a provision.

Of course, adding redundancy into the system also increases its cost and complexity. We hope that the information contained in this book can help ease your fears about managing such a complex environment.

Common Solutions for HA

Depending on your budget, you can arrive at a number of solutions for providing high availability. Using clustering servers has been a common way to build a highly available and scalable solution. You can provide increasing levels of HA by adopting one of the higher levels of protection described earlier. In most current datacenters, RAID disks (usually in SANs) provide at least a basic level of disk protection. Failover servers at the third level provide some protection from server failure. At the highest level, the disaster recovery site protects against drastic site failure.

Oracle technology can be used to provide all these levels of protection. For example, you can use Automatic Storage Management (ASM) to provide protection at the disk level, Oracle RAC to provide failover protection at the database level (in addition to database-level load balancing), and Oracle DataGuard and Oracle replication to provide site failure protection.

You can use Oracle RAC–related features such as Flex ASM for better ASM redundancy, HANFS for ACFS availability with NFS, Flex Cluster for availability for applications, Application Continuity with Transaction Guard for failover, Memory Guard with workload management, an in-memory persistent store to move data across nodes, as well as storage availability using Exadata and ZFS.

https://t.me/learningnets

Cluster, Cold Failover, and Hot Failover

Although we will be dealing with clustering in detail in subsequent chapters, we will define it here. A *cluster* is a set of two or more similar servers that are closely connected to one another and usually share the same set of disks. The theory is that in the case of failure of one of the servers, the other surviving server (or servers) can take up the work of the failed server. These servers are physically located close to one another and connected via a "heartbeat" system. In other words, they check one another's heartbeats or live presence at closely defined intervals and are able to detect whether the other node is "dead" within a short period of time. When one of the nodes is deemed nonresponsive to a number of parameters, a failover event is initiated and the service of the nonresponsive node is taken over by other node(s). Additional software may also allow a quick takeover of one another's functions.

Clusters can be implemented in many configurations. When one or more servers in a cluster sit idle, and takeover from another server (or servers) occurs only in the case of a failure, a *cold failover* occurs. When all servers in a cluster are working, and the load is taken on by the surviving server (or servers), this is called a *hot failover*. Assuming that all the servers in the cluster are similar in configuration, in a cold failover, the load carried by the surviving server is the same. In the case of a hot failover, however, the load taken on by the surviving server may be more than it can handle, and thus you will need to design both the servers and the load carefully.

The three general approaches to system failover, in order of increasing availability, are *no failover*, *cold failover*, and *hot failover*. Each strategy has a varying recovery time, expense, and user impact, as outlined in Table 1-2.

Approach	Recovery Time	Expense	User Impact
No failover	Unpredictable	No to low cost	High
Cold failover	Minutes	Moderate	Moderate
Hot failover	Immediate or in seconds	Moderate to high	None*

*To be precise, saying there is no user impact in a hot failover scenario is inaccurate. Very few systems are truly "hot" to the point of no user impact; most are somewhat "lukewarm," with a transient brownout.

TABLE 1-2. *Failover Approach and Impacts*

Variations on these strategies do exist. For example, many large enterprise clients have implemented hot failover and also use cold failover for disaster recovery. It is important to differentiate between failover and disaster recovery. *Failover* is a methodology used to resume system availability in an acceptable period of time, whereas *disaster recovery* is a methodology used to resume system availability when all failover strategies have failed.

https://t.me/learningnets

Technet24

No Failover

If a production system fails due to a hardware failure, the database and application are generally unaffected. Disk corruption and disk failures, of course, are an exception. Therefore, disk redundancy and good backup procedures are vital to mitigate problems arising from disk failure.

With no failover strategy in place, system failures can result in significant downtime, depending on the cause and your ability to isolate and resolve them. If a CPU has failed, you replace it and restart, while application users wait for the system to become available. For many applications that are not business critical, this risk may be acceptable.

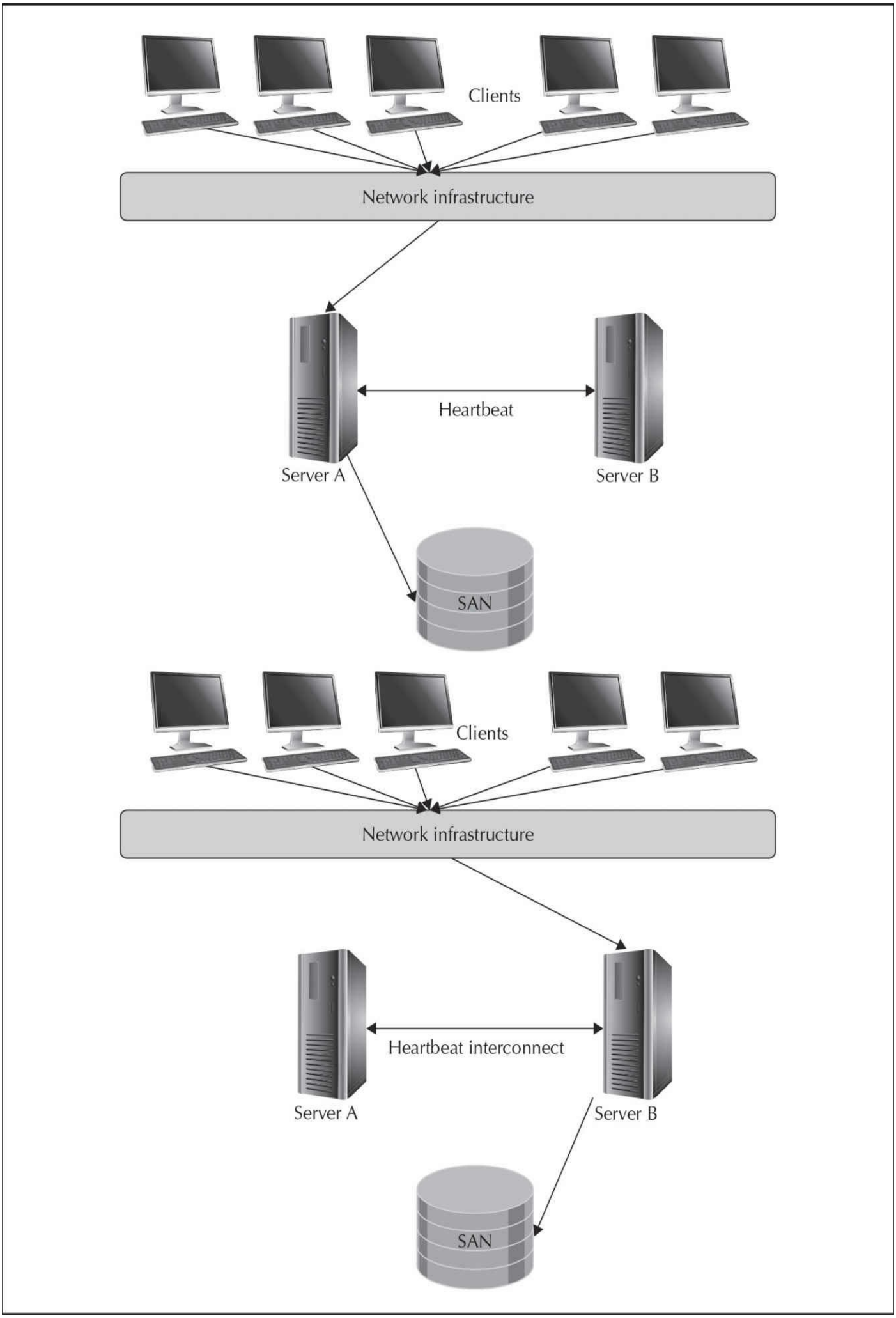
Cold Failover

A common and often inexpensive approach to recovery after failure is to maintain a standby system to assume the production workload in the event of a production system failure. A typical configuration has two identical computers with shared access to a remote disk subsystem.

After a failure, the standby system takes over the applications formerly running on the failed system. In a cold failover, the standby system senses a heartbeat from the production system on a frequent and regular basis. If the heartbeat consistently stops for a period of time, the standby system assumes the IP address formerly associated with the failed system. The standby can then run any applications that were on the failed system. In this scenario, when the standby system takes over the application, it executes a preconfigured start script to bring the databases online. Users can then reconnect to the databases that are now running on the standby server.

Customers generally configure the failover server to mirror the main server with an identical CPU and memory capacity to sustain production workloads for an extended period of time. Figure 1-2 depicts server connections before and after a failover.

https://t.me/learningnets



https://t.me/learningnets

FIGURE 1-2. *Server connections before (top) and after (bottom) a failover*

Hot Failover

The hot failover approach can be complicated and expensive, but it comes closest to ensuring 100 percent uptime. It requires the same degree of failover used for a cold failover but also requires that the state of a running user process be preserved to allow the process to resume on a failover server. One approach, for example, uses a three-tiered configuration of clients and servers. Hot failover clusters are normally capable of client load balancing. Oracle RAC supports hot failover configuration by transparently routing the incoming connections to the services in surviving nodes.

Table 1-3 shows the load distribution of a 3,000-user workload in a three-node cluster. During normal operation, all nodes share approximately an equal number of connections; after failover, the workload from the failed node will be distributed to surviving nodes.

State	A	B	C
Normal	1,000 users	1,000 users	1,000 users
B fails	1,000 users	0 users	1,000 users
Users previously connected to B log on again	1,500 users	0 users	1,500 users

TABLE 1-3. *Workload Distribution During Cluster Failovers*

The 1,000 users on servers A and C are unaware of server B's failure, but the 1,000 users who were on the failed server are affected. However, systems A, B, and C should be appropriately configured to handle the additional load during unexpected node failures. This is one of the key elements during the capacity planning for clusters.

Table 1-4 summarizes the most common aspects of cold failover versus hot failover.

https://t.me/learningnets

Aspects	Cold Failover	Hot Failover
Scalability/ number of nodes	Scalability limited to the capacity of the single node.	Because nodes can be added on demand, it provides infinite scalability. High number of nodes supported.
User interruption required	Required up to a minimal extent. The failover operation can be scripted or automated to a certain extent.	Not required. Failover is automatic.
Transparent failover of applications	Not possible.	Transparent application failover will be available where the sessions can be transferred to another node without user interruption.
Load balancing	Not possible; only one server will be used.	Incoming load can be balanced between both nodes.
Usage of resources	Only one server at a time; the other server will be kept idle.	Both the servers will be used.
Failover time	Several minutes because the other system must be cold started.	A few minutes (depends on the load).

TABLE 1-4. *Cold Failover vs. Hot Failover*

NOTE

When a RAC cluster supports thousands of processes on databases accessing hundreds of table using a large System Global Area (in excess of 500GB), failure results in brownouts of two minutes during RAC reconfiguration. The overhead of starting a few thousand connections can also add two to three minutes on top, leading to a total of five minutes or so. It's worth pointing out here that a "99.999" availability goal will be breached with just one such hot-failover incident!

HA Option Pros and Cons

Each HA option has its own advantages and disadvantages. Costs of setup and running the service are important to consider when deciding which HA option to use. At the end of the day, as an administrator or system architect, you are responsible for costing out the various options and helping management decide what is best. What's more,

https://t.me/learningnets

Technet24

you will need to figure in the additional complexity of maintaining various configurations, remembering that as you add more redundancy into the system, you are also increasing the options for failure when handling these now complex configurations. In addition, employing consultants or engaging third-party vendor professional services to set up these complex configurations, deploying additional hardware and software, and maintaining these systems can also quickly add to the basic costs.

Scalability

As mentioned at the beginning of the chapter, even powerful servers cannot always handle database load and capacity requirements. Server scalability can be improved using one or both of the following methods:

- Increase the processor count on the system. That is, *scale up* the computing resources.
- Increase the amount of work done in a given period of time via application tuning. That is, *speed up* the processing.

The most common view of scaling is that of hardware scaling, which has at least as much to do with the software components as with the hardware. But what do you do when you cannot increase the processor count because you have reached the maximum capacity for that line of servers, or when you have tuned all the workloads as best you can and no more tuning opportunities exist?

Initial solutions to these problems include the use of multiple application copies and databases, but these result in data-sync problems and other process issues. The best solution, of course, is the use of clustered servers that can collectively perform much better than a single server for many applications. In other words, we can use clusters of servers to *scale out* (also known as *horizontal scalability*) rather than *scale up* (also known as *vertical scalability*). It is in the provision of horizontal scalability where Oracle Real Application Clusters (RAC) excels.

The Oracle RAC Solution

Essentially, Oracle RAC provides the ability for multiple servers to consistently access a single copy of the data. Theoretically, as the requirement to access this single copy increases, you keep adding nodes to the cluster. This ability to provide consistent access is not simple—the process requires a lot of coordination between the various nodes in the cluster. Oracle RAC does this efficiently, and you will see exactly how this occurs in later chapters.

Although Oracle RAC scales well, there is an upper limit on horizontal scalability. In general, application scalability is based on how good the application works in a single instance. If the SQL statements executed by the application are efficient and use an expected and reasonable amount of resources (usually measured by logical I/O

https://t.me/learningnets

and/or physical I/O counts), you can generally expect this to scale well. In other words, you might compare Oracle RAC to a stereo amplifier: If the quality of the recording, whether on an audio tape or a digital device, is bad, placing even the best amplifier in front of it will not solve the problem. Instead, it will amplify the problem and make the situation unpleasant. This is also applicable for Oracle RAC or any other scalability solution. Hence, you will need to make sure application-level tuning is performed to remove bottlenecks before using clustering to scale out.

With the constant pressure on improving total cost of ownership (TCO), businesses have chosen to move away from "Big Iron," monolithic servers to smaller sets of lower-cost "commodity" servers. And this is where Oracle RAC is truly in its element, because it helps businesses realize this paradigm shift by enabling large workloads to run on clusters of lower-cost servers rather than single, monolithic boxes. Also, such servers are able to scale up or down to the workload easily. Many features in Oracle Database 12*c* RAC, such as server pools and SCAN (Single Client Access Name) listeners (discussed in later chapters), provide the ability to perform this scaling seamlessly without interruption to the business.

Along with near-linear scalability, Oracle RAC–based systems can be configured to eliminate SPOF as far as the database layer is concerned. When database servers fail, applications based on Oracle RAC systems simply keep running by shifting the load to surviving nodes. Optionally, you can use preferred/available nodes combined with service-level resource management to provide the required level of service to specific applications. When designed and coded properly, this application failover is mostly transparent to users.

When combined with Oracle Data Guard, Oracle RAC is protected from major site failures. Oracle RAC enables horizontal scalability and thus the ability to support large, global, single-instance computing that hosts thousands of users. When protected via various HA options, such single global instances significantly reduce costs via consolidation in terms of servers, datacenters, software licenses, and skilled staff to maintain them. Oracle Database 12*c* extends these capabilities using a plethora of new features, which were briefly mentioned previously.

Agility

In addition to the twin paradigms of availability and scalability, we need to address the need for agility. Organizational agility is the ability of an organization to move quickly and efficiently in all aspects of that business's operations—and this depends heavily on the ability of the business to acquire and process data related to its operations. Businesses today not only need to scale up but also to scale back, and they need to perform such scale-ups and scale-downs quickly, in a matter of a few hours or even minutes—and do all this without impact or downtime. This has placed an enormous demand on IT and datacenters to provide infrastructure not in a matter of days or weeks, but in terms of minutes. In other words, IT organizations should be able to commission and decommission computing services on the fly—something that was a pipe dream a few years ago. We will look at how Oracle Database 12*c* and Oracle

https://t.me/learningnets

Technet24

RAC can help you achieve this.

IT organizations, aided by vendors, have been trying to provide this scaling quickly and easily using a combination of first "virtualizing" computing resources and then building the ability to expose these resources in a metered and controlled fashion. This is achieved first by carving out virtual machines (VMs) from physical servers and presenting them as a service to both internal and external consumers. Although this is a good first response to the need for agility, IT organizations are finding it difficult to perform this at scale.

As Figure 1-3 shows, increasingly, IT organizations have been outsourcing various pieces of this puzzle. As you can see, various components are required to run IT: servers and storage, an optional virtualization layer on these servers, hosting a variety of guest operating systems, the familiar Oracle database, and, finally, the middleware and application layers. Although these components remain the same in any deployment, "cloud" vendors have been successfully providing these same components in a flexible manner. Here are the various options:

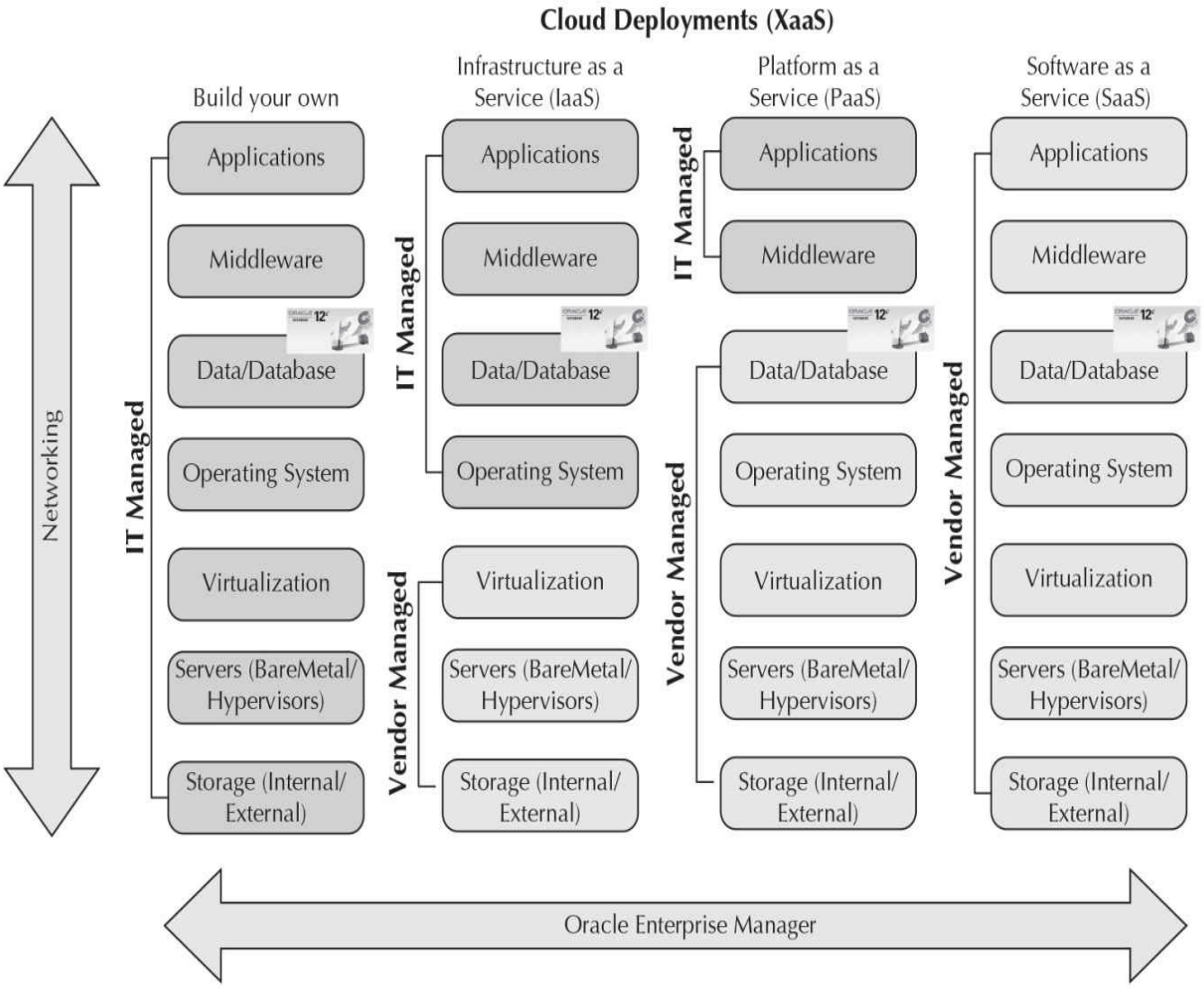


FIGURE 1-3. *Cloud deployment architecture*

https://t.me/learningnets

- On one end of the scale, IT organizations build all the various layers, including the database layer, themselves. Usually, these are in-house deployments built for specific reasons, such as the need to satisfy extreme security requirements.
- In the Infrastructure as a Service (IaaS) model, cloud vendors provide just the hardware and networking infrastructure. Cloud vendors such as Amazon EC2 and Rackspace provide this as a basic service.
- In the Platform as a Service (PaaS) model, IT organizations can build their own applications on top of databases and the hardware/networking infrastructure provided by cloud vendors.
- In the Software as a Service (SaaS) model, cloud vendors provide full service to organizations, including applications, database and file storage, and more.

In these deployments, IT vendors today are able to virtualize environments using products such as an Oracle virtual machine (a hypervisor based on open-source Xen technology) to spin up computing resources on demand. Prime examples of the ability of vendors to provide cloud computing include Amazon's Elastic Cloud Computing (EC2) and Salesforce.com's Sales Cloud 2. The former has quickly become sophisticated to the extent that Amazon is now even able to provision complete Oracle E-Business Suite environments in minutes. New models where such vendors provide isolation and security via virtual private clouds (VPCs) are evolving, so IT organizations are overcoming their apprehension and starting to deploy into such environments.

On the backend database side, this means that Oracle technologies should be able to scale database services as well. Oracle Database 12*c* and Oracle RAC play a key role here because they provide ultimate scalability. However, the challenge is to perform this scaling dynamically without any interruption to the availability, as mentioned previously. Oracle RAC utilizes innovative features such as Flex ASM, Flex Cluster, and Cloud File systems to enable these features. You will see how these features work in detail later in this book.

The Oracle Database 12*c* RAC Solution

Oracle Database 12*c* RAC takes this challenge head on. The key requirement for dynamic provisioning is that the technology should be able to support resource movement and reassignments easily and dynamically from a pool, along with support for such functionality at all layers and components. We'll cover this in more detail in later chapters, but briefly, Oracle ASM provides the total abstraction at the storage layer, allowing multiple hosts to not just see database storage but to share disk files as well, and dynamically adjust them to changing requirements. Oracle ASM also provides a complete suite of fully functioning Dynamic Volume Manager and File System for Oracle storage needs.

In addition, server pools in the Oracle Grid Infrastructure foundation provide the capability of dynamically allocating resources within a Grid or Oracle RAC

https://t.me/learningnets

Technet24

environment, thus providing flexibility at the database layer. SCAN IP provides a way to access a cluster using a single IP address, thus simplifying naming and administration.

Starting in Oracle Database 11*g*R2, Oracle provides the ability to spin up single, non–Oracle RAC instances for smaller loads and yet provide the high availability and "hot failover" to another node using RAC One. Edition-based redefinition completes the high availability scenario, because this feature can be used to provide application transparency during software changes as well (that is, online hot patching—the holy grail of downtime optimizations).

Summary

Modern business requirements have great impact on database and application availability, and vice versa. With ever-growing requirements and extreme dependence on information availability, information systems are expected to remain fully functional and survive all external failures. The key to designing highly available systems relies on eliminating single points of failure in all critical components.

Oracle technologies are always a leap ahead on the current trends, making sure the enterprise requirements, including the ability to deploy private database clouds, are met. Current versions of Oracle Clustering and Oracle Grid Infrastructure components operating underneath Oracle Database 12*c* allow us to increase and shrink the capacity on demand seamlessly. Oracle ASM and ACFS completely virtualized the storage infrastructure for the datacenter and have components built in to support continuous availability and transparent scalability.

Clusters provide an enterprise with uninterrupted access to their business-critical information, enabling the nonstop functions of the business. Clusters can be configured with various failover modes, depending on the requirements of the business. When designed and implemented judiciously, clusters also provide infinite scalability to business applications.

https://t.me/learningnets

CHAPTER 2

Oracle Database Clustering Basics and Its Evolution

As defined in the previous chapter, a *cluster* is a group of interconnected nodes that act like a single large server capable of growing and shrinking on demand. In other words, clustering can be viewed logically as a method for enabling multiple standalone servers to work together as a coordinated unit called a *cluster*. In general, the servers participating in the cluster are *homogenous*—that is, they use the same platform architecture, operating system, and almost identical hardware architecture and software patch levels (homogenous servers are common but not an absolute requirement, as several organizations use a nonhomogenous set of machines for running Oracle RAC). All machines are independent and respond to the same requests from a pool of client requests.

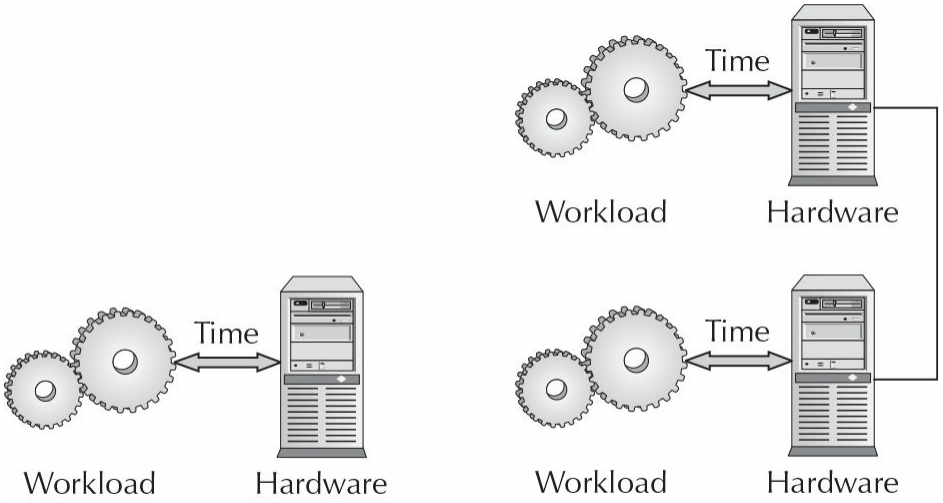
From another perspective, clustering can also be viewed as host virtualization in the modern cloud computing architectures—the end-user applications do not specifically connect to "a server" but rather connect to a logical server that is internally a group of physical servers. The clustering layer wraps the underlying physical servers and only the logical layers are exposed externally, thus offering the simplicity of a single node to the outside world.

https://t.me/learningnets

Technet24

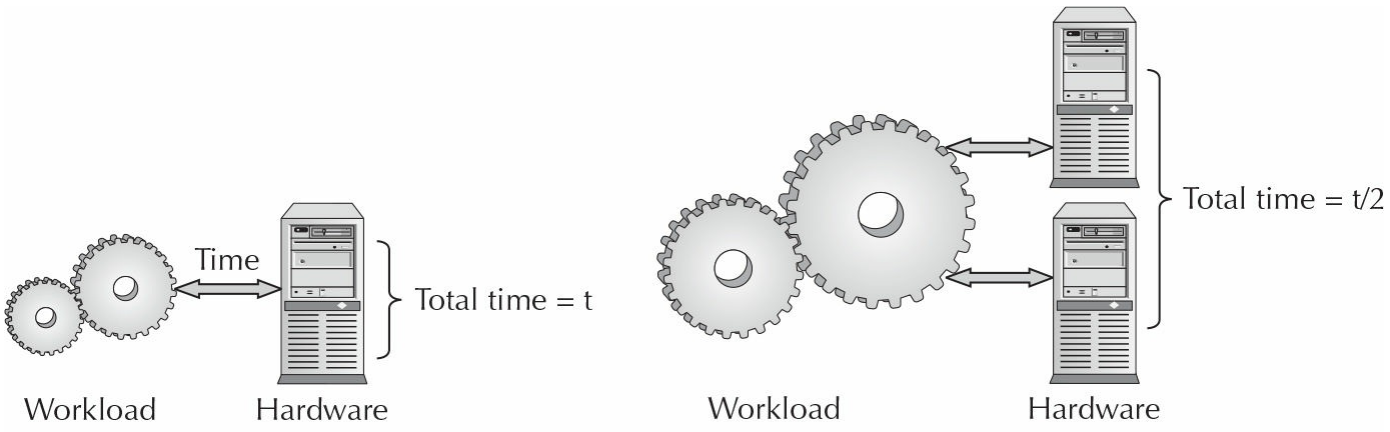
Traditionally, clustering has been used to scale up systems, to speed up systems, and to survive failures.

Scaling is achieved by adding extra nodes to the cluster group, thus enabling the cluster to handle progressively larger workloads. Clustering provides horizontal on-demand scalability without incurring any downtime for reconfiguration.



Original system (100% task) Clustered system (200% task)

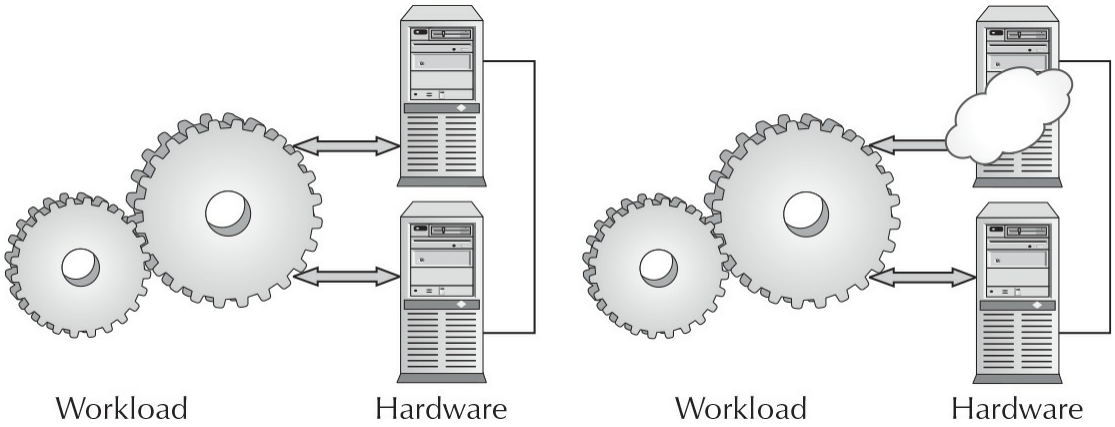
Speeding up is accomplished by splitting a large workload into multiple smaller workloads and running them in parallel in all the available CPUs. Parallel processing provides massive improvement in performance for jobs that can be run in parallel. The simple “divide and conquer” approach is applied to the large workloads, and parallel processing uses the power of all the resources to get work done faster.



Original system (time) Clustered system speedup

Because a cluster is a group of independent hardware nodes, failure in one node does not halt the application from functioning on the other nodes. The application and the services are seamlessly transferred to the surviving nodes, and with proper design the application continues to function normally, just as it was before the node failure. In some cases, the application or the user process may not even be aware of such failures because the failover to the other node is transparent to the application.

https://t.me/learningnets



When a uniprocessor system reaches its processing limit, it imposes a big threat to scalability. Traditional symmetric multiprocessing (SMP)—the use of multiple processors (CPUs) that share memory (RAM) within a single computer to provide increased processing capability—solved this problem. SMP machines achieved high performance via *parallelism*, where the processing job is split up and run on the available CPUs. Higher scalability is achieved by adding more CPUs and memory.

Figure 2-1 shows the basic architectural similarities between symmetric multiprocessing and clustering. However, keep in mind that the two architectures maintain cache coherency at totally different levels and latencies.

https://t.me/learningnets

Technet24

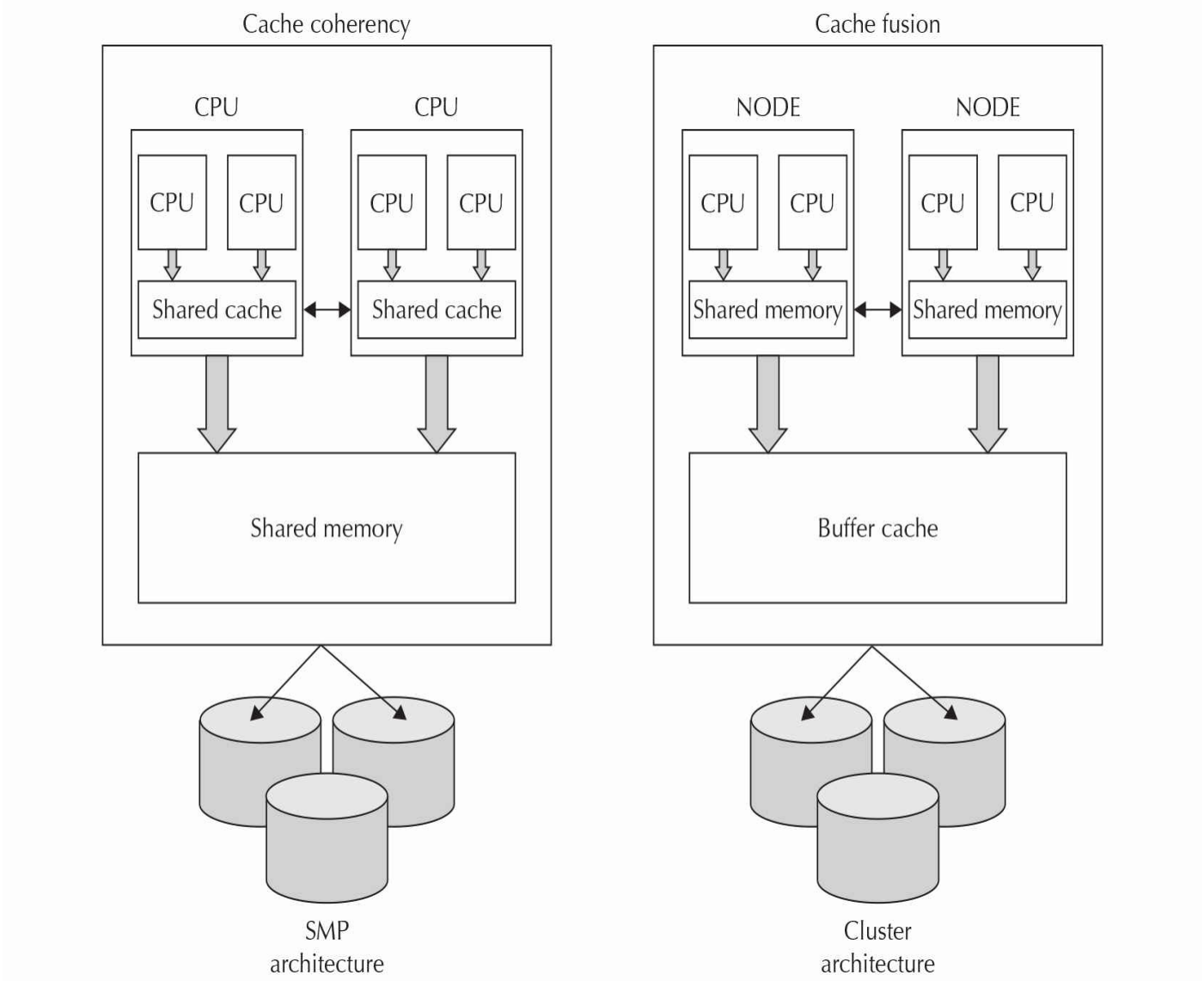


FIGURE 2-1. *SMP vs. clusters*

NOTE

Uniprocessors aren't used any longer. Also, newer technologies such as InfiniBand and Silicon Photonics from Intel make for faster network connectivity, and other technologies such as Oracle Data Connectors with Hadoop, Oracle Big Data Appliance, R, and Big Data SQL all support parallelism.

Cloud Computing with Clusters

Clustering is part of Oracle's cloud computing methodology, by which several low-cost commodity hardware components are networked together to achieve increased computing capacity. Adding additional nodes and distributing the workload to all the

https://t.me/learningnets

available machines are done to achieve scalability on demand.

Scalability and application performance can be improved via three methods:

- By working harder
- By working smarter
- By getting help

Working harder means adding more CPUs and more memory so that the processing power increases to handle any amount of workload. This is the usual approach and often helps as additional CPUs address the workload problems. However, this approach is not quite economical, because the average cost of the computing power does not always increase in a linear manner. Adding computing power for a single SMP box increases the cost and complexity at a logarithmic scale. Also, the performance (and scalability) is often constricted by the bottlenecks in the infrastructure layer, such as the available bandwidth and speed of the wires connecting storage to servers.

Working smarter is accomplished by employing intelligent and efficient algorithms at either the application layer or storage layer. By introducing "smartness" in the storage layer, you can greatly reduce the total amount of work to be done to achieve the desired results. Working smarter at the application layer often requires rewriting the application or changing the way it works (or sometimes changing the application design itself), which is quite impossible for a running application and requires unacceptable downtime. This option can be almost impossible for a third-party vendor and for packaged applications, because getting everyone onboard can become a tedious and time-consuming task.

Working smarter at the storage layer is accomplished by introducing intelligent storage servers, where the storage servers offload some amount of processing. This requires specially designed storage servers such as Oracle Exadata Storage Servers (used in Oracle Exadata Database Machine) to process some amount of critical workload close to the storage. Processing the workload close to where it is stored greatly enhances the performance of the application because it largely limits the number of round-trips between the storage and hosts as well as limits the size of data transferred to the database cluster infrastructure.

Getting help can be as simple as using other machines' computing power to do the work. In other words, getting help simply involves clustering the hardware, using the spare processing capacity of the idle nodes, and combining the processing transaction results at the end. More importantly, this approach does not require any changes to the application because it is transparent to the application. Another advantage to using this approach is that it allows for on-demand scalability—you can choose to get help whenever required, and you do not need to invest in massive hardware.

Oracle Exadata and Smart Scans

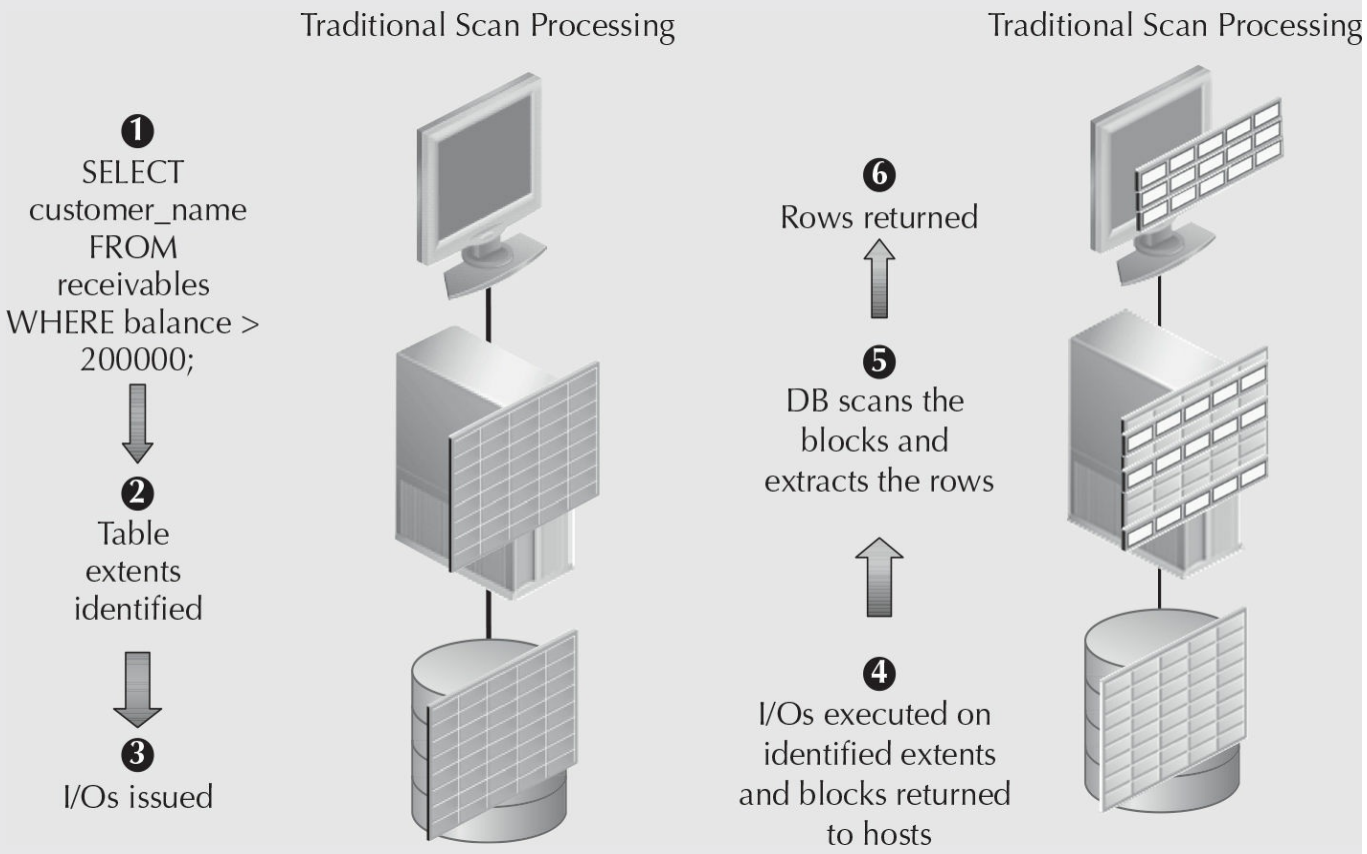
https://t.me/learningnets

Technet24

Oracle Exadata Storage Servers rightly fit into the "working smarter" method of improving performance and scalability. Traditional database implementations use the storage as plain containers to dump and retrieve data, and the storage containers are relatively "dumb" about the data they store. All the database processing is handled in database host memory, and often this involves transferring huge amounts of data from the storage to the host machines that do the actual number crunching. The summarized results are then passed to the upper layers of end-user applications. An extremely large volume of raw data is processed to validate the required business intelligence operations by applying filtering conditions on the column-level data.

Special-purpose storage servers are built with additional intelligence at the storage layer, where the storage is fully "aware" of the data that is stored in the disks. While processing the complex business intelligence reports, the host machines talk to the storage servers and provide additional information about the dataset being requested from the storage servers. The storage servers filter the data in the storage (known as "smart scan" processing) and pass the records matching the conditions specified by the host machines. Some amount of the processing is handled in the storage layer, and the database host machines do not need to process the huge amount of raw data to process the queries.

In a traditional storage-hosted database, the SQL query processing is handled at the database server level. The data stored in the disks is retrieved as blocks for the database server and loaded in database buffer cache memory for processing. The following illustrates the processing of the table scans in a conventional storage architecture:



1. The application layer (end-user client) issues a SQL query. For simplicity,

https://t.me/learningnets

we assume this is a SELECT statement with a WHERE condition.

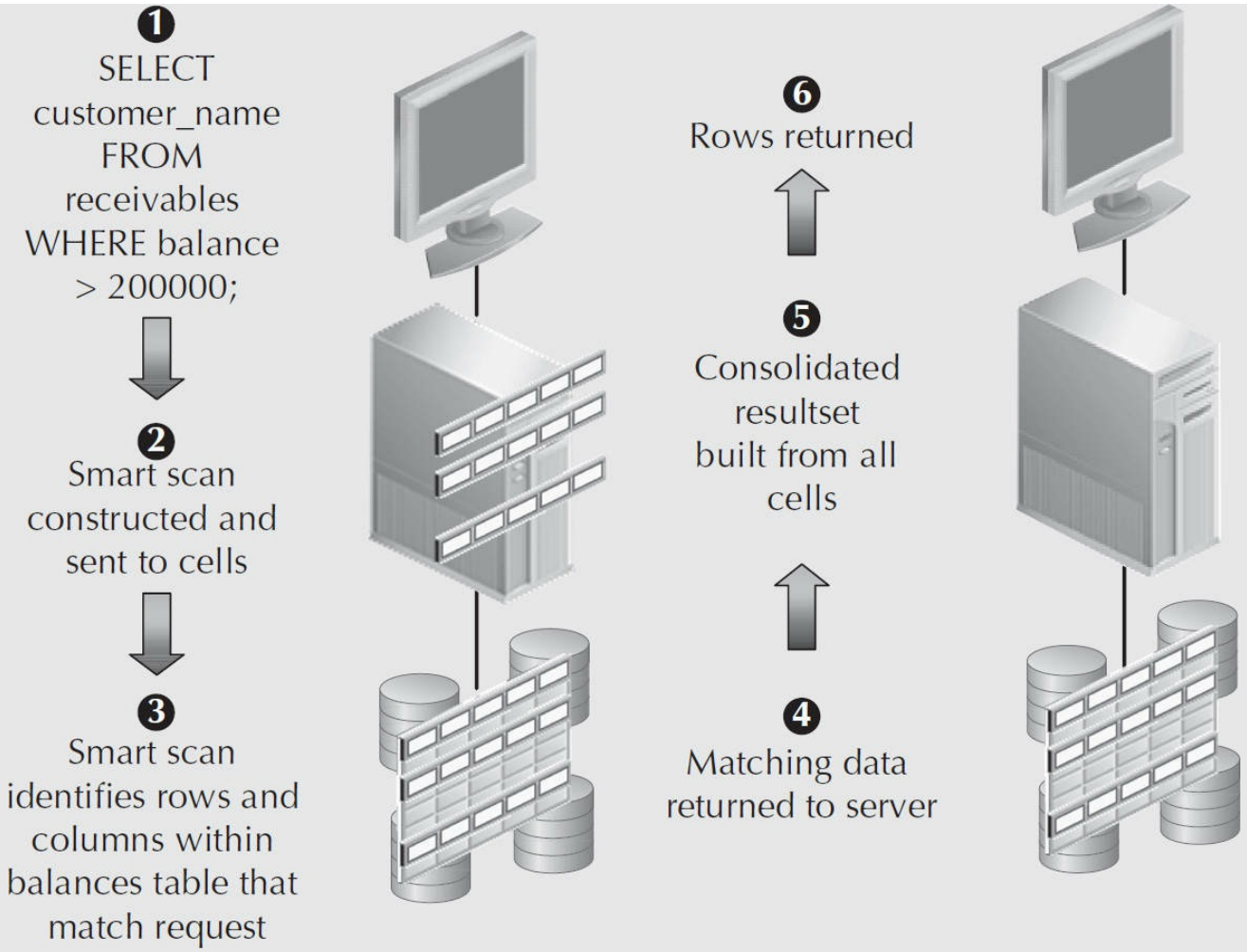
2. The database kernel queries the data dictionary and identifies the file and extents where the table data is stored.
3. The server process issues I/O calls to read all the physical blocks from the disk that happen to hold the table data.
4. The physical data blocks from disk are loaded into the database server's memory (buffer cache in the SGA).
5. The database server reads the memory buffers and filters the rows satisfied by the predicate (WHERE condition).
6. The matching rows are returned to client.

Traditional SQL processing works better if the tables are relatively small and well indexed. However, for complex business intelligence queries involving multiples of larger tables, the task of reading of all the data from disk and transferring it to host memory is very expensive. It requires a huge amount of raw data transferred between storage to host memory over the network. Moreover, the records that do not meet the filter conditions are simply discarded at the host level. This is a totally unproductive I/O operation that impacts the query response time. In this case, we often read multiple times the data required to match the filter conditions; poorly constructed queries put unnecessary overhead on the storage subsystem and affect the total system performance.

However, in the Exadata smart scan model, the entire workflow is handled intelligently and completely differently. The queries that perform the table scans are offloaded to the Exadata Storage Server, and only records meeting the filter criteria are returned to the database server. Tasks such as row filtering, column filtering, and some amount of join processing are performed in the storage server. It's important to note that Exadata's smart scanning works in some but not all cases. For instance, it works for direct-path queries only. Exadata uses a highly efficient, unbuffered, direct-read mechanism for scanning the tables that is similar to Oracle parallel query operations. The following illustrates this mechanism's operation:

https://t.me/learningnets

Technet24



1. The application layer (end-user application) issues a SELECT query.
2. When the database server detects Exadata, it builds an Exadata construct (known as an iDB command) representing the SQL and then sends this to the Exadata Storage Server. iDB is a data-transfer mechanism based on the InfiniBand protocol (low latency, high bandwidth) used for communications between an Exadata Storage Server and the database server.
3. The Exadata Storage Servers perform a "smart scan" on the tables and extract the rows of interest by applying the WHERE conditions directly on the storage.
4. The resultset is transferred directly to the database instance. The resultset is the query result and not the blocks satisfying the filter condition. This is sent directly to the Program Global Area (PGA) of the user session and is not cached in the System Global Area (SGA).
5. A consolidated result set is built from all cells.
6. Matching rows are returned to the client.

Offloading SQL processing to the Exadata server tremendously increases the speed of the query processing and frees up the CPU cycles in the database server. Also, processing the data closer to the storage eliminates a massive amount of unproductive I/O and improves the scalability of the storage subsystem.

https://t.me/learningnets

Shared Storage in Clustering

One of the key components in clustering is shared storage. Storage is accessible to all nodes, and all nodes can read and write in parallel to the shared storage, depending on the configuration of the cluster. Some configurations allow the storage to be accessible by all the nodes at all times; some allow sharing only during failovers.

More importantly, the application will see the cluster as a single system image and not as multiple connected machines. The cluster manager does not expose the system to the application, and the cluster is transparent to the application. Sharing the data among the available nodes is a fundamental concept in scaling and is achieved using several types of architectures.

Types of Clustering Architectures

Clustering architecture can be broadly categorized into three types, based on how storage is shared among the nodes:

- Shared nothing architecture
- Shared disk architecture
- Shared everything architecture

Table 2-1 compares the most common functionalities across the types of database clustering architectures and lists their pros and cons, along with implementation details and examples. Note that the shared disk architecture and shared everything architecture slightly differ in the number of nodes and storage sharing.

https://t.me/learningnets

Technet24

Function Shared	Shared Nothing	Shared Disks	Shared Everything
Disk ownership/ sharing	Disks are owned by individual nodes and are not shared among any of the nodes at any time.	Active nodes usually own disks, and the ownership is transferred to the surviving node during failure of the active node—that is, disks are shared only during failures.	Disks are always shared, and all nodes have equal rights to the disks. Any node can read/write data to any of the disks because no nodes exclusively own any disk.
Number of nodes	Typically a very high number.	Normally two nodes, with only one node active at any time.	Two or more nodes, depending on the configuration. The number of nodes is sometimes limited to 100 in Oracle 12*c*.
Data partitioning	Strictly partitioned. In normal operations, one node cannot access the data from the other nodes. Local nodes can access data local to that node.	Data partitioning is not required because only one instance will access the complete set of data.	No data partitioning is required because the data can be accessed from any nodes of the cluster.
Client coordinator	An external server or any group member.	No coordinator is required because the other node is used only during failover.	Not required. Any node can be accessed for any set of data.
Performance overhead	No performance overhead because no external coordination is involved or required.	No performance overhead.	No additional overhead after three nodes. Very negligible overhead up to three nodes.
Lock manager	Not required because data is not shared among the members.	Not required.	A distributed lock manager (DLM) is required for managing the resources.
Initial/on-demand scalability	Initially highly scalable, but limited to the capacity of the individual node for local access. On-demand scalability is not possible.	Not very scalable. Scalability is limited to the computing capacity of the single node.	Infinitely scalable because nodes can be added on demand.
Write access	Each node can write but only to its own disks. One node cannot write to the disks owned by another instance.	Only one node at a time. One node can write to all the disks.	All nodes can write to all disks simultaneously because the lock manager controls the orderly writes.
Load balancing	Not possible.	Not possible because there is only one active node at any point in time.	Near perfect load balancing, and a variety of configuration choices is offered.
Application partitioning	Strictly required because the member nodes only see a subset of data.	Not required because only one node is active at any time.	Not required, but can be configured if required.

https://t.me/learningnets

Function Shared	Shared Nothing	Shared Disks	Shared Everything
Dynamic node addition	Not possible because adding nodes requires data redistribution.	Possible, but does not make sense because only one node will be active at any time.	Very much possible. This is a key strength of this architecture type.
Failover capacity	No failover because the nodes/disks are tied to specific nodes.	Can be failed over to other nodes, with minimal loss of service.	Very capable. Often failover is transparent.
I/O fencing	Not required because there is no concurrent data access for multiple nodes.	Not required.	Provided by the DLM and cluster manager.
Failure of one node	Makes a subset of data inaccessible momentarily: 100/*N*% of the data is inaccessible, with *N* being the number of nodes in the cluster. Then the ownership is transferred to the surviving nodes.	Data access is momentarily disrupted, until the applications are failed over to the other node.	Because connections spread across all the nodes, a subset of sessions may have to reconnect to another node, but all the data is accessible to all the nodes. No data loss due to node failures.
Addition of nodes	You can add nodes and redistribute the data while keeping the same architecture.	Not possible because the addition of nodes does not help anything.	Nodes can be added and removed on the fly. Load balancing is done automatically during reconfiguration.
Examples	IBM DB2 DPF, Teradata, Tandem NonStop, Informix XPS, Microsoft Cluster Server	HP M/C ServiceGuard, Veritas Cluster Servers	Oracle Real Application Clusters, Oracle Exadata Database Machine, IBM DB2 Pure Scale

TABLE 2-1. *Functionalities of Clustering Architectures*

Shared Nothing Architecture

Shared nothing architecture is built using a group of independent servers, with each server taking a predefined workload (see Figure 2-2). If, for example, a number of servers are in the cluster, the total workload is divided by the number of servers, and each server caters to a specific workload. The biggest disadvantage of a shared nothing architecture is that it requires careful application partitioning. Adding a node would require complete redeployment, so it is not a scalable solution. Oracle does not support the shared nothing architecture.

https://t.me/learningnets

Technet24

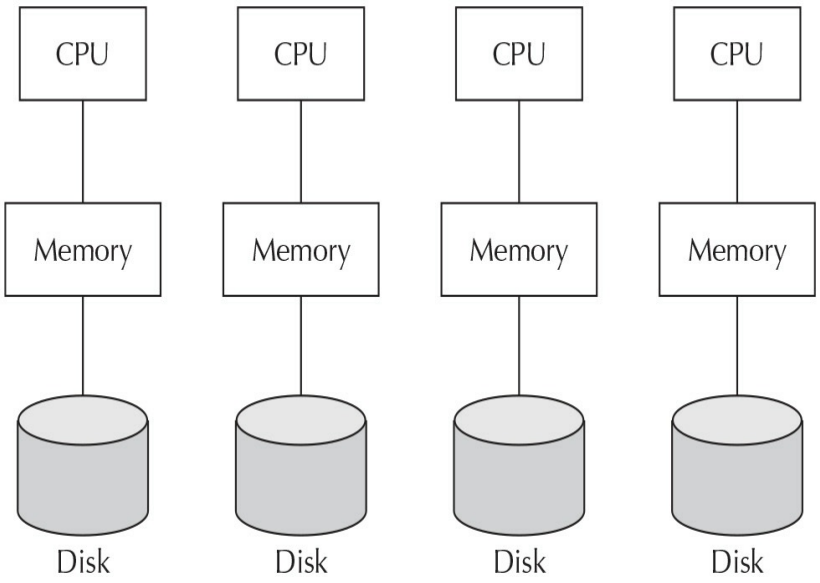


FIGURE 2-2. *Shared nothing clusters*

In shared nothing database clusters, data is typically divided across separate nodes. The nodes have little need to coordinate their activity because each is responsible for different subsets of the overall database. But in strict shared nothing clusters, if one node is down, its fraction of the overall data is unavailable.

The clustered servers neither share disks nor mirror the data—each of the servers has its own resources. Servers transfer ownership of their respective disks among one another in the event of a failure. A shared nothing cluster uses software to accomplish these transfers. This architecture avoids the distributed lock manager (DLM) bottleneck issue associated with shared disks while offering comparable availability and scalability. An example of a database that belongs to the shared nothing clustering solution model is the Cassandra database, which is a highly popular NoSQL database.

One of the major issues with shared nothing clusters is that they require very careful deployment planning in terms of data partitioning. If the partitioning is skewed, it negatively affects the overall system performance. Also, the processing overhead is significantly higher when the disks belong to the other node, which typically happens during the failure of any member nodes.

The biggest advantage of shared nothing clusters is that they provide linear scalability for data warehouse applications—they are ideally suited for that. However, they are unsuitable for online transaction processing (OLTP) workloads, and they are not totally redundant, so a failure of one node will make the application running on that node unavailable. Still, most major databases, such as IBM DB2 Data Partitioning Edition, Informix XPS, and Teradata, do implement shared nothing clusters.

https://t.me/learningnets

NOTE
Some shared nothing architectures require data to be replicated between the servers so that it is available on all nodes. This eliminates the need for application partitioning but brings up the requirement of a high-speed replication mechanism, which is almost always impossible to attain, compared to high-speed memory-to-memory transmission.

Shared Disk Architecture

For high availability, some shared access to the data disks is needed (see Figure 2-3). In shared disk storage clusters, at the low end, one node can take over the storage (and applications) if another node fails. In higher-level solutions, simultaneous access to data by applications running on more than one node at a time is possible, but typically a single node at a time is responsible for coordinating all access to a given data disk and serving that storage to the rest of the nodes. That single node can become a bottleneck for access to the data in busier configurations.

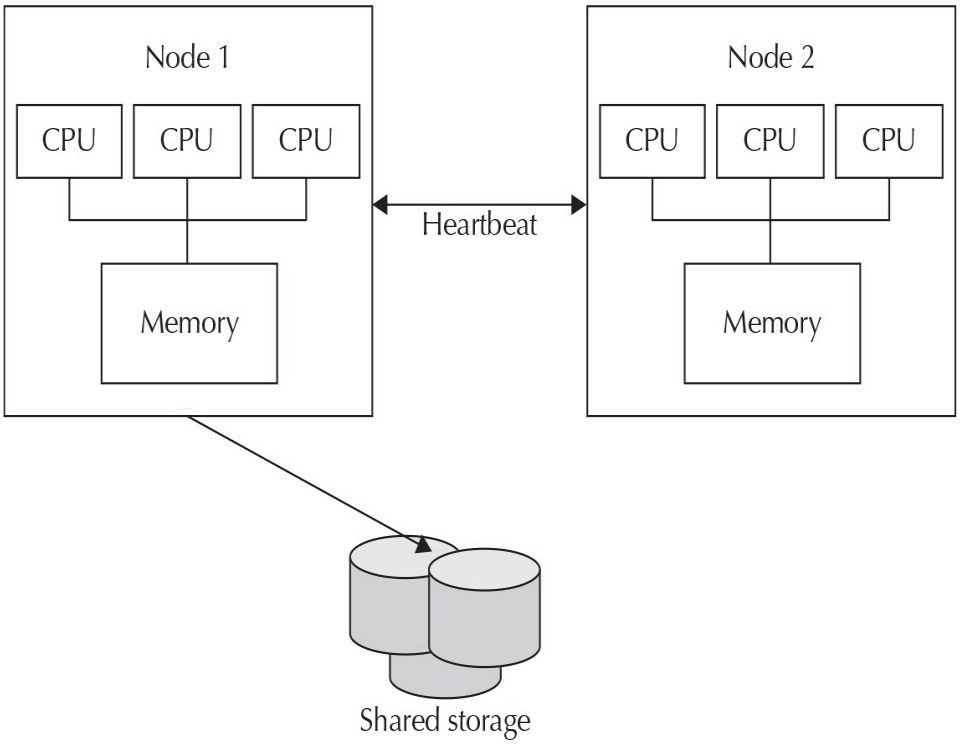


FIGURE 2-3. *Shared disk cluster*

In simple failover clusters, one node runs an application and updates the data while another node stands idle, until needed, and then takes over completely. In more sophisticated clusters, multiple nodes may access data, but typically one node at a time serves a file system to the rest of the nodes and performs all coordination for that file system.

https://t.me/learningnets
Technet24

Shared Everything Architecture

Shared everything clustering utilizes the disks accessible to all computers (nodes) within the cluster. These are often called “shared disk clusters” because the I/O involved is typically disk storage for normal files and/or databases. These clusters rely on a common channel for disk access because all nodes may concurrently write or read data from the central disks. Because all nodes have equal access to the centralized shared disk subsystem, a synchronization mechanism must be used to preserve coherence of the system. An independent piece of cluster software called the DLM (Distributed Lock Manager) assumes this role.

In a shared everything cluster, all nodes can access all the data disks simultaneously, without one node having to go through a separate peer node to get access to the data (see Figure 2-4). Clusters with such capability employ a cluster-wide file system (CFS), so all the nodes view the file system(s) identically, and they provide a DLM to allow the nodes to coordinate the sharing and updating of files, records, and databases.

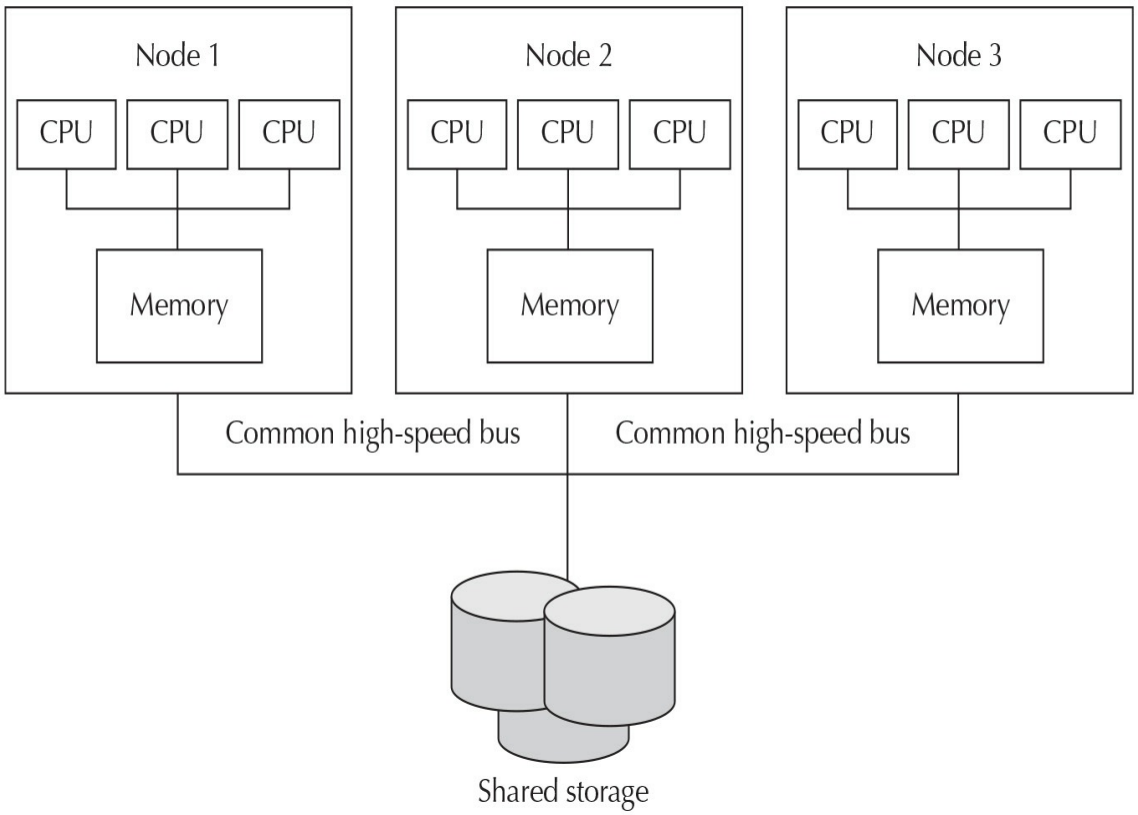


FIGURE 2-4. *Shared everything cluster*

A CFS provides the same view of disk data from every node in the cluster. This means the environment on each node can appear identical to both the users and the application programs, so it doesn’t matter on which node the application or user happens to be running at any given time.

Shared everything clusters support higher levels of system availability: If one node

https://t.me/learningnets

fails, other nodes need not be affected. However, higher availability comes at a cost of somewhat reduced performance in these systems because of the overhead in using a DLM and the potential bottlenecks that can occur in sharing hardware. Shared everything clusters make up for this shortcoming with relatively good scaling properties.

Oracle RAC is the classic example of the shared everything architecture. Oracle RAC is a special configuration of the Oracle database that leverages hardware-clustering technology and extends the clustering to the application level. The database files are stored in the shared disk storage so that all the nodes can simultaneously read and write to them. The shared storage is typically networked storage, such as Fibre Channel SAN or IP-based Ethernet NAS, which is either physically or logically connected to all the nodes.

Hadoop Clusters

Database clustering often solves the problems posed by cloud computing on structured data processing. However, conventional data processing methods using clusters are often unable to meet the demands of huge amounts of data generated by social media and machine data. Structured data processing requires proper schema definitions and a prebuilt data model. Enormous amounts of data with high velocity (often referred to as *web scale*) require totally different types of clusters, with high storage and computing capacity on demand.

Another challenge posed by web scale processing is the variety of data (such as text, images, audio, video, and geographic data) with very limited time to process. Often, the data is processed when it is in motion. Some of this data is never stored on disk; it lives and dies in the wire. Database clusters often process data at rest (meaning the data is stored on disk before/after processing) and are not properly equipped to process the web scale data and data in motion.

Internet and social media corporations were the first to bite the bullet and required a new data processing technique. Apache Hadoop is one of the frameworks on which massive volumes of unstructured data (also known as *big data*) is processed.

Architecture of Hadoop

The section provides a simple overview of the architecture of Apache Hadoop. The discussion here does not get into the nitty-gritty details. The storage is local to the nodes; however, the software layer, which integrates the storage from member nodes, pools the storage from all the nodes and exposes it as one logical storage unit that can be accessed from any of the nodes.

Hadoop clusters are architecturally similar to the typical shared nothing architecture, with the nodes having their own local disks. The file system, which runs on top of the Hadoop, is known as Hadoop Distributed File System (HDFS), which is a distributed, scalable, and portable file system. A Hadoop cluster basically has a single name node (which serves as the master) and a cluster of data nodes, although

https://t.me/learningnets

Technet24

redundancy options are available for the name node due to its criticality.

The HDFS layer pools all the local storage and builds one large logical file system to store the data. Figure 2-5 shows the architecture of a Hadoop cluster. The following are the fundamental principles of Hadoop clustering:

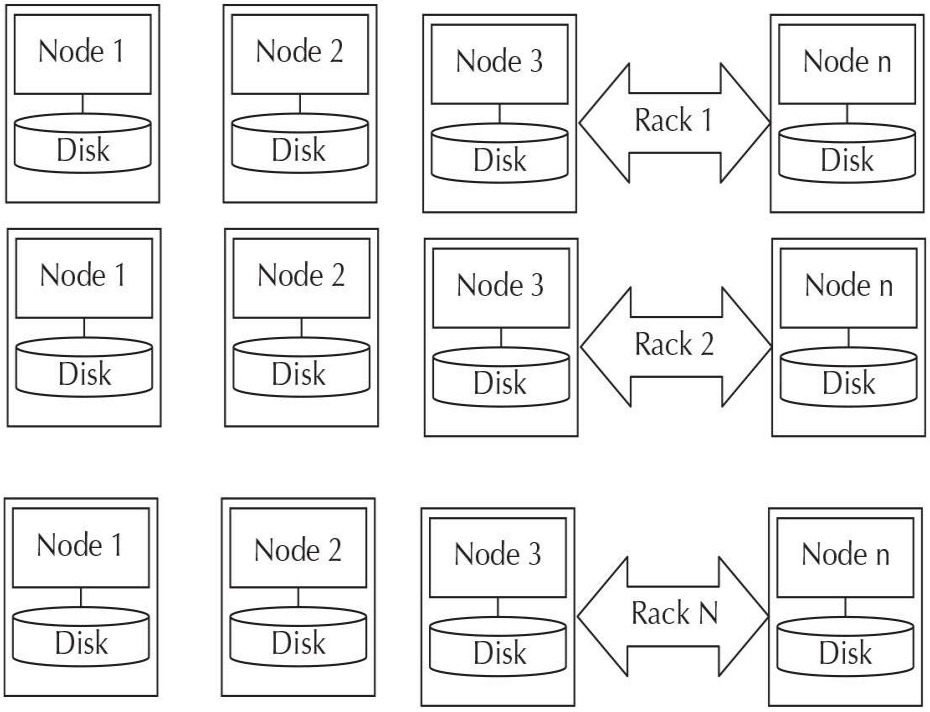


FIGURE 2-5. *Architecture of Hadoop clusters*

- Hadoop infrastructures use commodity-class hardware, and the hardware is fully expected to fail. Hardware component failures are tolerated well, since the software masks the hardware failures.
- Moving computation is cheaper than moving data compute nodes and only the result is transferred to compute nodes. Hadoop does this by sending the code to the nodes with the data (Datanodes in Hadoop parlance) and running tasks on those nodes. Hadoop always attempts to schedule computation as close to the data as possible.

Because the hardware infrastructure is built using commodity-class hardware, the disks are expected to fail. To overcome the challenge of lost data, files that are coming to Hadoop clusters are replicated to three different nodes (and they are configurable). The replication algorithm is implemented in such a way that no two copies of the replicas are stored in the same node. This is called *node awareness*. Also, when rack awareness is configured, the three copies are not all stored in the same rack. This protects against node failure and rack failure.

https://t.me/learningnets

Historical Background of Oracle RAC

Oracle was the first commercial database that supported clustering at the database level, in the early 1980s. Oracle 6.2 gave birth to Oracle Parallel Server (OPS), which used Oracle's own DLM and worked very well with Digital's VAX clusters. Oracle was the first database to run the parallel server.

In the early 1990s, when open systems dominated the computer industry, many UNIX vendors started clustering technology, mostly based on Oracle's DLM implementation. Oracle 7 Parallel Server (OPS) used vendor-supplied clusterware.

In the Oracle Database 8 release, Oracle introduced a generic lock manager. Oracle's lock manager is integrated with Oracle code with an additional layer called OSD (Operating System Dependent). Oracle's lock manager soon integrated with the kernel and became known as the IDLM (Integrated Distributed Lock Manager) in later versions of Oracle.

Oracle Real Application Clusters version 9*i* used the same IDLM and relied on external clusterware. Oracle provided its own clusterware for all operating systems starting with 10*g* and enhanced it quite a bit in Oracle Database 11*g* with the introduction of server pools and additional APIs to manage third-party applications. Oracle Clusterware is the de facto clusterware and a requirement for running Oracle RAC.

Oracle Parallel Storage Evaluation

Along with the cluster software infrastructure, another key component, which grew along with Oracle RAC, is the development of parallel storage management systems. Earlier versions of Oracle clusters (known as Oracle Parallel Servers until Oracle 8*i*) depended on third-party software for parallel storage management. Cluster file systems such as Veritas Cluster File Systems (VxFS) and RAW disk partitions were used as foundations for parallel storage in Oracle RAC.

The unavailability of general-purpose clusterware and a cluster-wide file system was hampering the adoption of Oracle RAC. Oracle provided generic clusterware with Oracle 9*i* RAC called Oracle Cluster Manager (OraCM) for Linux and Windows. Along with the clusterware, Oracle also developed a cluster file system called Oracle Cluster File System (OCFS), which was used to build the shared storage for Linux and Windows environments. Since Oracle provided OraCM with OCFS, the adaption of Oracle RAC significantly advanced. However, OCFS had inherent limitations and was not very scalable.

Oracle introduced a similar product in Oracle Database 9*i* Release 2 called Oracle Disk Manager (ODM) for Solaris operating systems. ODM was capable of handling file management operations (which were handled outside Oracle prior to this, including creating and destroying files), and file identifiers in ODM replaced file descriptors.

With the success and wisdom gained from Oracle Disk Manager, Oracle was ready for a fully functioning cluster-aware database file system. Enter Automatic Storage

https://t.me/learningnets

Technet24

Management (ASM). Starting with Oracle Database 10*g*, Oracle introduced ASM for mainline storage and promoted for the use of ASM extensively. Oracle ASM quickly gained traction in the market and performed flawlessly in Oracle benchmarks and large customer environments.

While Oracle ASM was being used for mainline storage of data, additional shared storage (outside ASM) was still required for some of the Oracle RAC components used to house cluster-configuration details, as well as special-purpose storage used during cluster configurations commonly known as "voting disks." The initial versions of ASM lacked the capacity to store cluster components such as OCR and voting disks inside ASM, because clusterware should be started to access the data inside ASM-managed storage. However, current versions of ASM implement an intelligent trick to access the ASM data (the details of which are discussed in the later chapters).

Custom applications built using the Oracle database and even some commercial applications (such as Oracle E-Business Suite) used a wide variety of external files during data extraction and loading. A classic example is processing Call Data Records (CDR) in a telecom billing application or output files in Oracle Applications' concurrent manager processing. Database applications similar to these required a scalable cluster-wide file system so that all the data could be accessed concurrently among the nodes.

To address the requirement for a cluster-wide file system, Oracle ASM was enhanced further to serve as a general-purpose file system, called ASM Cluster File System (ACFS). ACFS offered a seamless way to provide access to non-database files across the nodes and was used widely to store the database binaries, log files, and non-data files used in the Oracle database. The application programming interfaces (APIs) built on Oracle ASM allowed for the use of ACFS volumes such as traditional Network File System (NFS) mount points across the servers.

Oracle ASM with ACFS addresses the total storage solution for parallel storage requirements for Oracle Clusterware and eliminates the need for expensive third-party file systems or maintenance-heavy raw devices. Oracle ASM also provides volume manager and file system functionalities for an Oracle database and is an integrated part of the Oracle Clusterware architecture.

Oracle Parallel Server Architecture

An Oracle parallel or cluster database consists of two or more physical servers (nodes) that host their own Oracle instances and share a disk array. Each node's instance of Oracle has its own System Global Area (SGA) and its own redo log files, but the data files and control files are common to all instances. Data files and control files are concurrently read and written by all instances; however, redo logs can be read by any instance but written to only by the owning instance. Some parameters, such as db_block_buffers and log_buffer, can be configured differently in each instance, but other parameters must be consistent across all instances. Each cluster node has its own set of background processes, just as a single instance would. Additionally, OPS-specific processes are also started on each instance to handle cross-instance

https://t.me/learningnets

communication, lock management, and block transfers.

Figure 2-6 shows the architecture of OPS. It is important to understand the inner workings of OPS in order to understand Oracle RAC because it provides a solid foundation for visualizing the under-the-hood operations of Oracle RAC. Although this is not mandatory knowledge for managing Oracle RAC, a deeper understanding of the OPS components makes you an expert in Oracle RAC because Oracle RAC evolved from an OPS foundation.

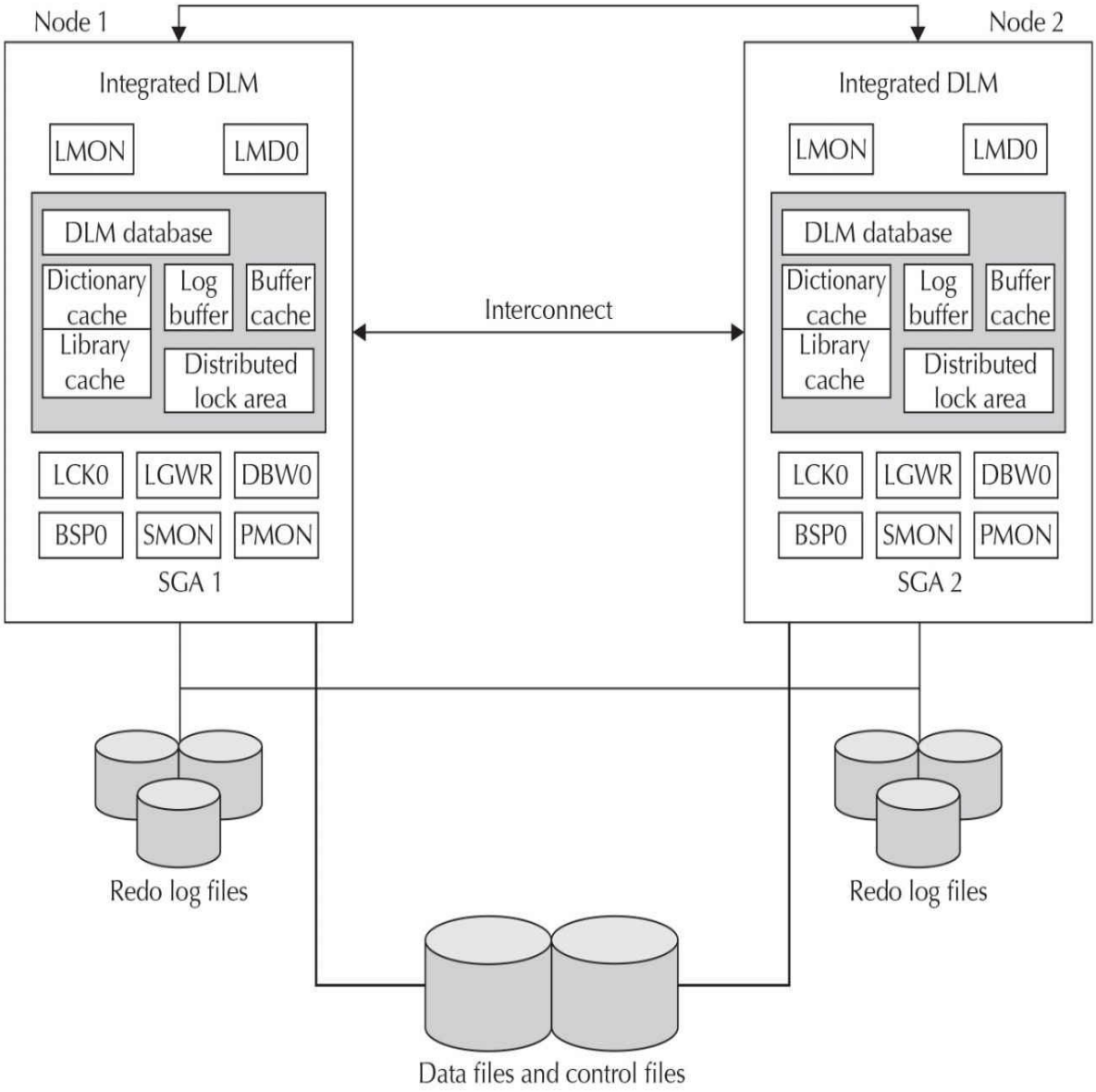


FIGURE 2-6. *OPS architecture*

Components of an OPS Database

You can find the following components on every instance of an Oracle 8*i* Parallel Server database:

https://t.me/learningnets

Technet24

- A cluster manager. This is OS vendor specific (except for Windows) and includes a node-monitoring facility and failure-detection mechanism.
- A distributed lock manager (DLM). The DLM includes deadlock detection and resource mastering.
- Cluster interconnect.
- A shared disk array.

NOTE
Vendor-specific cluster management software is not discussed here and is mentioned only for the sake of completeness. The vendor product is the basic mandatory cluster software needed before installing and configuring OPS and some variants of earlier versions of RAC. It is generally called the cluster manager (CM) and has its own group membership services, node monitor, and other core layers.

Cluster Group Services (CGS)

One of the key "hidden" or lesser-known components of OPS is Cluster Group Services (CGS). CGS had some OSD components (such as the node monitor interface), and the rest of it is the GMS part built into the Oracle kernel. CGS holds a key repository used by the DLM for communication and network-related activities. This layer in the Oracle 8*i* kernel (and beyond) provides the following key facilities without which an OPS database cannot operate:

- Internode messaging
- Group membership consistency
- Cluster synchronization
- Process grouping, registration, and deregistration

A whole set of cluster communication interfaces and APIs became an internal part of the Oracle code in OPS 8*i*. GMS provided many services (such as member status and node evictions) that were external in Oracle 8 but were made internal in Oracle 8*i*.

Distributed Lock Manager (DLM)

The DLM keeps an inventory of all the locks and global enqueues that are held by all the instances in the OPS database. Its job is to track every lock granted to a resource. The requests from various instances for lock acquisition and release are coordinated by the DLM. The memory structures required by DLM to operate are allocated out of the shared pool. The lock resources, messages buffers, and so on are all in the shared pool of each instance. The DLM is designed such that it can survive node failures in all but

https://t.me/learningnets

one node of the cluster.

The DLM is always aware of the current holders or requestors of locks and the grantee. In case locks are not available, the DLM queues the lock requests and informs the requestor when a lock resource becomes available. Some of the resources that the DLM manages are data blocks and rollback segments. Oracle resources are associated with the DLM locks by instance, using a complex hashing algorithm. The lock and enqueue functionality in OPS is the same as in a single-instance RDBMS server, except that OPS takes a global view.

DLM relies on the core RDBMS kernel for locking and enqueues services. The DLM coordinates locking at the global level, and this is a service that the core layers don't provide.

Locking Concepts in Oracle Parallel Server

In an OPS database, users must acquire a lock before they can operate on any resource. This is also applicable to a single-instance scenario. In pure DLM terminology, a *resource* is any object accessed by users, and a *lock* is a client operational request of a certain type or mode on that resource.

Parallel Cache Management (PCM) means that the coordination and maintenance of data blocks occur within each data buffer cache (of an instance) so that the data viewed or requested by users is never inconsistent or incoherent. The access to data is controlled via the PCM framework using data blocks with global coordinated locks. In simple terms, PCM ensures that only one instance in a cluster can modify a block at any given time. Other instances have to wait.

Broadly speaking, locks in OPS are either PCM locks or non-PCM locks. PCM locks almost exclusively protect the data blocks, and non-PCM locks control access to data files, control files, data dictionary, and so on. PCM locks are static in OPS, and non-PCM locks are dynamically set using the init.ora settings of certain parameters.

Locks on PCM resources are referred to as "lock elements" and non-PCM locks are called "enqueues." DLM locks are acquired on a resource and are typically granted to a process. PCM locks and row-level locks operate independently.

PCM Lock and Row Lock Independence

PCM locks and row locks operate independently. An instance can disown a PCM lock without affecting row locks held in the set of blocks covered by the PCM lock. A row lock is acquired during a transaction. A database resource such as a data block acquires a PCM lock when it is read for update by an instance. During a transaction, a PCM lock can therefore be disowned and owned many times if the blocks are needed by other instances.

In contrast, transactions do not release row locks until changes to the rows are either committed or rolled back. Oracle uses internal mechanisms for concurrency control to isolate transactions so modifications to data made by one transaction are not visible to other transactions until the transaction modifying the data commits. The row

https://t.me/learningnets

Technet24

lock concurrency control mechanisms are independent of parallel cache management: concurrency control does not require PCM locks, and PCM lock operations do not depend on individual transactions committing or rolling back.

IDLM lock modes and Oracle lock modes are not identical, although they are similar. In OPS, locks can be local or global, depending on the type of request and operations. Just as in a single instance, locks take the form

<Type, ID1, ID2>

where *Type* consists of two characters, and *ID1* and *ID2* are values dependent on the lock type. The ID is a 4-byte, positive integer.

Local locks can be divided into latches and enqueues. These could be required for local instance-based operations. A shared pool latch is a simple example of a local latch, irrespective of whether or not OPS is present. Enqueues can be local or global. They take on a global role in an OPS environment and remain local in a single instance. A TX (transaction) enqueue, control file enqueue (CF), DFS (Distributed File System) enqueue lock, and a DML/table lock are examples of global enqueues in an OPS database. The same enqueue is local in a single-instance database. Similarly, data dictionary and library cache locks are global in an OPS environment.

Local locks provide transaction isolation or row-level locking. Instance locks provide for cache coherency while accessing shared resources. GV$LOCK and GV$LOCK_ELEMENT are important views that provide information on global enqueues and instance locks.

Originally, two background processes—Lock Manager Daemon (LMD) and Lock Monitor (LMON)—implemented the DLM (refer to Figure 2-6). Each instance had its own set of these two processes. The DLM database stored information on resources, locks, and processes. In Oracle 9*i*, the DLM was renamed Global Cache Services (GCS) and Global Enqueue Services (GES).

DLM Lock Compatibility Matrix

Every resource in an Oracle RAC environment is identified by its unique resource name. Each resource can potentially have a list of locks currently granted to users. This list is called the “Grant Q.” Locks that are in the process of converting or waiting to be converted from one mode to another are placed on the “Convert Q” of that resource. For each lock, a resource structure exists in memory that maintains a list of owners and converters. Each owner, waiter, and converter has a lock structure, as shown in Table 2-2.

https://t.me/learningnets

Lock	NL	CR	CW	PR	PW	EX
NL	Grant	Grant	Grant	Grant	Grant	Grant
CR	Grant	Grant	Grant	Grant	Grant	Queue
CW	Grant	Grant	Grant	Queue	Queue	Queue
PR	Grant	Grant	Queue	Grant	Queue	Queue
PW	Grant	Grant	Queue	Queue	Queue	Queue
EX	Grant	Queue	Queue	Queue	Queue	Queue

TABLE 2-2. *DLM Lock Compatibility Matrix*

Every node has directory information for a set of resources it manages. To locate a resource, the DLM uses a hashing algorithm based on the name of the resource to find out which node holds the directory information for that resource. Once this is done, a lock request is given directly to this "master" node. The directory area is nothing but a DLM memory structure that stores information about which node masters which blocks.

The traditional lock-naming conventions (such as SS, SX, X) are provided in Table 2-3 along with the corresponding DLM mode.

Conventional naming	NL	CR	CW	PR	PW	EX
DLM naming	NL	SS	SX	S	SSX	X

TABLE 2-3. *Conventional Naming vs. DLM Naming*

Note that in this table, NL means null mode; CR/SS means concurrent read mode; CW/SX means concurrent write mode; PR/S means protected read mode; PW/SSX means protected write mode; and EX/X means exclusive mode.

Lock Acquisition and Conversion

Locks granted on resources are in the Grant Q (as discussed previously). Locks are placed on a resource when a process acquires a lock on the Grant Q of that resource. It is only then that a process owns a lock on that resource in a compatible mode.

A lock can be acquired if there are no converters and the mode the Oracle kernel requires is compatible with the modes already held by others. Otherwise, it waits on the Convert Q until the resource becomes available. When a lock is released or converted, the converters run the check algorithm to see if they can be acquired.

Converting a lock from one mode to another occurs when a new request arrives for a resource that already has a lock on it. *Conversion* is the process of changing a lock

https://t.me/learningnets

Technet24

from the mode currently held to a different mode. Even if the mode is NULL, it is considered as holding a lock. Conversion takes place only if the mode required is a subset of the mode held or the lock mode is compatible with the modes already held by others, according to a conversion matrix within the IDLM.

Processes and Group-Based Locking

When lock structures are allocated in the DLM memory area, the operating system process ID (PID) of the requesting process is the key identifier for the requestor of the lock. Mapping a process to a session is easier inside Oracle, and the information is available in V$SESSION. However, in certain clients, such as Oracle Multithreaded Server (MTS) and Oracle XA, a single process may own many transactions. Sessions migrate across many processes to make up a single transaction. This would disable identification of the transaction and the origin of the same. Hence, lock identifiers had to be designed to have session-based information, where the transaction ID (XID) is provided by the client when lock requests are made to the DLM.

Groups are used when group-based locking is used. This is preferred particularly when MTS is involved, and when MTS is used the shared services are implicitly unavailable to other sessions when locks are held. From Oracle 9*i* onward, process-based locking no longer exists. Oracle 8*i* OPS (and later) uses group-based locking irrespective of the kind of transactions. As mentioned, a process within a group identifies itself with the XID before asking Oracle for any transaction locks.

Lock Mastering

The DLM maintains information about the locks on all nodes that are interested in a given resource. The DLM nominates one node to manage all relevant lock information for a resource; this node is referred to as the “master node.” Lock mastering is distributed among all nodes.

Using the Interprocess Communications (IPC) layer enables the distributed component of the DLM to share the load of mastering (administering) resources. As a result, a user can lock a resource on one node but actually end up communicating with the LMD processes on another node. Fault tolerance requires that no vital information about locked resources be lost, regardless of how many DLM instances fail.

Asynchronous Traps

Communication between the DLM processes (LMON, LMD) across instances is implemented using the IPC layer across the high-speed interconnect. To convey the status of a lock resource, the DLM uses asynchronous traps (ASTs), which are implemented as interrupts in the OS handler routines. Purists may differ on the exact meaning of AST and the way it is implemented (using interrupts or other blocking mechanism), but as far as OPS and Oracle RAC are concerned, it is an interrupt. AST can be a *blocking* AST or an *acquisition* AST.

When a process requests a lock on a resource, the DLM sends a blocking asynchronous trap (BAST) to all processes that currently own a lock on that same

https://t.me/learningnets

resource. If possible and necessary, the holder(s) of the lock may relinquish it and allow the requester to gain access to the resource. An acquisition AST (AAST) is sent by DLM to the requestor to inform him that it now owns the resource (and the lock). An AAST is generally regarded as a “wakeup call” for a process.

How Locks Are Granted in a DLM

To illustrate how locking works in OPS’s DLM, consider a simple two-node cluster with a shared disk array:

1. Process p1 needs to modify a data block on instance 1. Before the block can read into the buffer cache on instance 1, p1 needs to check whether a lock exists on that block.
2. A lock may or may not exist on this data block, and hence the LCK process checks the SGA structures to validate the buffer lock status. If a lock exists, LCK has to request that the DLM downgrade the lock.
3. If a lock does not exist, a lock element (LE) has to be created by LCK in the local instance, and the role is local.
4. LCK must request the DLM for the LE in exclusive mode. If the resource is mastered by instance 1, DLM continues processing. Otherwise, the request must be sent to the master DLM in the cluster.
5. Assuming the lock is mastered on instance 1, the DLM on this instance does a local cache lookup in its DLM database and finds that a process on instance 2 already has an exclusive (EX) lock on the same data block.
6. DLM on instance 1 sends out a BAST to DLM on instance 2 requesting a downgrade of the lock. DLM on instance 2 sends another BAST to LCK on the same instance to downgrade the lock from EX to NULL.
7. The process on instance 2 may have updated the block and may not have committed the changes. The Dirty Buffer Writer (DBWR) is signaled to write out the block to disk. After the write confirmation, the LCK on instance 2 downgrades the lock to NULL and sends an AAST to DLM on the same instance.
8. DLM on instance 2 updates its local DLM database about the change in lock status and sends an AAST to DLM on instance 1.
9. The master DLM on instance 1 updates the master DLM database about the new status of the lock (EX), which can now be granted to the process on its instance. DLM itself upgrades the lock to EX.
10. DLM on instance 1 now sends another AAST to the local LCK process informing it about the lock grant and that the block can be read from disk.

Cache Fusion Stage 1, CR Server

OPS 8*i* introduced Cache Fusion Stage 1. Until version 8.1, cache coherency was

https://t.me/learningnets

Technet24

maintained using the disk (ping mechanism). Cache Fusion introduced a new background process called the *Block Server Process* (BSP). The major use or responsibility of BSP was to ship a consistent read (CR) version of a block (or blocks) across instances in a read/write contention scenario. The shipping was done using the high-speed interconnect and not the disk. This was called Cache Fusion Stage 1 because it was not possible to transfer all types of blocks to the requesting instance, especially with the write/write contention scenario.

Cache Fusion Stage 1 laid the foundation for Oracle RAC Cache Fusion Stage 2, in which both types of blocks (CR and CUR) can be transferred using the interconnect, although a disk ping is still required in some circumstances.

Oracle 8*i* also introduced the GV$ views, or "global views." With the help of GV$ views, DBAs could view cluster-wide database and other statistics sitting on any node/instance of the cluster. This was of enormous help to DBAs, because previously they had to group together data collected on multiple nodes to analyze all the statistics. GV$ views have the instance_number column to support this functionality.

Block Contention

Block contention occurs when processes on different instances need access to the same block. If a block is being read by instance 1 or is in the buffer cache of instance 1 in read mode, and another process on instance 2 requests the same block in read mode, *read/read* contention results. This situation is the simplest of all cases and can easily be overcome because there are no modifications to the block. A copy of the block is shipped across by BSP from instance 1 to instance 2 or read from the disk by instance 2 without the need to apply an undo to get a consistent version. In fact, PCM coordination is not required in this situation.

Read/write contention occurs when instance 1 has modified a block in its local cache, and instance 2 requests the same block for a read. In Oracle 8*i*, using Cache Fusion Stage 1, instance locks are downgraded, and the BSP process builds a CR copy of the block using the undo data stored in its own cache and ships the CR copy across to the requesting instance. This is done in coordination with DLM processes (LMD and LMON).

If the requesting instance (instance 2) needs to modify the block that instance 1 has already modified, instance 1 has to downgrade the lock, flush the log entries (if not done before), and *then* send the data block to disk. This is called a "ping." Data blocks are pinged only when more than one instance needs to modify the same block, causing the holding instance to write the block to disk before the requesting instance can read it into its own cache for modification. Disk ping can be expensive for applications in terms of performance.

A *false ping* occurs each time a block is written to disk, even if the block itself is not being requested by a different instance, but another block managed by the same lock element is being requested by a different instance.

A *soft ping* occurs when a lock element needs to be down-converted due to a request of the lock element by another instance, and the blocks covered by the lock are

https://t.me/learningnets

already written to disk.

Write/Write Contention

This situation occurs when both instances have to modify the same block. As explained, a disk ping mechanism results when the locks are downgraded on instance 1 and the block is written to disk. Instance 2 acquires the exclusive lock on the buffer and modifies it.

The Oracle RAC Solution

Although OPS was good for its time, it had several limitations, especially with scalability. In addition, it required careful setup, configuration, and a heavy amount of administration. It also required raw devices for storage, which were not easy for inexperienced system administrators to set up and administer.

Oracle RAC is a natural evolution of OPS. The limitations of the OPS are addressed through improvements in the code, the extension of Cache Fusion, and dynamic lock remastering.

Dynamic lock remastering allows frequently used database objects to be managed in the local instance, which greatly reduces the interconnect traffic and increases performance.

Oracle RAC also comes with an integrated clusterware and storage management framework, removing the dependency on the vendor clusterware and providing ultimate scalability and availability for database applications.

Availability

Oracle RAC systems can be configured to have no single point of failure, even when running on low-cost, commodity hardware and storage. If a monolithic server fails for any reason, the applications and databases running on the server are completely unavailable. With Oracle RAC, if any of the database servers fail, applications simply keep running and the services are continuously available through the remaining servers. The underlying clusterware makes sure the services are relocated to other running servers, and failover is frequently transparent to applications and occurs in seconds.

Oracle RAC has proven time and again in all “near death” situations that it can save the day with its controlled reliability. The paradigm has shifted from reliability to maximum reliability. Analyzing and identifying the subsystem and system failure rates in accordance with the appropriate standard ensures reliability, which is a key component in the Oracle RAC technology.

Scalability

Oracle RAC allows multiple servers in a cluster to manage a single database

https://t.me/learningnets

Technet24

transparently and without any interruptions to database availability. Additional nodes can be added when the database is up and running. With dynamic node addition and effective workload management, Oracle RAC allows database systems to scale out in both directions, with an ability to run an increased workload by increasing the number of nodes as well as the ability to reduce the amount of time to complete a workload. This means that the previous ceilings on scalability have been removed. Collections of servers can work together transparently to manage a single database with linear scalability.

Oracle RAC requires no changes to existing database applications. A clustered Oracle RAC database appears to applications as a traditional single-instance database environment. As a result, customers can easily migrate from single-instance configurations to Oracle RAC without needing to change their applications. Oracle RAC also requires no changes to existing database schemas.

Affordability

Oracle RAC allows organizations to use collections of low-cost computers to manage large databases rather than needing to purchase a single large, expensive computer. Clusters of small, commodity-class servers can now satisfy any database workload. For example, a customer needing to manage a large Oracle database might choose to purchase a cluster of eight industry-standard servers with four CPUs each, rather than buying a single server with 32 CPUs.

Oracle RAC allows clusters of low-cost, industry-standard servers running Linux to scale to meet workloads that previously demanded the use of a single, larger, more expensive computer. In the scale-up model of computing, sophisticated hardware and operating systems are needed to deliver scalability and availability to business applications. But with Oracle RAC, scalability and availability are achieved through functionality delivered by Oracle above and beyond the operating system.

Summary

Clustering is a great solution for scaling up and speeding up the database workload processing with extreme availability. Because the storage is shared among the nodes, all the members of a cluster can concurrently access the storage, which helps the system be available during node failures. Oracle databases implement the shared everything architecture to achieve on-demand scalability and ultimate availability.

The right clustering technology can achieve infinite scalability on demand and uninterrupted availability. Oracle databases have a good track record of clusters with OPS and its successor, Oracle RAC. The limitations of OPS are addressed in RAC, with the introduction of the Cache Fusion framework. Later versions further improved Cache Fusion with the introduction of the read-only and read-mostly locking frameworks.

With the introduction of the Grid Infrastructure, Oracle integrated Automatic Storage Management, a true volume manager and file system for the Oracle database

https://t.me/learningnets

infrastructure, with its clustered database solution. This eliminated the requirement of having a third-party file system manager for the Oracle infrastructure. Oracle RAC 12*c* provides impressive scalability, availability, and flexibility at a low cost. It makes the consolidation of databases affordable and reliable by leveraging scale across the architecture.

https://t.me/learningnets

Technet24

CHAPTER 3

Oracle RAC Architecture

The first two chapters of this book have set the stage for an in-depth discussion of Oracle RAC databases. This chapter introduces the architecture as well as the key concepts behind Oracle RAC databases.

The goal of this chapter is to show how an Oracle Real Application Clusters (RAC) database differs from a non-RAC, single-instance Oracle database. It explains the architecture of Oracle Clusterware, the foundation on which an Oracle RAC database runs, as well as that of an Oracle RAC database. You'll also learn about the key background processes that support Oracle Clusterware and the Oracle RAC database.

Introduction to Oracle Real Application Clusters

Unlike a normal Oracle database, where there's a single instance for the entire database, Oracle Real Application Clusters allows multiple instances to access a single database, and an Oracle RAC database can have a maximum of 100 instances. Instances running on multiple server nodes access the common set of datafiles comprising a single database. In a single-instance environment, one Oracle database is

https://t.me/learningnets

used by only one instance running on a server machine; thus, users accessing the database can connect to the database via that single server only. The processing resources (CPU memory and so on) available for the database work are limited to the single server's processing resources.

A *cluster* is a set of connected servers that appear as one entity to the end users and applications. Oracle RAC relies on Oracle Clusterware for making this possible. Oracle Clusterware connects the servers behind the scenes so that they work together as one logical entity. Oracle Clusterware provides the backbone, or the infrastructure, that makes Oracle RAC possible.

In an Oracle RAC environment, because multiple instances use the same database, database users gain from a higher throughput and the stability offered by a cluster—if one of the instances goes down, the database continues to remain accessible through the remaining running instances. As far as the users are concerned, the cluster, which consists of a set of servers hosting each of the instances, appears as a single entity. In an Oracle RAC database, there's an additional thread of redo for each instance in the cluster, as well as a dedicated undo tablespace for each of the instances.

NOTE

An instance is a set of background processes and memory structures operating in a machine associated with the database. A database is a collection of physical files. A database and an instance have a one-to-many relationship. A database can be concurrently mounted by more than one instance in Oracle RAC, and at any point, one instance will be part of only one database.

The nonvolatile storage for datafiles comprising the database is equally available to all the nodes for read and write access. Oracle RAC needs to coordinate and regulate simultaneous data accesses from multiple server nodes. Hence, an efficient, reliable, and high-speed private network must exist among the nodes of the cluster for sending and receiving data. Figure 3-1 shows the configuration of a single-instance database and an Oracle RAC database.

https://t.me/learningnets

Technet24

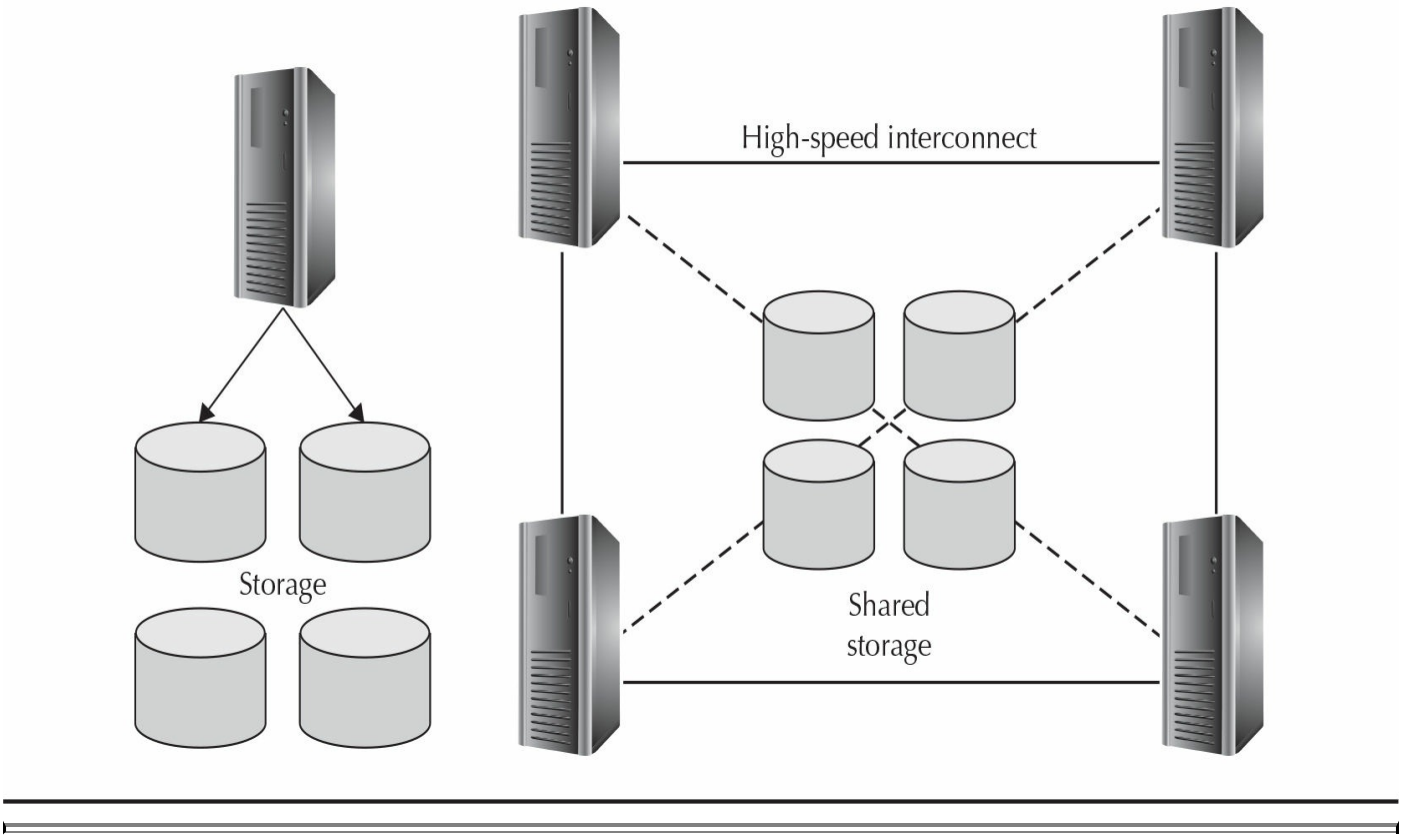


FIGURE 3-1. *Single-instance database and an Oracle RAC setup*

Single-Instance Environment vs. Oracle RAC Environment

Similar to in a single-instance environment, each instance in an Oracle RAC environment has its own System Global Area (SGA) and background processes. However, all the datafiles and control files are equally accessible to all the nodes, so these files must be placed on a shared disk subsystem. Each instance also has its own dedicated set of online redo log files. The online redo log files can be written to only by the instance to which they belong. However, these files must be readable by other instances during instance crash recovery. This requires that online redo log files reside on a shared disk subsystem and not on a node's local storage, because the files would be lost if the node crashed.

NOTE

An Oracle RAC database can have up to 100 instances.

Table 3-1 compares single-instance components to components of an instance in an Oracle RAC environment.

https://t.me/learningnets

Component	Single-Instance Environment	RAC Environment
SGA	Instance has its own SGA.	Each instance has its own SGA.
Background processes	Instance has its own set of background processes.	Each instance has its own set of background processes.
Datafiles	Accessed by only one instance.	Shared by all instances, so must be placed on shared storage.
Control files	Accessed by only one instance.	Shared by all instances, so must be placed on shared storage.
Online redo log file	Dedicated for writing and reading to only one instance.	Only one instance can write, but other instances can read during recovery and archiving. If an instance is shut down, log switches by other instances can force the idle instance redo logs to be archived.
Archived redo log	Dedicated to the instance.	Private to the instance, but other instances will need access to all required archive logs during media recovery.
Flash recovery log	Accessed by only one instance.	Shared by all the instances, so must be placed on shared storage.
Alert log and other trace files	Dedicated to the instance.	Private to each instance; other instances never read or write to those files.
Undo tablespace	Dedicated to the instance.	Dedicated to the instance.
ORACLE_HOME	Multiple instances on the same machine accessing different databases can use the same executable files.	Same as single instance, but can also be placed on shared file system, allowing a common ORACLE_HOME for all instances in an Oracle RAC environment.

TABLE 3-1. *Components of a Single-Instance Environment vs. an Oracle RAC Environment*

Instances in an Oracle RAC environment share the data because the same set of data might be needed by multiple instances simultaneously. There is no danger in multiple instances reading the same data simultaneously; however, data integrity issues may arise if one instance modifies (via insert or update) data while one or more other instances read or multiple instances modify the same data concurrently. This concurrent read/write or write/write behavior, if not coordinated correctly, might lead

https://t.me/learningnets

Technet24

to data corruption or inconsistent representation of the data. Single-instance Oracle already ensures that readers never block writers and that “dirty” reads are never allowed.

Oracle RAC ensures that all the instances across the clusters see a consistent image of the database. The Distributed Lock Manager (DLM), also called the *Global Resource Directory (GRD)*, coordinates resource sharing among the instances.

Oracle Flex Clusters

When you’re running an Oracle RAC with a large number of nodes, an Oracle Flex Cluster is a good solution. All nodes in such a cluster belong to a single Oracle Grid Infrastructure cluster.

You can use the Oracle Flex Clusters architecture to centralize policy decisions when deploying resources for various applications, as well as to manage responses to service-level failures and to manage recovery.

An Oracle Flex Cluster consists of two types of nodes:

- **Hub nodes** The hub nodes are tightly connected to other nodes and have direct access to shared storage. Hub nodes are similar to Oracle RAC Databases in previous releases of Oracle.
- **Leaf nodes** Leaf nodes are loosely coupled with hub nodes and may not have direct access to shared storage.

NOTE
You can convert a hub node to a leaf node, and vice versa, without any downtime.

You can have up to 64 hub nodes in an Oracle Flex Cluster, and you can have a large number of leaf nodes. You can use the two types of nodes to run different types of applications.

The Oracle RAC database instances you run on the leaf nodes in an Oracle Flex Cluster are called *reader nodes.* You can run a large number of reader nodes (up to 64 nodes) on a hub node in order to run massive parallel queries for crunching through large datasets. Oracle recommends that you allocate as much memory as possible to the reader nodes for parallel queries to use.

Once you set up the reader nodes and the read-only database instances, you can create services that point queries to these nodes for better parallel query performance. The database instances that run on reader nodes are quite different from instances running on the hub node. Note the following about these instances:

- The database instances run in a read-only mode.
- As long as the hub node to which a set of reader nodes is connected is still part

https://t.me/learningnets

of a cluster (that is, not evicted), the reader nodes can continue running without any performance slowdown.

Oracle Extended Clusters

Normally, you run the Oracle RAC environment from a single datacenter. However, you can configure Oracle RAC on an Oracle extended cluster, where the nodes of the cluster are geographically dispersed between multiple datacenters in the same area or even in different cities.

An extended-distance (or simply, extended) cluster is a special Oracle RAC architecture that allows you to provide fast recovery from a site failure by letting all nodes in all the sites process transactions as part of a single database cluster. For disaster recovery, Oracle recommends that you run Oracle Data Guard along with Oracle RAC. You must consider whether an extended cluster is suitable for you due to the additional latency from the cluster being spread out across a large area.

Although an extended cluster provides higher availability than a local Oracle RAC setup, it cannot protect you against all types of outages. For comprehensive data protection, Oracle recommends that you use Oracle Data Guard along with Oracle RAC.

Oracle Multitenant and Oracle RAC

The Oracle Multitenant option of Oracle Database 12*c* helps you consolidate databases and simplify provisioning and upgrades. Oracle Multitenant enables you to create a container database (CDB) that holds multiple pluggable databases (PDBs).

You can convert your normal Oracle databases into PDBs. You can create an Oracle RAC database as a CDB and then plug all your PDBs into the CDB. In an Oracle RAC environment, the multitenant CDB is based on Oracle RAC, with multiple instances of Oracle RAC CDB running. You can choose to make each of the PDBs available on some or all of the instances of the Oracle RAC CDB.

Because a PDB isn’t automatically started in a CDB instance, you must manually start the PDB on an Oracle RAC CDB, The PDB services will also be made available when you assign the first dynamic database services to a PDB. The PDB will then be available on the instances on which your service runs.

Administrator- vs. Policy-Managed Databases

Until Oracle Database 11*g* Release 2 (11.2), Oracle RAC deployment was limited to what’s called *administrator-managed deployment*. Under this type of deployment, you configured each database instance to run on a predetermined node in the cluster. You could also specify which instance a database service would run by designating the instance as preferred and available when creating (or modifying) a database service.

https://t.me/learningnets

Technet24

Starting from the Oracle 11.2 release, you have the choice of using the newer *policy-managed deployment*, where there's no direct link between database services and the instances on which they run. Instead of designating the preferred and available instances for a database service, you determine in which server pool a service will run. A *server pool* is a logical grouping of servers containing the resources managed by Oracle Clusterware to support applications. These resources include instances, services, application VIPs, and other application components. In order to use policy-managed deployment, you specify a service as "singleton" or "uniform" when creating or modifying a service. A singleton service will run only on a specific server in a server pool, whereas a service designated as uniform can run on any instance on any server in the server pool. The number of database instances is determined by the size of the server pool.

Server pools logically divide a cluster into groups of servers that host applications. Note that these "applications" can include a database service, for example, or even a non-database application such as an application server.

Policy-managed administration offers online server relocation based on policies you configure, in order to satisfy workload requirements. This ensures that the resources required for critical work are allocated as needed. A server can belong to a specific server pool at any given point in time. You can configure the policies to change the server pools based on the workload demand, to ensure that resource allocation matches changing workload requirements. You can also dedicate specific cluster servers to various applications and databases. Oracle uses the server pool attribute IMPORTANCE to direct the placement and prioritization of servers. Besides the IMPORTANCE attribute, you can configure a number of server pool attributes.

Dynamic Services

Dynamic database services enable you define characteristics and rules for users and applications in an Oracle RAC database. Oracle recommends that you create database *services*, which are an automatic workload management facility, to take advantage of various workload-balancing and failover options, as well as that you configure high availability.

A database service is a named representation of a single database instance or a set of instances. You can use services to group database workloads and ensure that client requests are serviced by the optimal instance(s). Services also help provide high availability by failing over client connections to surviving instances when one of the instances crashes or is otherwise unavailable.

Chapter 9 shows how you can use services along with the High Availability Framework, Fast Application Notification (FAN), Transaction Guard, Connection Load Balancing, Load Balancing Advisory, and Runtime Connection Load Balancing (RCLB) to provide highly reliable and optimal performance for users and applications.

Oracle Database Quality of Service

https://t.me/learningnets

Management

Oracle Database Quality of Service (QoS) Management is a policy-based product that helps monitor and manage the Oracle RAC workload. QoS enables you to automatically adjust resources such as server pools.

QoS Management automatically adjusts the configuration to keep applications running at peak performance levels. It also automatically adjusts to any changes in the system configuration and demand, thus helping smooth out application performance.

You can manage QoS through the Oracle QoS Management page in the Oracle Enterprise Manager Cloud Control.

Oracle RAC Components

Oracle Clusterware is a required component for an Oracle RAC database and provides the essential infrastructure to run the database. In an Oracle Database RAC database, the clustering software is integrated into the relational database management system (RDBMS) kernel.

Oracle Clusterware manages not only the database but also other resources necessary for the functioning of the cluster, such as the Virtual Internet Protocol (VIP) addresses, listeners, and database services. Oracle Clusterware together with Oracle's Automatic Storage Management is known as *Oracle Grid Infrastructure*.

NOTE

Although Oracle Clusterware is all you need for most operating systems in order to create and manage an Oracle RAC database, if your database applications need third-party vendor clusterware, you can use that clusterware, so long as it's certified by Oracle for use with Oracle Clusterware.

Here are the major components of Oracle RAC:

- A shared disk system
- Oracle Clusterware
- Cluster interconnects
- Oracle kernel components

Figure 3-2 shows the basic architecture.

https://t.me/learningnets

Technet24

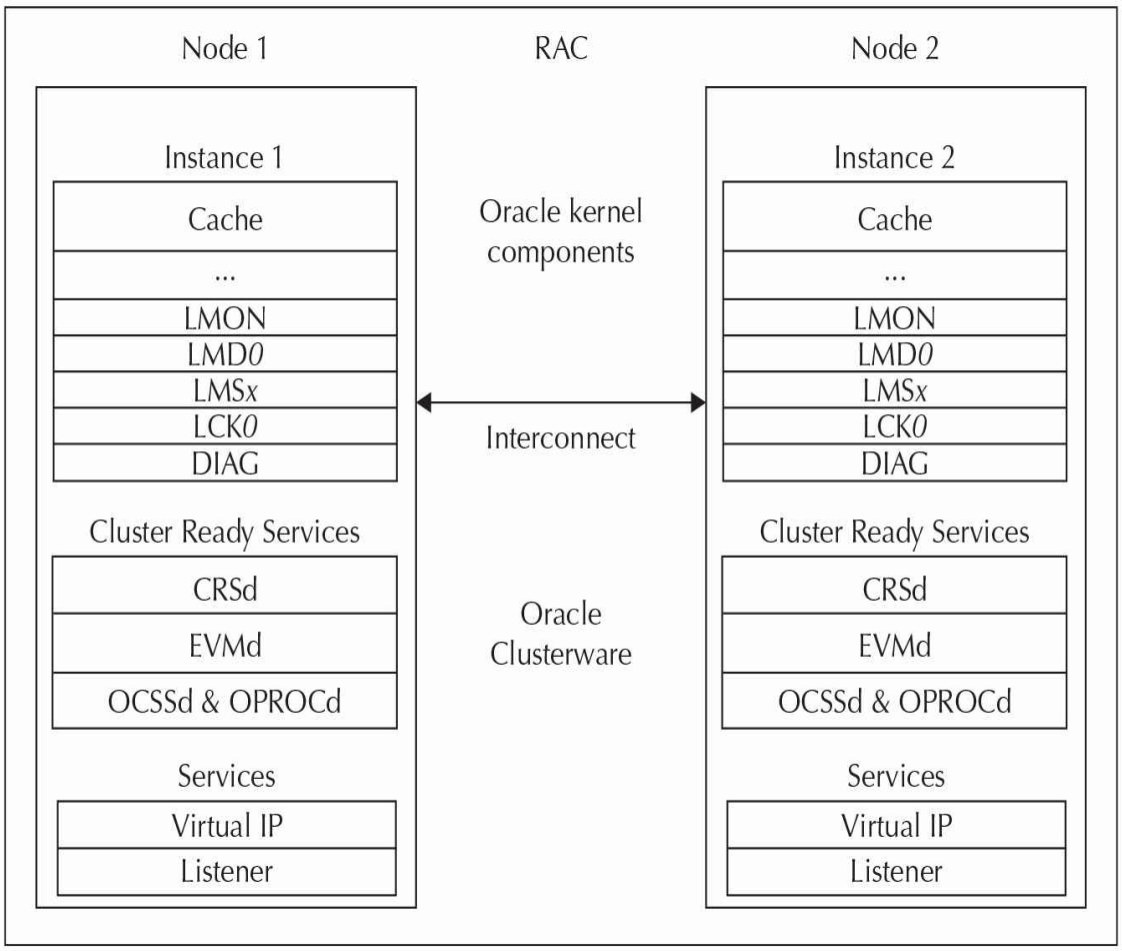


FIGURE 3-2. *Oracle 12c architecture*

Shared Disk System

Scalable shared storage is a critical component of an Oracle RAC environment. Traditionally, storage was attached to each individual server using SCSI (Small Computer System Interface) or SATA (Serial ATA) interfaces local to the hosts. Today, more flexible storage is popular and is accessible over storage area networks (SANs) or network-attached storage (NAS) using regular Ethernet networks. These new storage options enable multiple servers to access the same set of disks through a network, simplifying provisioning of storage in any distributed environment. SANs represent the evolution of data storage technology to this point.

Traditionally, in single-instance client/server systems, data was stored on devices either inside or directly attached to the server. This kind of storage is known as *directly attached storage (DAS)*. Next in the evolutionary scale came network-attached storage, which removed the storage devices from the server and connected them to the network. Storage area networks take the principle a step further by allowing storage devices to exist on their own separate networks and communicate with each other over very fast media, such as a high-speed Fibre Channel network. Users can gain access to these storage devices through server systems that are connected to the local area

https://t.me/learningnets

network (LAN) and SAN.

In shared storage, database files should be equally accessible to all the nodes concurrently. Generic file systems do not allow disks to be mounted in more than one system. Also, the regular UNIX file systems (UFS) do not allow the files to be shared among the nodes because of the obvious file-locking (inode locks) issues and the unavailability of a coherent file system cache. One option is to use the Network File System (NFS), but it is unsuitable because it relies on a single host (which mounts the file system) and for performance reasons. Because the disks in such an implementation are attached to one node, all the write requests must go through that particular node, thus limiting the scalability and fault tolerance. The total available I/O bandwidth is dependent on the bandwidth provided by that single host through which all I/O can be serviced. Because that node may become a single point of failure (SPOF), it is another threat for the high availability (HA) architecture.

The choice of file system is critical for Oracle RAC deployment. Traditional file systems do not support simultaneous mounting by more than one system. Therefore, you must store files in either raw volumes without any file system or on a file system that supports concurrent access by multiple systems.

Oracle RAC uses a "shared everything" architecture, which means that all datafiles, control files, redo log files, and SPFILEs must be stored on cluster-aware shared disks. Storing these files in shared storage ensures that all the database instances that are part of a cluster are able to access the various files necessary for instance operation.

In prior releases, Oracle allowed you to use raw devices as one way to provide the shared storage for Oracle RAC. However, in Oracle Database 12*c*, raw devices aren't certified for use in a RAC environment. Instead, you must use one of the following Oracle-supported cluster storage solutions:

- **Automatic Storage Management (ASM)** ASM is a portable, dedicated, and optimized cluster file system for Oracle database files. ASM is the only supported storage if you use Oracle Standard Edition with Oracle RAC. ASM is discussed in Chapter 6. Oracle recommends ASM as a storage option for Oracle RAC environments.
- **Cluster File System** One or more cluster file systems can be used to hold all Oracle RAC datafiles. Oracle recommends the use of the Oracle Cluster File System (OCFS). Alternatively, you can use a certified third-party cluster file system such as IBM GPFS, which works only on IBM AIX systems. Cluster file systems aren't widely used, so we will not be discussing them in great detail.

Oracle Clusterware

Oracle Clusterware software provides the underlying infrastructure that makes it possible for Oracle RAC to function. It is Oracle Clusterware that manages entities such as databases, listeners, services, and virtual IP addresses, which are referred to as *resources* in Oracle Clusterware parlance. Oracle Clusterware uses processes or

https://t.me/learningnets

Technet24

services that run on the individual nodes of the cluster to provide the high availability capabilities for a RAC database. It enables nodes to communicate with each other and forms the cluster that makes the nodes work as a single logical server.

Oracle Clusterware, which is integrated into the Oracle Grid Infrastructure, consists of several background processes that perform different functions to facilitate cluster operations. Oracle Clusterware is a mandatory piece of software required to run the Oracle RAC option. It provides the basic clustering support at the OS level and enables Oracle software to run in the clustering mode.

In the current release, Oracle has formally divided the Oracle Clusterware technology stack into two separate stacks. The Cluster Ready Services (CRS) stack is the *upper technology stack* and the Oracle High Availability Services (OHAS) stack constitutes the *foundational or lower layer*. Let's review the processes in each of the two technology stacks.

The Cluster Ready Services Stack

The CRS stack is the upper-level stack in Oracle Clusterware, and depends on the services provided by the lower-level High Availability Cluster Service stack. The CRS stack is comprised of the following key services:

- **CRS** This is the key service responsible for managing Oracle RAC high availability operations. The CRS daemon, named CRSd, manages the cluster's start, stop, and failover operations. CRS stores its configuration data in the Oracle Cluster Registry (OCR).
- **CSS** The CSS service manages the cluster's node membership and runs as the ocssd.bin process in a Linux-based system.
- **EVM** The Event Manager service (EVMd) manages FAN (Fast Application Notification) server callouts. FAN provides information to applications and clients about cluster state changes and Load Balancing Advisory events for instances services and nodes.
- **Oracle Notification Service** The ONSd daemon represents the Oracle Notification Service, which, among other things, helps publish the Load Balancing Advisory events.

The key background processes and services that comprise Oracle Clusterware are CRSd, OCSSd, EVMd, and ONSd. The operating system's init daemon starts these processes by using the Oracle Clusterware wrapper scripts, which are installed by Oracle Universal Installer during the installation of Oracle Clusterware. Oracle installs three wrapper scripts: init.crsd, init.evmd, and init.cssd. These are configured with the respawn action so that they will be restarted whenever they fail—except that the init.cssd wrapper script is configured with the **fatal** parameter, causing the cluster node to reboot to avoid any possible data corruption, as explained later in this chapter.

Table 3-2 summarizes the details of the key Oracle Clusterware processes and their functionality.

https://t.me/learningnets

CRS Process	Functionality	Failure of the Process	Runs As
Cluster Ready Services daemon (CRSd)	Resource monitoring, resource failover, and node recovery	Process restarted automatically depending on the startup mode. Node reboot if it is running in reboot mode. Does not cause node restart when running in restart mode.	root
Cluster Synchronization Services daemon (CSSd)	Basic node membership, Group Services, and basic locking	Node restart.	oracle
Oracle Notification Service (ONS)	Used to extend the HA notification to clients	Restarts automatically during failures.	oracle
Event Management daemon (EVMd)	Spawns a child process event logger and generates callouts	Automatically restarted; does not cause a node reboot.	oracle
Cluster Logging Services (ologgerd)	Stores information sent by the nodes in a cluster in the management database (MGMTDB)	Automatically restarted; does not cause a node reboot.	root

TABLE 3-2. *CRS Processes and Functionalities*

Cluster Ready Services (CRSd) Cluster Ready Services, which is represented by the CRSd process (or the CRSd daemon), manages all operations for all cluster resources (databases, listeners, and so on), such as starting and stopping the resources. The CRSd process also monitors all resources while they're running and can automatically restart any failed services.

Failure or death of the CRSd can cause node failure, and it automatically reboots the nodes to avoid data corruption because of possible communication failure among the nodes. The CRSd runs as the superuser *root* in the UNIX platforms and runs as a background service in Microsoft Windows platforms.

The Cluster Ready Services daemon process is spawned by the init.crsd wrapper script, provides the high availability framework for Oracle Clusterware, and manages the states of cluster resources by starting, stopping, monitoring, and relocating the failed cluster resources to the available cluster nodes within the cluster. A cluster resource can be a network resource such as a virtual IP, database instance, listener, database, or any third-party application, such as a web server. The CRSd process retrieves the cluster resource's configuration information stored in the Oracle Cluster Registry (OCR) before taking any action on the cluster resource. CRSd also uses OCR

https://t.me/learningnets

Technet24

to maintain the cluster resource profiles and statuses. Every cluster resource has a resource profile, which is stored in the Oracle Cluster Registry.

CRS is installed and run from a different Oracle home, known as GRID_HOME, that is independent from ORACLE_HOME. Here's a list of the key functions performed by CRS:

- CRSd manages resources such as starting and stopping the services and failovers of the application resources. It spawns separate processes to manage application resources.
- CRSd has two modes of running during startup and after a shutdown. During a planned clusterware startup, it is started in reboot mode. It is started in restart mode after an unplanned shutdown. In reboot mode, CRSd starts all the resources under its management. In restart mode, it retains the previous state and returns the resources to their previous states before shutdown.
- CRS manages the Oracle Cluster Registry (OCR) and stores the current known state in the OCR.
- CRS runs as root on UNIX and LocalSystem on Windows and automatically restarts in case of failure.
- CRS requires a public interface, private interface, and the virtual IP (VIP) for operation. Public and private interfaces should be up and running and should be pingable from each other before the CRS installation is started. Without this network infrastructure, CRS cannot be installed.

How Cluster Ready Services (CRS) Works The Oracle Grid Infrastructure uses CRS for interaction between the OS and the database. CRS is the background engine for the Oracle 12*c* RAC high availability framework, which provides the standard cluster interface for all platforms. In this discussion, we use the Oracle Grid Infrastructure and Oracle Clusterware interchangeably.

The Oracle Grid Infrastructure must be installed in a separate Oracle home before the standard RAC installation. This separate Oracle home is known as GRID_HOME. The Oracle Grid Infrastructure is a mandatory component for Oracle 12*c* RAC.

When Oracle Clusterware is installed on the cluster where third-party clusterware is integrated, CRS relies on the vendor clusterware for the node membership functionality and just manages Oracle services and resources. CRS manages the node membership functionality along with managing regular RAC-related resources and services.

Cluster Membership Decisions Oracle Clusterware contains logic for determining the health of other nodes of the cluster and whether they are alive. After a certain period of inactivity of a node, Oracle Clusterware declares the node dead and evicts it. If a new node needs to be added, it is allowed to join immediately.

Resource Management Frameworks Resources represent applications or

https://t.me/learningnets

system components (both local and remote to the cluster) whose behavior is wrapped and monitored by a cluster framework. Thus, the application or system component becomes highly available. Correct modeling of resources stipulates that they must be managed by only one cluster framework. Having multiple frameworks managing the same resource can produce undesirable side effects, including race conditions in start/stop semantics.

Consider an example in which two cluster frameworks manage the same shared storage, such as a raw disk volume or cluster file system. In the event that the storage component goes down, both frameworks may compete in trying to bring it back up and may decide to apply totally different recovery methods (for example, restarting vs. using a notification for human intervention, or waiting before restarting vs. retrying in a tight loop).

In the case of single application resources, both frameworks may even decide to fail over the component to a totally different node. In general, clusterware software systems are not prepared to handle resources that are managed by multiple HA frameworks.

Starting and Stopping Oracle Clusterware When a node fails, Oracle Clusterware is brought up at boot time via the init daemon (on UNIX) or Windows Service Management (on Windows). Therefore, if the init process fails to run on the node, the OS is broken and Oracle Clusterware does not start.

You can start, stop, enable, and disable Oracle Clusterware with the following commands (which must be run as the superuser):

```
crsctl stop crs  # stops Oracle Clusterware
crsctl start crs  # starts Oracle Clusterware
crsctl enable crs  # enables Oracle Clusterware
crsctl disable crs  # disables Oracle Clusterware
```

The commands to start and stop Oracle Clusterware are asynchronous, but while it is stopped, a small wait time may occur before control is returned. Only one set of CRSs can be run on one cluster. Adding the **-cluster** option to these commands makes them global across the RAC environment.

Oracle Clusterware APIs are documented, and customers are free to use these programmatic interfaces in their custom and non-Oracle software to operate and maintain a coherent cluster environment. You must use SRVCTL to start and stop Oracle resources named with “ora.” Oracle does not support third-party applications that check Oracle resources and take corrective actions on those resources. Best practice is to leave Oracle resources controlled by Oracle Clusterware. For any other resource, either Oracle or the vendor clusterware (not both) can manage it directly.

The Clusterware Startup Process The Oracle High Availability Service daemon (OHASd) starts all other Oracle Clusterware daemons. During installation of Oracle Grid Infrastructure, Oracle adds the following entry into the /etc/inittab file:

https://t.me/learningnets

Technet24

The /etc/inittab file executes the /etc/init.d/init.ohasd control script with the **run** argument, which spawns the ohasd.bin executable. The cluster control files are stored at the location /etc/oracle/scls_scr/<hostname>/root. The /etc/init.d/init.ohasd control script starts the OHASd based on the value of the ohasdrun cluster control file. The value **restart** causes Oracle to restart the crashed OHASd using **$GRID_HOME/bin/ohasd restart**. The value **stop** indicates a scheduled shutdown of the OHASd, and the value **reboot** causes Oracle to update the ohasdrun file with a value of **restart** so that Oracle will restart the crashed OHASd.

The OHASd daemon uses cluster resources to start other Clusterware daemons. The OHASd daemon will have one cluster resource for each Clusterware daemon, and these resources are stored in the Oracle Local Registry.

Agents help manage the Clusterware daemons, by performing actions such as starting, stopping, and monitoring the Clusterware daemons. The four main agents are oraagent, orarootagent, cssdagent, and cssdmonitor. These agents perform start, stop, check, and clean actions on their respective Clusterware daemons.

So, to put everything into context, the OHASd daemon will start all other Clusterware daemons. Once these are started, OHASd will start the daemon resources, and then the daemon resources using their respective agents will start the underlying Clusterware agents.

Oracle Cluster Registry Oracle Clusterware uses the Oracle Cluster Registry to store the metadata, configuration, and state information of all the cluster resources defined in Oracle Clusterware. Oracle Cluster Registry (OCR) must be accessible to all nodes in the cluster, and Oracle Universal Installer will fail to install all nodes in the installer that do not have proper permissions to access the Oracle Cluster Registry files, which are binary files and cannot be edited by any other Oracle tools. The OCR is used to bootstrap CSS for port information, nodes in the cluster, and similar information.

The Cluster Synchronization Services daemon (CSSd) updates the OCR during cluster setup, and once the cluster is set up, OCR is used for read-only operations. Exceptions are any change in a resource status that triggers an OCR update, services going up or down, network failovers, ONS and a change in application states, and policy changes.

OCR is the central repository for CRS and it stores details about the services and status of the resources. OCR is the registry equivalent of Microsoft Windows, which stores name/value pairs of information, such as resources that are used to manage the resource equivalents by the CRS stack. Resources with the CRS stack are components that are managed by the CRS and need to store some basic demographic information about the resources—the good state, the bad state, and the callout scripts. All such information makes it into the OCR. The OCR is also used to bootstrap CSS for the port information, nodes in the cluster, and similar information. This is a binary file and cannot be edited by any other Oracle tools.

https://t.me/learningnets

Oracle Universal Installer provides the option to mirror the OCR file during installation of Oracle Clusterware. Although this is important, it is not mandatory. The OCR file should be mirrored if the underlying storage does not guarantee continuous access to the OCR file. The mirror OCR file has the same content as the primary OCR file and must be housed on shared storage, such as a cluster file system or raw device.

Oracle stores the location of the OCR file in a text file called ocr.loc, which is located in different places depending on the operating system. For example, on Linux-based systems the ocr.loc file is placed under the /etc/oracle directory, and for UNIX-based systems the ocr.loc is placed in /var/opt/oracle. Windows systems use the registry key Hkey_Local_Machine\software\Oracle\ocr to store the location of the ocr.loc file.

Oracle Universal Installer (OUI) also uses OCR during installation time. All the CSS daemons have read-only access during startup. Oracle uses an in-memory copy of OCR on each cluster node to optimize the queries against the OCR by various clients, such as CRS, CSS, SRVCTL, NETCA, Enterprise Manager, and DBCA. Each cluster node has its own private copy of the OCR, but to ensure the atomic updates against the OCR, no more than one CRSd process in a cluster is allowed to write into the shared OCR file.

This master CRSd process refreshes the OCR cache on all cluster nodes. Clients communicate with the local CRSd process to access the local copy of the OCR and to contact the master CRSd process for any updates on the physical OCR binary file. OCR also maintains the dependency hierarchy and status information of the cluster resources defined within the cluster. For example, there is a service resource that cannot be started until the dependent database resource is up and running, and OCR maintains this information.

The Cluster Synchronization Services daemon (CSSd) updates OCR during the cluster setup. Once the cluster is set up, OCR will be used by read-only operations. During node additions and deletions, CSS updates the OCR with the new information. The CRS daemon will update the OCR about the status of the nodes during failures and reconfigurations. Other management tools such as NetCA, DBCA, and SRVCTL update the services information in the OCR as they are executed. OCR information is also cached in all nodes, and the OCR cache will benefit more of the read-only operations.

The OCR file is automatically backed up in the OCR location every four hours. These backups are stored for a week and circularly overwritten. The last three successful backups of OCR (a day old and a week old) are always available in the directory $ORA_CRS_HOME/cdata/<cluster name>. The OCR backup location can also be changed using the ocrconfig command-line utility.

You can also take a manual backup of the OCR binary file by executing the **ocrconfig -manualbackup** command, provided that Oracle Clusterware is up and running on the cluster node. The user running the **ocrconfig** command must have administrative privileges. The **ocrconfig** command can also be used to list the existing backups of the OCR and the backup location, as well as to change the backup location. Chapter 8 provides detailed discussion of the management of OCR.

https://t.me/learningnets

Technet24

Oracle Local Registry The Oracle Local Registry (OLR) is similar to the Oracle Cluster Registry, but it only stores information about the local node. The OLR is not shared by other nodes in the cluster and is used by the OHASd while starting or joining the cluster.

The OLR stores information that is typically required by the OHASd, such as the version of Oracle Clusterware, the configuration, and so on. Oracle stores the location of the OLR in a text file named /etc/oracle/olr.loc. This file will have the location of the OLR configuration file $GRID_HOME/cdata/<hostname.olr>. Oracle Local Registry is similar to OCR in terms of internal structure because it stores information in keys, and the same tools can be used to either check or dump the data of OLR.

Voting Disk A *voting disk* is a shared disk that will be accessed by all the member nodes in the cluster during an operation. The voting disk is used as a central reference for all the nodes and keeps the heartbeat information between the nodes. If any of the nodes is unable to ping the voting disk, the cluster immediately recognizes the communication failure and evicts the node from the cluster group to prevent data corruptions. The voting disk is sometimes called a “quorum device” because the split-brain resolution is decided based on the ownership of the quorum device.

The voting disk manages the cluster membership and arbitrates the cluster ownership during communication failures between the nodes. Oracle RAC uses the voting disk to determine the active instances of the cluster, and inactive instances are evicted from the cluster. Because the voting disk plays a vital role, you should mirror it. If Oracle mirroring is not used for the voting disk, external mirroring should be used.

Voting is perhaps the most universally accepted method of arbitration. It has been used for centuries in many forms for contests, for selecting members of government, and so forth. One problem with voting is plurality—the leading candidate gains more votes than the other candidates, but not more than half of the total votes cast. Other problems with voting involve ties. Ties are rare, and the node contained in the master sub cluster will survive. Starting with the 11.2.0.2 release, we can evict the specific sub cluster that has the smaller number of nodes all the time rather than making a guess based on the voting map.

Occasionally, voting is confused with a quorum. They are similar but distinct. A *vote* is usually a formal expression of opinion or will in response to a proposed decision. A *quorum* is defined as the number, usually a majority of officers or members of a body, that, when duly assembled, is legally competent to transact business. Both concepts are important; the only vote that should ratify a decision is the vote of a quorum of members. For clusters, the quorum defines a viable cluster. If a node or group of nodes cannot achieve a quorum, they should not start services because they risk conflicting with an established quorum.

Cluster Synchronization Services (CSS) Cluster Synchronization Services, represented by the Oracle Cluster Synchronization Services daemon (OCSSd), manages the cluster configuration by specifying which nodes are members of the RAC

https://t.me/learningnets

cluster. OCSSd provides synchronization services among the nodes. It provides the access to the node membership and enables basic cluster services, including cluster group services and cluster locking. It can also run without integrating with vendor clusterware.

Failure of OCSSd causes the machine to reboot to avoid a “split-brain” situation (when all the links of the private interconnect fail to respond to each other, but the instances are still up and running; each instance thinks that the other instance is dead and tries to take over ownership). This is also required in a single instance if Automatic Storage Management (ASM) is used. (ASM is discussed in detail in Chapter 6.) OCSSd runs as the *oracle* user.

The OCSSd process is spawned by the init.cssd wrapper script, runs as a non-root operating system user with real-time priority, and manages the configuration of Oracle Clusterware by providing node cluster and group membership services. OCSSd provides these services via two types of heartbeat mechanisms: network heartbeat and disk heartbeat. The main objective of the network heartbeat is to check the viability of the Oracle cluster, whereas the disk heartbeat helps to identify the split-brain situation. Due to this fact, it is very important that OCSSd is always running; hence, the init.cssd wrapper script is configured with the **fatal** parameter. Failure of OCSSd causes the machine reboot to avoid a split-brain situation. This is also required in the single instance if Automatic Storage Management (ASM) is used.

The following list summarizes the functionalities of OCSSd:

- CSS provides basic Group Services support. Group Services is a distributed group membership system that allows applications to coordinate activities to achieve a common result.
- Lock Services provides the basic cluster-wide serialization locking functions. It uses the First In, First Out (FIFO) mechanism to manage locking.
- Node Services uses OCR to store data and updates the information during reconfiguration. It also manages the OCR data, which is otherwise static.
- The *cssdagent* process is part of Oracle Synchronization Services, and its job is to continuously monitor the cluster to provide the I/O fencing solution for Oracle Clusterware. The prime objective of the cssdagent process is to identify potential cluster node hangs and to reboot any hanged node so that processes on that node cannot write to the storage. The cssdagent process is locked in memory and runs as a real-time process. It sleeps for a fixed time and runs as the root user. Failure of the cssdagent process causes the node to automatically restart.

The Event Manager Process The third key component in OCS is called the Event Management Logger, which runs the Oracle Event Manager daemon process EVMd. This daemon process spawns a permanent child process called *evmlogger* and generates the events when things happen. The evmlogger spawns new children processes on demand and scans the callout directory to invoke callouts.

https://t.me/learningnets

Technet24

The EVMd process receives the FAN events posted by clients and distributes the FAN events to the clients that have subscribed to them. The EVMd process will restart automatically on failure, and the death of the EVMd process does not halt the instance. The EVMd process runs as the oracle user.

Imagine an Oracle cluster composed of two cluster nodes: Node A and Node B. When Node B leaves the cluster, the OCSSd process on Node A posts a leave FAN event, which the EVMd process on Node A publishes to the CRSd process on Node A because the CRS process is the subscriber for the leave FAN event. Oracle provides the evmwatch and evmget utilities to view the FAN events in the standard output. These utilities are also useful to test the functionality of the EVMd process.

I/O Fencing

Fencing is an important operation that protects processes from other nodes modifying the resources during node failures. When a node fails, it needs to be isolated from the other active nodes. Fencing is required because it is impossible to distinguish between a real failure and a temporary hang. Therefore, we assume the worst and always fence (if the node is really down, it cannot do any damage; in theory, nothing is required. We could just bring it back into the cluster with the usual join process). Fencing, in general, ensures that I/O can no longer occur from the failed node.

Other techniques can be used to perform fencing. The most popular are reserve/release (R/R) and persistent reservation (SCSI3). SAN Fabric fencing is also widely used both by Red Hat Global File System (GFS) and Polyserv. Reserve/release by its nature works only with two nodes. (That is, one of the two nodes in the cluster upon detecting that the other node has failed will issue the reserve and grab all the disks for itself. The other node will commit suicide if it tries to do I/O in case it was temporarily hung. The I/O failure triggers some code to kill the node.)

In general, in the case of two nodes, R/R is sufficient to address the split-brain issue. For more than two nodes, the SAN Fabric fencing technique does not work well because it would cause all the nodes but one to commit suicide. In those cases, persistent reservation, essentially a match on a key, is used. In persistent reservation, if you have the right key, you can do I/O; otherwise, your I/O fails. Therefore, it is sufficient to change the key on a failure to ensure the right behavior during failure.

All of the Oracle clients that are I/O capable can be terminated by the CSS before it takes a node out, which avoids the I/Os from going through because they will be intercepted and not completed. Hence, the database will not treat those transactions as committed and errors will be returned to the app; for those that did get committed and did not make it to the datafiles, the database will recover them on startup of the same instance or another instance that is running.

https://t.me/learningnets

Oracle Notification Services The Oracle Notification Services (ONS) process is configured during the installation of Oracle Clusterware and is started on each cluster node when CRS starts. Whenever the state of a cluster resource changes, the ONS process on each cluster node communicates with one another and exchanges the HA event information. CRS triggers these HA events and routes them to the ONS process, and then the ONS process publishes the HA event information to the middle tier. To use ONS on the middle tier, you need to install ONS on each host where you have client applications that need to be integrated with FAN. Applications use these HA events for various reasons, especially to quickly detect failures. The whole process of triggering and publishing the HA events is called Fast Application Notification (commonly known as FAN in the Oracle community). Alternatively, the HA events are called FAN events.

These FAN events alone are of no use until applications do not have the logic to respond to the FAN events published by the ONS process. The best way to receive and respond to these FAN events is by using a client that is tightly integrated with FAN, such as Java Database Connectivity (JDBC) Implicit Connection Cache, Universal Connection Pool for Java, or database resources. Using a user-defined callout script is another way to publish these FAN events to the AQ (Advance Queue) tables. FAN events contain important information such as the event type, reason, and status, and users can write their own callout scripts to act upon certain types of events instead of taking action on each FAN event.

In addition to the CRS, CSS, EVM and ONS services, the Oracle Clusterware Technology Stack also utilizes the following processes:

- **Cluster Time Synchronization Services (CTSS)** Performs time management tasks for the cluster.
- **Grid Naming Service (GNS)** If you choose to use GNS, the GNS services process is in charge of name resolution and it handles requests sent from DNS servers to the cluster nodes.
- **Oracle Agent (oraagent)** The oraagent process replaces the RACG process in Oracle 11.1 and supports Oracle-specific requirements.
- **Oracle Root Agent (orarootagent)** Lets the CRSd manage root-owned resources, such as the network.

The Oracle High Availability Services Technology Stack

The Oracle High Availability Services (OHAS) stack is the foundational layer of Oracle Clusterware and is responsible for supporting high availability in an Oracle RAC environment. The OHAS stack consists of the following key processes:

- **Cluster Logging Services (ologgerd)** This process stores information it receives from the various nodes in a cluster in an Oracle Grid Infrastructure

https://t.me/learningnets

Technet24

Management Repository, which is a default new Oracle 12*c* database named MGMTDB that's automatically created for you when you create an Oracle 12*c* RAC database. In Oracle Grid Infrastructure 12.1.0.1, you had to choose if you wanted the installer to create the MGMTDB database. Starting with the Oracle Grid Infrastructure 12.1.0.2 release, this database is mandatory and is automatically installed. The database is created as a CDB, with a single PDB. This new database, which is a small Oracle 12*c* database, runs on only one of the nodes in an Oracle RAC cluster. Note that the ologgerd service will run on only one of the nodes in a cluster.

- **Grid Plug and Play (GPNPd)** This process manages the Grid Plug and Play profile and ensures that all nodes in a cluster have a consistent profile.
- **Grid Interprocess Communication (GIPC*d*)** This is a support daemon that underlies Redundant Interconnect Usage, which facilitates IP failover natively, without the need for tools such as bonding. This is also the underlying protocol used by the clusterware to communicate.
- **Oracle Agent (oraagent)** Not to be mistaken with the oraagent process on the CRS technology stack, this process also supports Oracle-specific requirements.
- **Oracle Root Agent (orarootagent)** This process is the counterpart to the similarly named agent process in the CRS technology stack and is meant to help manage root-owned services that run in the Oracle High Availability Services Technology stack layer. This process is required for running resources that need root-elevated privileges.

Oracle RAC Networking Concepts and Components

Oracle RAC environments involve concepts such as a virtual IP and SCAN (Single Client Access Name). In addition, you need to understand networking components such as network cards, network IP addresses, network bonding, and cluster interconnects. The following sections describe the key Oracle RAC–related networking concepts and components.

Key Networking Concepts

The most important Oracle RAC networking concepts you need to understand are

- Oracle virtual IPs
- Application virtual IP
- SCAN

The following sections explain each of these concepts.

https://t.me/learningnets

Oracle Virtual IP

A virtual IP is required to ensure that applications can be designed to be highly available. A system needs to eliminate all single points of failure. In Oracle, clients connected to an Oracle RAC database must be able to survive a node failure. Client applications connect to the Oracle instance and access the database through the instance. Therefore, a node failure will bring down the instance to which the client might have connected.

The first design available from Oracle was Transparent Application Failover (TAF). With TAF, a session can fail over to the surviving instances and continue processing. Various limitations exist with TAF; for instance, only query failover is supported. Also, to achieve less latency in failing over to the surviving node, Oracle tweaked the TCP timeout (platform dependent; defaults to 10 minutes in most UNIX ports). It wouldn't be a good idea to design a system in which a client takes 10 minutes to detect that there is no response from the node to which it has connected.

To address this, Oracle uses a feature called the *cluster VIP*—a cluster virtual IP address that would be used by the outside world to connect to the database. This IP address needs to be different from the set of IP addresses within the cluster. Traditionally, listeners would be listening on the public IP of the box and clients would contact the listeners on this IP. If the node dies, the client would take the TCP timeout value to detect the death of the node. Each node of the cluster has a VIP configured in the same subnet of the public IP. A VIP name and address must be registered in the DNS in addition to the standard static IP information. Listeners would be configured to listen on VIPs instead of the public IP.

When a node is down, the VIP is automatically failed over to one of the other nodes. During the failover, the node that gets the VIP will "re-ARP" to the world, indicating the new MAC address of the VIP. Clients who have connected to this VIP will immediately get a reset packet sent. This results in clients getting errors immediately rather than waiting for the TCP timeout value. When one node goes down in a cluster and a client is connecting to the same node, the client connection will be refused by the down node, and the client application will choose the next available node from the descriptor list to get a connection. Applications need to be written so that they catch the reset errors and handle them. Typically for queries, applications should see an ORA-3113 error.

NOTE

In computer networking, the Address Resolution Protocol (ARP) is the method of finding the host's hardware addresses (MAC address) when only the IP address is known. The hosts use ARP when they want to communicate with each other in the same network. ARP is also used by routers to forward a packet from one host through another router. In cluster VIP failovers, the new node that gets the VIP advertises the new ARP address to the world. This is typically known as gratuitous ARP, and during

https://t.me/learningnets

Technet24

this operation, the old hardware address is invalidated in the ARP cache, and all the new connections will get the new hardware address.

Application VIP

Oracle uses a virtual IP to allow Oracle database clients to rapidly recognize that a node has died, thus improving the reconnect time. Oracle also extends this capability to user applications. An application VIP is a cluster resource for managing the network IP address. An application VIP is mostly used by an application that is accessed over the network. When an application is built dependent on an application VIP (AVIP), whenever a node goes down, the application VIP fails over to the surviving node and restarts the application process on the surviving node.

The steps for creating an application VIP are the same as creating any other cluster resource. Oracle supplies the standard action program script usrvip, which is located under the <GRID_HOME>/bin directory. Users must use this action script to create an application VIP in an Oracle cluster. DBAs always wonder about the difference between a normal VIP and an application VIP. Upon failover to a surviving node, a normal VIP does not accept connections and forces clients to reconnect using another address, whereas an application VIP remains fully functional after it is relocated to another cluster node and it continuously accepts connections.

Single Client Access Name

SCAN is a single network name that resolves to three different IP addresses registered either in DNS or GNS. During installation of Oracle Grid Infrastructure, Oracle creates three SCAN listeners and SCAN VIPs. If Grid Name Server (GNS) is not used, then SCAN must be registered in DNS before installation of the Oracle Grid Infrastructure. Oracle fails over the SCAN VIP and the listener to another node in case of failure. Each database instance registers itself to the local listener and the SCAN listener using the database initialization parameter REMOTE_LISTENER, because this is the only way the SCAN listener knows about the location of the database instances. The SRVCTL utility can be used to manage and monitor the SCAN resources in the cluster.

SCAN allows users to connect to the database in the cluster using the EZconnect or jdbc thin driver, as shown here:

```
sqlplus system/password@my-scan:1521/oemdb
jdbc:oracle:thin:@my-scan:1521/oemdb
```

SCAN eases client connection management because there is no need to change the client connection strings even if the client's target database is moved to a different node in the cluster. With more and more customers deploying Oracle RAC and building shared database services, there's always a demand to simplify the client connection management and failover capabilities between moved database instances. For example, prior to Oracle 11*g* Release 2, there was no way other than modifying the service to notify the client connection that the database instance it is intending to

https://t.me/learningnets

connect to using the Oracle RAC service has been moved to another cluster node. SCAN allows failover to the moved database instances without modifying the Oracle RAC service. The following steps explain how a client connects to the database in a cluster using SCAN:

1. The TNS layer retrieves the IP address of the SCAN from the Domain Name Server or Grid Name Server; it then load-balances and fails over across the IP address.
2. The SCAN listener is now aware of databases in the cluster and will redirect the connection to the target database node VIP.

Oracle provides self-management of the network requirements of the cluster by supporting Dynamic Host Configuration Protocol (DHCP) for private interconnect and virtual IP addresses. Oracle needs an optimized way to resolve the IP address to a name; hence, it developed its own Grid Name Service, which is linked to the Domain Name Service and allows users to connect to the cluster and the databases in the cluster. Oracle uses a dedicated subdomain and a virtual IP GNS (aka GNS VIP) that is registered in the DNS. Each cluster will have its own GNS, GNS virtual IP, and a dedicated subdomain, and the subdomain will forward all requests for addresses in the subdomain to the GNS virtual IP. Oracle fails over the GNS and GNS virtual IP to another node in case of node failure.

When using GNS and DHCP, Oracle obtains the virtual IP address from the DHCP server and configures the SCAN name during cluster configuration. Oracle dynamically obtains new IP addresses from the DHCP server and configures the cluster resource whenever a node joins the cluster. Oracle does not use DHCP for the GNS VIP for obvious reasons—it must be known to the cluster prior to the Oracle Grid Infrastructure being installed.

The Networking Stack Components

The Oracle Grid Infrastructure requires fast communication over the public and private networks between the cluster nodes; hence, all the network components enabling this communication between the cluster nodes are very important, and it is not possible to build the Oracle Real Application Cluster database without these network components.

Network Interface Cards

Oracle requires each cluster node to have at least two network interface cards—one for the public network and one for the private network. Four are preferred when bonded and for HA. Otherwise, single interfaces are a point of failure unless one creates a VLAN on the two interfaces bonded and exposes it as a separate interface name associated with each network. The interface cards must be the same on all cluster nodes in an Oracle cluster. The network interface card for the public network must support TCP/IP and must have a valid IP address and an associated hostname registered in the domain name server.

https://t.me/learningnets

Technet24

You can complete the Oracle Grid Infrastructure installation by just registering the hostname in the /etc/hosts file; however, it is *strongly* recommended for an associated hostname to be registered in the domain name server. The network interface card for the private network must support UDP (User Datagram Protocol). Performance of the Oracle Real Application Cluster depends on the speed of the private network; hence, a high-speed network interface card such as 10 Gigabit Ethernet Network Card is recommended.

Network IP Addresses

The public network interface on each cluster node must have a valid public IP address to identify the cluster node on the public network, and the public IP address must be associated with a hostname registered in the Domain Name Service (DNS). Sometimes more public IP addresses are required, depending on the redundancy technology used to team or bond network interface cards. For example, IPMP (IP network multipathing) on the Solaris operating system would require three public IP addresses.

Each cluster node must have assigned an unused virtual IP address associated with a virtual hostname registered in DNS. Oracle will configure this virtual IP address during installation of the Oracle Grid Infrastructure. This virtual IP address must have the same subnet as the public IP address configured for the public network interface card.

Each cluster node must have a private IP address for the private interface card. It is recommended that the private IP address be from a nonroutable private network. This is because we want to avoid reaching outside for "any" routing that may result in some inadvertent routing. Oracle does not require a private hostname and IP address to be registered in DNS because this private IP address must be known to cluster nodes only, so a private hostname can be registered in the /etc/hosts file of each cluster node. An Oracle cluster requires three IP addresses for SCAN that have the same subnet as the public IP address.

Cluster Interconnects

The cluster interconnect is another important component in Oracle RAC. It is a communication path used by the cluster for the synchronization of resources and is also used in some cases for the transfer of data from one instance to another. Typically, the interconnect is a network connection that is dedicated to the server nodes of a cluster (and thus is sometimes referred to as a *private* interconnect) and has a high bandwidth and low latency. Different hardware platforms and different clustering software have different protocol implementations for the high-speed interconnect.

Oracle RAC is all about scalability and high availability, so every effort must be made to configure redundant hardware components to avoid any single point of failure. Therefore, it is highly recommended that you configure redundant network interface cards for private and public networks. Different technologies are available to bond network interface cards together, such as teaming on Linux and IPMP on the Solaris operating system. Active/Passive teaming of the network interface cards works fine,

https://t.me/learningnets

but users can configure the redundant network interface cards in Active/Active configuration, where each network interface card transmits and receives network packets.

Table 3-3 lists the various interconnects used by the implementations based on the clusterware used and the network hardware. Using vendor-specific interconnects is often discouraged by Oracle because they make troubleshooting more complex compared with the standard open-system protocols.

Operating System	Clusterware	Network Hardware	RAC Protocol
HP OpenVMS	HP OpenVMS	Memory Channel	TCP
HP OpenVMS	HP OpenVMS	Gigabit Ethernet	TCP
HP Tru64	HP TruCluster	Memory Channel	RDG
HP Tru64	HP TruCluster	Memory Channel	UDP
HP Tru64	HP TruCluster	Gigabit Ethernet	RDG
HP Tru64	HP TruCluster	Gigabit Ethernet	UDP
HP-UX	Oracle Clusterware	Hyper fabric	UDP
HP-UX	Oracle Clusterware	Gigabit Ethernet	UDP
HP-UX	HP Serviceguard	Hyper fabric	UDP
HP-UX	HP Serviceguard	Gigabit Ethernet	UDP
HP-UX	Veritas Cluster Server	Gigabit Ethernet	LLT
HP-UX	Veritas Cluster Server	Gigabit Ethernet	UDP
IBM AIX	Oracle Clusterware	Gigabit Ethernet (FDDI)	UDP
IBM AIX	HACMP	Gigabit Ethernet (FDDI)	UDP
Linux	Oracle Clusterware	Gigabit Ethernet	UDP
Microsoft Windows	Oracle Clusterware	Gigabit Ethernet	TCP
Sun Solaris	Oracle Clusterware	Gigabit Ethernet	UDP
Sun Solaris	Fujitsu Primecluster	Gigabit Ethernet	ICF
Sun Solaris	Sun Cluster	SCI Interconnect	RSM
Sun Solaris	Sun Cluster	Firelink interconnect	RSM
Sun Solaris	Sun Cluster	Gigabit Ethernet	UDP
Sun Solaris	Veritas Cluster Server	Gigabit Ethernet	LLT
Sun Solaris	Veritas Cluster Server	Gigabit Ethernet	UDP

https://t.me/learningnets

Technet24

TABLE 3-3. *Interconnects Based on Clusterware and Hardware*

In addition to the interconnects described in this table, newer interconnects such as Reliable Datagram Sockets (RDG) over InfiniBand can provide a horizontally scalable, high-speed, low-latency alternative to traditional vertical scaling.

Network Bonding

At the network level, a failure in a NIC can cause an outage in the cluster, especially if the failure occurs at the interface on which the interconnect is configured. To achieve high availability at this layer, network bonding is recommended. Bonding allows a node to see multiple physical NICs as a single logical unit.

The Linux kernel includes a bonding module that can be used to achieve software-level NIC teaming. The kernel-bonding module can be used to team multiple physical interfaces to a single logical interface, which is used to achieve fault tolerance and load balancing. The bonding driver is available as part of Linux kernel version 2.4 or later. Because the bonding module is delivered as part of the Linux kernel, it can be configured independently from the interface driver vendor (different interfaces can constitute a single logical interface).

Various hardware vendors provide different types of NIC bonding solutions for the network interconnect resiliency. Typically, bonding offers the following benefits:

- **Bandwidth scalability** Adding a network card doubles the network bandwidth. It can be used to improve aggregate throughput.
- **High availability** Provides redundancy or link aggregation of computer ports. Failure of a NIC does not induce a cluster outage because the traffic is routed through the other network.
- **Load balancing** Port aggregation supports true load-balancing and failure recovery capabilities as well as distributes traffic evenly across the aggregated links.
- **Single MAC address** Because port-aggregated networks share a single, logical MAC address, there is no need to assign individual addresses to aggregated ports.
- **Flexibility** Ports can be aggregated to achieve higher performance whenever network congestion occurs.

Interconnect Switch

A private network between cluster nodes must be configured using a high-bandwidth network switch (gigabit or higher) that supports TCP/IP. The Oracle requirement is to have a high-bandwidth, nonroutable, private network between the cluster nodes. Generally, big organizations share the network switches and are mostly reluctant to have a dedicated network switch for the Oracle Real Application Cluster databases. Users can use the shared network switches for the private interconnect and can follow the given guidelines to choose the right private interconnect:

https://t.me/learningnets

- Oracle prefers using a dedicated high-bandwidth network switch.
- If you're using the shared network switch, then use a dedicated, untagged, private, nonroutable VLAN. VLANs should not span the switch blades. Using a VLAN may have lots of advantages, but it is not superior to a dedicated switch. An overloaded backplane or processor on the physical switch will adversely impact the performance of the Cache Fusion traffic over the private network; this is why Oracle prefers using the dedicated physical switch for private interconnects.
- Always configure the network interface card on the fastest PCI bus available on the cluster nodes.
- Make sure that network interface cards are switched to "autonegotiate" and are configured for the maximum supported bandwidth.
- Jumbo frames can be configured for private network interface cards provided the private interconnect switch supports the jumbo frames.
- Consider configuring redundant network switches for the interconnect because failure in a network switch can bring the whole Oracle RAC database down.

The basic requirement of an interconnect is to provide reliable communication between nodes, but this cannot be achieved by a crossover cable between the nodes. However, using a crossover cable as the interconnect may be appropriate for development or demonstration purposes. Substituting a normal crossover cable is not officially supported in production Oracle RAC implementations for the following reasons:

- Crossover cables do not provide complete electrical insulation between nodes. Failure of one node because of a short circuit or because of electrical interference will bring down the surviving node.
- Using crossover cables instead of a high-speed switch greatly limits the scalability of the clusters because only two nodes can be clustered using a crossover cable.
- Failure of one node brings down the entire cluster because the cluster manager cannot exactly detect the failed/surviving node. Had there been a switch during split-brain resolution, the surviving node could easily deduct the heartbeat and take ownership of the quorum device, and node failures could be easily detected.
- Crossover cables do not detect split-brain situations as effectively as a communication interface through switches. Split-brain resolution is the effective part in cluster management during communication failures.

Node Time Synchronization

Time on each cluster node must be synchronized because different time stamps on the cluster nodes can lead to false node eviction in the cluster. All operating systems

https://t.me/learningnets

Technet24

provide a Network Time Protocol feature that must be configured and used before the Oracle Grid Infrastructure is installed.

Split-Brain Resolution

In the Oracle RAC environment, server nodes communicate with each other using high-speed private interconnects. A high-speed interconnect is a redundant network that is exclusively used for inter-instance communication and some data block traffic. A *split-brain* situation occurs when all the links of the private interconnect fail to respond to each other, but the instances are still up and running. Therefore, each instance thinks that the other instances are dead, and that it should take over the ownership.

In a split-brain situation, instances independently access the data and modify the same blocks, and the database will end up with changed data blocks overwritten, which could lead to data corruption. To avoid this, various algorithms have been implemented.

In the Oracle RAC environment, the Instance Membership Recovery (IMR) service is one of the efficient algorithms used to detect and resolve the split-brain syndrome. When one instance fails to communicate with the other instance, or when one instance becomes inactive for some reason and is unable to issue the control file heartbeat, the split-brain situation is detected and the detecting instance will evict the failed instance from the database. This process is called *node eviction*. Detailed information is written in alert log and trace files. (Refer to Chapter 14 for more information.)

Oracle Kernel Components

The Oracle kernel components in the Oracle RAC environment are the set of additional background processes in each instance of an Oracle RAC database. The buffer cache and shared pool become global in the Oracle RAC environment, and managing the resources without conflicts and corruptions requires special handling. The additional background processes in the Oracle RAC environment, along with those normal background processes that usually exist in single instances, manage the global resources effectively.

Global Cache and Global Enqueue Services

In Oracle RAC, because more than one instance is accessing the resource, the instances require better coordination at the resource management level. Otherwise, data corruption may occur. Each instance will have its own set of buffers but will be able to request and receive data blocks currently held in another instance’s cache. Buffer manipulation in the Oracle RAC environment is quite different from a single-instance environment because at any time only one set of processes may be accessing

https://t.me/learningnets

the buffer. In Oracle RAC, the buffer cache of one node may contain data that is requested by another node. The management of data sharing and exchange in this environment is done by the Global Cache Services.

Each instance has its own buffer cache in the System Global Area (SGA). Oracle uses Cache Fusion to logically combine the buffer cache of multiple instances in a RAC cluster, so the instances can process the data as if it was stored in a single buffer cache.

Oracle RAC uses two key processes to ensure that each instance in a cluster gets the block it needs from the buffer cache: the Global Cache Services (GCS) and the Global Enqueue Services (GES). The GCS and GES together form and manage the Global Resource Directory (GRD) to maintain the status of all datafiles and all cached blocks. The contents of the GRD are distributed across all the active instances of a cluster. The following section explains the Global Resource Directory and its components in more detail.

Global Resource Directory

All the resources in the cluster group form a central repository of resources called the Global Resource Directory (GRD). Each instance masters some set of resources and together all instances form the GRD. The resources in the cluster group are equally distributed among the nodes based on their weight.

The GRD is part of the shared pool, and an undersized shared pool in an Oracle RAC environment can severely impact performance because the GRD competes with the Library cache and the Data Dictionary, as well as the other components of the shared pool.

When one instance departs the cluster, the GRD portion of that instance needs to be redistributed to the surviving nodes. When an instance enters or leaves a cluster, the components of the GRD need to be redistributed among all instances. The GRD is frozen during the redistribution process, in order to enable the atomic distribution of the GRD among the instances in the cluster. Especially in large RAC installations with large SGAs and large active datasets, this redistribution process will cause a brownout. When a new instance enters the cluster, the GRD portions of the existing instances must be redistributed to create the GRD portion of the new instance. The components of the GRD and management issues are discussed in Chapter 11.

Oracle RAC Background Processes

Oracle RAC databases have two or more instances, each with its own memory structures and background processes. In addition to the usual single-instance background processes, Oracle RAC employs other processes to manage the shared resources. Thus, the Oracle RAC database has the same structure as that of a single-instance Oracle database, plus additional processes and memory structures that are specific to Oracle RAC. These additional processes maintain cache coherency across the nodes.

https://t.me/learningnets

Technet24

Maintaining cache coherency is an important part of an Oracle RAC. *Cache coherency* is the technique of keeping multiple copies of a buffer consistent among different Oracle instances on different nodes. Global cache management ensures that access to a master copy of a data block in one buffer cache is coordinated with the copy of the block in another buffer cache. This ensures the most recent copy of a block in a buffer cache contains all changes that are made to that block by any instance in the system, regardless of whether those changes have been committed on the transaction level.

The Importance of Coordination

It's important to understand why inter-instance cache coordination is necessary in an Oracle RAC environment. Consider a two-instance environment without any cache coordination and communication among the instances, as shown in Figure 3-3:

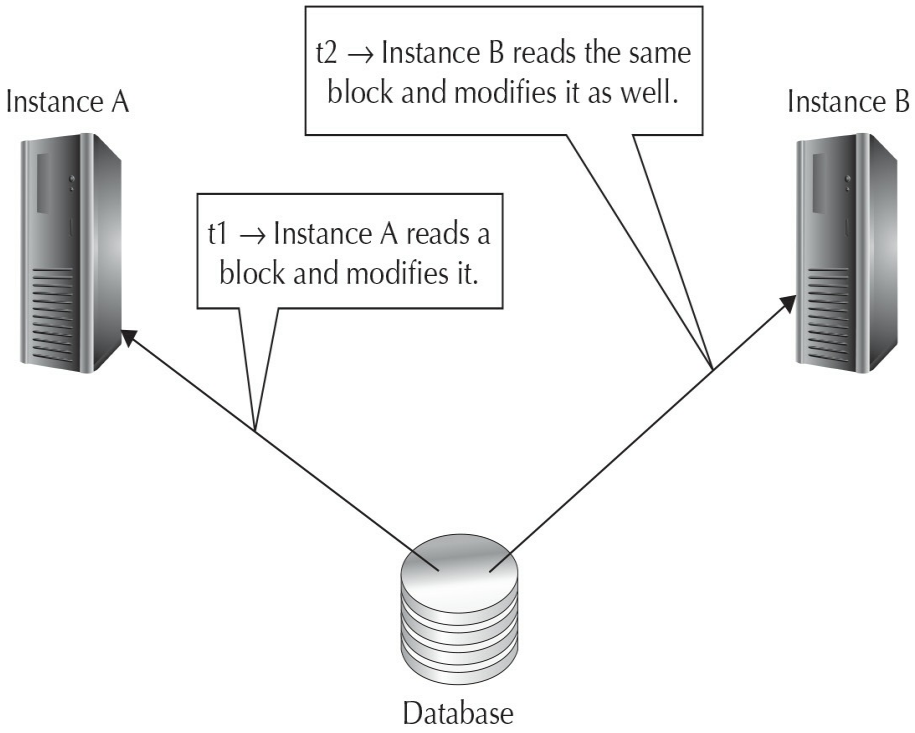


FIGURE 3-3. *Instances read a block without any coordination.*

1. Referring to Figure 3-3, consider at time t1, instance A reads a block in its buffer cache and modifies row 1 in it. The modified block is still in its buffer cache and has not yet been written to disk.

2. Sometime later at time t2, instance B reads the same block in its buffer cache and modifies another row in that block. Instance B also has not written the block to disk, so the disk still contains the old version of the block.
3. Now at time t3, instance A writes the block to disk. At this stage, modifications

https://t.me/learningnets

from instance A are written to disk (see Figure 3-4).

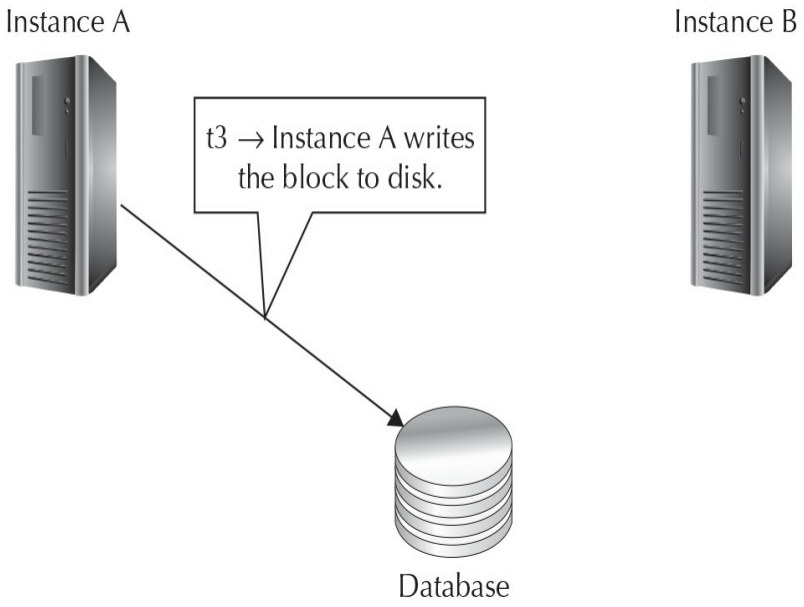


FIGURE 3-4. *Instance A writes the block to disk without coordination.*

4. Later at time t4, instance B writes the block to disk. It overwrites the block written by instance A in step 3. As you can easily infer, the changes made to the block by instance A are lost (see Figure 3-5).

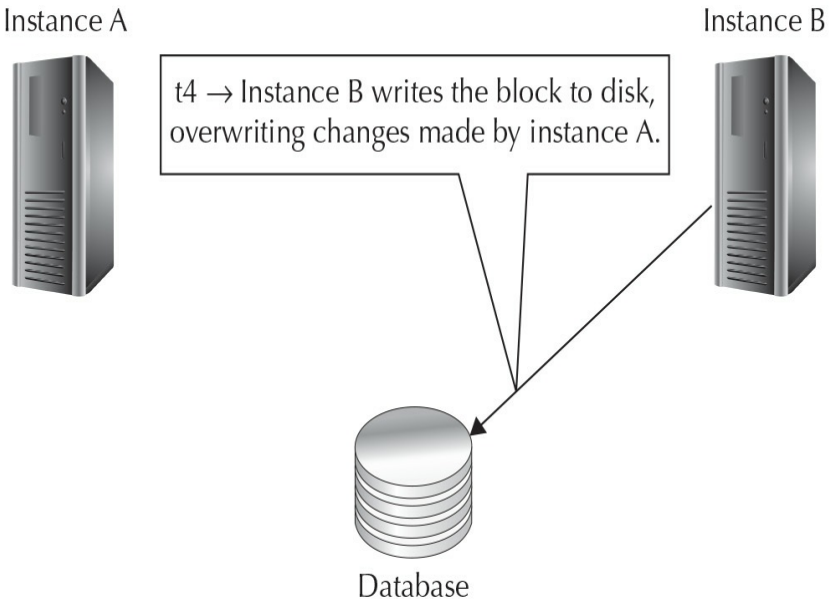


FIGURE 3-5. *Instance B overwrites the changes made by instance A.*

This scenario and many other similar situations require that when data is

https://t.me/learningnets
Technet24

simultaneously accessed by multiple machines, the read and (especially) write activities must be coordinated among these machines; otherwise, data integrity problems will result that may manifest as data corruption.

Now let's repeat the preceding operation sequence in the presence of coordination:

1. At time t1, when instance A needs a data block with an intent to modify, it reads the block from disk. However, before reading, it must inform the GCS (DLM) of its intention to do so. GCS keeps track of the lock status of the block being modified by instance A by keeping an exclusive lock against the block on behalf of instance A.

2. At time t2, instance B wants to modify the same block. Before doing so, it must inform the GCS of its intention to modify the block. When GCS receives the request from instance B, it asks the current lock holder instance A to release the lock. Thus, GCS ensures that instance B gets the latest version of the block and also passes on the write privilege to it (exclusive lock).
3. At time t3, instance B gets the latest (current) version of the block that has the changes made by instance A and modifies it.
4. At any point in time, only one instance has the current copy of the block. Only that instance can write the block to disk, thereby ensuring that all the changes to the block are preserved and written to disk when needed.

The GCS thus maintains data coherency and coordination by keeping track of the lock status of each block that is read and/or modified by the server nodes in the cluster. GCS guarantees that only one copy of the block in memory can be modified and that all the modifications are written to disk at the appropriate time. It maintains the cache coherence among the nodes and guarantees the integrity of the data. GCS is an in-memory database that contains information about the current locks on blocks and also keeps track of instances that are waiting to acquire locks on blocks. This is known as *Parallel Cache Management (PCM)* and has been a central feature of Oracle clustered databases since the introduction of Oracle Parallel Server (OPS) in the early 1990s.

PCM uses distributed locks on the resources to coordinate access to resources by different instances of an Oracle RAC environment. The GCS helps to coordinate and communicate the lock requests from Oracle processes between instances in the Oracle RAC environment.

Each instance has a buffer cache in its SGA. To ensure that each Oracle RAC database instance obtains the block that it needs to satisfy a query or transaction, Oracle RAC instances use two processes: the GCS and the GES. The GCS and GES maintain records of the lock status of each data file and each cached block using a GRD. The GRD contents are distributed across all of the active instances as part of the shared pool on each instance. Hence, it is good to increase the SGA size by some factor, but not more than 5 percent of the total SGA size. Tests have found that the larger your block size, the lower the memory overhead for the extra GCS, GES, and GRD components in the SGA. For large SGAs that exceed 20GB, it has been noted

https://t.me/learningnets

that the overhead is dependent on the block size used and could be around 600MB to 700MB for a 16KB-block-sized database.

The *cost* (or overhead) of cache coherency is defined as the need to check with other instances if a particular access is permitted before granting any access to a specific shared resource. Algorithms optimize the need to coordinate on each and every access, but some overhead is incurred. *Cache coherency* means that the contents of the caches in different nodes are in a well-defined state with respect to each other. Cache coherency identifies the most up-to-date copy of a resource, also called the "master copy." In case of node failure, no vital information is lost (such as committed transaction state) and atomicity is maintained. This requires additional logging or copying of data but is not part of the locking system.

A *resource* is an identifiable entity—that is, it has a name or reference. The entity referred to is usually a memory region, a disk file, or an abstract entity. A resource can be owned or locked in various states, such as exclusive or shared. By definition, any shared resource is lockable. If it is not shared, no access conflict will occur. If it is shared, access conflicts must be resolved, typically with a lock. Although the terms *lock* and *resource* refer to entirely separate objects, the terms are sometimes (unfortunately) used interchangeably.

A *global resource* is visible and used throughout the cluster. A *local resource* is used by only one instance. It may still have locks to control access by the multiple processes of the instance, but no access to it occurs from outside the instance. Data buffer cache blocks are the most obvious and most heavily used global resource. Other data item resources are also global in the cluster, such as transaction enqueues and database data structures.

The non-data-block resources are handled by Global Enqueue Services (GES), also called Non-Parallel Cache Management (Non-PCM). The Global Resource Manager (GRM), also called the Distributed Lock Manager (DLM), keeps the lock information valid and correct across the cluster.

All caches in the SGA are either global (and must therefore be coherent across all instances) or local. The library, row cache (also called dictionary cache), and buffer caches are global. The large and Java pool buffers are local. For Oracle RAC, the GRD is global in itself and also used to control the coherency.

After one instance caches data, in some cases other instances within the same cluster database can acquire a block image from another instance in the same database faster than by reading the block from disk. Therefore, Cache Fusion moves current copies of blocks between instances rather than rereading the blocks from disk under certain conditions. When a consistent block is needed or a changed block is required on another instance, Cache Fusion can transfer the block image between the affected instances. RAC uses the private interconnect for inter-instance communication and block transfers. GCS manages the block transfers between the instances.

The GRD manages the locking or ownership of all resources that are not limited to a single instance in Oracle RAC. The GRD is composed of the GCS, which handles the data blocks, and the GES, which handles the enqueues and other global resources.

https://t.me/learningnets

Technet24

NOTE
Cache Fusion doesn't require you to tune any parameters. Oracle Database dynamically allocates all Cache Fusion resources. The dynamic mastering of resources by keeping them local to the data blocks means a high level of performance.

Each process has a set of roles, and we will study them in detail in the following sections. In Oracle RAC, the library cache and shared pool are globally coordinated. All the resources are managed by locks, and a key background process also manages the locks. GCS and GES use the following processes to manage the resources. These Oracle RAC–specific background processes and the GRD collaborate to enable Cache Fusion:

- **LMS** Global Cache Services process
- **LMON** Global Enqueue Services Monitor
- **LMD** Global Enqueue Services daemon
- **LCK0** Instance Enqueue process
- **ACMS** Atomic Controlfile to Memory Service (ACMS)
- **RMSn** Oracle RAC Management Processes (RMSn)
- **RSMN** Remote Slave Monitor

The LMON and LMD processes communicate with their partner processes on the remote nodes. Other processes may have message exchanges with peer processes on the other nodes (for example, PQ). The LMS process, for example, may directly receive lock requests from remote foreground processes.

The following sections explain the key functions of the various Oracle RAC–specific background processes.

LMS: Global Cache Services Process

LMS is a process used in Cache Fusion. LMS maintains records of the status of the datafiles and the cached block, and it stores that information in the GSD. The acronym is derived from the Lock Manager Server process. It enables consistent copies of blocks to be transferred from a holding instance's buffer cache to a requesting instance's buffer cache without a disk write under certain conditions. It also retrieves requests from the server queue that are queued by LMD to perform requested lock operations.

LMS also rolls back any uncommitted transactions for any blocks that are being requested for a consistent read by the remote instance. LMS processes also control the flow of messages between instances. Each instance can have up to 10 LMS processes, though the actual number of LMS processes varies according to the amount of messaging traffic between nodes. You can control the number of LMNON processes

https://t.me/learningnets

by setting the GCS_SERVER_PROCESSES initialization parameter. If this parameter is not set manually, the number of LMS processes automatically started during instance startup is a function of the CPU_COUNT of that node and is usually adequate for most types of applications. It is only under special circumstances that you may need to tweak this parameter to increase the default number of LMS processes.

LMS processes can also be started dynamically by the system based on demand, and this is controlled by the parameter **_lm_dynamic_lms**. By default, this parameter is set to FALSE. In addition, LMS processes manage Lock Manager Server requests for GCS resources and send them to a service queue to be handled by the LMS process. It also handles global lock deadlock detection and monitors for lock conversion timeouts.

LMON: Global Enqueue Services Monitor

LMON is the Lock Monitor process and is responsible for managing the Global Enqueue Services (GES). It maintains the consistency of GCS memory in case of process death. LMON is also responsible for cluster and lock reconfiguration when an instance joins or leaves the cluster. It also checks the instance death and listens for local messages. The LMON process also generates a detailed trace file that tracks instance reconfigurations.

The background LMON process monitors the entire cluster to manage global resources. LMON manages instance deaths and the associated recovery for any failed instance. In particular, LMON handles the part of recovery associated with global resources. LMON-provided services are also known as Cluster Group Services (CGS).

LMD: Global Enqueue Services Daemon

LMD is the daemon process that manages Enqueue Manager Services requests for the GCS. The acronym *LMD* refers literally to the Lock Manager Daemon, the term used for the process in OPS. The resource agent process manages requests for resources to control access to blocks. The LMD process also handles deadlock detection and remote resource requests. Remote resource requests originate from another instance.

LCK0: Instance Enqueue Process

The Lock (LCK) process manages non–Cache Fusion instance resource requests and cross-instance call operations for shared resources. It also builds a list of invalid lock elements and validates lock elements during recovery. An instance can use only a single LCK process because primary functionality is handled by the LMS process.

RMSn

The RMSn (Oracle RAC Management) processes are responsible for performing the Oracle RAC manageability functions, such as creating the necessary RAC resources (for example, when you add a new instance to a cluster).

https://t.me/learningnets

Technet24

RSMN

The RSMN (Remote Slave Monitor) process monitors and manages the background slave process creation and communications on remote instances.

Summary

This chapter introduced the various building blocks of Oracle RAC. Oracle Grid Infrastructure provides the foundation for Oracle RAC scalability and availability. Because storage is shared across all the nodes, it is very important to establish redundancy at the disk level, and ASM helps provide a very scalable and highly available storage subsystem.

Next we discussed storage and how the network components play a vital role in building the cluster. We looked into the different options available for configuring the interconnects, which are key to the systems in supporting scalability and availability. Bonded network interfaces at the operating system level provide the redundancy and required scalability for cluster operations. Redundant network switches should also be part of high availability architecture.

It’s important to understand the importance of resource coordination and the kernel components involved in the cluster-wide operations. Global Cache Services and Global Enqueue Services ensure the resources are properly queued, and they synchronize the database operations. With this background in mind, we’ll begin preparing the hardware to install Oracle RAC in the next chapter.

https://t.me/learningnets

PART II

Installation, Configuration, and Storage

https://t.me/learningnets

Technet24

CHAPTER 4

Oracle Grid Infrastructure Installation

This chapter focuses on installing the Oracle Grid Infrastructure. In this chapter, you'll learn the essentials of satisfying the requirements for the hardware, storage, operating system, and networking layer components required for a successful Oracle Grid Infrastructure installation.

The term *Grid Infrastructure* primarily includes two components—Oracle Clusterware and Automatic Storage Management (ASM)—both of which are installed into a single directory called *Grid home*. Note that this was known as Oracle Clusterware in Oracle 11*g* Release 1 and earlier versions.

Grid Infrastructure is the foundation for Oracle Real Application Clusters (RAC), and you must install it before you can install Oracle RAC or RDBMS software. Oracle Grid Infrastructure is a building block for RAC databases. It consists of additional processes and/or agents that allow participating cluster nodes to communicate and to provide a unified image to clients. The primary function of clusterware is to provide high availability and scalability, manage node membership changes, offer a single or unified view of shared resources, and protect and maintain the integrity of the cluster.

In Oracle 12*c*, the term *Grid Infrastructure* encompasses the following features or components:

https://t.me/learningnets

- Oracle Clusterware (Flex Clusters)
- Oracle Automatic Storage Management (Flex ASM)
- Oracle ASM Cluster File System (CloudFS)
- Oracle Database Quality of Service (QoS) Management
- Cluster Health Monitor (CHM)
- Grid Naming Service (GNS)

The requirements you need to fulfill in preparing for the Grid Infrastructure installation depend on the optional components you plan on using. For example, if you want to install and configure Flex Clusters, the mandatory requirement is the configuration of GNS on one of the core nodes using a fixed virtual IP. As a result, you would be using Flex ASM.

The following features need additional consideration, planning, or investigation before you install the Grid Infrastructure:

- Flex Clusters / Flex ASM
- GNS
- Role separation
- Shared GI_Home or local GI_Home
- ASM or Shared File System for storing the voting disk, OCR, and datafiles
- Use of Grid Infrastructure Management Repository Database (MGMTDB), which is new in Oracle Database 12*c*
- Automated execution of the root.sh script on all nodes (which requires you to have the same password on *all* the nodes in the cluster)

Figure 4-1 shows the installation process for Oracle 12*c* RAC.

https://t.me/learningnets

Technet24

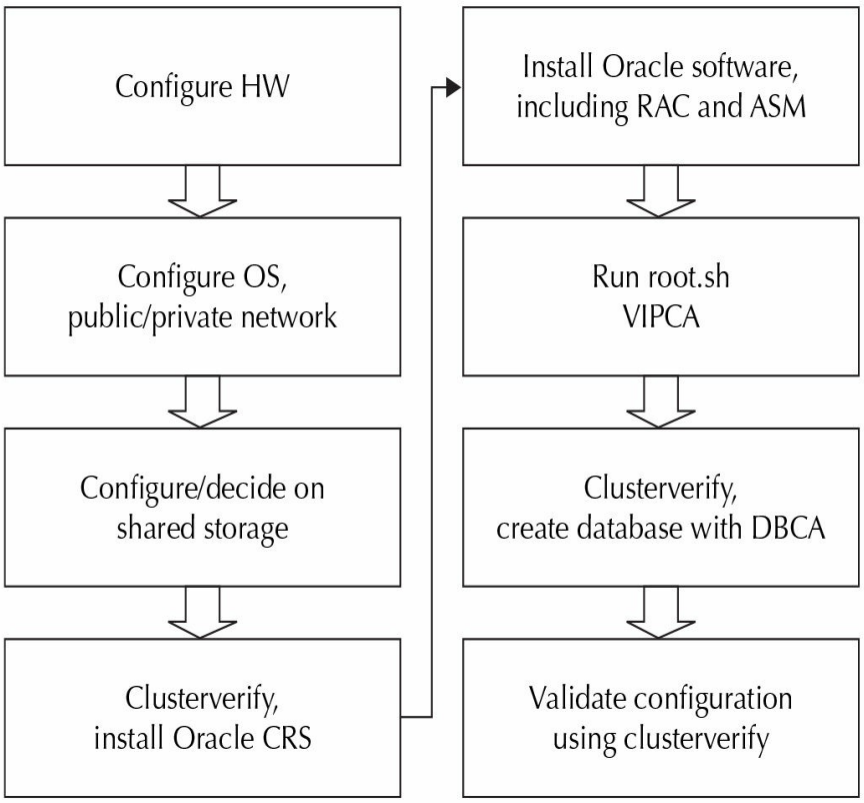


FIGURE 4-1. *Oracle RAC installation flowchart*

In Oracle Grid Infrastructure 12*c* Release 2 (12.2), you have the following new capabilities:

- You can deploy Oracle Domain Services clusters and Oracle Member clusters.
- You can configure the RAC cluster nodes in multiple locations called *sites*, in an architecture called an *extended cluster*. Oracle Grid Infrastructure deployment supports an off-the-cluster global repository named the Grid Infrastructure Management Repository (GIMR). The global GIMR runs in a Domain Services cluster and is based on a multitenant database with a pluggable database (PDB) for storing the GIMR of each cluster in the domain cluster.
- You can get proactive diagnostic support, including early warnings about potential performance bottlenecks as well as the root causes and corrective actions for various issues, through the Cluster Health Advisor.
- The Oracle Flex Cluster architecture uses hub nodes and leaf nodes. Oracle RAC reader nodes facilitate this architecture by allocating a set of read/write instances that run OLTP (Online Transaction Processing) workloads on the hub nodes and a set of read-only database instances across the leaf nodes. Updates to the read-write instances running on the hub nodes are propagated quickly to the read-only instances running on the leaf nodes, for querying and reporting functions.

https://t.me/learningnets

An Overview of the Grid Infrastructure Installation Process

The first step in the Grid Infrastructure installation process is the configuration of the operating system and hardware for the Oracle Grid Infrastructure and RDBMS software.

Each server in the cluster will have a minimum of one public network interface, which should support TCP/IP and one private network interface and must support UDP (User Datagram Protocol). Here is how the public and the network interfaces work:

- The *public network interface* is the standard network connection that links the server to all the other computers in your network.
- The *private network interface* is normally a nonroutable private network connection shared by only the servers within the cluster. Some servers have additional network interfaces for the management framework, and they are not critical or required for the installation or functioning of the Oracle Grid Infrastructure. The Oracle Grid Infrastructure and Oracle RAC software use the private network interface to communicate with the other servers in the cluster and for shipping blocks between the instances required for Cache Fusion.

Note that the use of crossover cables for interconnects is not supported. If you are planning on using multiple interfaces for a private network, each private network interface must be on a different subnet. Up to four interfaces are allowed for Oracle RAC HAIP (Highly Available IP). You can enable HAIP functionality using OUI during the installation process. Grid Infrastructure will utilize all the HAIP addresses/interfaces for the private network for load-balancing traffic.

Oracle 12*c* supports the use of IPv6-based addresses for the public network, and database clients can connect to either IPv4 or IPv6 addresses. IPv6 is not supported in Windows as of this writing. You should continue to use IPv4-based addressing for private network.

Configuring the *shared storage* for datafiles is an important step in hardware preparation. As a fundamental requirement for high availability, all the database files in the cluster will be housed in shared storage that is separate from the server nodes in the cluster. The shared storage allows multiple database instances, running on different servers, to access the same database information and ensures that if one or more server nodes fail, all the remaining nodes will continue to have access to all database files.

After you define the shared storage for the datafiles, the next step is installing the Oracle Grid Infrastructure, which logically binds multiple servers into a cluster.

During the Oracle Grid Infrastructure install, you specify the location at which to create two Oracle Grid Infrastructure components: a voting disk to record node membership information and the Oracle Cluster Registry (OCR) to record cluster configuration information.

https://t.me/learningnets

Technet24

The voting disk and OCR can be stored inside the Oracle ASM. However, a local copy of the OCR, known as Oracle Local Registry (OLR), is stored along with the Oracle Grid Infrastructure binaries. However, you're allowed to store the password files in the Oracle ASM disk group, and this is the default location if you use the Database Configuration Assistant (DBCA) to create a database.

You perform the Oracle Grid Infrastructure install on one server, and the Grid software will be automatically installed on all the other servers in the cluster during the remote operation phases of the Oracle Universal Installer.

At the end of the Oracle Grid Infrastructure install, Oracle configures the virtual IP (VIP) addresses, ASM, the local listener, and the SCAN listeners for the servers in the cluster. Clients use a Single Client Access Name (SCAN) to connect to the database service in the cluster. The SCAN is a domain name that resolves to three IP addresses from the same subnet of the public network in round-robin fashion. Because the SCAN is not associated with any particular server in the cluster, there's no need to reconfigure the clients if a server is added or removed from the existing cluster.

Prior to Oracle 11*g* Release 2, clients connecting to the database using a virtual IP required reconfiguration if the connecting database service was moved onto another server in the cluster. Clients can still connect to the database service using a virtual IP, but this is not a preferred way to connect to a database service with Oracle 11*g* Release 2 RAC. This continues to be true with Oracle 12*c* as well.

During the installation process, you will be able to create an ASM disk group that can be used for storing voting and OCR files. After the Oracle Grid Infrastructure is installed, you'll be able to create additional ASM disk groups using the ASM Configuration Assistant (asmca) to store the Oracle datafiles. Following this, you install the Oracle Database 12*c* Release 2 RAC or RDBMS software and then create the cluster database. The installer will recognize that the Oracle Grid Infrastructure has already been installed.

As with the Oracle Grid Infrastructure install, the Oracle RAC database install is performed on one server, and the installer will automatically place the database software on the other participating servers in the cluster.

After the database software is installed, you will create a cluster database using the Database Configuration Assistant (DBCA). The last step is the installation of the Enterprise Manager Agent. The Enterprise Manager Agent connects to the Enterprise Manager Grid Control, where you administer the Oracle RAC environment. You install the agent on just one node in the cluster, and from there it's automatically installed on the other nodes.

We use the 64-bit Oracle Linux operating system and Oracle Database 12*c* Release 2 (12.2.0.1) for the installation. Also, we assume that the cluster servers have the minimum required hardware. Port-specific documentation and installation guides should be consulted during installation.

This chapter is *not* intended to replace the Oracle installation guide, and we strongly recommend that you consult the support notes from Oracle Support. You should use the installation procedures we show in this chapter in conjunction with the Oracle Grid

https://t.me/learningnets

Control and RAC database installation manuals.

Before you dive into the Grid Infrastructure software installation, however, you'll need to perform several preinstallation tasks.

Preinstallation Tasks

Installing the Oracle Grid Infrastructure starts with some hardware preparation steps, such as configuring network names and addresses, updating relevant configuration files, and setting up specific operating system users, groups, and privileges. Installation of the Oracle Grid Infrastructure will be smooth sailing when the required prerequisites are satisfied ahead of time.

In the following sections, we review the essentials of how to configure the following key components before you can perform the installation of the Grid Infrastructure software:

- The Installer, CVU, and ORAchk
- The operating system
- The network
- The Network Time Protocol (NTP)
- Groups and users
- Secure Shell and user limits
- Shared storage
- Kernel parameters
- The Cluster Verification Utility (cluvfy)

Understanding the Installer, CVU, and ORAchk

You'll be using the Oracle Universal Installer to install the Oracle Grid Control (and the Oracle RAC database). In addition to the Installer, you need to be familiar with two other related tools: the Cluster Verification Utility (CVU) and the ORAchk utility. Here are some points to keep in mind:

- Internally, the Installer uses the CVU utility to perform many of the prerequisite checks before the installation begins. The Installer runs the prerequisite checks and creates the fix-up scripts as well. You can also run the CVU commands at the command line to check the readiness of your system before an installation.
- Fixup scripts are shell scripts that the Installer creates when a system doesn't meet the minimum requirements for an installation. The fix-up scripts can set kernel parameters, create and set permissions on the Oracle inventory directory, and create/reconfigure primary and secondary group membership for the Oracle

https://t.me/learningnets

Technet24

user, as well as set shell limits for the required values.

- Whereas the CVU utility helps check the readiness of your system, the ORAchk utility helps you check your system after the installation to ensure there aren't any post-installation issues. These checks include kernel requirements and the OS resource allocations. In addition, you can run the **orachk -u -c pre** command to check the system's health before an upgrade. The ORAchk Upgrade Readiness Assessment automates both the pre- and post-upgrade checks you'll need to perform.

Configuring the Operating System

In this book, we use the Oracle Linux OS, so our discussion here is pertinent to this operating system. You can choose from the minimal installation options to complete the Linux installation.

Because OpenSSH is a requirement for Oracle Grid Infrastructure installation, you can confirm that the SSH packages are installed by running the following command:

```
$ sudo rpm -qa | grep ssh
```

Once your Linux software is installed, you need to complete several preinstallation configuration tasks. Our recommendation is to use the Oracle-provided Preinstallation RPM, which completes most of the preinstallation configuration tasks for you, as explained in the following section.

Automatically Configuring Oracle Linux with Oracle Preinstallation RPM

Oracle recommends that you use the Oracle Preinstallation RPMs to simplify your OS configuration. You can use either Oracle Linux 6 or 7 as well as the Oracle Preinstallation RPM to automatically configure the Oracle Linux OS for both the Oracle Grid Infrastructure and the Oracle Database installations. Once you install the Oracle Linux software on all nodes of the cluster, you can either register your Linux distribution with Oracle Unbreakable Linux Network (ULN) or download the Oracle Preinstallation RPM using the Oracle public yum repository.

NOTE

Oracle recommends that you run the Oracle Preinstallation RPM if you're using Oracle Linux. This will configure your OS for both the Oracle Grid Infrastructure and the Oracle Database installations. It saves you a bunch of time and keeps your installation from stalling due to misconfiguration of OS configuration items.

Once you install the Oracle Preinstallation RPM, you can use it to update your

https://t.me/learningnets

Linux release by having it automatically do the following for you:

- Download and install any additional RPM packages necessary for installing Oracle Grid Infrastructure and Oracle Database.
- Create necessary OS users (oracle) and groups such as orainventory (oinstall) and OSDBA (DBA) for the user oracle.
- Create the operating system directories for the Oracle Grid and Database software installations.
- Set hard and soft resource limits.
- Set any other recommended OS parameters.

The simplest way to install the Oracle Preinstallation RPM and have it automatically perform all the Oracle Linux–related checks and required tasks is to use yum, as shown here:

```
$ yum install oracle-database-server-12cR2-preinstall -y
```

The **-y** flag tells yum to skip the package confirmation prompts.

After the OS Configuration

Following the Oracle Linux configuration with the Oracle RPM, you need to complete the network interface configuration for all the cluster nodes, as well as complete the system configuration for shared storage access for all the nodes. At this point, you'll be ready to install Oracle Grid Infrastructure and Oracle RAC.

Configuring the Network

The preinstallation tasks start with the configuration of the network. The following sections describe the key tasks you need to perform in configuring the network for an Oracle RAC database.

Enabling Multicast Communication

You must ensure that multicast communication is enabled for the private network. This requirement was introduced with Oracle 11*g* Release 2 and continues to be a requirement with Oracle 12*c* as well. This feature provides redundancy for the cluster interconnect (private network interfaces) without using any OS-specific teaming or bonding drivers. Oracle uses multicast communication on the private network to establish initial communication with the peer servers in the cluster and later on switches to unicast communication once the communication is established. Therefore, it is very critical that you enable multicast network communication for the private network for a successful install.

If you're upgrading from Oracle 11*g* Release 2, you have already met the requirement for multicast communication. Oracle has published My Oracle Support

https://t.me/learningnets

Technet24

Note 1212703.1, which provides scripts to test and verify that the servers in the cluster can communicate with each other using multicasting on the private network.

In addition, you can use the Cluster Verification Utility to perform validation checks for multicasting requirements. Note that the multicast communication should be enabled not only on the private network but also the network's switches used for private interconnect. Otherwise, the root.sh script will succeed on the first node but will fail on the rest of the nodes in the cluster.

The following shows how the Cluster Verification Utility (cluvfy) performs the multicasting network checks:

```
$ ./cluvfy stage -pre crsinst -n oltracn01,oltracn02 -verbose -r 12.2
Performing pre-checks for cluster services setup
..
…
Checking multicast communication...
Checking subnet "10.10.10.0" for multicast communication with multicast group
"224.0.0.251"...
Check of subnet "10.10.10.0" for multicast communication with multicast group
"224.0.0.251" passed.
Check of multicast communication passed.
```

NOTE

In computer networking, unicast routing is the communication protocol used for sending messages between a single sender (tx) and single receiver (rx) over a network. Multicast, on the other hand, is communication between a single sender and multiple receivers. Point-to-point communication is unicast communication, and point-to-multipoint is multicast communication.

Large installations and organizations generally use Internet Group Management Protocol (IGMP) snooping to monitor the multicast traffic. If IGMP is not properly configured, it may interfere with the server's communication over the private network and break the whole Oracle RAC system.

Required Network Interfaces

You can designate network interfaces for use as public, private, or Oracle ASM interfaces. Oracle Flex ASM can use its own dedicated private network or the same private networks as Oracle Clusterware. You can classify the networks as follows:

- PUBLIC
- PRIVATE
- ASM
- PRIVATE+ASM

https://t.me/learningnets

The public and private interfaces on all cluster servers must have the same name. For example, in a two-node Oracle RAC, if the public interface name on node 1 is eth0, then the public interface name on node 2 should also be eth0.

SCAN and the Grid Naming Service

Oracle database clients connect to the database using a Single Client Access Name (SCAN). SCAN and its IP addresses provide a stable name that clients use for connecting to a RAC database, regardless of the nodes that constitute the cluster. Because SCAN addresses resolve to a cluster rather than a specific node address, you can add nodes to and remove nodes from your cluster, leaving the SCAN address unchanged.

Oracle Grid Infrastructure 12*c* Release 2 allows you to assign IP addresses dynamically using the Oracle Grid Naming Service (GNS), which requires a Dynamic Host Configuration Protocol (DHCP) service running on the public network to provide one IP address for each cluster server's VIP and three IP addresses for the SCAN addresses.

When you add new nodes, for example, your organization's DHCP server dynamically assigns addresses for those nodes. The addresses are automatically registered in GNS, and GNS resolves all cluster node addresses registered with it.

The SCAN is a virtual IP name, but unlike a typical virtual IP address, SCAN isn't associated with just one single node/IP address—it's associated with all nodes in the cluster and their IP addresses.

In order to enable GNS, you must configure your network to provide a set of IP addresses to a subdomain assigned to your cluster, such as grid.example.com. You delegate Domain Name Service (DNS) requests for that subdomain of the GNS virtual IP address for that cluster. DHCP provides the set of IP addresses that the cluster needs within this subdomain.

In order to use GNS, you must configure your DNS so it delegates requests for cluster name resolution (any names in the subdomain delegated to the cluster) to GNS. To use GNS, you must specify a static IP address for the DNS VIP address. If you enable GNS, any naming requests made to the cluster are delegated to the GNS daemon, which listens on the GNS virtual IP address. GNS processes the naming requests and hands them the appropriate address.

SCAN resolves to several IP addresses, reflecting multiple SCAN listeners running across the cluster and listening on the SCAN IP addresses and SCAN ports. All services in the cluster are registered with the SCAN listener. When clients submit a request to a service running in the cluster, the SCAN listener replies with the address of the local listener on the node that's carrying the least load when compared to the other nodes. The client uses this local listener to connect to the service.

You can configure SCAN so it can be resolved either by GNS or the DNS. SCANs should resolve to at least one address, but Oracle recommends that you configure the SCAN name so it resolves to three IP addresses.

GNS uses the multicast Domain Name Service (mDNS), so the cluster can

https://t.me/learningnets

Technet24

dynamically assign the hostnames and IP addresses as you add and remove nodes to and from the cluster. When you use GNS, you don't need to perform any configuration in the DNS.

The Grid Naming Service virtual IP address (GNS VIP) is a static IP address that's configured in the DNS. If you're using GNS, you must configure a GNS name and fixed address in your Domain Name Service for the GNS VIP. In addition, you must configure a subdomain in your DNS dedicated to the GNS VIP for resolving the cluster address.

If you're using GNS, don't configure the SCAN names and addresses in your DNS. GNS will manage the SCAN names in this case. If you are using GNS, you can configure the leaf nodes on private/public networks during the installation. The leaf nodes on public networks don't use any Oracle Clusterware services. You can configure the network resources and listeners for the leaf nodes after the installation by executing SRVCTL commands.

Configuring the Node Names and the IP Addresses

For each hub node, if you aren't using GNS (which means you're using DNS instead), you must configure the following:

- A public node name and address in your DNS, and in the /etc/hosts file on each node. The public IP address is dynamically assigned (via DHCP), or you can define it in a DNS or a hosts file. The public node name is the same as the hostname for the node. For example, you can have the public node name rac1.myhost.com and the address 192.0.2.10.
- A private node address on the private interface for each node in the cluster. The private IP address is used for internode communication. Ensure that all the nodes in the cluster are connected by the private subnet used by the private interfaces.

TIP

Oracle strongly recommends that you use a physically separate private network for the private interconnect due to the potential of heavy traffic among the nodes, including the Cache Fusion traffic.

- A public node virtual IP name and address. The IP name and address must be in the format *hostname-vip*, as in node01-vip.example.com, with the address 192.0.2.11. Because you aren't using a dynamic network with GNS, you must specify a virtual hostname for each node in the cluster. The virtual node name, which is a public node name, reroutes client requests sent to a node when that node becomes unavailable. Because the VIP is used for client to database connections, the VIP address must be publicly accessible.

https://t.me/learningnets

If you're using manual configuration and DNS resolution, you must configure the SCAN so it resolves to three addresses on your DNS. You need to register three IP addresses in the DNS for the Single Client Access Name resolving to a domain name in round-robin fashion. Register one virtual IP for each cluster node in the DNS as well. The virtual IP must be unplumbed (*unplumbing* is the detaching of an IP address from an interface) because the Oracle Grid Infrastructure will configure the virtual IP during the installation process. In other words, make sure when you ping the virtual IP or the IPs associated with SCAN that they are inactive.

Configuring the /etc/hosts File

You also need to configure the /etc/hosts file, which contains the Internet Protocol (IP) hostnames and addresses for the local host and other hosts in the Internet network. This file is used to resolve a name to an address (that is, to translate a hostname into its Internet address). When your system is using DNS for address resolution, this file is accessed only if the name server cannot resolve the hostname.

NOTE

You must use the same private adapters for Oracle RAC and Oracle Grid Infrastructure (Clusterware). Ensure that the network interfaces are at least 1 GbE. Oracle recommends 10 GbE. You can alternatively use InfiniBand for the interconnect.

For simplicity, we manually assign the IP addresses during our installation. Following is a sample hosts file that shows how to configure the network addresses:

```
127.0.0.1       localhost.localdomain   localhost
#Public Host Names
10.1.1.161      alpha1.us.oracle.com    alpha1
10.1.1.171      alpha2.us.oracle.com    alpha2
10.1.1.181      alpha3.us.oracle.com    alpha3
10.1.1.191      alpha4.us.oracle.com    alpha4
```

You can execute the **ping** (which stands for Packet Internet Groper) command to verify the configuration. Except for the VIP interface, all other interfaces, including SCAN, should be up and running now.

Do *not* register the Single Client Access Name in the /etc/hosts configuration file because doing so will not allow SCAN to resolve with three different IP addresses and will configure the Grid Infrastructure to use only one IP address instead; therefore, it won't be able to use the SCAN functionality.

In the /etc/hosts file, you specify the IP address in either the dotted decimal or octal format, and you specify the hostname in either relative or absolute domain name

https://t.me/learningnets

Technet24

format. You can’t specify multiple hostnames (or aliases).

NOTE
You can configure SCANs in a DNS or a host file. Oracle strongly recommends that you not configure a SCAN VIP address in the hosts file and use DNS resolution instead.

After you configure the hosts file properly on one of the nodes, copy it to the other nodes so that they are exactly the same. The hosts file on each node needs to contain the private, public, and virtual IP (VIP) addresses and node names of each node in the cluster. If you’re using GNS, you could exclude entries for VIP because that will be managed by the GNS.

Testing That the SCAN Is Correctly Configured

You can check that the DNS is correctly associating the SCAN with the appropriate addresses by executing the **nslookup** command:

```
root@node1]$ nslookup mycluster-scan
Server:         dns.example.com
Address:        192.0.2.001

Name:   mycluster-scan.example.com
Address: 192.0.2.201
Name:   mycluster-scan.example.com
Address: 192.0.2.202
Name:   mycluster-scan.example.com
Address: 192.0.2.203
```

Once you install everything, when clients send a request to the cluster, SCAN listeners will redirect those requests to the cluster nodes.

Configuring NTP

You need to edit the /etc/sysconfig/ntpd configuration file of the Network Time Protocol (NTP) service on each cluster node to use the slewing option to synchronize the time on each cluster node, as shown next. It is very important that the system time is synchronized on all servers in the cluster to avoid false node evictions.

```
OPTIONS="-x -u ntp:ntp -p /var/run/ntpd.pid"
```

The Cluster Time Synchronization Service will start the CTSSd daemon in exclusive mode if NTP is not started on the cluster nodes. If the NTP service is

https://t.me/learningnets

configured and running on all nodes in the cluster, the CTSSd daemon will run in the observer mode.

Oracle 12*c* Grid Infrastructure also requires that you disable “avahi-daemon” on all the nodes. Note that the avahi-daemon is relevant only if you’re using the Linux operating system, as we are in this book.

Setting Up the Groups and Users

In a typical Oracle RAC installation, the operating system user *oracle* will own the RDBMS binaries and all Oracle-related files. Normally this user will be part of the DBA group. Any user who is part of the DBA group and has sysoper/sysdba privileges can connect to the oracle instance without supplying a password. This is called *operating system authentication*. This privileged user can also start up and shut down instances.

Oracle RAC allows using different operating system users to own different components of Oracle RAC so that people with different job responsibilities and accountabilities can manage their respective components of the whole Oracle RAC system. For example, the system admin team may use the operating system account *grid* to manage the Oracle Grid Infrastructure, whereas the database team may use the *oracle* user to manage the databases. The user *grid* owns only Oracle Grid Infrastructure software installation and is also called the Grid user. The user oracle owns all Oracle installations, or one or more Oracle database installations, and is also called the Oracle user.

NOTE

Although you can have just a single Oracle Grid Infrastructure owner, you can have a different Oracle user that owns different installations.

We’ll use different operating system accounts in the installation process explained in this chapter. Oracle products use an operating system group called *oinstall* that is associated with the central inventory. All operating system accounts used for Oracle product installation will have oinstall as a primary group so that the Oracle central inventory is shared by all Oracle product owners.

Oracle databases have two standard administration groups: OSDBA (typically, dba) and OSOPER (typically, oper), which is optional. In addition, you can have an extended set of database groups to grant task-specific system privileges for administrating the database. These task-specific system privileges groups consist of OSBACKUPDBA (typically backupdba), OSDGDBA (typically dgdba), OSKMDBA (typically kmdba), and OSRACDBA (typically, racdba).

There are two main Oracle Grid Infrastructure OS groups:

https://t.me/learningnets

Technet24

- **OSASM** This group is for administering Oracle ASM (typically asmadmin).
- **OSOPER** This group is for Oracle ASM (typically, asmoper).

In addition, you can create the ASNMSNMP user with privileges to monitor Oracle ASM instances.

You'll need to create the following operating system users and groups on each cluster node before installing the Oracle Grid Infrastructure and RDBMS software:

```
#/usr/sbin/groupadd -g 501 oinstall
#/usr/sbin/groupadd -g 502 osdba
#/usr/sbin/groupadd -g 504 asmadmin
#/usr/sbin/groupadd -g 506 asmdba
#/usr/sbin/groupadd -g 507 asmoper
```

NOTE

You must make sure that the OINSTALL operating system group is the primary group for all Oracle software installation owners.

Verify that the attributes (uid and gid) of the users and groups created here are identical on all cluster nodes. Internal to the operating system and cluster layer, the user IDs (UIDs) are compared to the UIDs of the member nodes while authorizations and permissions occur. Therefore, it's important to keep the user IDs and group IDs consistent across the nodes.

```
# id oracle
uid=502(oracle) gid=501(oinstall) groups=502(dba),501(oinstall)
x# id grid
uid=501(grid) gid=501(oinstall)
 groups=501(oinstall),504(asmadmin),506(asmdba),507(asmoper)
```

Starting with Oracle Database 12*c* Release 2 (12.2), you can follow the separation-of-duty best practices when administering Oracle RAC by using the new SYSRAC administrative privilege or the Oracle Clusterware agent. This obviates the need to use the much more powerful SYSDBA privilege. As with the SYSDBA, SYSBACKUP, and SSYSKM privileges introduced in earlier releases, the SYSRAC privilege helps enforce separation of duties. By default, the Clusterware agent uses this privilege to connect to an Oracle RAC database on behalf of SRVCTL and other RAC-related utilities.

Creating the Required Linux Directories

Before you can install the Grid Infrastructure and Oracle Database software and create

https://t.me/learningnets

RAC databases (explained in Chapter 5), you must set up certain required directories on the Linux file system.

Oracle Base Directory for the Grid Infrastructure (Grid User)

You need two Oracle base directories: one for the Oracle Grid Infrastructure installation and the other for the Oracle Database installation. The Oracle base directory for the Grid is where the Grid Infrastructure stores all diagnostic and administrative logs, as well the logs for the ASM and Oracle Clusterware operations. Here's an example of an Oracle Grid Infrastructure base directory:

/u01/app/grid

You must create this directory on each node of your cluster.

Oracle Home Directory for the Grid Infrastructure

You must create an Oracle home directory for Oracle Grid Infrastructure (Grid home) on all the cluster's nodes. You can't place the Grid home directory under an Oracle base directory or in the home directory of an installation owner.

Our Grid home is located here:

/u01/app/12.2.0/grid

Creating the Oracle Home and Oracle Base Directories

Oracle recommends that you create the Oracle Grid Infrastructure Grid home and the Oracle base home directories manually on each of your cluster's nodes, as shown here:

```
# mkdir -p /u01/app/12.1.0/grid
# mkdir -p /u01/app/grid
# mkdir -p /u01/app/oracle
# chown -R grid:oinstall /u01
# chown oracle:oinstall /u01/app/oracle
# chmod -R 775 /u01/
```

Configuring Shared Storage

In Oracle Database 12*c* Release 2 (12.2), you have several potential storage options for the various files and binaries related to the database and Clusterware:

- Oracle Automatic Storage Management (ASM)
- Oracle Automatic Storage Management Cluster File System (Oracle ACFS)
- Local file system
- OCFS2
- Network File System (NFS)
- Direct-attached storage (DAS)

https://t.me/learningnets

Technet24

Oracle ACFS is an extension of the Oracle ASM technology, and it enables you to support all your application data. ACFS provides a general-purpose file system; therefore, you can place Oracle database binaries and administrative files (such as trace files) on Oracle ACFS. Oracle Automatic Storage Management Dynamic Volume Manager (Oracle ADVM) provides volume management services.

Entities You Can Store on the Various Storage Options

Not everything can be stored on each of the storage options listed in the previous section. The following list shows you which storage options let you store which type of database/Clusterware files and binaries.

- **OCR and voting (Clusterware) files** You can store these only on Oracle ASM.
- **Oracle Clusterware binaries** You can store these only on the local file system (or on NFS).
- **Oracle RAC Database binaries** You can't store these on Oracle ASM. You can store them on Oracle ACFS, the local file system, OCFS2, DAS, or NFS.
- **Oracle RAC Database files (and Oracle RAC Database recovery files)** You can store these on ASM, ACFS, OCFS2, DAS, or NFS.

If you decide to use Automatic Storage Management (ASM) as shared storage for the datafiles (as we do in our installation later in this chapter), you should configure partitions on the external shared disks. ASM requires raw disks for storage because file systems cannot be used as ASM devices. The disk partitions are without any file systems and initially owned by the root user. You should only create one partition per disk presented to the ASM, and each ASM disk should be of the same size for better I/O performance.

System vendors use a logical volume manager (LVM) to present the external shared storage to the servers in the form of logical volumes to utilize the available storage space. However, using logical volumes for Oracle ASM to be used with Oracle RAC is not supported because logical volumes encapsulate the physical disk architecture and do not allow Oracle ASM to optimize the I/O on the available physical devices. More importantly, LVMs aren't allowed in an Oracle RAC environment because they can't be shared among the instances.

NOTE

Oracle does not support raw or block devices as shared storage for Oracle RAC in Oracle 12c. Oracle 11g Release 2 allowed you to upgrade while the voting disk and the OCR remained on raw or block devices. This is no longer allowed with Oracle 12c, and you have to plan on migrating to supported devices before you upgrade.

https://t.me/learningnets

Oracle provides ASM libraries to ease the management of the storage devices. The ASMLib library provides persistent paths and permissions for storage devices used with ASM. Once the ASMLib library is installed and configured on all cluster nodes, there is no need to update udev or any other device label files that typically store device paths and permissions.

Oracle supplies ASMLib for each Linux distribution, and it can be downloaded from the Oracle Technology Network website (www.oracle.com/technetwork/topics/linux/asmlib/index-101839.html). If you are using Oracle Linux, the Oracle ASMLib kernel driver is now included in the Unbreakable Enterprise Kernel (UEK); therefore, no driver package is required to be installed using UEK. Similarly, if you are on Red Hat Linux, you can download the kernel driver from the Red Hat support channel because it is no longer delivered by Oracle. Additional information on ASMLib drivers for Red Hat 6.0 and above is available at http://www.oracle.com/technetwork/server-storage/linux/asmlib/rhel6-1940776.html.

If you are using Linux and do not use Oracle Preinstallation RPM, the CVU tool will fail to detect shared disks. Therefore, it is recommended that you install the cvuqdisk RPM, which is available in the Grid Infrastructure installation media. You must copy the cvuqqdisk package to all nodes of your cluster and run it on those nodes, as shown here:

```
# pwd
/nfs/12c/grid/rpm
# export CVUQDISK_GRP=oinstall
# rpm -ivh cvuqdisk-1.0.10-1.rpm
Preparing...                ########################################### [100%]
   1:cvuqdisk               ########################################### [100%]
#
```

ASMLib

ASMLib is a support library for Automatic Storage Management. This library provides a layer of abstraction for the devices used by ASM. The ASMLib library also allows an Oracle database using ASM more efficient access to the disk groups it is using.

ASMLib primarily consists of three RPMs;

- **Kernel library** Typically named oracleasm-`uname –r`.rpm
- **Oracle Support Tools library/RPM** oracleasm-support-`uname –r`.rpm
- **oracleasmlib** `uname –r`.rpm

Oracle has stopped delivering the Kernel library for Red Hat versions 6.0 and above, but the same is included in Oracle’s Unbreakable Enterprise Kernel (UEK) and is not required for UEK/Oracle Linux 6.0 and above kernels. For Red Hat–

https://t.me/learningnets

Technet24

based operation systems, you have to contact Red Hat Support, and they would make the library available via their delivery channels. You can then download the remaining two RPMs from the Oracle OTN website.

Download the ASM library oracleasmlib-2.0 as well as the Oracle ASM utility oracleasm-support-2.0 from the aforementioned web location and install them on each cluster node as the root user. Here is an example of installing ASM libraries on a Linux machine:

```
# rpm -ivh oracleasm-support-2.1.3-1.el5x86_64.rpm \
oracleasmlib-2.0.4-1.el5.x86_64.rpm
```

Guidelines for Using Oracle ASM Disk Groups

If you configure only a single ASM disk group, Oracle will place the OCR and voting files, as well as the database and recovery files, all on that one disk group. By creating multiple disk groups, you can choose to place the various files on different disk groups.

When you install the Grid Infrastructure, you can create one or two ASM disk groups. Later on, you can use the Oracle Automatic Storage Management Configuration Assistant (ASMCA).

It's good to create two ASM groups during the installation of the Grid Infrastructure. Here's what the database will use the two ASM groups for:

- The first ASM disk group will store the OCR files as well as the Oracle ASM password file.
- The second ASM disk group will store the Grid Infrastructure Management Repository (GIMR) datafiles and the OCR backup files, which Oracle strongly recommends storing on a different disk group from where you store the OCR files.

Later on, when you install the Oracle Database following the installation of the Oracle Grid Infrastructure, you use the same disk groups for the database files as the Oracle Clusterware files, or you can create new disk groups for the database files.

Setting Up the Grid Infrastructure Management Repository

The Grid Infrastructure Management Repository (GIMR) is a multitenant database with a pluggable database (PDB) for the GIMR of each cluster. The database is called the Grid Infrastructure Management Repository Database (MGMTDB).

The GIMR stores the following cluster information:

- Performance data collected by the Cluster Heath Monitor
- Diagnostic data collected by the Cluster Health Advisor

https://t.me/learningnets

- CPU architecture data for Quality of Service (QoS) management
- Metadata for rapid home provisioning

In an Oracle standalone cluster, the GIMR is hosted on an Oracle ASM disk group as a multitenant database with one PDB. In an Oracle Domain Services cluster, you'll have a global GMIR, and client clusters (Oracle Member Cluster for Database) use this remote GIMR. Remote hosting of the GIMR reduces the overhead of running the Management Database on the individual clusters that are members of the Oracle Domain Services cluster.

During the installation, if you choose to configure an Oracle Domain Services cluster, you'll be prompted to configure a dedicated ASM disk group (with the default name MGMT) for the GIMR.

Configuring ASM

Before Oracle Grid Infrastructure and the Oracle RAC database can use ASM storage, you must configure ASM. This involves configuring the ASM library driver and creating the ASM devices, as explained in the following sections.

Configuring the ASM Driver You will have to configure the Oracle ASM library driver on all nodes in the cluster. While configuring this driver, choose the grid user and the asmdba group as the owner of the ASM driver because the grid user will own the Oracle Clusterware and ASM components of the Oracle RAC.

Oracle loads and runs the ASM driver on system startup according to this configuration. Here is an example of configuring ASM on a Linux machine:

```
#/etc/init.d/oracleasm configure
Configuring the Oracle ASM library driver.
This will configure the on-boot properties of the Oracle ASM library driver.
The following questions will determine whether the driver is loaded on boot
and what permissions it will have. The current values will be shown in
brackets ('[]'). Hitting <ENTER> without typing an answer will keep
that current value. Ctrl-C will abort.
Default user to own the driver interface []: grid
Default group to own the driver interface []: asmdba
Start Oracle ASM library driver on boot (y/n) [n]: y
Scan for Oracle ASM disks on boot (y/n) [y]: y
Writing Oracle ASM library driver configuration: done
Initializing the Oracle ASMLib driver: [ OK ]
Scanning the system for Oracle ASMLib disks: [ OK ]
```

Repeat this step on each cluster node. If you don't perform this step, the oracleasm service will not start on the cluster node and you will not be able to use Oracle ASM as shared storage for the Oracle RAC database.

Creating the Oracle ASM Devices Once you configure the Oracle ASM driver

https://t.me/learningnets

Technet24

on each cluster node, your next step is to mark the shared disk partitions on the first cluster node and scan the marked disk partitions on the other nodes in the cluster. Marking a shared disk partition is also called “creating the Oracle ASM device.”

You will perform the following step on the first cluster node only to mark all candidate disk partitions to be used with Oracle ASM. This ensures the disk headers are wiped clean because ASM will not select the devices as candidates if it finds the Volume Table of Contents (VTOC) information of any volume manager or file system.

```
#dd if=/dev/zero of=/dev/sda1 bs=1M count=10
10+0 records in
10+0 records out
10485760 bytes (10 MB) copied, 0.400074 seconds, 26.2 MB/s
# dd if=/dev/zero of=/dev/sdb1 bs=1M count=10
10+0 records in
10+0 records out
10485760 bytes (10 MB) copied, 0.431521 seconds, 24.3 MB/s
# dd if=/dev/zero of=/dev/sdc1 bs=1M count=10
10+0 records in
10+0 records out
10485760 bytes (10 MB) copied, 0.414814 seconds, 25.3 MB/s
```

Once the disk headers are wiped clean, the next step is to stamp the ASM header information on the devices. The following sequence of operations mark the devices as ASM disks:

```
# /usr/sbin/oracleasm init
Creating /dev/oracleasm mount point: /dev/oracleasm
Loading module "oracleasm": oracleasm
Mounting ASMlib driver filesystem: /dev/oracleasm
[# /usr/sbin/oracleasm createdisk DISK1 /dev/sda1
Writing disk header: done
Instantiating disk: done
[
# /usr/sbin/oracleasm createdisk DISK2 /dev/sdb1
Writing disk header: done
Instantiating disk: done
# /usr/sbin/oracleasm createdisk DISK3 /dev/sdc1
Writing disk header: done
Instantiating disk: done
```

Once all the candidate disk partitions are marked as candidate disks for Oracle ASM, scan the newly marked disks on the other nodes in the cluster because there is no need to mark the candidate disk partitions on each cluster node. Here is an example of scanning the marked disks on other nodes in the cluster:

https://t.me/learningnets

```
# /usr/sbin/oracleasm scandisks
Scanning system for ASM disks [ OK ]
# /usr/sbin/oracleasm scandisks
Reloading disk partitions: done
Cleaning any stale ASM disks...
Scanning system for ASM disks...
# /usr/sbin/oracleasm listdisks
DISK1
DISK2
DISK3
```

When using ASMLib devices for ASM, make sure not to use the discovery string of /dev/oracleasm/disk/*. Instead use ORCL:* (if required); otherwise, Oracle won't use the underlying ASMLib libraries and will make traditional I/O calls instead. This results in your system not benefiting from the advantages offered by ASMLib.

NOTE

When using Oracle ASM devices on multipath devices, update the ORACLEASM_SCANORDER and ORACLEASM_SCANEXCLUDE variables in the /etc/sysconfig/oracleasm file on all nodes in the cluster with the expected scan ordering and exclusion of the multipath devices, respectively. You should refer to My Oracle Support Note 394956.1, which explains this process in detail.

Secure Shell and User Limits Configuration

Oracle Universal Installer (OUI) installs the binaries in one node and then propagates the files to the other nodes in the cluster. The Installer primarily uses the **ssh** and **scp** commands in the background during installation to run remote commands and copy files to the other nodes in the cluster.

This requires noninteractive file copying across the nodes. You must configure SSH so that these commands do not prompt for a password. Setting up SCP can enable this across the nodes. You also need to turn off the banner for SSH. Unlike in previous releases of Oracle RAC, there is no need to manually configure the user equivalence for the Oracle software owners because Oracle allows you to set this up during installation of the Oracle Grid Infrastructure.

If you want to configure SSH, which would be required for the Cluster Verification Utility to succeed, you can use the sshUserSetup.sh script available in the Oracle 12*c* install media under ../grid/sshsetup/sshUserSetup.sh. Here is an example:

```
# ./sshUserSetup.sh -user grid -hosts "oltracn01 oltracn02" -advanced -verify
-noPromptPassphrase
```

https://t.me/learningnets

Technet24

NOTE

Oracle recommends that you disable Transparent HugePages and instead use standard HugePages or enhanced performance.

Setting User Limits

The default hard limits for Oracle software owners aren't sufficient to install and configure Infrastructure. You should edit the /etc/security/limits.conf file as shown here:

```
grid soft nproc 2047
grid hard nproc 16384
grid soft nofile 1024
grid hard nofile 65536
oracle soft nproc 2047
oracle hard nproc 16384
oracle soft nofile 1024
oracle hard nofile 65536
```

You'll also need to add the following line to the /etc/pam.d/login configuration file if it has not already been added:

```
session    required    pam_limit.so
```

This configuration causes the user login process to load the pam_limits.so module of PAM to further enforce/set the hard limits for the login user as defined in the /etc/security/limits.conf configuration file, because /etc/security/limits.conf is the configuration file of the pam_limits.so module of PAM.

Pluggable Authentication Module (PAM)

PAM was invented by Sun, and in the Linux operating system it provides library modules to supply a flexible authentication mechanism. Programs such as login and su can use these modules for authentication. You can add the updated authentication schemes into the PAM modules rather than changing the programs that depend on the authentication. Each PAM module has its own text-configuration file, and PAM consults with these text-configuration files before taking any security-related action for the application. In a nutshell, PAM provides account, authentication, and session management.

Configuring the Kernel Parameters

https://t.me/learningnets

As the *root* user, update the kernel parameters in the /etc/sysctl.conf file on *both* nodes. If these settings are already configured, make sure they are at least set to the following values (it's okay if the values are set higher):

```
kernel.shmmni = 4096
kernel.sem = 250 32000 100 128
fs.file-max = 512 x processes (for example 6815744 for 13312 processes)
net.ipv4.ip_local_port_range = 9000 65500
net.core.rmem_default = 262144
net.core.rmem_max = 4194304
net.core.wmem_default = 262144
net.core.wmem_max = 1048576
```

NOTE

You should always refer to Oracle My Support Note 169706.1 for the latest information on the updated prerequisites (kernel, memory, swap, OS packages, and so on) prior to installing Oracle databases. Oracle regularly updates this My Oracle Support note, which covers the system and software requirements for each platform that supports the Oracle RAC database.

Running the Cluster Verification Utility

Deploying Oracle RAC is a multistage operation, and components can malfunction at any stage due to the various dependencies at each layer. Testing and verification are required at each stage before proceeding to the next stage. Oracle provides a tool you can use to verify the cluster setup at every stage: the Cluster Verification Utility, called cluvfy.

The Cluster Verification Utility is a standalone tool used to verify a well-formed Oracle RAC cluster. It can be used at any stage—preinstallation, configuration, or operation—because it verifies the entire cluster stack and does not perform any configuration changes in cluster or Oracle RAC operations.

You can use cluvfy to verify the stage-by-stage progress during Oracle RAC installation, as each stage comprises a set of operations during Oracle RAC deployment. Each stage has its own set of entry (pre-checks) and/or exit (post-checks) criteria.

Oracle internally runs cluvfy during the installation process to verify the prerequisites and provide the fix-up scripts for fixable issues. It's good to run this utility to verify the prerequisites prior to the installation of the Oracle Grid Infrastructure, but this isn't compulsory. Oracle includes this tool in the install media, but we strongly recommend that you download the latest version from Oracle Technology Network (OTN) available at http://www.oracle.com/technetwork/database/options/clustering/downloads/cvu-

https://t.me/learningnets

Technet24

download-homepage-099973.html.

Cluster Verification Utility Stages

Following are the 14 stages in the Oracle RAC deployment where you can run the Cluster Verification Utility. Note that the utility automatically checks all nodes that are specified, so it's necessary to run the utility from only one node. This utility is available in the Oracle Grid Infrastructure software media and is named runcluvfy.sh.

```
[oracle@rac2 grid_1]$ ./runcluvfy.sh stage -list
USAGE:
runcluvfy.sh stage {-pre|-post} <stage-name> <stage-specific options>
  [-verbose]
Valid Stages are:
      -pre cfs         : pre-check for CFS setup
      -pre crsinst     : pre-check for CRS installation
      -pre acfscfg     : pre-check for ACFS Configuration.
      -pre dbinst      : pre-check for database installation
      -pre dbcfg       : pre-check for database configuration
      -pre hacfg       : pre-check for HA configuration
      -pre nodeadd     : pre-check for node addition.
      -pre appcluster  : pre-check for Oracle Clusterware Application Cluster
Installation
      -post hwos       : post-check for hardware and operating system
      -post cfs        : post-check for CFS setup
      -post crsinst    : post-check for CRS installation
      -post acfscfg    : post-check for ACFS Configuration.
      -post hacfg      : post-check for HA configuration
      -post nodeadd    : post-check for node addition.
      -post nodedel    : post-check for node deletion.
      -post appcluster : post-check for Oracle Clusterware Application Cluster
 Installation
[oracle@rac2 grid_1]
```

At the first stage in preparing the hardware for Oracle RAC installation, you can use cluvfy to check the hardware for basic network connectivity and the most common operating system requirements. It also checks the software owner privileges and access to shared storage from both the nodes. It can be invoked by using the following command-line option. The output text (edited for clarity and simplicity) will be similar to the following:

https://t.me/learningnets

```
oracle@rac1 grid_1]$ ./runcluvfy.sh stage -post hwos -n rac1,rac2 -verbose
Verifying Node Connectivity ...
  Verifying Hosts File ...
  Node Name                             Status
  ------------------------------------  ------------------------
  rac1                                  passed
  rac2                                  passed
  Verifying Hosts File ...PASSED
Interface information for node "rac1"
…
Interface information for node "rac2"
…
Check: MTU consistency on the private interfaces of subnet "192.168.10.0"
  Node              Name          IP Address    Subnet        MTU
  ----------------  ------------  ------------  ------------  ----------------
  rac1              eth1          192.168.10.1  192.168.10.0  1500
  rac2              eth1          192.168.10.2  192.168.10.0  1500
  Verifying Check that maximum (MTU) size packet goes through subnet ...PASSED
  Verifying subnet mask consistency for subnet "192.168.56.0" ...PASSED
  Verifying subnet mask consistency for subnet "192.168.10.0" ...PASSED
Verifying Node Connectivity ...PASSED
Verifying Multicast check ...
Verifying ASM Integrity ...
  Verifying Node Connectivity ...
Verifying ASM Integrity ...PASSED
Verifying Users With Same UID: 0 ...PASSED
Verifying Time zone consistency ...PASSED
Verifying Shared Storage Discovery ...
  ASM Disk Group                        Sharing Nodes (2 in count)
  ------------------------------------  ------------------------
  DATA                                  rac1 rac2
Verifying Shared Storage Discovery ...PASSED
Verifying DNS/NIS name service ...PASSED
Post-check for hardware and operating system setup was successful.
CVU operation performed:      stage -post hwos
Date:                         Apr 25, 2017 5:48:13 PM
CVU home:                     /u01/app/12.2.0.1/grid_1/
User:                         oracle
[oracle@rac1 grid_1]$
```

An individual subsystem or a module of the Oracle RAC cluster is known as a *component* in the utility. You can also use the Cluster Verification Utility to verify the component level. You can verify specific properties such as availability, integrity, liveliness, and sanity for any individual cluster component.

The **cluvfy comp –list** command shows the cluster components:

https://t.me/learningnets

```
[oracle@rac2 bin]$ pwd
/u01/app/12.2.0.1/grid_1/bin
[oracle@rac2 bin]$ ./cluvfy comp -list

USAGE:
cluvfy comp  <component-name> <component-specific options>  [-verbose]
Valid Components are:
      nodereach        : checks reachability between nodes
      nodecon          : checks node connectivity
      ssa              : checks shared storage accessibility
      space            : checks space availability
      sys              : checks minimum system requirements
      clu              : checks cluster integrity
      clumgr           : checks cluster manager integrity
      ocr              : checks OCR integrity
      olr              : checks OLR integrity
      ha               : checks HA integrity
      freespace        : checks free space in CRS Home
      crs              : checks CRS integrity
      nodeapp          : checks node applications existence
      admprv           : checks administrative privileges
      peer             : compares properties with peers
      software         : checks software distribution
      acfs             : checks ACFS integrity
      asm              : checks ASM integrity
      gpnp             : checks GPnP integrity
      gns              : checks GNS integrity
      scan             : checks SCAN configuration
      ohasd            : checks OHASD integrity
      clocksync        : checks Clock Synchronization
      vdisk            : checks Voting Disk configuration and UDEV settings
     healthcheck      : checks mandatory requirements and/or best practice
 recommendations
      dhcp             : checks DHCP configuration
      dns              : checks DNS configuration
      baseline         : collect and compare baselines
[oracle@rac2 bin]$
```

Once you verify the hardware, operating system, and network-related components, the cluster is ready for the Oracle Grid Infrastructure installation.

Oracle Cluster Ready Services (CRS) is the foundation for the Oracle Grid Infrastructure. You can check whether the cluster is ready for the installation of CRS by specifying the following options for the Cluster Verification Utility. You should

https://t.me/learningnets

invoke this command as the OS user *grid* and list the cluster nodes as command-line parameter arguments (note that the output has been edited for clarity).

```
[oracle@rac1 bin]$ ./cluvfy stage -pre crsinst -n rac1,rac2 -r 12.2

Verifying Physical Memory ...PASSED
Verifying Free Space:
 rac2:/usr,rac2:/var,rac2:/etc,rac2:/u01/app/12.2.0.1/grid_1,
rac2:/sbin,rac2:/tmp ...PASSED
Verifying User Existence: oracle ...
  Verifying Users With Same UID: 54321 ...PASSED
Verifying User Existence: oracle ...PASSED
Verifying Group Existence: oinstall ...PASSED
```

If the Cluster Verification Utility doesn't report any errors, you've successfully set up your cluster for the Oracle Grid Infrastructure installation. If the check reveals errors, you must fix them before starting the Oracle Grid Infrastructure installation.

If the Cluster Verification Utility reports a success, your next step is to install Oracle Grid Infrastructure on the cluster nodes, as we explain in the following sections.

Installing Oracle Grid Infrastructure with OUI

Oracle Grid Infrastructure installation is the foundation for the Oracle RAC database installation. The Oracle Grid Infrastructure is the backbone of the remainder of the installation, because more than one database can share the same Oracle Grid Infrastructure foundation.

You run the Oracle Universal Installer (OUI) from one of the cluster's nodes as the grid OS user, and the Installer will propagate all the required files to the other nodes using **scp** commands.

The Installer works in the background and sometimes appears to be hanging in the foreground. For those who are interested in knowing the undercover operations of the Installer, we'll trace the OUI in the following section.

Tracing the Universal Installer

OUI is a simple Java program that copies the files from the staging area and relinks with the operating system libraries to create and update the Oracle inventory. The staging area can be a CD-ROM/DVD or a local mount point in the file system.

OUI puts minimal details about the installation's options and progress in the log file installations_<timestamp>.log. This log file is typically stored in the $ORACLE_HOME/orainventory directory. However, it doesn't contain a detailed level of operations to debug the installation issues.

You can invoke the OUI with the DTRACING (or Java tracing) option on the command line:

https://t.me/learningnets

Technet24

```
$ runInstaller -debug  -J-DTRACING.ENABLED=true -J-DTRACING.LEVEL=2
-DSRVM_TRACE_LEVEL=2
```

Optionally, you can redirect the tracing output to a file. We always start the Installer with tracing on so that we don't need to restart it if any failures occur. Here's an example showing the redirection of the output of the tracing to an output file:

```
$ script /tmp/install_screen.`date +%Y%m%d%H%M%S`.out
$ runInstaller -debug -J-DTRACING.ENABLED=true -J-DTRACING.LEVEL=2
-DSRVM_TRACE_LEVEL=2
```

Once the installation is complete, simply type **exit** to close the output file. This tracing option is useful during installation of Oracle RAC, where the files are copied to the remote nodes using the **scp** command. Tracing tells you what the Installer is doing at any point in time by providing detailed traces of the install actions. However, the limited information is also written to the trace file <installations_timestamp>log, but this information is very "high level" and does not get into the actual problems faced by the Installer.

OUI Command-Line Options and Exit Codes

Oracle Universal Installer also supports a few command-line parameters during the installation. You can use the **runInstaller -help** command to show the full list of command-line options and their descriptions, as well as command-line variables usage.

OUI also sends the exit codes after the installation, which can be useful in an unattended installation. The following table summarizes the OUI exit codes and what they stand for.

Code	Description
0	All installations were successful.
1	All installations were successful, but some optional configuration tools failed.
–1	At least one installation failed.

Installing Oracle Grid Infrastructure

It has been a long and hard road to this point, but we're ready to start installing the Oracle Grid Infrastructure. Since we've spent quite a bit of time going over the prerequisites and learning how to satisfy them, the installation should be smooth sailing (in most cases).

Starting with Oracle Grid Infrastructure 12*c* Release 2 (12.2), Oracle Grid Infrastructure software is available as an image file for download and installation. In order to install the Grid Infrastructure, you must run the following as the oracle user. Go to the Grid Control home directory and run the gridSetup.sh script you'll find there, as shown here:

https://t.me/learningnets

```
$ cd /u01/app/app/12.2.0.1/grid
$ ./gridSetup.sh
```

In the following sections, we take you step by step all the way to the end of the Grid Infrastructure installation process. We break the process down into chunks, so as to make it easy to understand the rather long series of steps.

Choosing the Installation Options and Naming Your Cluster

The very first screen in the Grid Infrastructure installation offers options to configure the software, as shown in Figure 4-2.

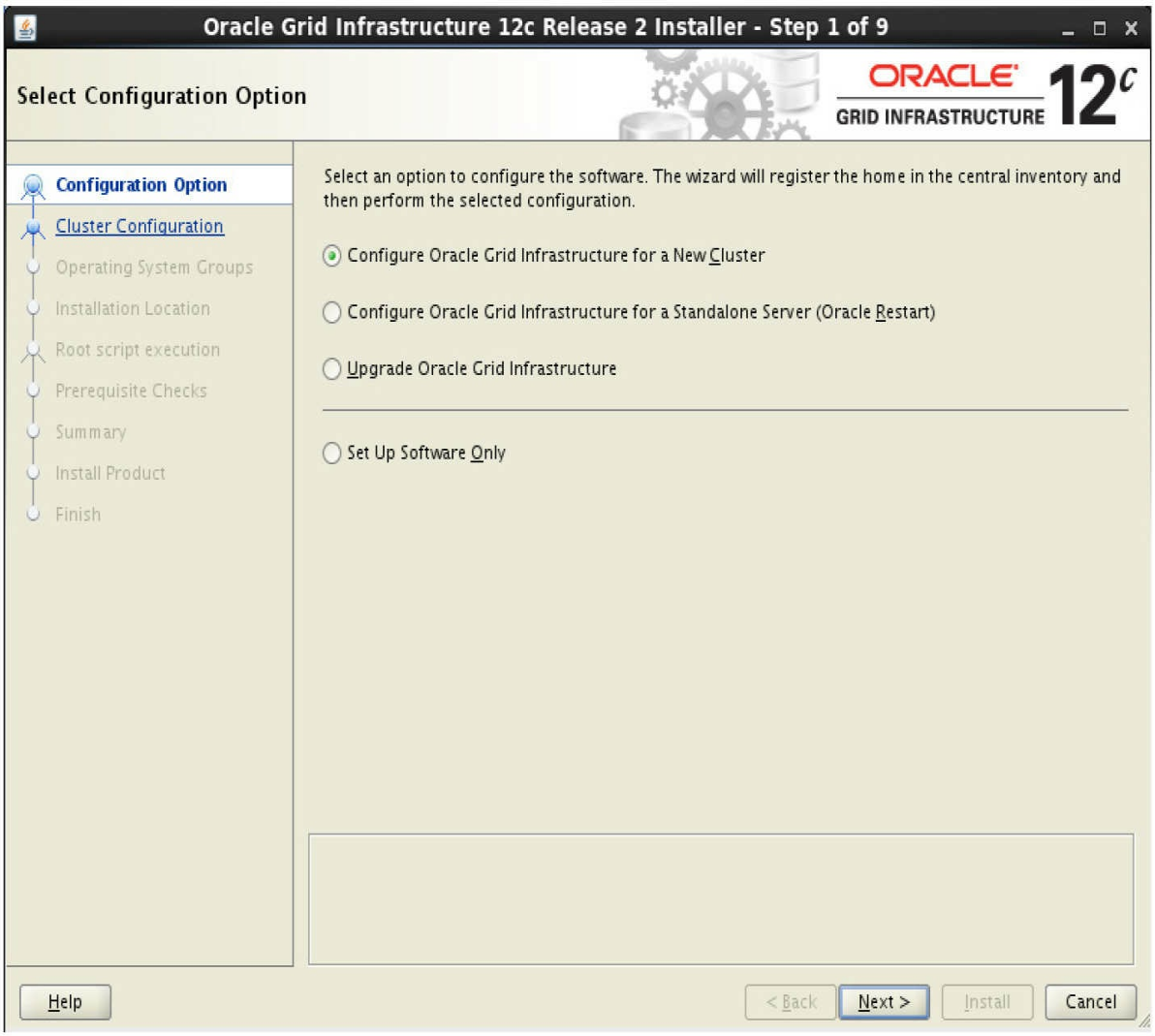


FIGURE 4-2. *The Select Configuration Option screen*

Following are the options you'll see on this screen:

- **Configure Oracle Grid Infrastructure for a New Cluster** This option (the

https://t.me/learningnets

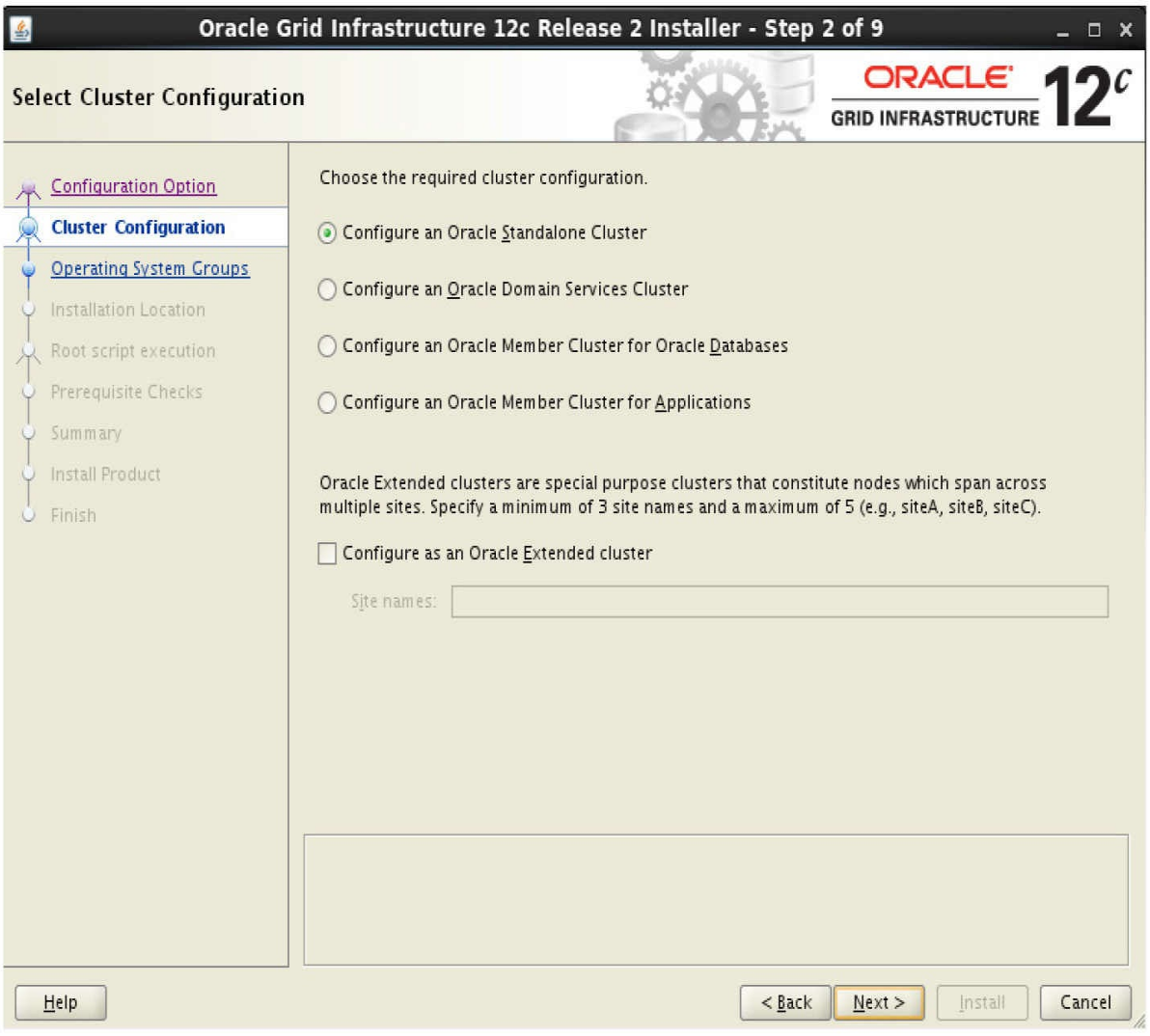
Technet24

default) installs and configures the Oracle Grid Infrastructure for the current installation.

- **Configure Oracle Grid Infrastructure for a Standalone Server (Oracle Restart)** This option installs and configures the Oracle Grid Infrastructure on a standalone server. This option is useful if you want to create an Oracle RAC One Node database because it requires Oracle Grid Infrastructure to be up and running on the server before it can be installed and configured.
- **Upgrade Oracle Grid Infrastructure** This option upgrades the existing Oracle Grid Infrastructure installed on the cluster nodes.
- **Set Up Software Only** This option installs the Oracle Grid Infrastructure binaries on the current node only. This option is rarely used because it will not perform remote operations or any other cluster configurations, such as configuring virtual IPs, the local listener, or SCAN listeners.

After choosing one of these installation options, click Next.

In the Select Cluster Configuration screen, shown in Figure 4-3, you can choose from among the following cluster types:



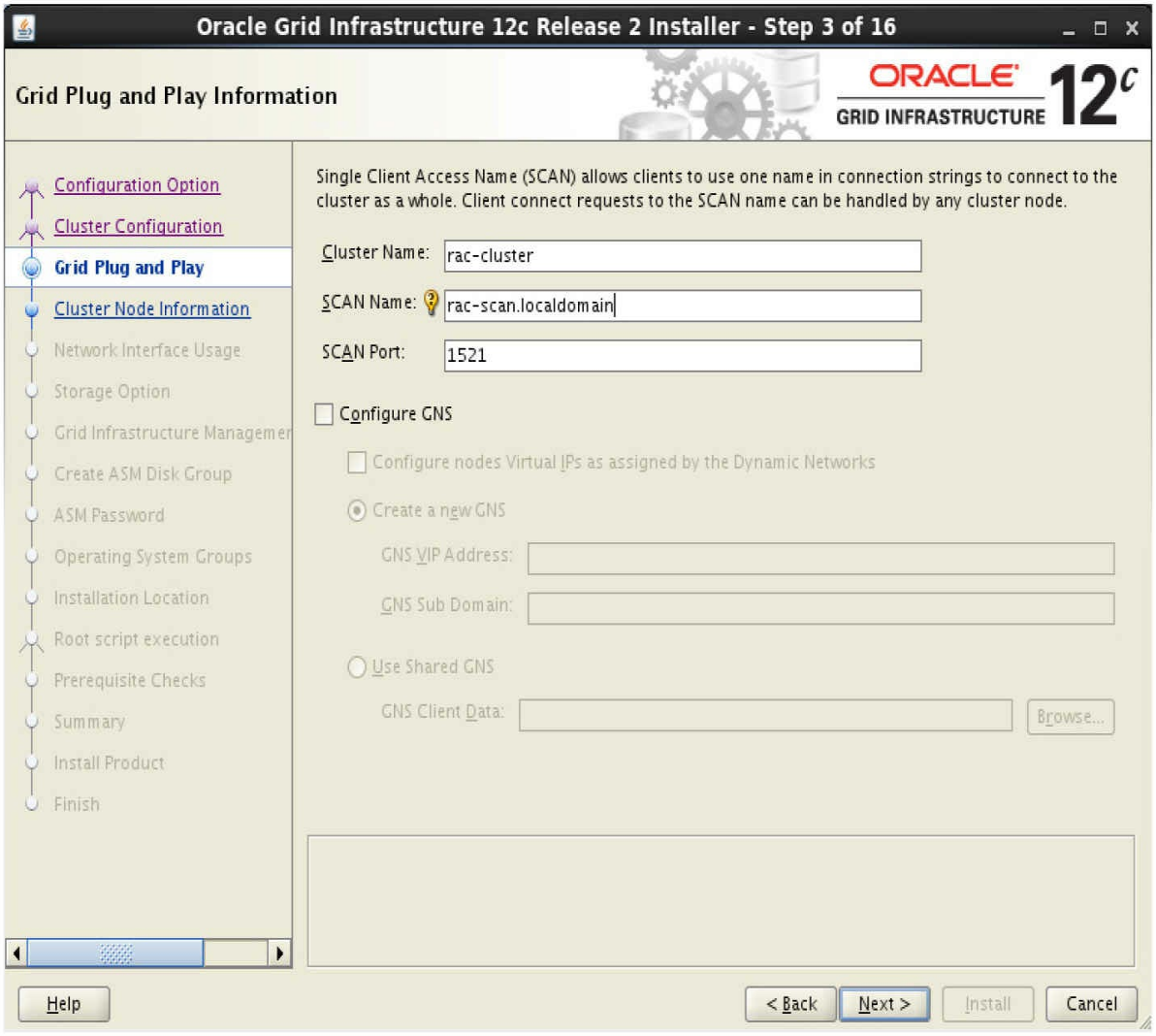
https://t.me/learningnets

FIGURE 4-3. *The Select Cluster Configuration screen*

- Oracle standalone cluster
- Oracle Domain Services cluster
- Oracle member cluster for Oracle databases
- Oracle member cluster for applications
- Oracle extended cluster

In this case, we'll select the first option, Configure an Oracle Standalone Cluster, and then click Next.

In the Grid Plug and Play Information screen, shown in Figure 4-4, you can specify configuration details for Grid Plug and Play functionality. On this screen, you specify the name of your cluster, SCAN, the port number to be used by the SCAN listeners, and the Grid Naming Service (GNS) information. You should uncheck the Configure GNS check box on this screen because we won't be configuring GNS in this installation.



https://t.me/learningnets

Technet24

FIGURE 4-4. *The Grid Plug and Play Information screen*

Grid Plug and Play

The Grid Plug and Play feature of the Oracle Grid Infrastructure eases the management of large grid systems by providing the ability to add and remove cluster nodes dynamically. Oracle requires a dynamic naming scheme to implement this feature; hence, it uses GNS (Grid Naming Service). GNS provides such a naming scheme by using a DHCP server to assign virtual IPs dynamically. Because GNS provides virtual IP assignments dynamically and also resolves the server names, this eliminates any manual configuration of IPs and server names before a node is added or removed from the existing cluster.

NOTE

Make sure the cluster name is unique in your network because Oracle Grid Control will not discover and allow you to add the cluster if another cluster with the same name exists in the network.

Specifying the Cluster Nodes and Verifying SSH Connectivity

Next up is the Cluster Node Information screen, shown in Figure 4-5, where you must specify the names of all the nodes in the cluster. For each node, you must specify the public and virtual node names.

https://t.me/learningnets

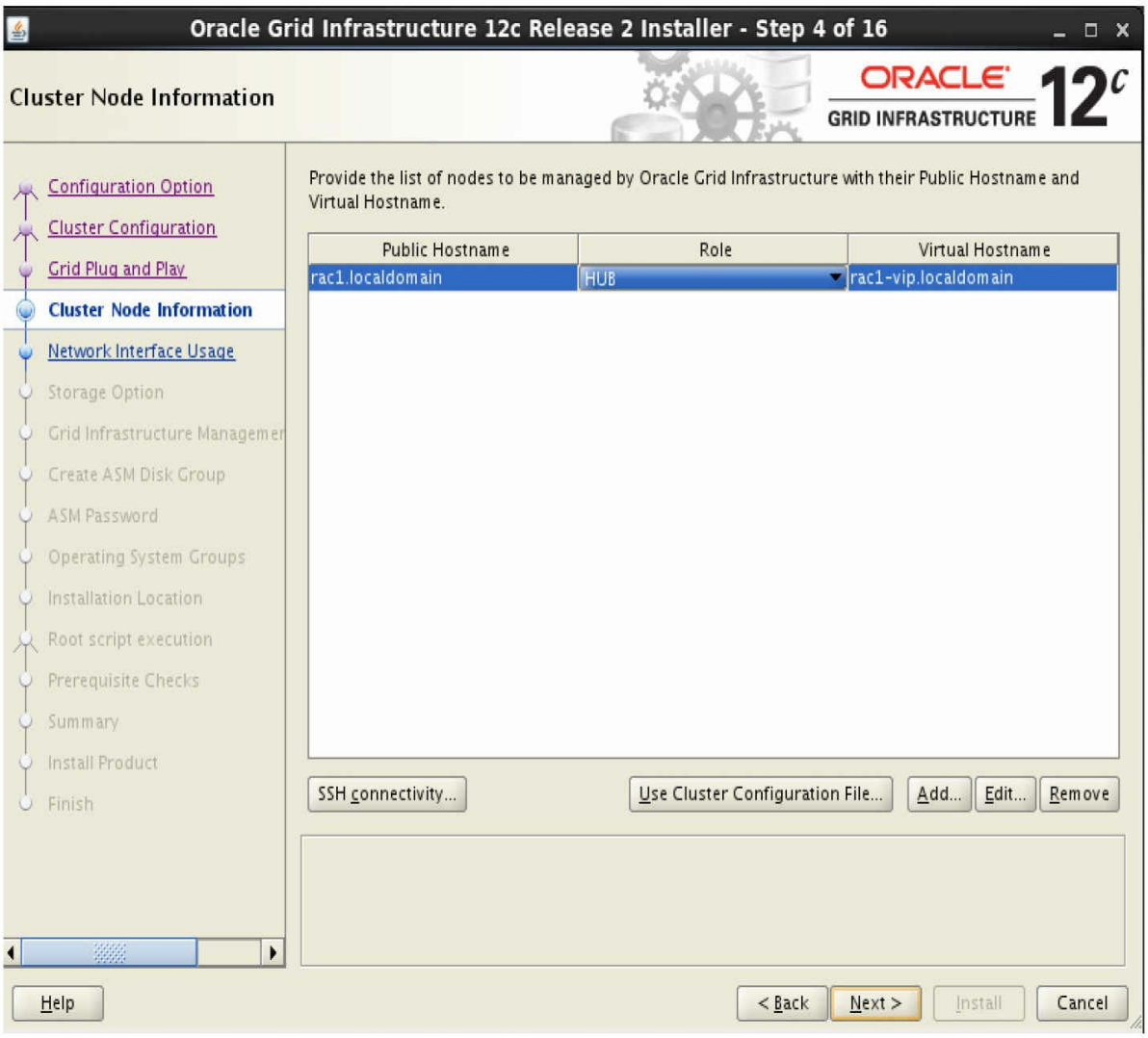


FIGURE 4-5. *The Cluster Node Information screen*

NOTE

Oracle assigns a virtual IP and virtual node name automatically if GNS is configured and selected in the Grid Plug and Play screen.

You can click the Add, Edit, or Remove button to add, modify, or remove node names, respectively. Figure 4-6 shows how you can add a single node or a range of nodes by specifying a node range expression for the nodes you wish to add to this cluster. In our case, we'll choose to add a single node, rac2, to the already existing node, rac1, making a total of two nodes for our cluster.

https://t.me/learningnets

Technet24

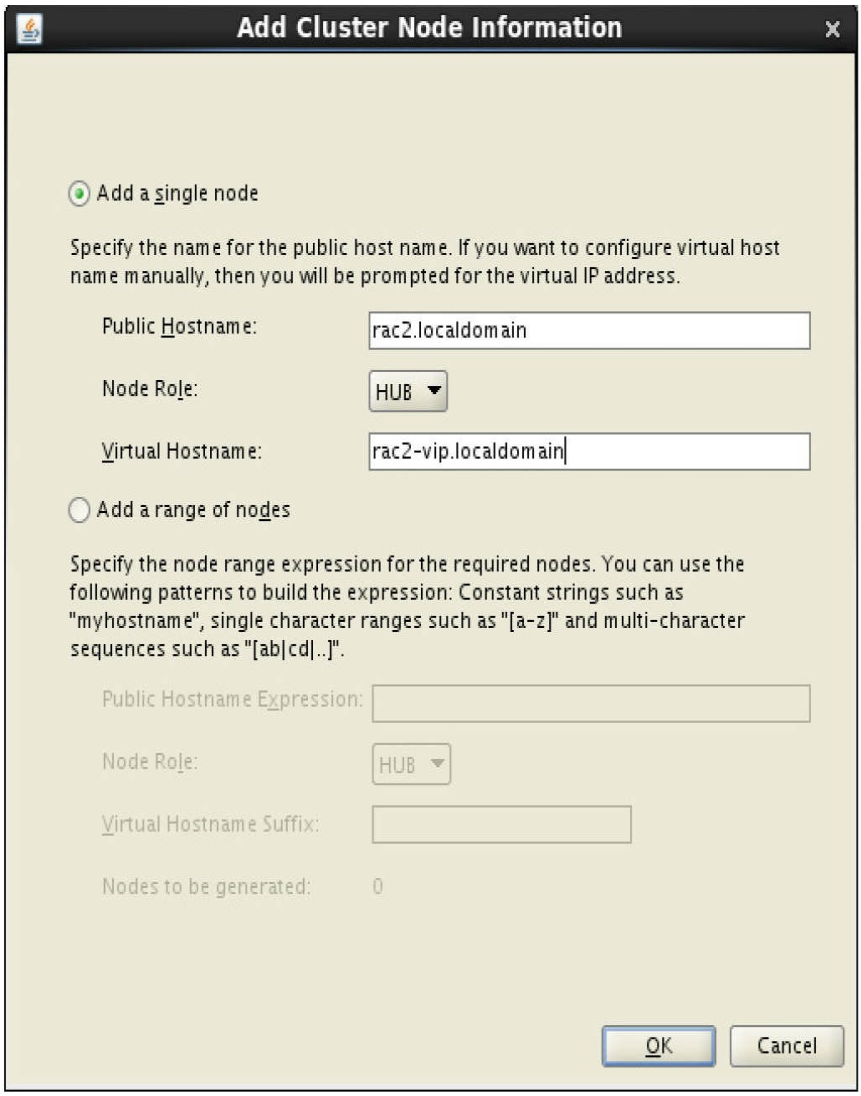


FIGURE 4-6. *Adding nodes to the cluster*

Also on this screen, you can configure the passwordless connectivity for the Oracle Grid software owner between the nodes you specified by clicking the SSH Connectivity button. Figure 4-7 shows how you can specify the password for the OS user oracle to test the SSH connectivity. Click Test once you specify the password for the oracle user.

https://t.me/learningnets

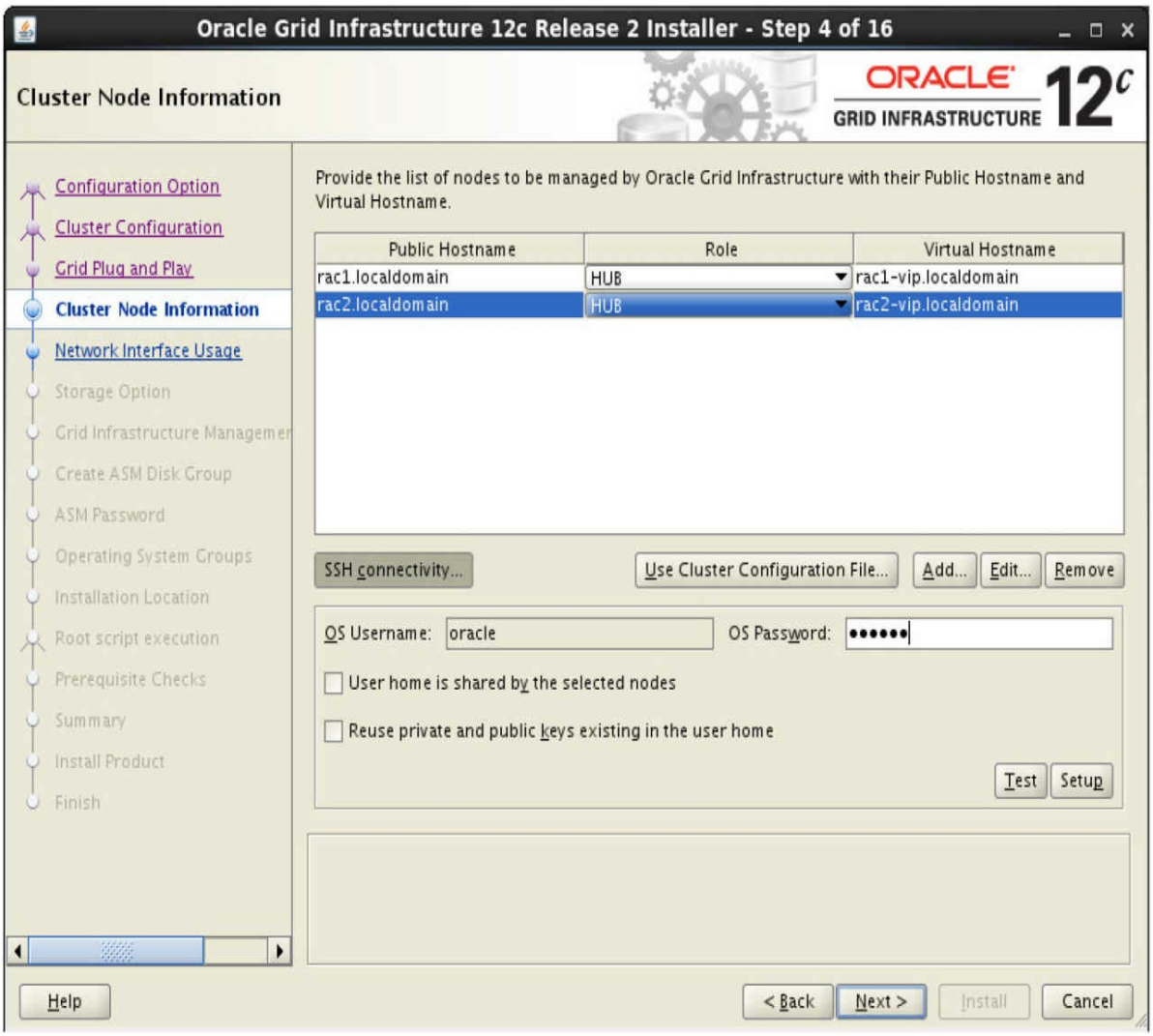


FIGURE 4-7. *Testing SSH connectivity*

Figure 4-8 shows that passwordless SSH connectivity is successfully established between the selected nodes.

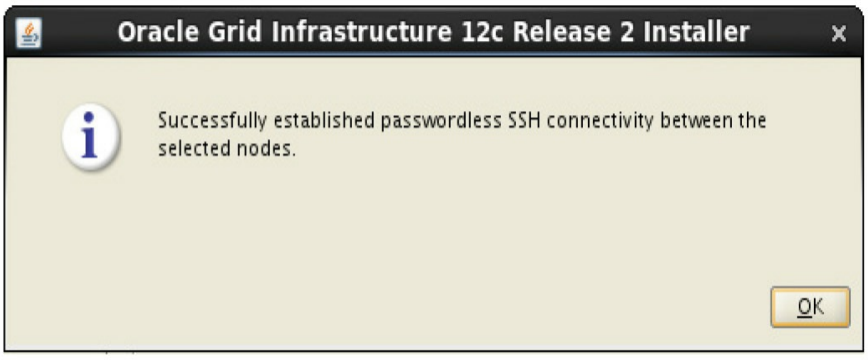


FIGURE 4-8. *Establishing passwordless SSH connectivity between the cluster's nodes*

https://t.me/learningnets
Technet24

After specifying the cluster node names and testing the SSH connectivity between all nodes of the cluster, click Next.

Highly Available IP (HAIP)

Highly Available IP is a private virtual IP that Oracle Grid Infrastructure configures to provide redundancy for private interconnect in Oracle Grid Infrastructure 12*c*. Oracle can configure up to four HAIPs per cluster server for private network redundancy.

Oracle utilizes IP range 169.254.0.0 and multicasting on the private network to implement this redundancy; hence, it's very important that multicasting is enabled on private interfaces and that you aren't using the 169.254.0.0 range on your network. Otherwise, the Oracle Grid Infrastructure 12*c* installation or an upgrade to Oracle 12*c* will fail.

Refer to My Oracle Support Bulletin 1212703.1, "Grid Infrastructure Install or Upgrade May Fail Due to Multicasting Requirement," for additional details and a test program to validate the requirement. With this feature, you don't have to employ any third-party or OS-level bonding to provide redundancy.

You can define multiple private network adapters during the installation process or subsequently using the **oifcfg** command. HAIP allows interconnect traffic to be load-balanced across all active interconnect interfaces, and corresponding HAIP addresses will be failed over transparently to other or available and configured adapters. If you use multiple interfaces during installation or configure them later using **oifcfg**, you will notice entries something like the following in alert logs of ASM and RDBMS instances:

```
Cluster communication is configured to use the following interface(s)
 for this instance
  169.254.92.149
  169.254.239.6
cluster interconnect IPC version: Oracle UDP/IP (generic)
IPC Vendor 1 proto 2
```

The Installer creates additional virtual IP addresses, one per each interface, as follows:

```
eth1:1    Link encap:Ethernet  HWaddr 08:00:27:97:DD:10
          inet addr:169.254.92.149  Bcast:169.254.127.255  Mask:255.255.128.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

eth2:1    Link encap:Ethernet  HWaddr 08:00:27:A2:6A:CB
          inet addr:169.254.239.6  Bcast:169.254.255.255  Mask:255.255.128.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

Specifying the Network Interfaces

In the Specify Network Interface Usage screen, shown in Figure 4-9, you can specify the planned usage of each network interface. If there are many network cards in the

https://t.me/learningnets

nodes, you can instruct Oracle Grid Infrastructure as to which network cards it must use for public traffic and which ones to reserve for private network traffic and which one to be used as the ASM network.

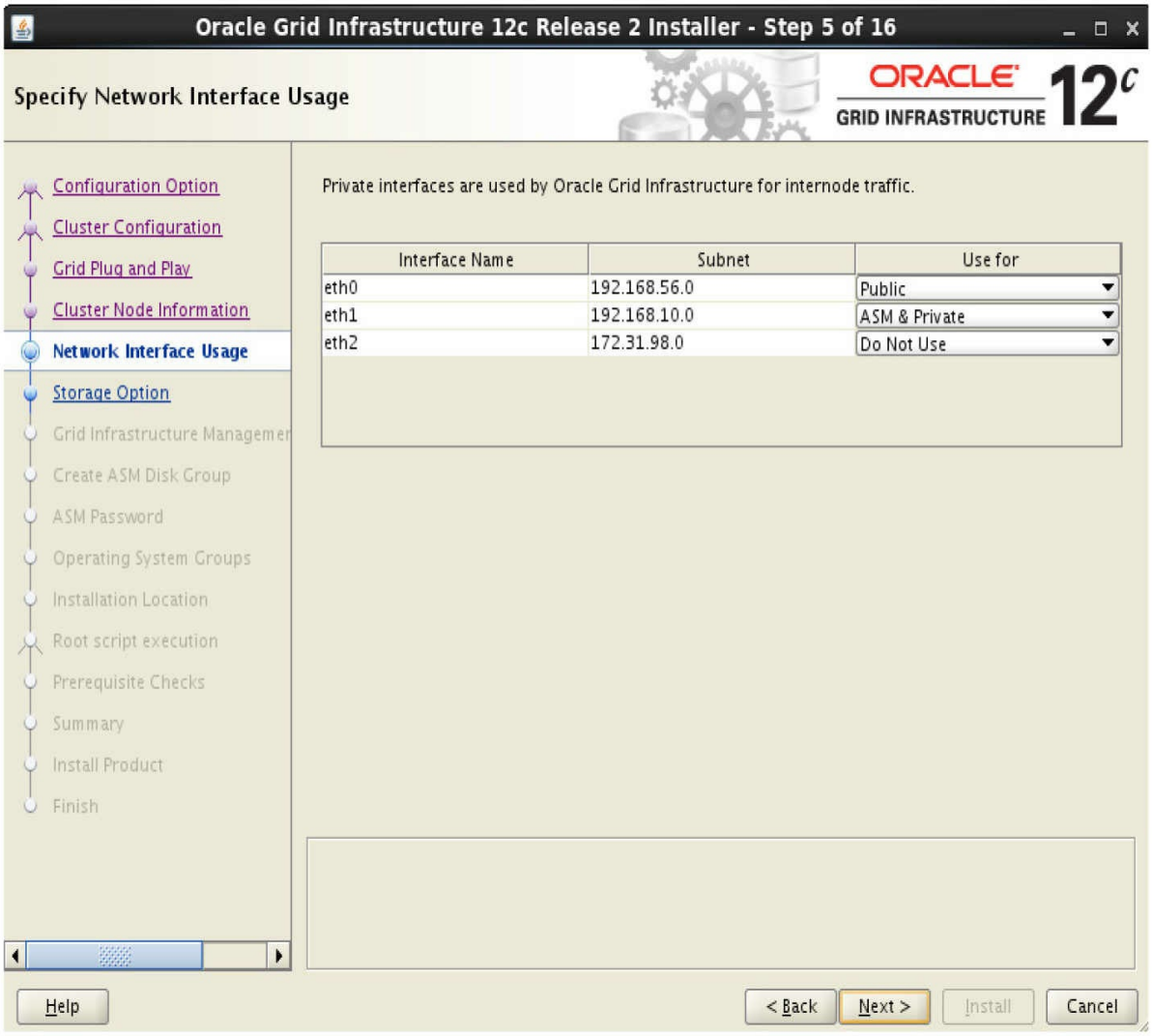


FIGURE 4-9. *The Specify Network Interface Usage screen*

Here, you can specify the interface name and the subnet. You'll also specify the public and private networks, and the VIP interface must be set to Do Not Use. If you don't want Oracle to use a specific network interface, you can select Do Not Use.

If the network IP address starts with *10.x* or *192.x*, it is assumed to be a private network by default. Starting with Oracle 11*g* Release 2 (11.2.0.2), you can select multiple private interfaces, and Oracle will automatically configure private interconnect for redundancy by using Highly Available IP (HAIP) cluster resources.

Specifying a separate network for ASM is a new feature in Oracle 12*c* that defines a dedicated network to provide the capability of isolating ASM's internal network traffic from other private network traffic. Oracle will use this network specifically as a communication path for all the traffic between databases instances and ASM instances,

https://t.me/learningnets
Technet24

which is typically metadata and extent maps.

If you choose the ASM & Private option, OUI expects that you would be using Oracle Flex ASM for Storage. If you are using local ASM, then select Private. In this screen, we are using two interfaces for the “ASM & Private” network, thus taking advantage of the HAIP feature instead of using OS-level bonding. Once the correct network is selected, click Next.

Selecting the Storage Options

Oracle ASM provides shared storage to the instances that are part of a database, which is a primary requirement of an Oracle RAC database. Figure 4-10 shows the next screen presented by the Installer, the Storage Option Information screen, where you’re asked to choose between configuring Oracle ASM on block devices or in an NFS location.

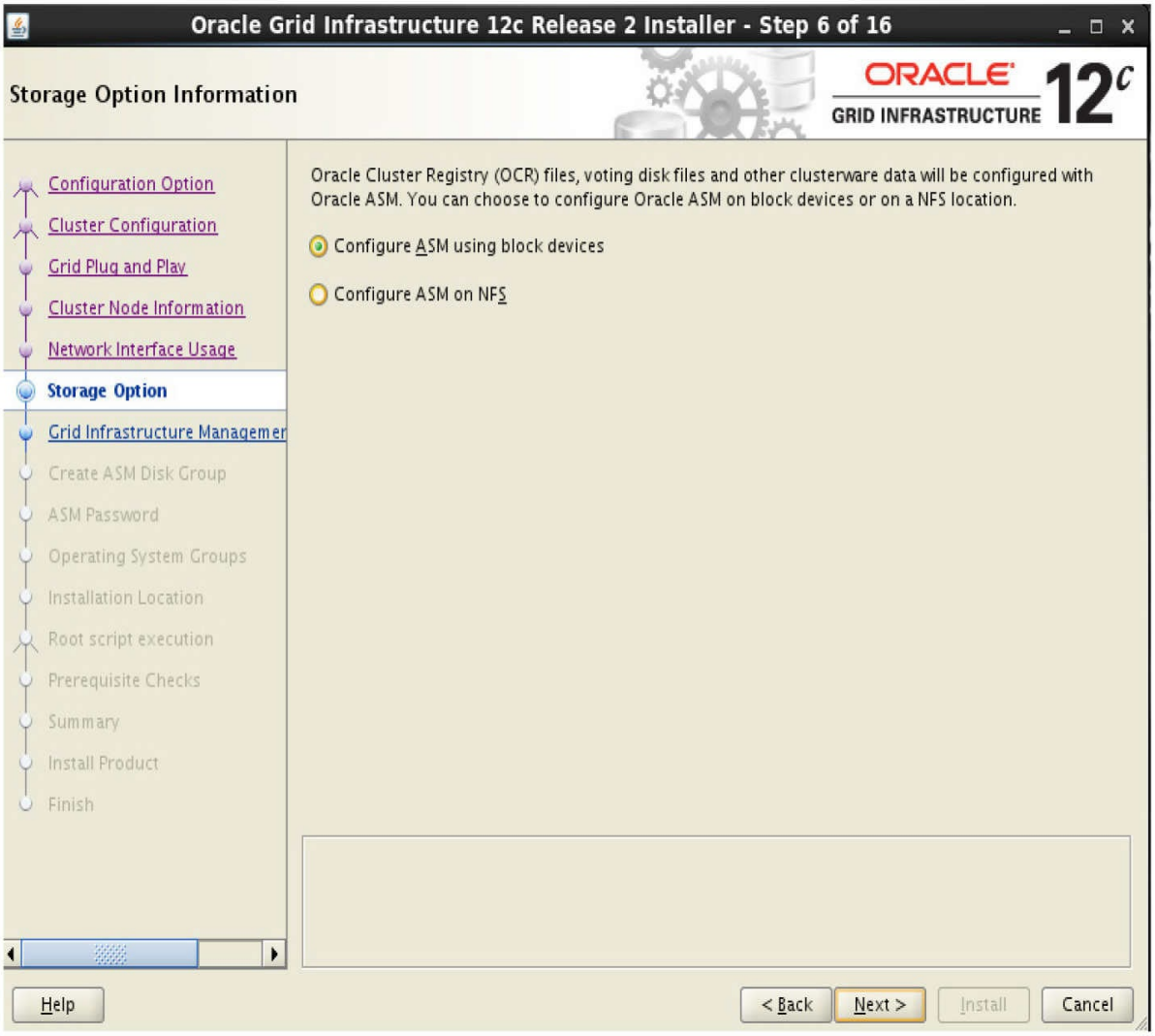


FIGURE 4-10. *The Storage Option Information screen*

https://t.me/learningnets

TIP

Select the cluster name carefully because the only way to change the cluster name after installation is by reinstalling Oracle Grid Infrastructure.

This screen shows that Oracle Universal Installer does not allow you to specify raw devices for shared storage since raw devices are no longer supported with Oracle RAC database. Oracle only supports ASM and cluster file system for shared storage to be used with Oracle RAC. With Oracle 11*g* Release 2, OUI allowed raw devices if you were upgrading from previous releases, but this is no longer allowed and you would be required to move from raw devices to supported storage. If you choose ASM on this screen, Oracle will start ASMCA (Automatic Storage Management Configuration Assistant) in the Create ASM Disk Group screen to configure the ASM before it can store the OCR and voting disk.

Select the option “Configure ASM using block devices” and then click Next.

The next screen, shown in Figure 4-11, is a new addition to Oracle 12*c* that lets you choose whether you wish to create a separate ASM group for the mandatory Grid Infrastructure Management Repository (GIMR) data. We’ll choose not to do so, so click Next to move along the installation procedure.

https://t.me/learningnets

Technet24

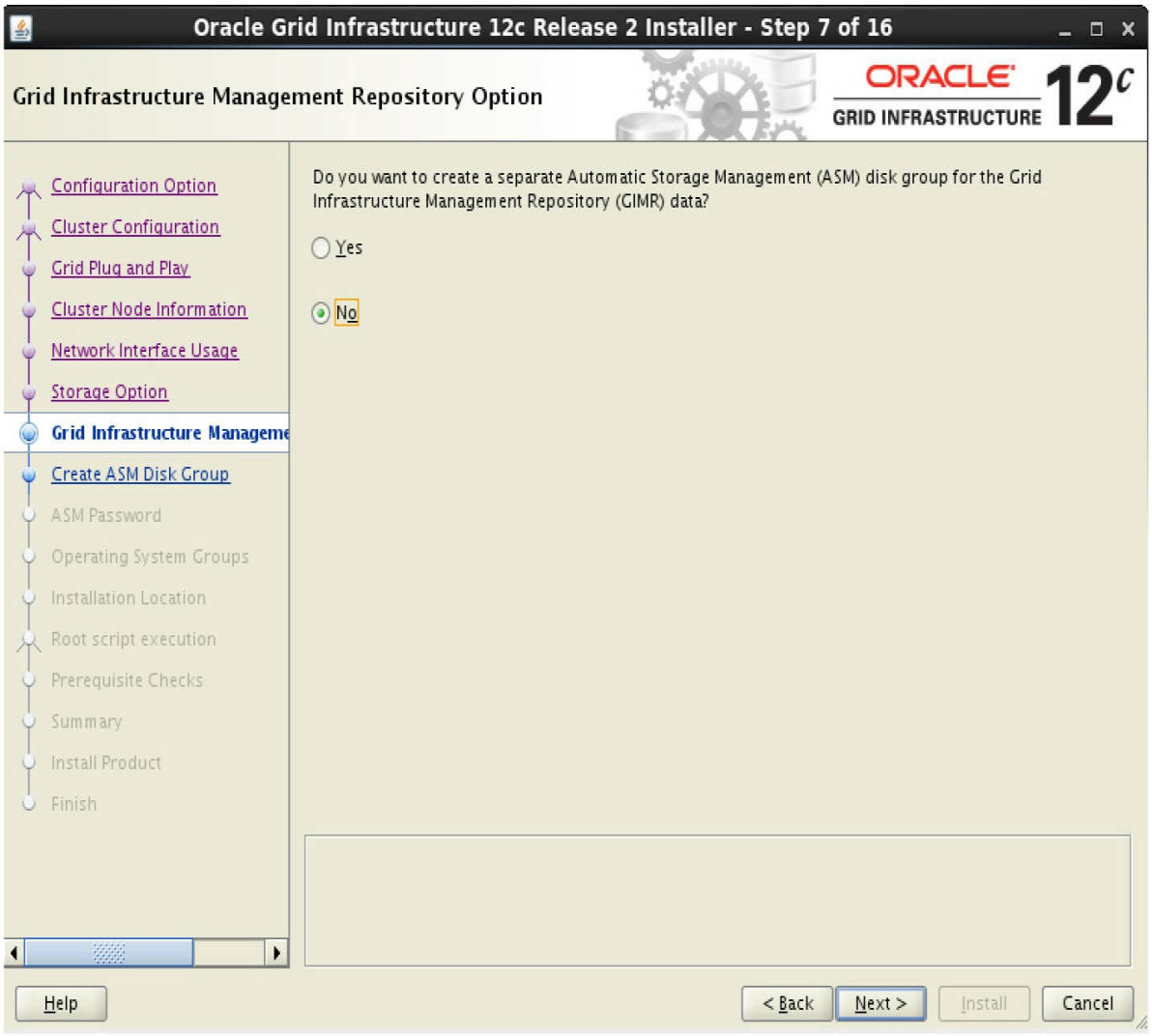


FIGURE 4-11. *Grid Infrastructure Management Repository Option screen*

The OCR contains critical information about the cluster configuration, including the public and private network configuration details. The minimum advised size of the OCR and voting disk is 280MB, because the reconfiguration of the OCR and voting disk may affect the entire cluster, although the actual information in these files is relatively small. The voting disk is used by the Cluster Synchronization Service (CSS) to resolve network splits, commonly referred to as *split brain*. It's used as the final arbiter on the status of configured nodes, either up or down, and to deliver eviction notices. It contains the kill block, which is formatted by the other node during the node eviction.

Oracle allows mirroring of the OCR and voting disks. If you decide to choose mirroring, select Normal Redundancy and provide the location of the mirror device. Otherwise, choose External Redundancy.

The OCR location can be either a clustered file system or ASM. If you use ASM to store the OCR and voting disk, Oracle decides the number of the OCR and voting disks based on the configuration of the ASM disk group being used to store the OCR

https://t.me/learningnets

and voting disks.

The number of voting disks in an ASM disk group is based on the redundancy level of the ASM disk group. Oracle allocates three voting disks in normal redundancy and five voting disks for a high-redundancy ASM disk group.

The OCR file is treated like any other database file in the ASM; hence, no special care is required.

Specify the location(s) of the voting disks and the OCR as Automatic Storage Management on this screen and click Next.

NOTE

Storing the OCR file and voting disks in an ASM disk group created with five ASM disks of 1GB each is sufficient for normal-sized grid environments. Oracle has improved the node fencing starting with Oracle Grid Infrastructure 11.2.0.2 onward as it tries to kill the processes capable of performing I/O operations and stopping the Oracle Grid Infrastructure on the failed node rather than rebooting the failed node as the first thing to stop the failed node issuing any I/O requests to the database files. This is also known as rebootless node fencing.

In the Create ASM Disk Group screen, shown in Figure 4-12, Oracle creates the ASM disk group for storing the OCR and voting disks. Oracle Universal Installer internally executes the ASMCA tool to create and configure the ASM disk group.

https://t.me/learningnets

Technet24

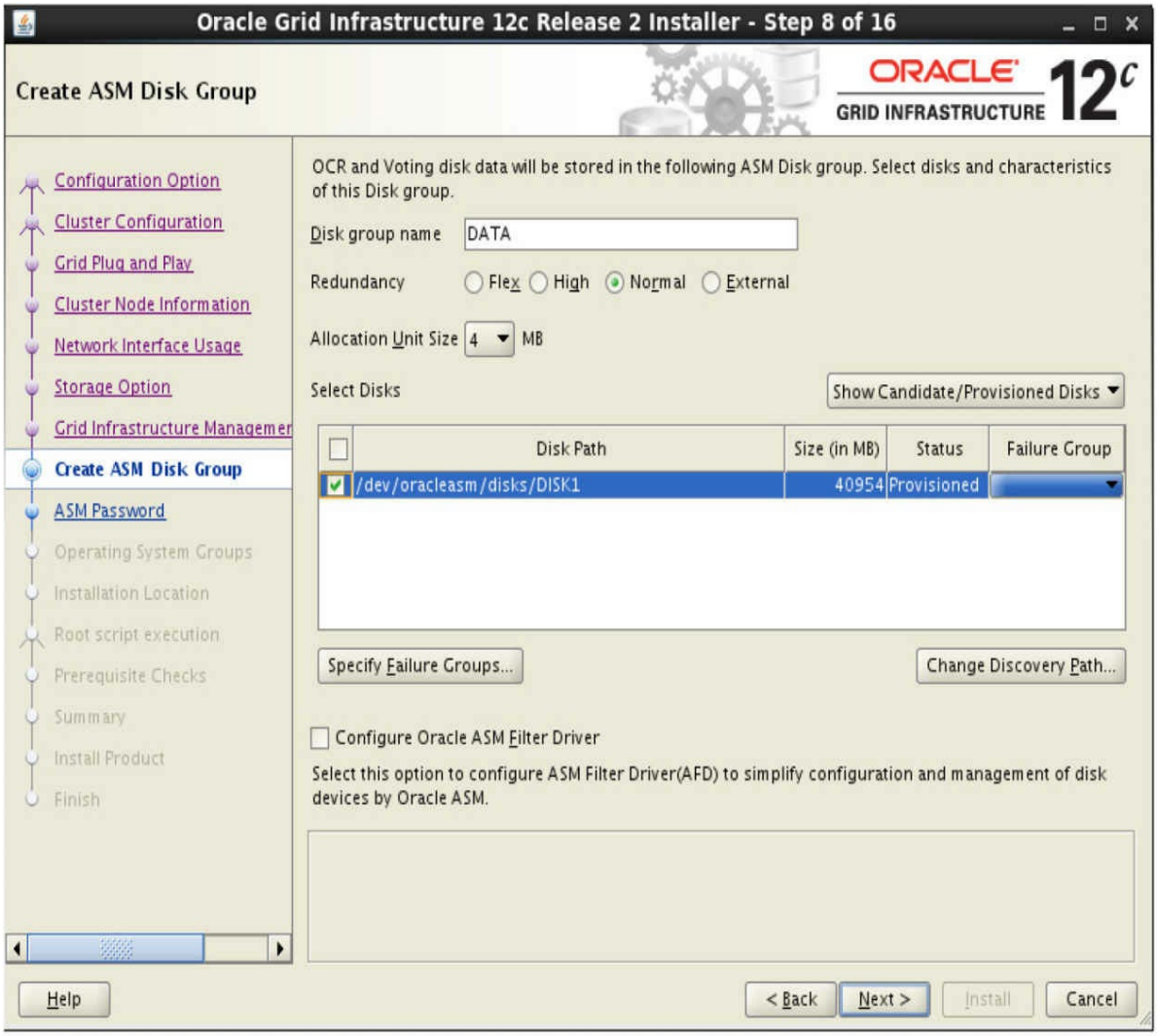


FIGURE 4-12. *The Create ASM Disk Group screen*

When you click the Show Candidate/Provisioned Disks button, the installer lists all the candidate disks that you can use to create an ASM disk group. You can type the name of the ASM disk group, choose the required redundancy for the disk group, and select the candidate disks for the disk group.

Specify DATA as the disk group name, choose external redundancy, select one candidate disk to create the DATA disk group, and then click Next. Before you click the Next button, however, confirm that the Status column shows Provisioned for the disk(s) you've selected.

NOTE

You may need to change the discovery string by clicking the Change Discovery Path button on this screen. Oracle also verifies the minimum number of candidate disks

https://t.me/learningnets

selected to create the ASM disk group for the OCR and voting disks and would not allow you to create the ASM disk group if the number of selected candidate disks is less than three or five for the ASM disk group with normal redundancy or high redundancy, respectively.

In the Specify ASM Password screen, shown in Figure 4-13, you can specify the password for the SYS and ASMSNMP users. We'll select the option "Use same passwords for these accounts" because we're going to use the same password for these users.

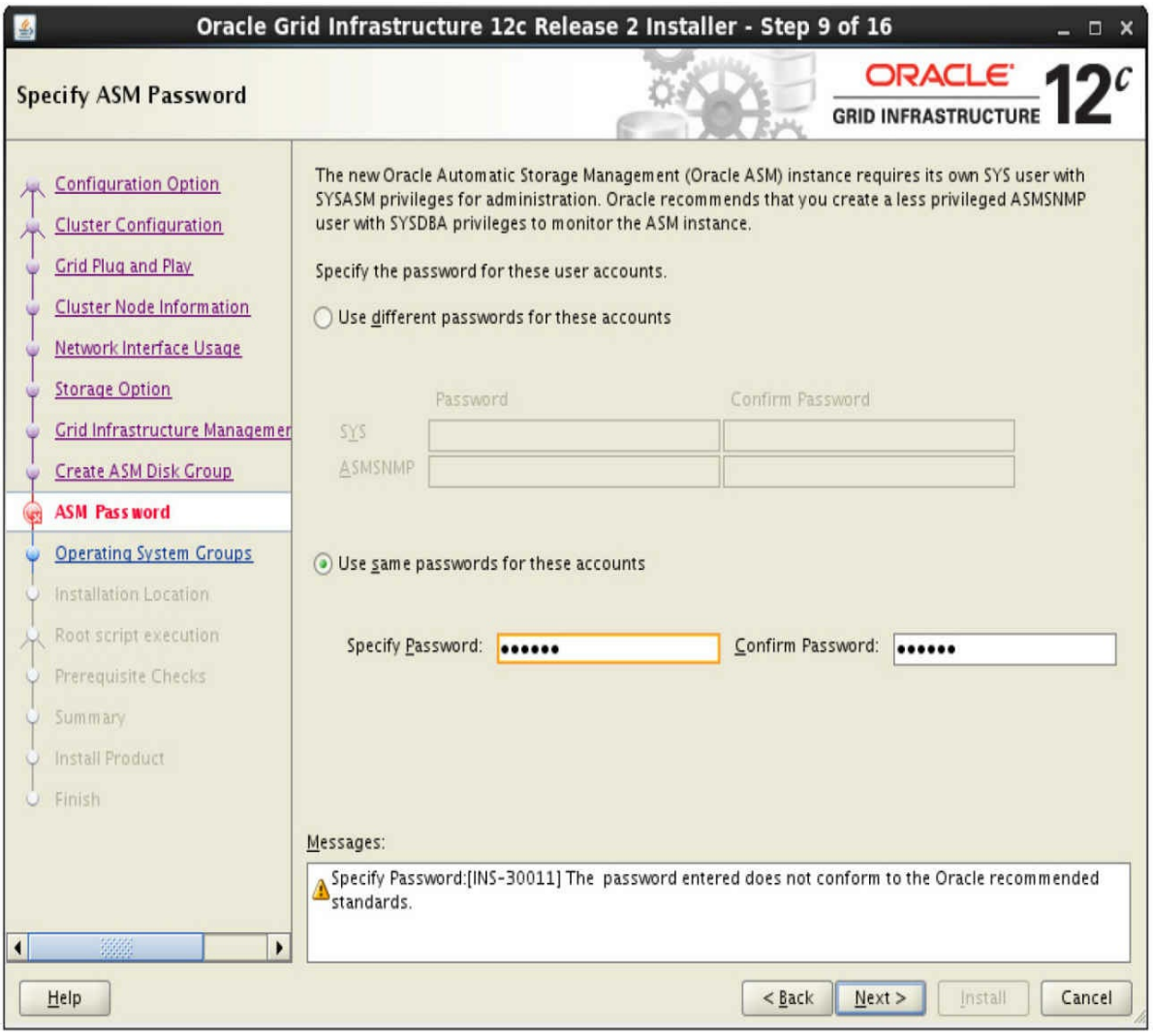


FIGURE 4-13. *The Specify ASM Password screen*

The ASMSNMP user is required to monitor the ASM targets in the Oracle Grid Control, also known as OEM. Enter the password and click Next.

In the Failure Isolation Support screen, shown in Figure 4-14, you can specify the IPMI credentials if you want Oracle to utilize Intelligent Platform Management Interface (IPMI) on the cluster servers to shut them down in the event of a failure. Oracle requires IPMI drivers and a Baseband Management Controller on each cluster

https://t.me/learningnets
Technet24

server to integrate Oracle Grid Infrastructure with IPMI. We won't use IPMI in this installation, so we'll select Do Not Use Intelligent Platform Management Interface (IPMI) on this screen and then click Next.

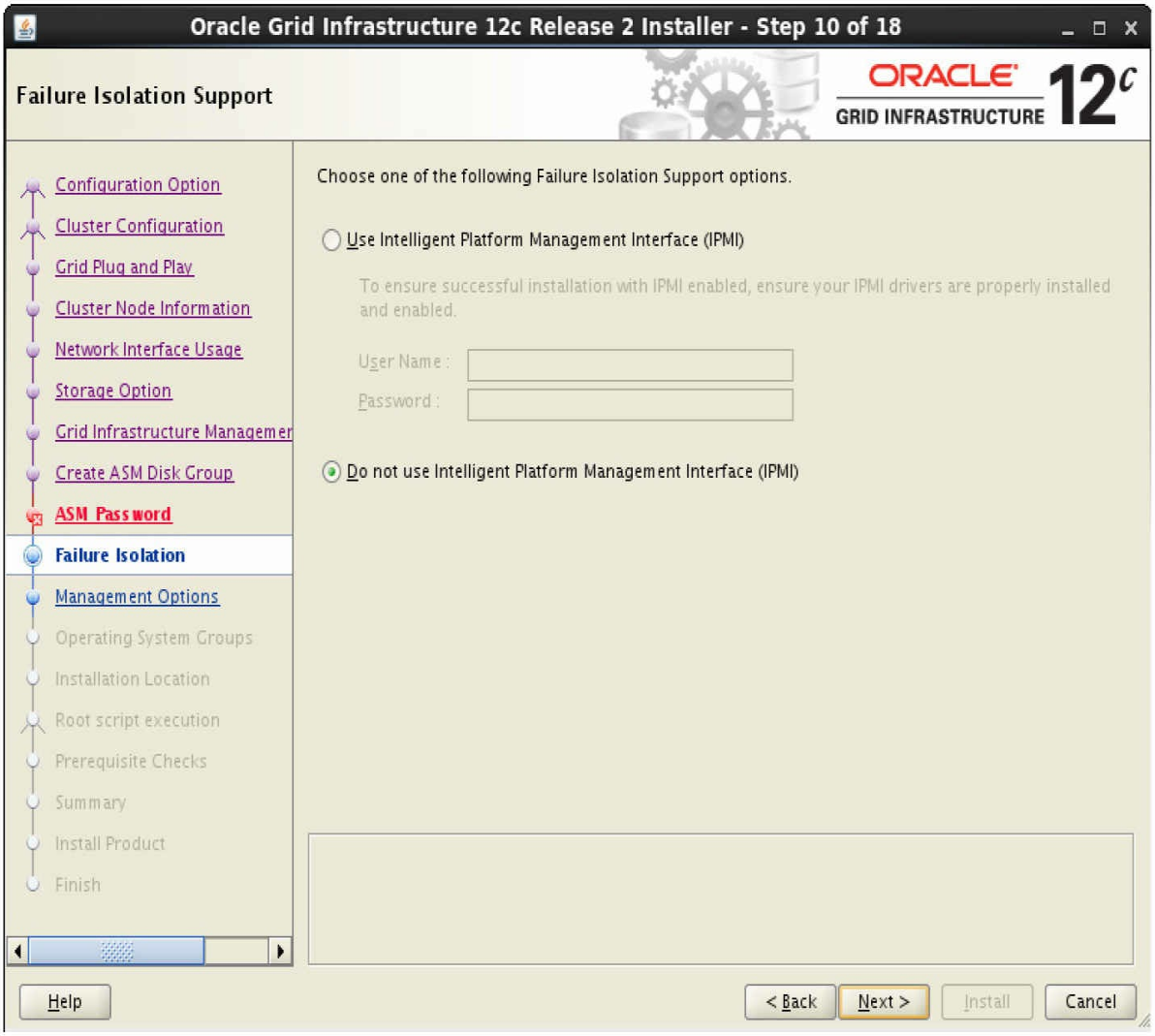


FIGURE 4-14. *The Failure Isolation Support screen*

Specifying Management Options and Privileged OS System Groups

The screen Specify Management Options, shown in Figure 4-15, enables you to configure the management of the Oracle Grid Infrastructure and Oracle ASM by Enterprise Manager Cloud Control. We'll choose not to do this at this juncture, so click Next.

https://t.me/learningnets

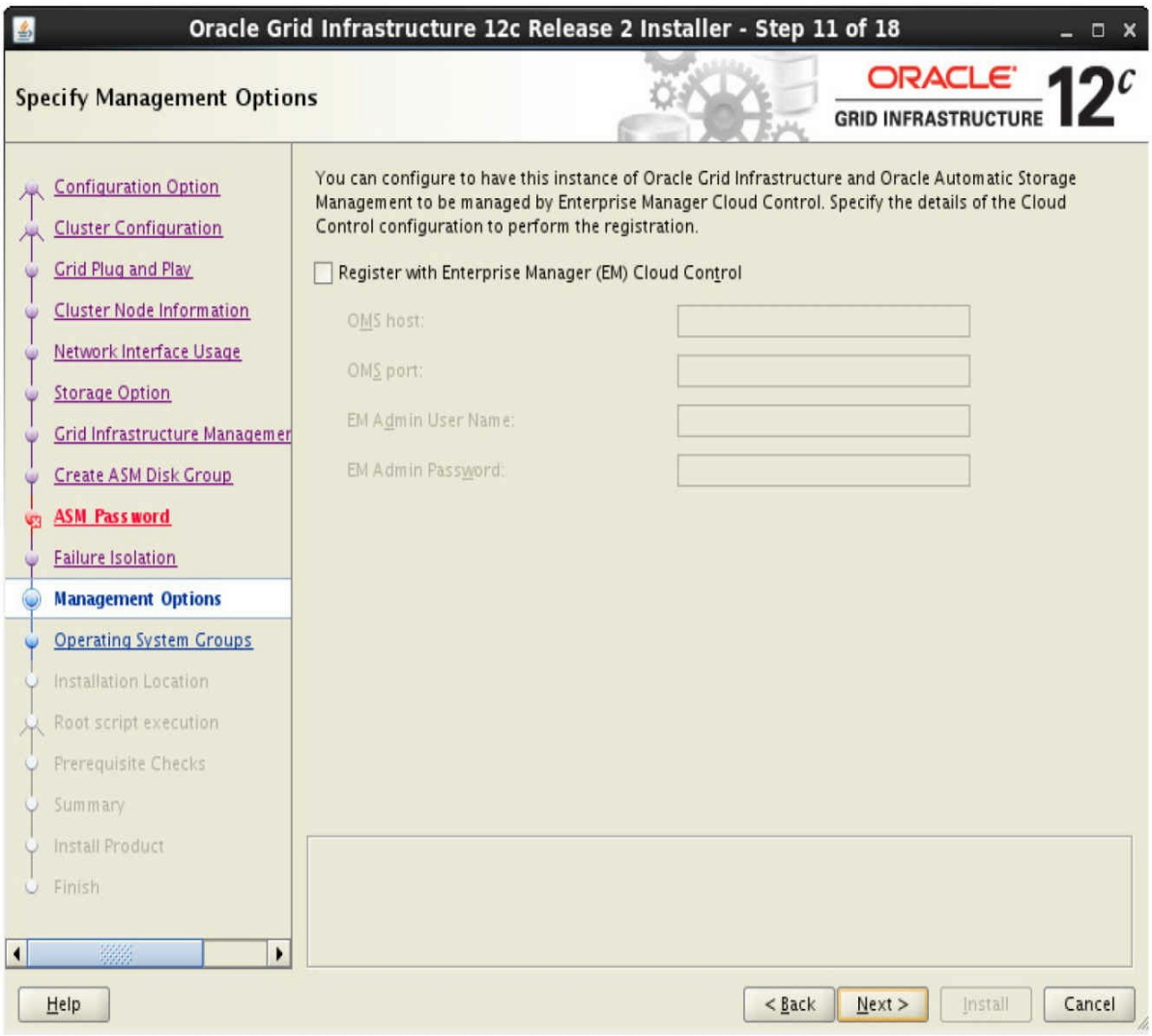


FIGURE 4-15. *Specify Management Options screen*

In the Privileged Operating System Groups screen, shown in Figure 4-16, you can select the operating systems groups to be used for operating system authentication for ASM. You can select asmdba for the OSBDA group, asmoper for the OSOPER group, and asmadmin for the OSASM group on this screen and then click Next (for simplicity, we'll choose dba as the OS group for both the OSASM and OSDBA groups).

https://t.me/learningnets

Technet24

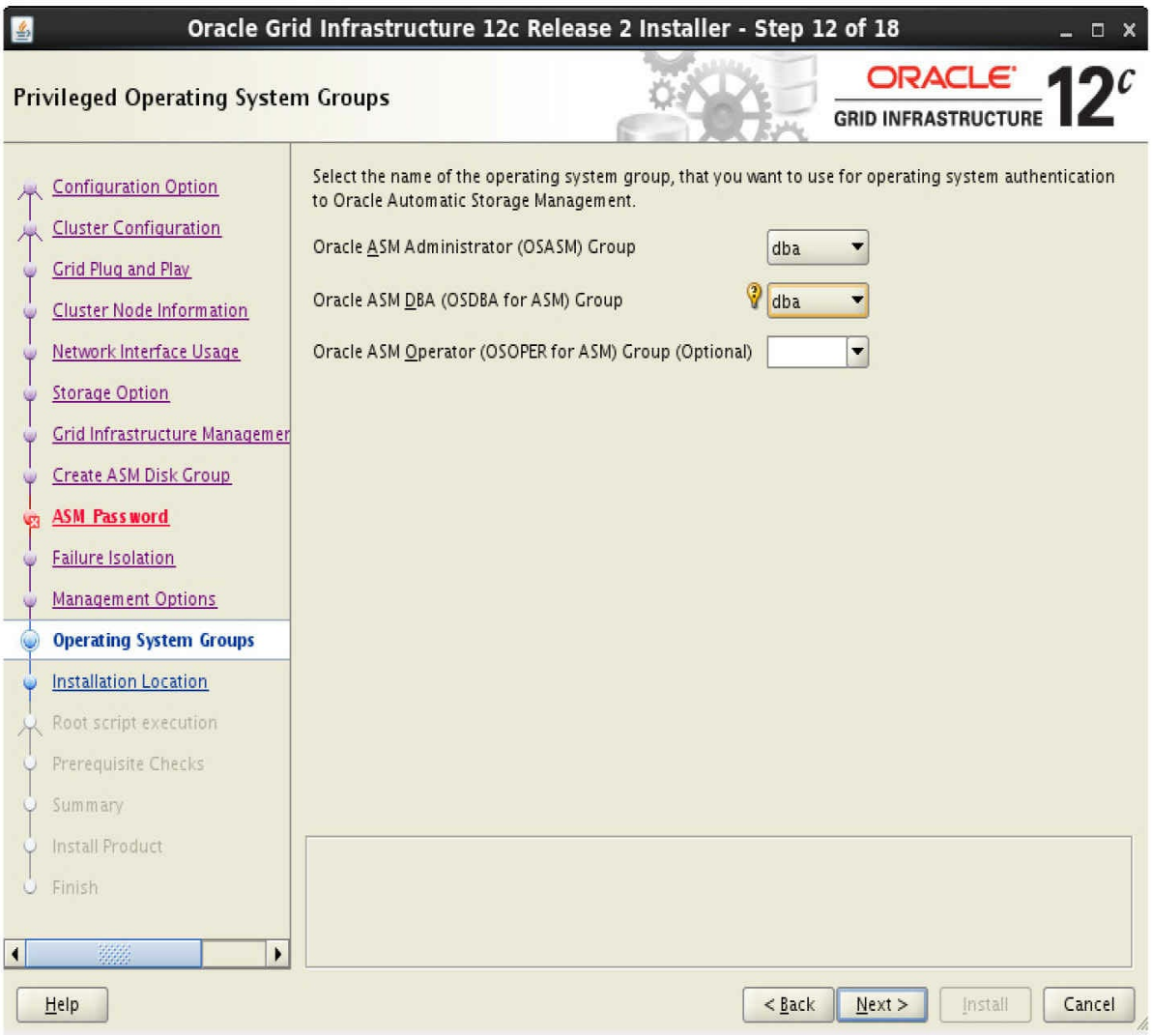


FIGURE 4-16. *The Privileged Operating System Groups screen*

In the Specify Installation Location screen, shown in Figure 4-17, you must specify the storage location for Oracle base and software installation. You should use a separate location for Oracle base for each software owner.

https://t.me/learningnets

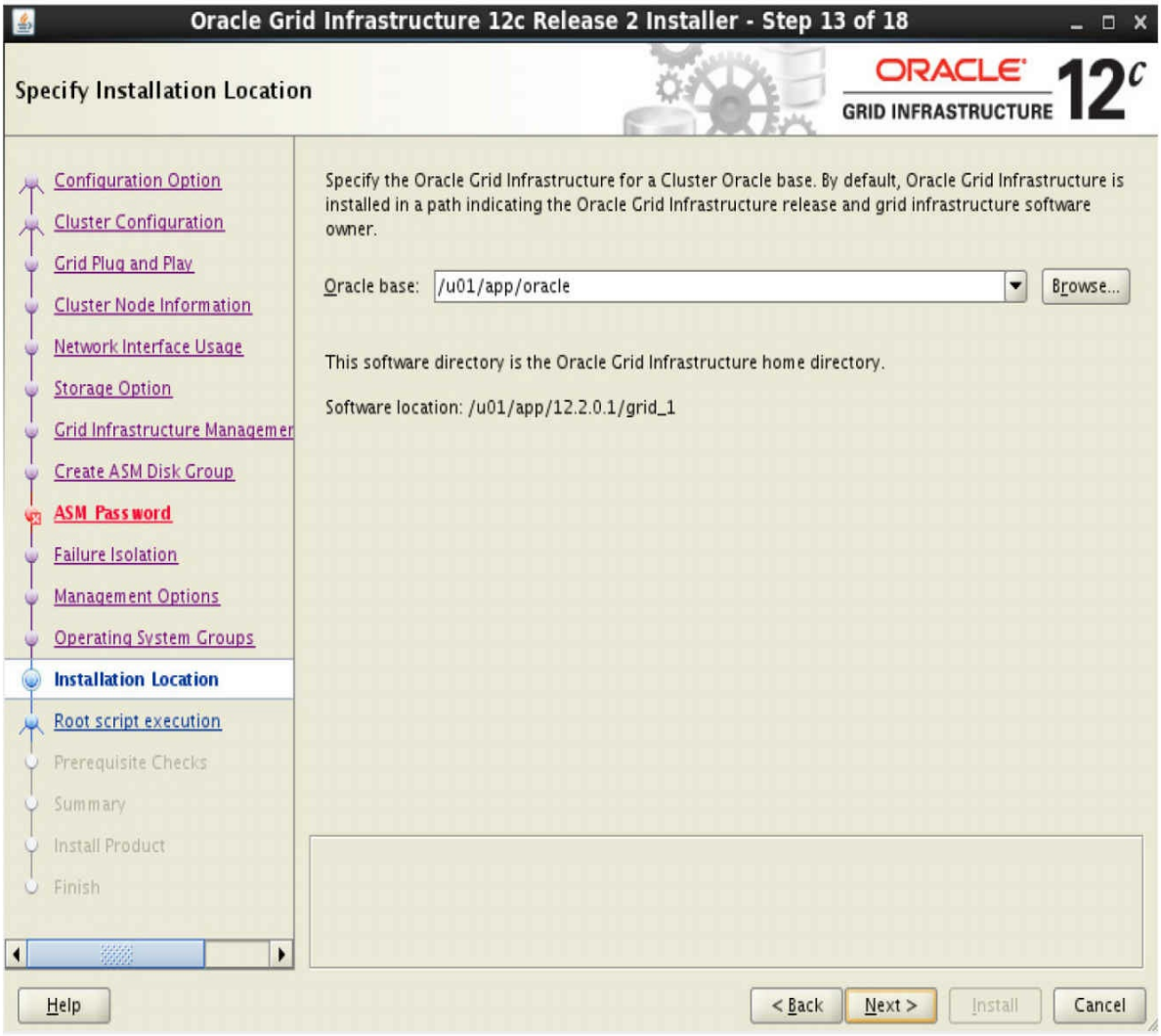


FIGURE 4-17. *The Specify Installation Location screen*

You must ensure that the software location is not inside the path specified for the Oracle base. Specify /u01/app/oracle as the Oracle base location and click Next. The installer shows the software directory that it's going to use as the Oracle Grid Infrastructure home directory: /u01/app/12.2.0.1/grid_1. It's common to specify the name of the Oracle Grid Infrastructure owner ("grid") at the end of the path, but we choose to specify the value "grid_1" instead.

The Create Inventory screen, shown in Figure 4-18, enables you to configure a directory for the installation metadata files, a directory called the "inventory directory" (oraInventory). Members of the oinstall OS system group will have write permissions on this directory. Specify a location such as /u01/app/oraInventory and click Next.

https://t.me/learningnets

Technet24

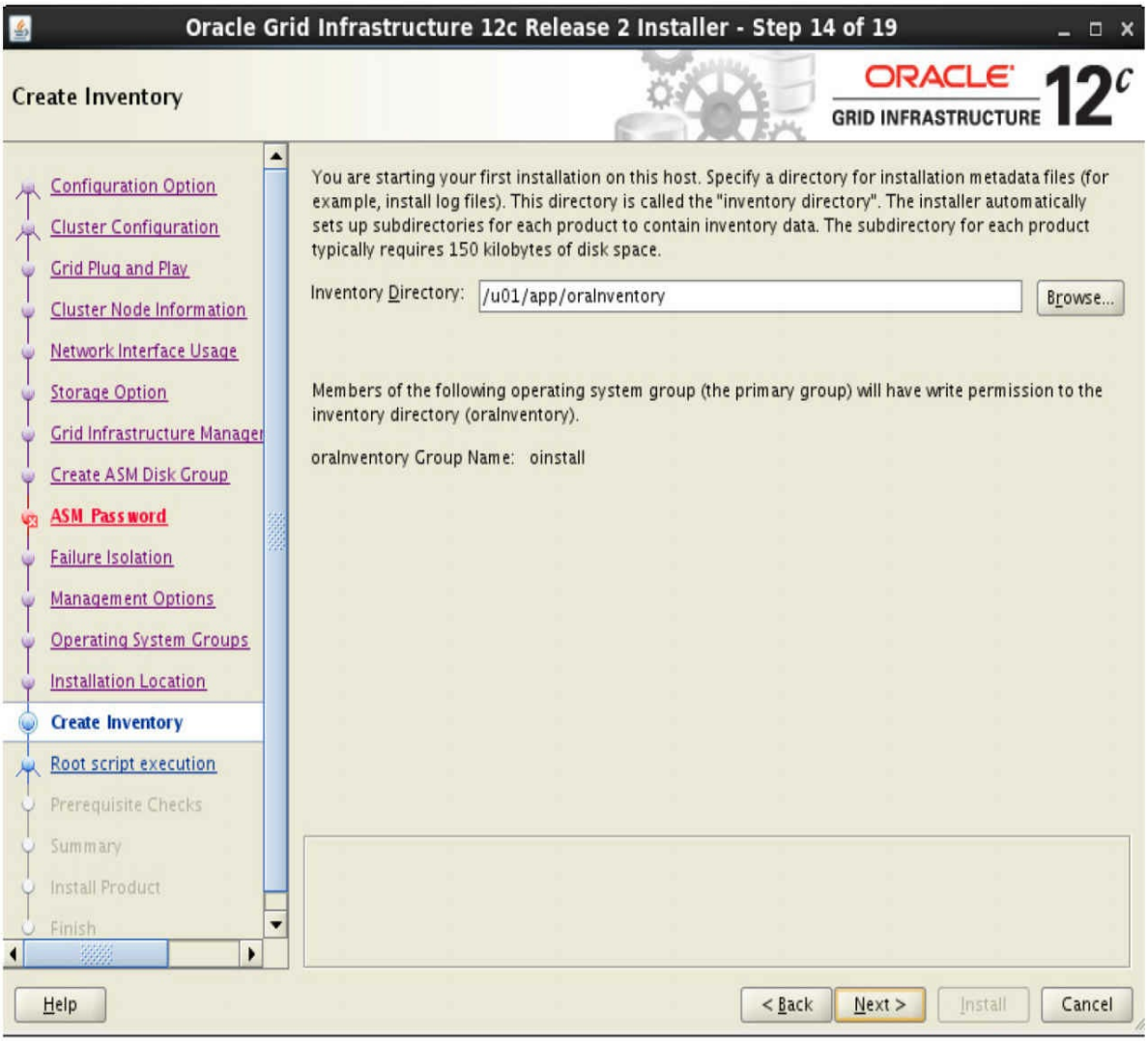


FIGURE 4-18. *The Create Inventory screen*

The Installer will now present you with a new screen (see Figure 4-19). This screen lets you decide if you want the Installer to automatically execute the required root.sh and the rootupgrade.sh scripts on all the nodes in the cluster. This is a useful feature if you have multiple nodes. You can specify the root password or specify the credentials for an OS user with sudo privileges. Note that the root password needs to be the same on all the nodes within the cluster. Although Figure 4-19 shows that the option "Automatically run configuration scripts" is checked, when we performed this install, we chose to manually execute the root scripts.

https://t.me/learningnets

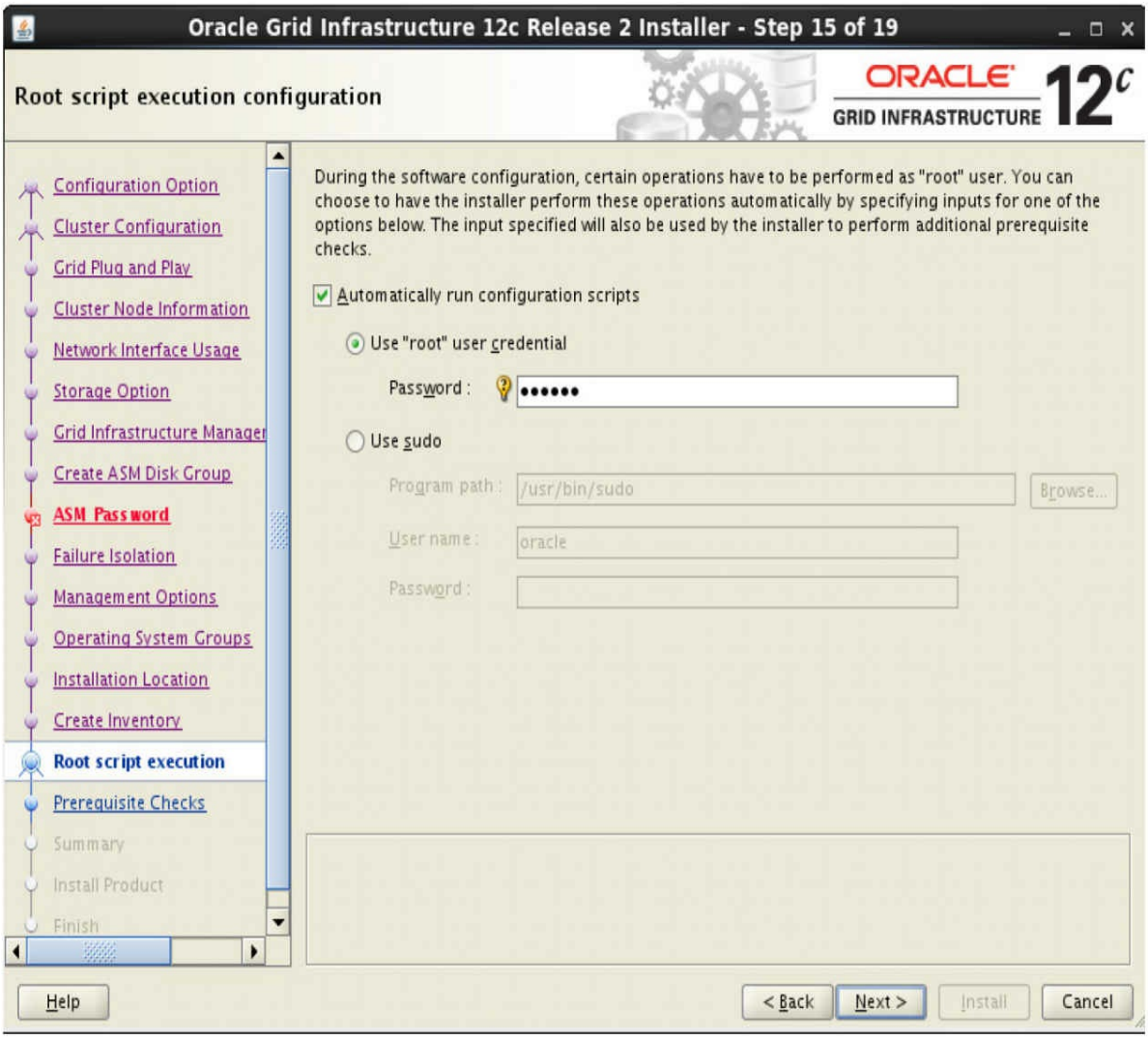


FIGURE 4-19. *Root Script Execution Configuration screen*

Performing the Prerequisite Checks

You'll now see the Perform Prerequisite Checks screen, shown in Figure 4-20, where the Oracle Universal Installer verifies the minimum requirements for installing the Oracle Grid Infrastructure. Oracle Universal Installer internally executes the Cluster Verification Utility to verify the operating system and hardware prerequisites. Based on the results of the verification tests performed by the Cluster Verification Utility, Oracle displays the failed prerequisites on this screen.

https://t.me/learningnets

Technet24

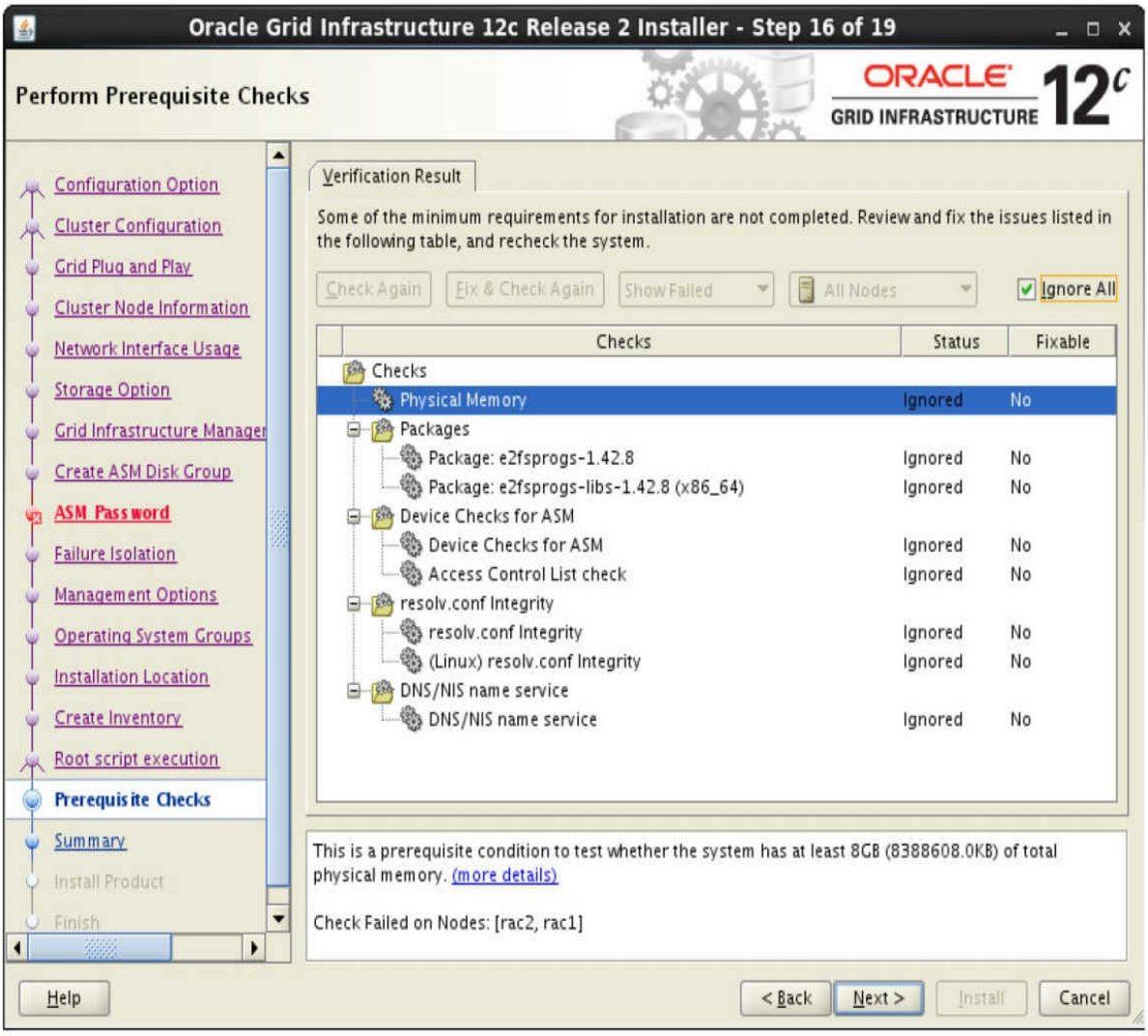


FIGURE 4-20. *The Perform Prerequisite Checks screen*

OUI provides fix-up scripts to repair any failed but fixable prerequisite requirements. Oracle marks each failed prerequisite with a Fixable status of Yes, which is also displayed on the screen. You can select the failed prerequisites with a Fixable status of Yes and then click the Fix & Check Again button on this screen to fix and verify the prerequisites again. Once the Installer verifies the required prerequisites successfully, click Next to continue.

Note that in our case, Figure 4-20 shows some failures, but we decided to move forward without fixing them (by selecting the Ignore All option). Of course, in a production environment, you would need to be very careful before choosing to ignore any checks that failed verification by the Installer.

In the Summary screen, shown in Figure 4-21, Oracle will display the installation information. You should verify this information. If the information displayed is correct, click Install to start the software installation.

https://t.me/learningnets

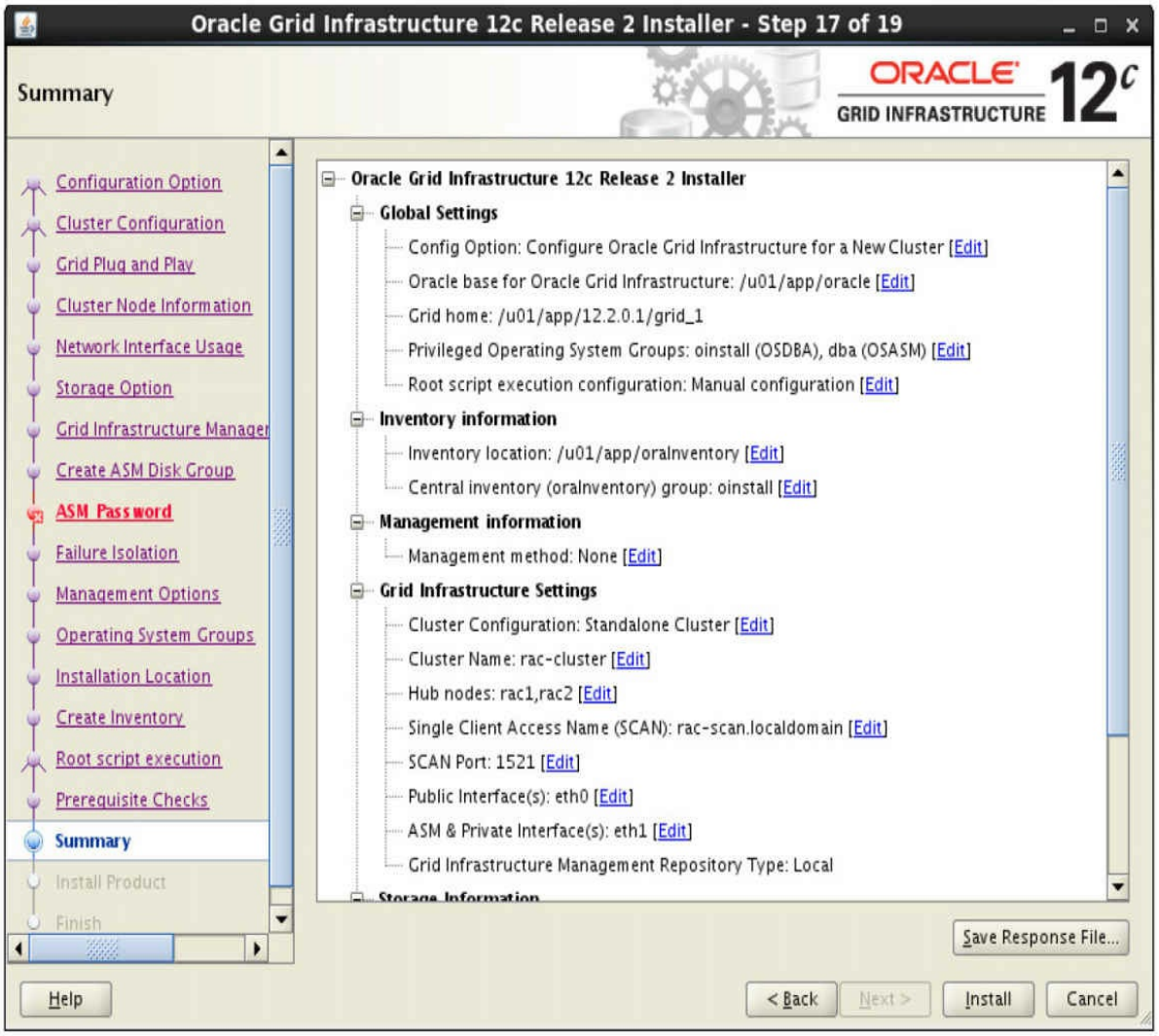


FIGURE 4-21. *The Summary screen*

Installation of the Oracle Grid Infrastructure may take up to 30 minutes, depending on the hardware configuration and the number of nodes in the cluster. The Installer will copy all the required files to the Oracle Grid Infrastructure home directory and link the files with the operating system libraries. Once the installation and linking is done on the local node, the Installer will copy the files to the remote node. During the entire process, you can view the status in the progress bar of the Install Product screen (see Figure 4-22), which shows the percentage completion.

https://t.me/learningnets

Technet24

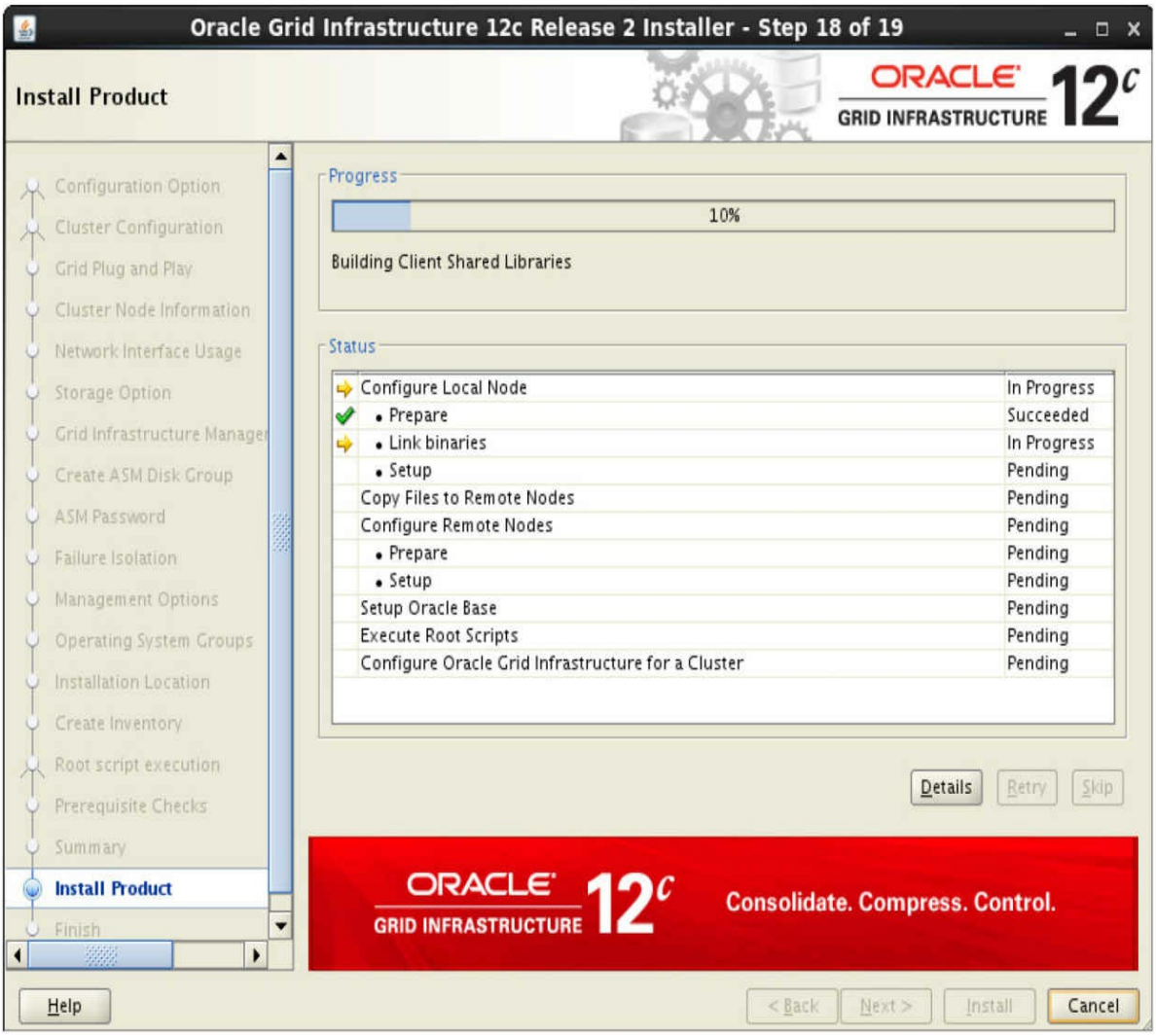


FIGURE 4-22. *Install Product screen showing the Oracle Grid Infrastructure installation progress*

Running the root Scripts

Because we chose the option of manually running the root.sh script on all nodes, after the installation of binaries on all the nodes, a dialog box pops up, as shown in Figure 4-23, prompting you to run the following two scripts as the root user on all the nodes in the cluster:

https://t.me/learningnets

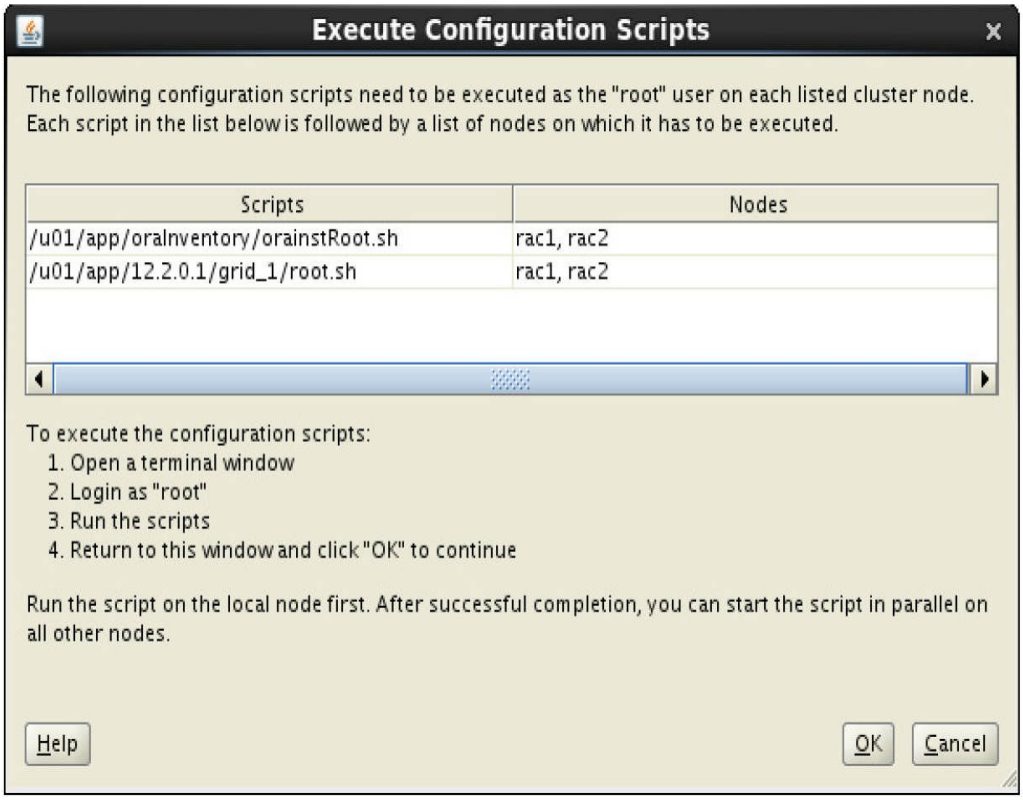


FIGURE 4-23. *The Execute Configuration Scripts screen*

./u01/app/orainventory/orainstRoot.sh
/u01/app/12.2.0/grid_1/root.sh

You must run the scripts first on the local node. After the scripts complete successfully on the local node, you can run the scripts in parallel in all the other nodes. If you attempt to run the scripts in parallel on all nodes, the scripts on the non-local nodes will fail.

The first script, orainstRoot.sh, sets the inventory settings. The second script, root.sh, sets the permission for the files inside the Oracle Grid Infrastructure home, configures the cluster, and starts Oracle Clusterware on the cluster node.

Following is the output for when we ran the orainstRoot.sh command:

```
[root@rac2 grid_1]#
[root@rac1 oraInventory]# ./orainstRoot.sh
Changing permissions of /u01/app/oraInventory.
Adding read,write permissions for group.
Removing read,write,execute permissions for world.

Changing groupname of /u01/app/oraInventory to oinstall.
The execution of the script is complete.
[root@rac1 oraInventory]#
```

The root.sh script internally calls other configuration scripts such as rootmacro.sh,

https://t.me/learningnets
Technet24

rootinstall.sh, and rootcrs.pl. The rootcrs.pl script does the majority of the post-installation configuration. The root.sh script will also set the write permission on the parent directory of the $GRID_HOME to the root user only. This is the reason why it's a good idea to use a separate storage location for the Oracle Grid Infrastructure home.

Running the root.sh script on the first cluster node, as shown next, will start the Oracle Grid Infrastructure daemons and ASM as well as mount the DATA disk group. Successful execution of the root.sh script is essential for successfully completing the installation of the Oracle Grid Infrastructure.

```
[root@rac1 oraInventory]# /u01/app/12.2.0.1/grid_1/root.sh
Performing root user operation.

The following environment variables are set as:
    ORACLE_OWNER= oracle
    ORACLE_HOME=  /u01/app/12.2.0.1/grid_1

Enter the full pathname of the local bin directory: [/usr/local/bin]:
   Copying dbhome to /usr/local/bin ...
   Copying oraenv to /usr/local/bin ...
   Copying Creating /etc/oratab file...
Using configuration parameter file: /u01/app/12.2.0.1/grid_1/crs/install/crsconfig_params
The log of current session can be found at:
  /u01/app/oracle/crsdata/rac1/crsconfig/rootcrs_rac1_2017-04-16_01-46-02PM.log
2017/04/16 13:48:47 CLSRSC-363: User ignored prerequisites during installation
2017/04/16 13:48:47 CLSRSC-594: Executing installation step 3 of 19:
'CheckFirstNode'.
'CreateOHASD'.2017/04/16 13:50:18 CLSRSC-594: Executing installation step 12 of
19: 'ConfigOHASD'.
CRS-2791: Starting shutdown of Oracle High Availability Services-managed resources
on 'rac1'
CRS-2793: Shutdown of Oracle High Availability Services-managed resources on
'rac1' has completed
CRS-4133: Oracle High Availability Services has been stopped.

CRS-4123: Oracle High Availability Services has been started.
CRS-2672: Attempting to start 'ora.crsd' on 'rac1'
CRS-2676: Start of 'ora.crsd' on 'rac1' succeeded
CRS-4266: Voting file(s) successfully replaced
##  STATE    File Universal Id                File Name Disk group
--  -----    -----------------                --------- ---------
 1. ONLINE   a2edb1b05c394ff9bf67311128832ffb (/dev/oracleasm/disks/DISK1) [DATA]
Located 1 voting disk(s).
CRS-2672: Attempting to start 'ora.storage' on 'rac1'
2017/04/16 13:57:28 CLSRSC-343: Successfully started Oracle Clusterware stack
2017/04/16 13:57:28 CLSRSC-594: Executing installation step 18 of 19: 'ConfigNode'.
```

https://t.me/learningnets

```
CRS-2672: Attempting to start 'ora.ASMNET1LSNR_ASM.lsnr' on 'rac1'
CRS-2676: Start of 'ora.ASMNET1LSNR_ASM.lsnr' on 'rac1' succeeded
CRS-2672: Attempting to start 'ora.asm' on 'rac1'
CRS-2676: Start of 'ora.asm' on 'rac1' succeeded

2017/04/16 14:02:58 CLSRSC-594: Executing installation step 19 of 19: 'PostConfig'.
2017/04/16 14:05:59 CLSRSC-325: Configure Oracle Grid Infrastructure
for a Cluster ... succeeded
[root@rac1 oraInventory]#
```

You can exit the Installer and look for the installActions.log file in the oraInventory home. If you enabled Installer tracing, you also get more information on the Installer actions with finer details. If you have selected to configure Grid Infrastructure Repository, it will be configured when root.sh is executed on the last node.

The orainstRoot.sh script creates the /etc/oracle directory and the oratab file entries. These entries are used by the Oracle Grid Infrastructure agents, which are executed during startup and shutdown of the operating system.

Open a terminal window, log in as root, and run the two root scripts. Once the two scripts finish running on all the non-local nodes, come back to the Execute Configuration Scripts dialog box shown in Figure 4-23, and click OK to continue.

The Product Installation

You can click the Details button in the Install Products screen to view the latest entries in the installation log file, as shown in Figure 4-24. Alternatively, you can tail the log file for the installation.

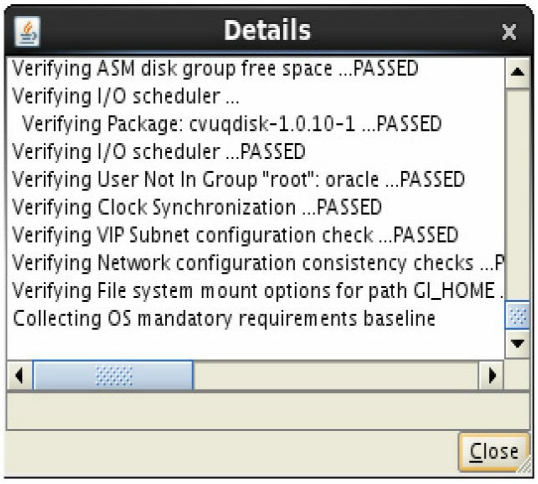


FIGURE 4-24. *The Details window*

Once all the Clusterware configuration is complete, click Next on the Install Product screen, shown in Figure 4-25.

https://t.me/learningnets
Technet24

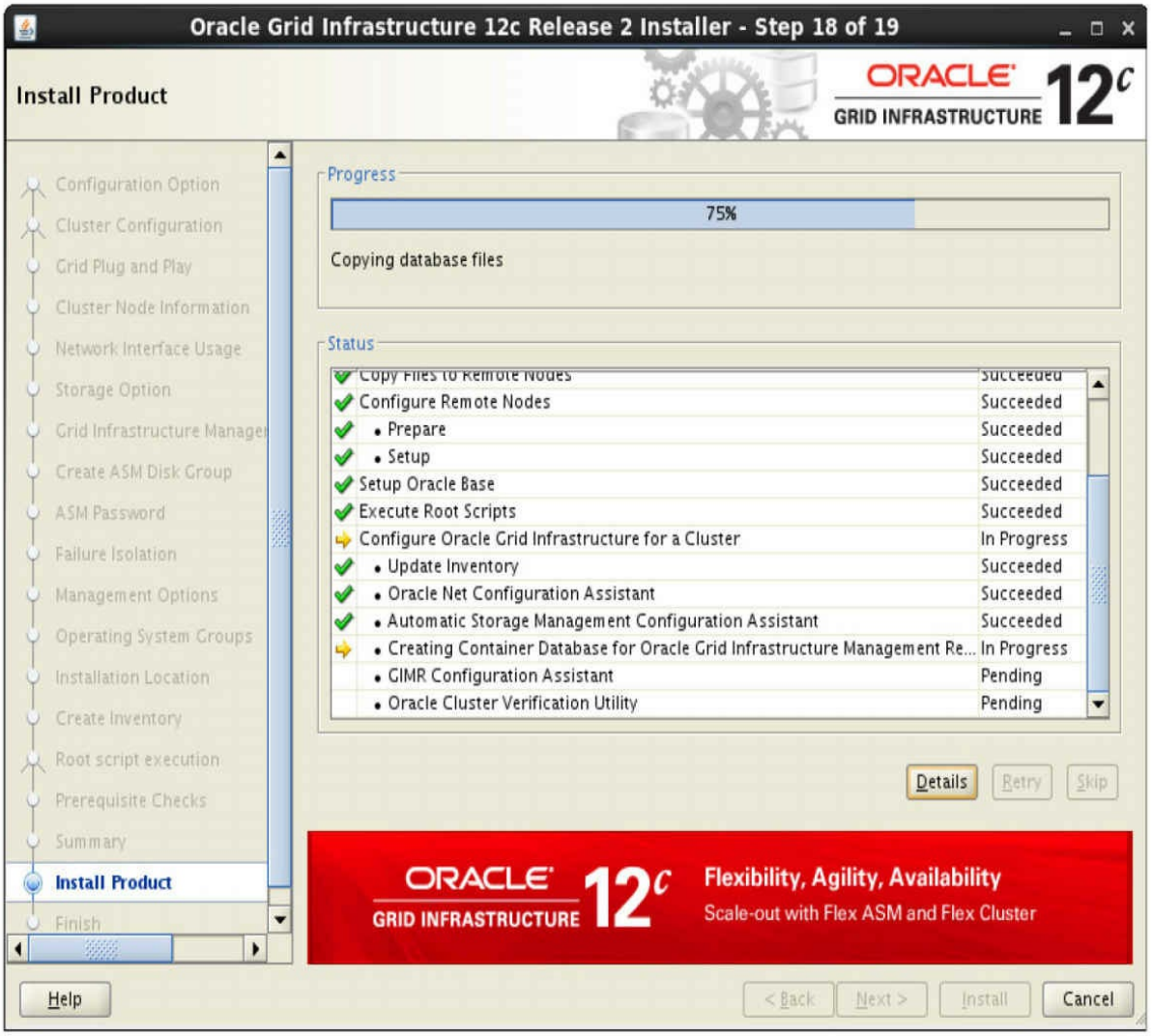


FIGURE 4-25. *The Install Product screen*

The Installer will show the Finish screen (see Figure 4-26). This screen shows the configuration of Oracle Grid Infrastructure was successful. In our case, because we chose to ignore certain minor prerequisites, it states that “some configuration assistants failed, were cancelled or skipped.” However, because we know exactly what this is referring to, there’s no need to worry. Click the Close button to end the installation of Oracle Grid Infrastructure.

https://t.me/learningnets

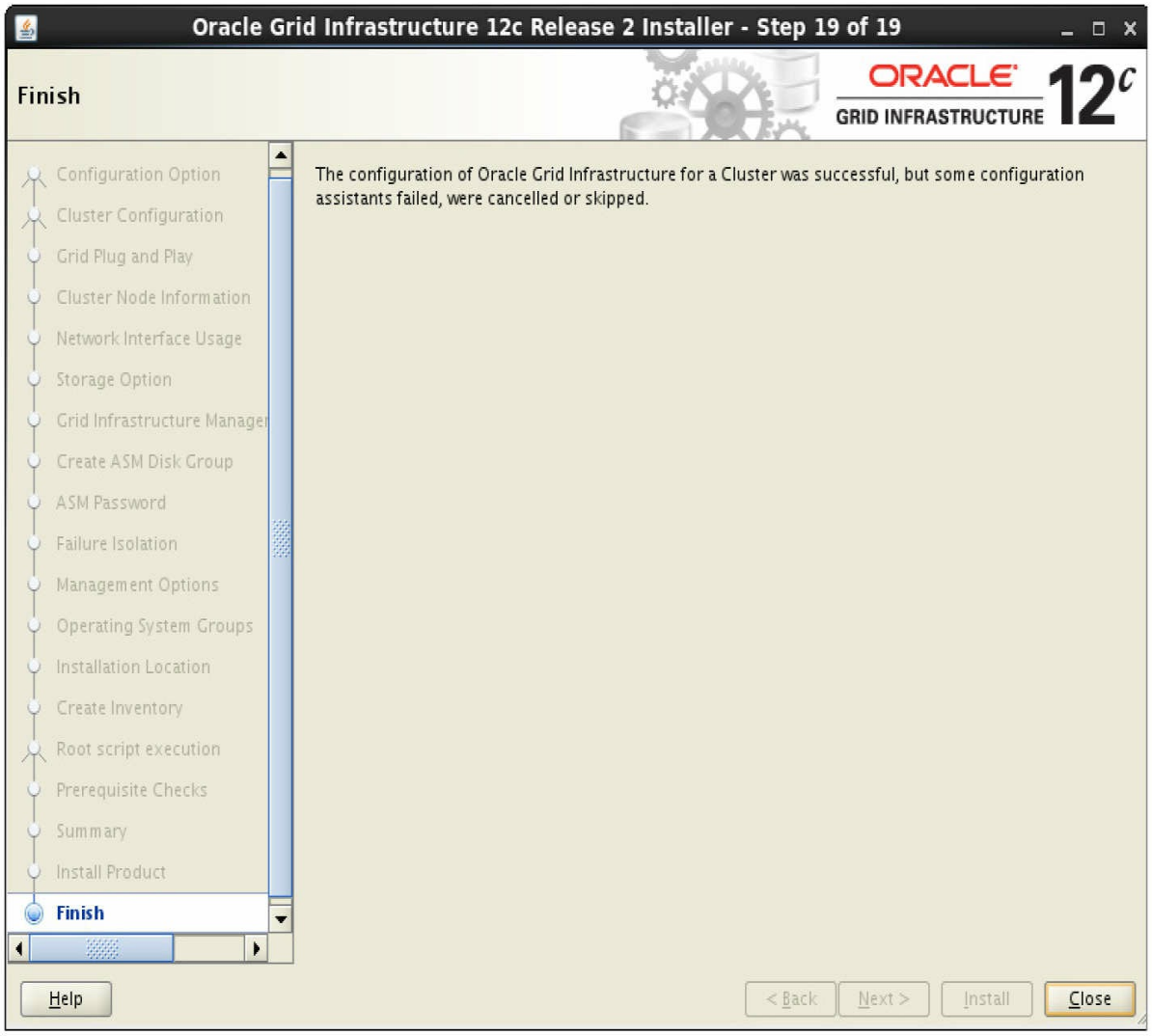


FIGURE 4-26. *End of installation*

Changes in root.sh Script Starting Oracle Grid Infrastructure 11.2.0.2

You don’t need to deconfigure the cluster nodes prior to restarting the root.sh script on a failed cluster node. You can simply restart the installation from the point where the script had failed on a node—that is, once you fix the issue that caused the failure. Oracle internally records the checkpoint of the root.sh execution and uses this to know the starting point of the root.sh script execution when it is restarted.

Oracle has also enabled the ability to execute the root.sh script in parallel on the cluster nodes, except for the first and last nodes in the cluster.

Verifying the Oracle Grid Infrastructure

https://t.me/learningnets

Technet24

Installation

After the installation, exit the Installer. Optionally, you can use the Cluster Verification Utility (runcluvfy.sh), as shown next, to verify the Oracle Grid Infrastructure installation. Actually, this step is automated, because the Installer runs the Cluster Verification Utility at both the start and the end of the install.

```
[oracle@rac1 grid_1]$ ./runcluvfy.sh stage -post crsinst -n rac1,rac2
Verifying Node Connectivity ...
  Verifying Hosts File ...PASSED
…
CVU operation performed:      stage -post crsinst
Date:                         Apr 25, 2017 5:58:49 PM
CVU home:                     /u01/app/12.2.0.1/grid_1/
User:                         oracle
[oracle@rac1 grid_1]$
```

The other way to verify whether the Oracle Grid Infrastructure is installed successfully is to display the cluster resources status. You should see that the cluster resources are started and that their status is ONLINE. Here’s how you do this with the help of the CRSCTL utility:

https://t.me/learningnets

```
[oracle@rac1 bin]$ ./crsctl status resource -t
--------------------------------------------------------------------------------
Name           Target  State        Server                   State details
--------------------------------------------------------------------------------
Local Resources
--------------------------------------------------------------------------------
ora.ASMNET1LSNR_ASM.lsnr
               ONLINE  ONLINE       rac1                     STABLE
               ONLINE  ONLINE       rac2                     STABLE
ora.DATA.dg
               ONLINE  ONLINE       rac1                     STABLE
               ONLINE  ONLINE       rac2                     STABLE
ora.LISTENER.lsnr
               ONLINE  ONLINE       rac1                     STABLE
               ONLINE  ONLINE       rac2                     STABLE
ora.chad
               ONLINE  ONLINE       rac1                     STABLE
               ONLINE  ONLINE       rac2                     STABLE
ora.net1.network
               ONLINE  ONLINE       rac1                     STABLE
               ONLINE  ONLINE       rac2                     STABLE
ora.ons
               ONLINE  ONLINE       rac1                     STABLE
               ONLINE  ONLINE       rac2                     STABLE
ora.proxy_advm
               OFFLINE OFFLINE      rac1                     STABLE
               OFFLINE OFFLINE      rac2                     STABLE
```

https://t.me/learningnets

```
--------------------------------------------------------------------------------
Cluster Resources
--------------------------------------------------------------------------------
ora.LISTENER_SCAN1.lsnr
      1        ONLINE  ONLINE       rac1                     STABLE
ora.LISTENER_SCAN2.lsnr
      1        ONLINE  ONLINE       rac2                     STABLE
ora.LISTENER_SCAN3.lsnr
      1        ONLINE  ONLINE       rac2                     STABLE
ora.MGMTLSNR
      1        ONLINE  ONLINE       rac2                     169.254.239.38 192.1
                                                             68.10.2,STABLE
ora.asm
      1        ONLINE  ONLINE       rac1                     Started,STABLE
      2        ONLINE  ONLINE       rac2                     Started,STABLE
      3        OFFLINE OFFLINE                               STABLE
ora.cvu
      1        ONLINE  ONLINE       rac2                     STABLE
ora.mgmtdb
      1        ONLINE  ONLINE       rac2                     Open,STABLE
ora.orcl.db
      1        ONLINE  ONLINE       rac1                     Open,HOME=/u01/app/o
                                                             racle/product/12.2.0
                                                             /dbhome_1,STABLE
      2        ONLINE  ONLINE       rac2                     Open,HOME=/u01/app/o
                                                             racle/product/12.2.0
                                                             /dbhome_1,STABLE
ora.qosmserver
      1        ONLINE  ONLINE       rac2                     STABLE
ora.rac1.vip
      1        ONLINE  ONLINE       rac1                     STABLE
ora.rac2.vip
      1        ONLINE  ONLINE       rac2                     STABLE
ora.scan1.vip
      1        ONLINE  ONLINE       rac1                     STABLE
ora.scan2.vip
      1        ONLINE  ONLINE       rac2                     STABLE
ora.scan3.vip
      1        ONLINE  ONLINE       rac2                     STABLE
--------------------------------------------------------------------------------
[oracle@rac1 bin]$
```

Before we wrap up this chapter on the installation of the Grid Infrastructure, it's good to note that starting with Oracle Database 12*c* Release 2 (12.2), you can run the installer with the **-executeConfigTools** option to perform post-installation configuration of Oracle products. This is true of the Oracle Database installation as well, which is the topic of the following chapter.

https://t.me/learningnets

Summary

Oracle Grid Infrastructure is the foundation of the Oracle Real Application Clusters deployment, and the installation process is quite simple and straightforward when the underlying storage and network components are configured appropriately. Oracle Grid Infrastructure is a software layer that tightly integrates with the operating system to provide clustering and high availability features to a RAC database on a set of servers. You enable high availability of the infrastructure by configuring redundant components at the network and storage layers.

The Oracle-provided Cluster Verification Utility helps a lot in preparing and validating the Oracle Grid Infrastructure and provides a lot of options to validate the installation process at various stages. The Installer also provides fix-up scripts to help in reconfiguring the kernel parameters and the required settings, thus eliminating mundane tasks and human errors in the preparation of the cluster nodes during the installation.

Starting with Oracle Grid Infrastructure 11*g* Release 2, Oracle has made significant improvements in the installation and configuration with the new Grid Plug and Play, the Typical Installation type, and restartable root scripts. Oracle Universal Installer performs the various prerequisite checks and provides a fix-up script for fixable issues, which makes Oracle Grid Infrastructure installation even easier for the end user.

The automation of the root.sh or rootupgrade.sh script on multiple nodes without manual intervention is a great feature if you are installing Oracle RAC on more than two nodes.

In the next chapter, we'll be installing the Oracle 12*c* Release 2 (12.2.0.1) Database with the Oracle RAC option.

https://t.me/learningnets

Technet24

CHAPTER 5

Installing Oracle RAC and Creating an Oracle RAC Database

In Chapter 4, you learned how to install Oracle Grid Infrastructure, which serves as the foundation for Oracle RAC. In this chapter, we review the Oracle Database 12*c* Release 2 (12.2.0.1) RAC installation process. Installing the Oracle RAC database software is the next logical step after installing Oracle Grid Infrastructure, and creating an Oracle 12.2 release database is your last step in the process.

As we've already completed most of the required groundwork for the database installation as part of the preparation for the installation of Oracle Grid Infrastructure in the previous chapter, our work is simple in this chapter. Installing the Oracle RAC software is quite similar to performing a single-instance Oracle Database software installation. Internally, the Oracle Universal Installer (OUI) installs the binaries on a single node, and it uses the underlying file-transfer mechanisms to propagate the files to other nodes, relinking them with the respective operating system binaries.

If you've previously worked with Oracle 11*g* Release 2, you might have noticed that Oracle has started delivering patch sets in the form of full-blown software instead of mere updates. In other words, they're full releases. This facilitates out-of-place database upgrades and also avoids you having to install Oracle base release binaries

https://t.me/learningnets

and then upgrade to the latest patch set. If you're planning to upgrade your databases that are running on previous releases, you'd be installing Oracle 12*c* RAC binaries into a different Oracle home. Installing Oracle 12*c* RAC binaries into pre-existing Oracle homes is not recommended, although it's possible to do so.

Before you perform the Oracle 12*c* Release 1 RAC installation, make sure you have an operational Oracle Grid Infrastructure stack with the same or higher version.

Oracle RAC One Node is a single-instance Oracle RAC database that runs on a single machine preconfigured with Oracle Grid Infrastructure, which provides a kind of active/passive functionality. With RAC One Node, a single-instance RAC database runs on a single node in a cluster at any point in time.

Oracle RAC One Node

Oracle RAC One Node is a single-instance Oracle RAC database running on a single node preconfigured with Oracle Grid Infrastructure. By utilizing the clustering technology, the Oracle RAC One Node database allows on-demand migration of database instances to other servers, conversion to Oracle RAC without any downtime, and rolling patches for single-instance databases. Oracle RAC One Node also provides high availability for single-instance databases. Oracle provides the OMOTION utility to migrate Oracle RAC One Node database instances to other servers in the cluster online without any downtime.

Oracle internally uses transaction shutdown to migrate single instances online without impacting the current transactions, but Oracle ensures that two servers do not provide the same services at the same time. This is really a great feature because it allows large organizations to consolidate smaller single-instance databases in one place and at the same time allows them to standardize the deployment of Oracle databases within the organization with the option to increase the scalability of single-instance databases by upgrading them to Oracle RAC databases without any downtime.

Although not required, you can use the Cluster Verification Utility (cluvfy) for the pre–database installation check. You can run the Cluster Verification Utility in preinstallation mode to verify the basic node reachability and integrity of Oracle Clusterware. This also checks the basic kernel parameters and required operating system libraries. At the end, the utility checks the status of the Oracle Clusterware daemons and the network infrastructure issues. To confirm the install readiness on the hardware, run cluvfy (as the user *grid*) as shown here:

https://t.me/learningnets

Technet24

```
[oracle@rac1 grid_1]$ ./runcluvfy.sh stage -pre dbinst -n rac1,rac2 -osdba dba
…
Verifying Physical Memory ...PASSED
Verifying Group Membership: oinstall(Primary) ...PASSED
Verifying Group Membership: dba ...PASSED
Verifying Run Level ...PASSED
CVU operation performed:      stage -pre dbinst
Date:                         Apr 25, 2017 6:04:58 PM
CVU home:                     /u01/app/12.2.0.1/grid_1/
User:                         oracle
[oracle@rac1 grid_1]$
```

That is it! Now you are ready to start the installation of RAC after deciding the storage location for the datafiles. If you are currently using raw or block devices as your storage, you should plan on migrating to one of the supported shared storage options, such as ASM, CFS, or NFS, since Oracle 12*c* no longer supports raw devices, even during upgrades.

There's no need to install and configure the ASM using Oracle database binaries before the database install because ASM is installed and configured with Oracle Grid Infrastructure. Because we'll be using Oracle ASM to build this Oracle RAC database, you'll need to create Oracle ASM disk groups to store the Oracle data and backups. This can be accomplished using the ASMCA utility running out of the Grid Infrastructure home.

Before the installation, you can adjust the environment settings for your favorite shell and start the Installer with the tracing option turned on (as discussed in detail in the previous chapter). Tracing will help you get the current stage of the Installer and debug the installation in case of a failure and/or hang.

In the following sections, we first describe the installation of the Oracle RAC software. Following this, we show how to use the Oracle Database Configuration Assistant (DBCA) to create an Oracle RAC database.

Installing Oracle Real Application Clusters

As with the Oracle Grid Infrastructure installation, you use the Oracle Universal Installer (OUI) to install the Oracle RAC binaries. OUI will install the Oracle RAC binaries on the first node and then copy them onto the other servers in the cluster.

You can run the cluvfy utility for the pre–database configuration checks, as shown here. You must run the utility as the user *oracle*.

https://t.me/learningnets

```
[oracle@rac1 grid_1]$ ./runcluvfy.sh stage -pre dbcfg -n rac1,rac2 -d orcl
      Verifying Physical Memory ...PASSED
      Verifying User Existence: oracle ...PASSED
      Verifying Group Existence: dba ...PASSED
      Verifying CRS Integrity ...PASSED
      Verifying DNS/NIS name service 'rac-scan ...
      Verifying ASM Integrity ...
        Verifying Node Connectivity ...PASSED

CVU operation performed:      stage -pre dbcfg
Date:                         Apr 25, 2017 6:08:53 PM
CVU home:                     /u01/app/12.2.0.1/grid_1/
User:                         oracle
[oracle@rac1 grid_1]$
```

It’s not mandatory to run the cluvfy utility to check the prerequisites because the Oracle Universal Installer will run cluvfy internally to verify all prerequisites before installing Oracle RAC 12*c* Release 2.

As mentioned earlier, the Oracle RAC installation is as simple and straightforward as a single-instance environment once you satisfy the prerequisites. You can install the binaries from the installation media (CD-ROM or DVD) or from the staging area if you’ve extracted the software to disk.

Run through the following steps in an X client on only the first node in the cluster:

1. Begin by invoking the Oracle Universal Installer (OUI), as shown here:

   ```
   $ cd /u01/stage/12c/database
   $ ./runInstaller
   ```

   The OUI will display the Configure Security Updates screen, shown in Figure 5-1, which allows you to specify your e-mail address registered with Oracle My Support and the password so that Oracle can notify you whenever a new security update is available.

https://t.me/learningnets

Technet24

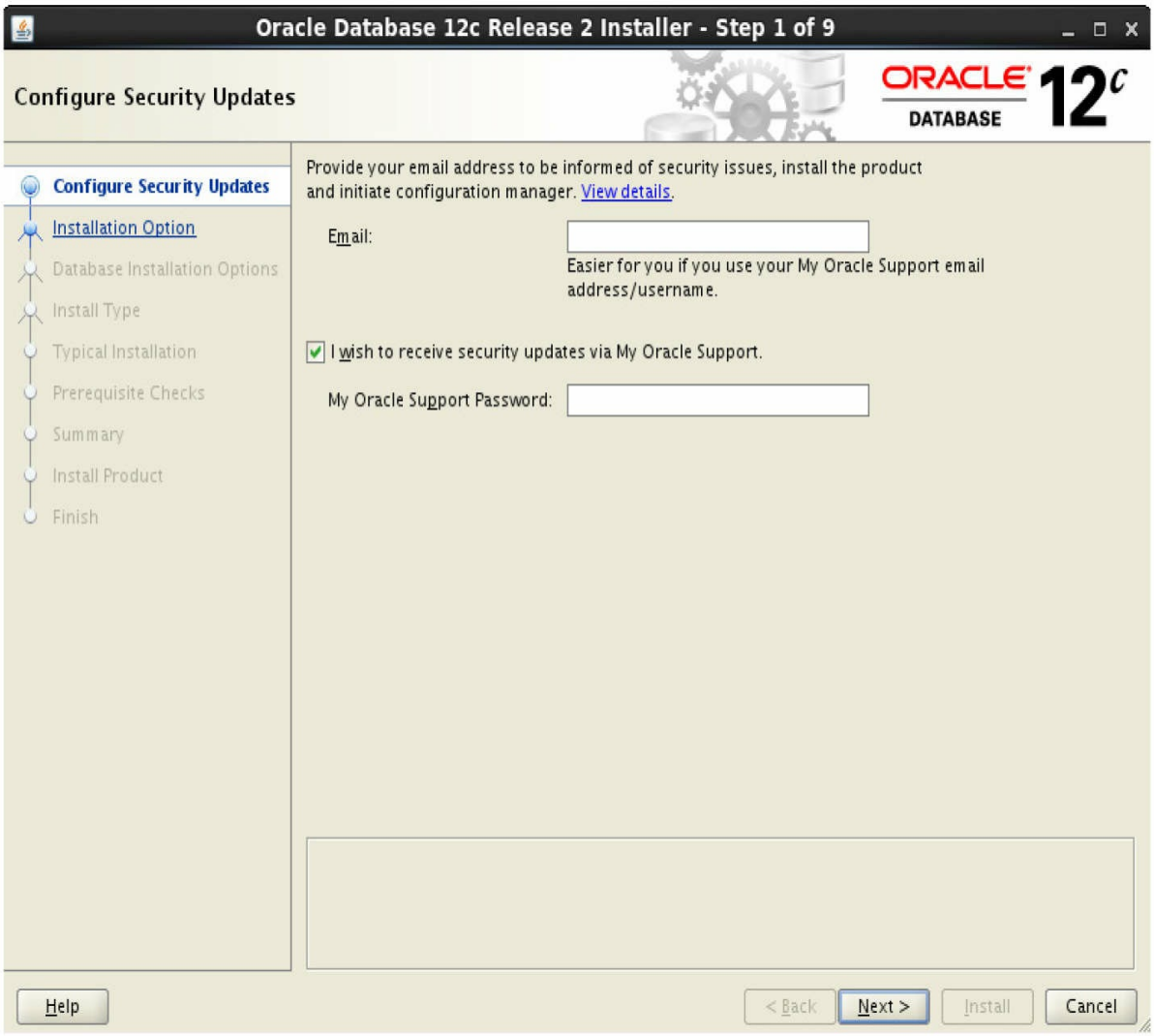


FIGURE 5-1. *The Configure Security Updates screen*

Note that this requires the database servers to be connected to the Internet, and most datacenters don't expose their database servers to the public network for security reasons. Uncheck the box labeled "I wish to receive security updates via My Oracle Support" on this screen. Click Next.

2. The OUI will present the Select Installation Option screen, shown in Figure 5-2, where you specify the required installation and configuration option for the current install. This screen provides the following three installation options:

https://t.me/learningnets

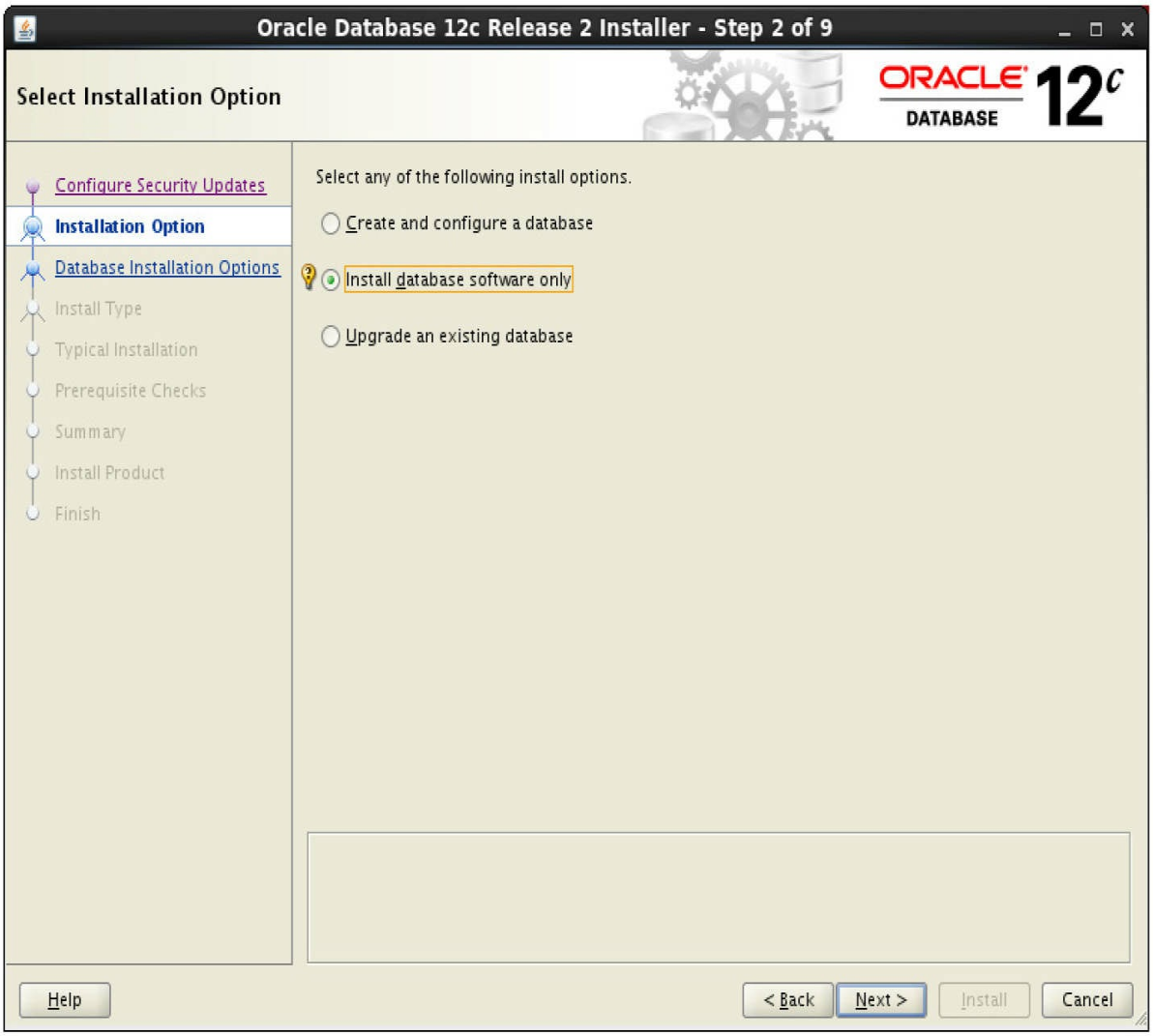


FIGURE 5-2. *The Select Installation Option screen*

- **Create and configure a database** This option is the default installation option, which installs Oracle RAC binaries and creates a database based on a preconfigured template. This option is useful especially for beginners because Oracle provides separate templates for different types of workloads, such as for OLTP and decision support systems.
- **Install database software only** This option installs only the Oracle RAC software on all the servers in the cluster. Database administrators mostly use this option, which allows them more flexibility while creating the database using the Database Configuration Assistant once Oracle RAC binaries are installed.
- **Upgrade an existing database** This option upgrades the existing Oracle RAC database in the cluster.

Because we're going to create a database later with the Oracle Database Configuration Assistant (DBCA), select the second option (Install database software only) and click Next.

https://t.me/learningnets

Technet24

3. In the Select Database Installation Option screen, shown in Figure 5-3, you can choose from among the following three options:

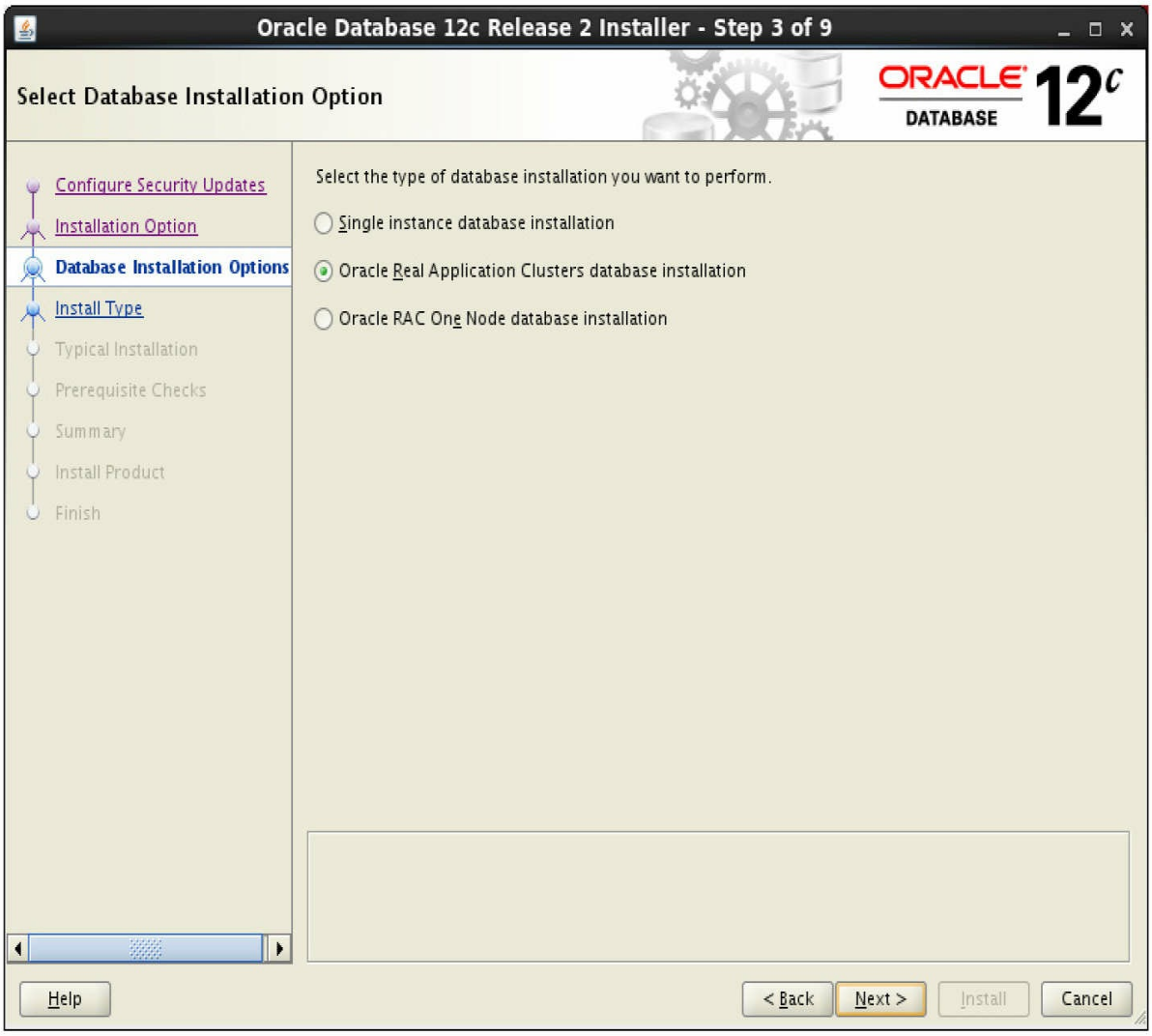


FIGURE 5-3. *The Select Database Installation Option screen*

- **Single instance database installation** This option allows you to install single-instance database software on the local node only.
- **Oracle Real Application Clusters database installation** This option allows you to select and install Oracle Real Application Clusters binaries on the selected nodes in the cluster.
- **Oracle RAC One Node database installation** This option installs Oracle RAC One Node database binaries on the selected node.

On this screen, select the option “Oracle Real Application Clusters database installation” and then click Next.

4. After selecting the grid installation option, you will be taken to the Select List of Nodes screen, shown in Figure 5-4, where you select the list of nodes on

https://t.me/learningnets

which you want to install the RAC binaries. The local node is preselected for you. In our case, the local node is rac1. We'll select the additional node named rac2. Click Next.

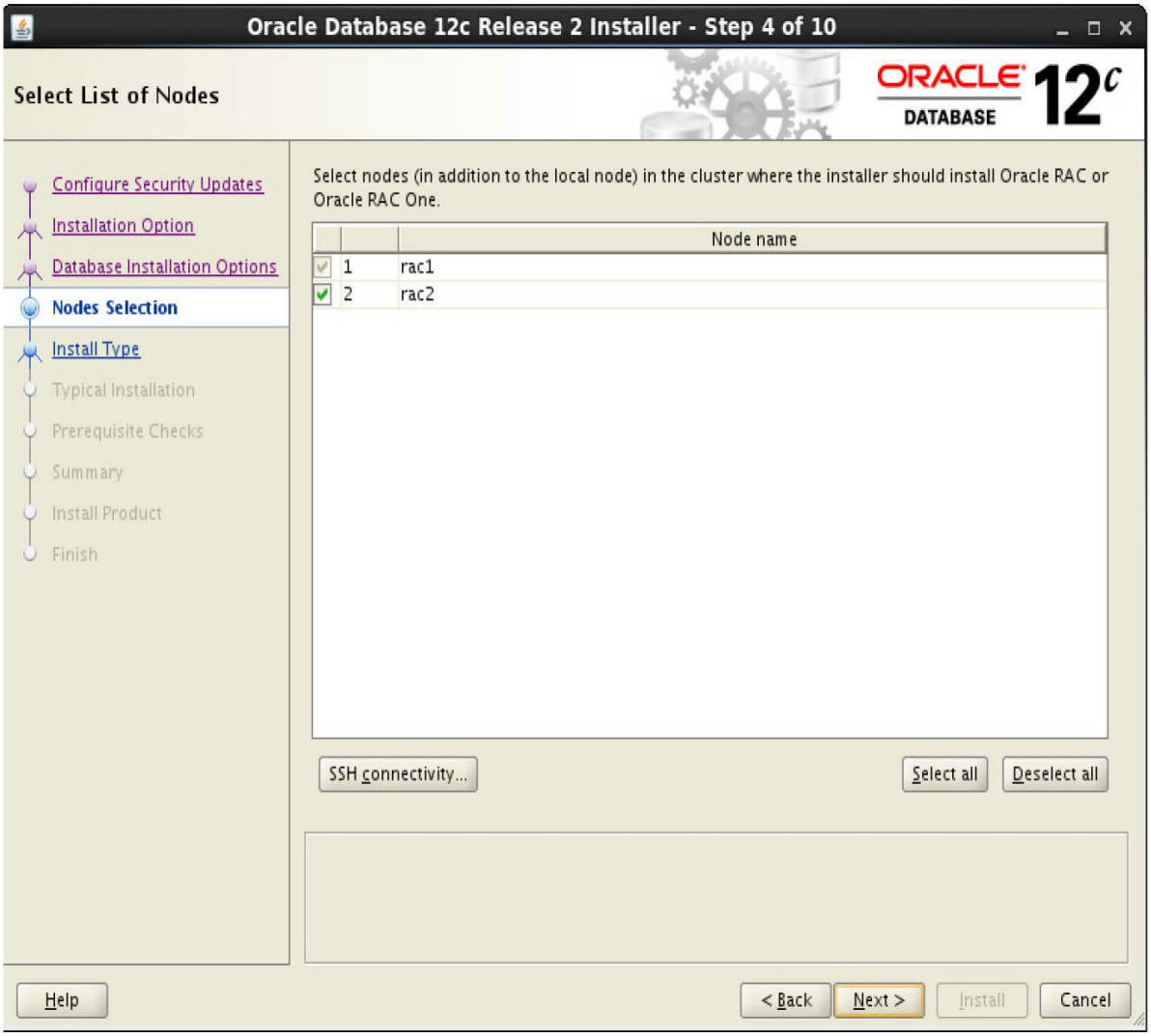


FIGURE 5-4. *The Select List of Nodes screen*

5. OUI will now show you the Select Database Edition screen, shown in Figure 5-5, where you can choose between the Enterprise and Standard Edition of the Oracle RAC database. Select the option on this screen carefully based on the purchased licenses. We'll choose the Enterprise Edition in this case. Click Next.

https://t.me/learningnets

Technet24

FIGURE 5-5. *The Select Database Edition screen*

6. In the Specify Installation Location screen, shown in Figure 5-6, you specify the storage location for ORACLE_BASE and ORACLE_HOME. The storage location for ORACLE_HOME will always be unique. Our Oracle base location is /u01/app/oracle, and the software location under the ORACLE_BASE location is /u01/app/oracle/product/12.2.0/dbhome_1.

https://t.me/learningnets

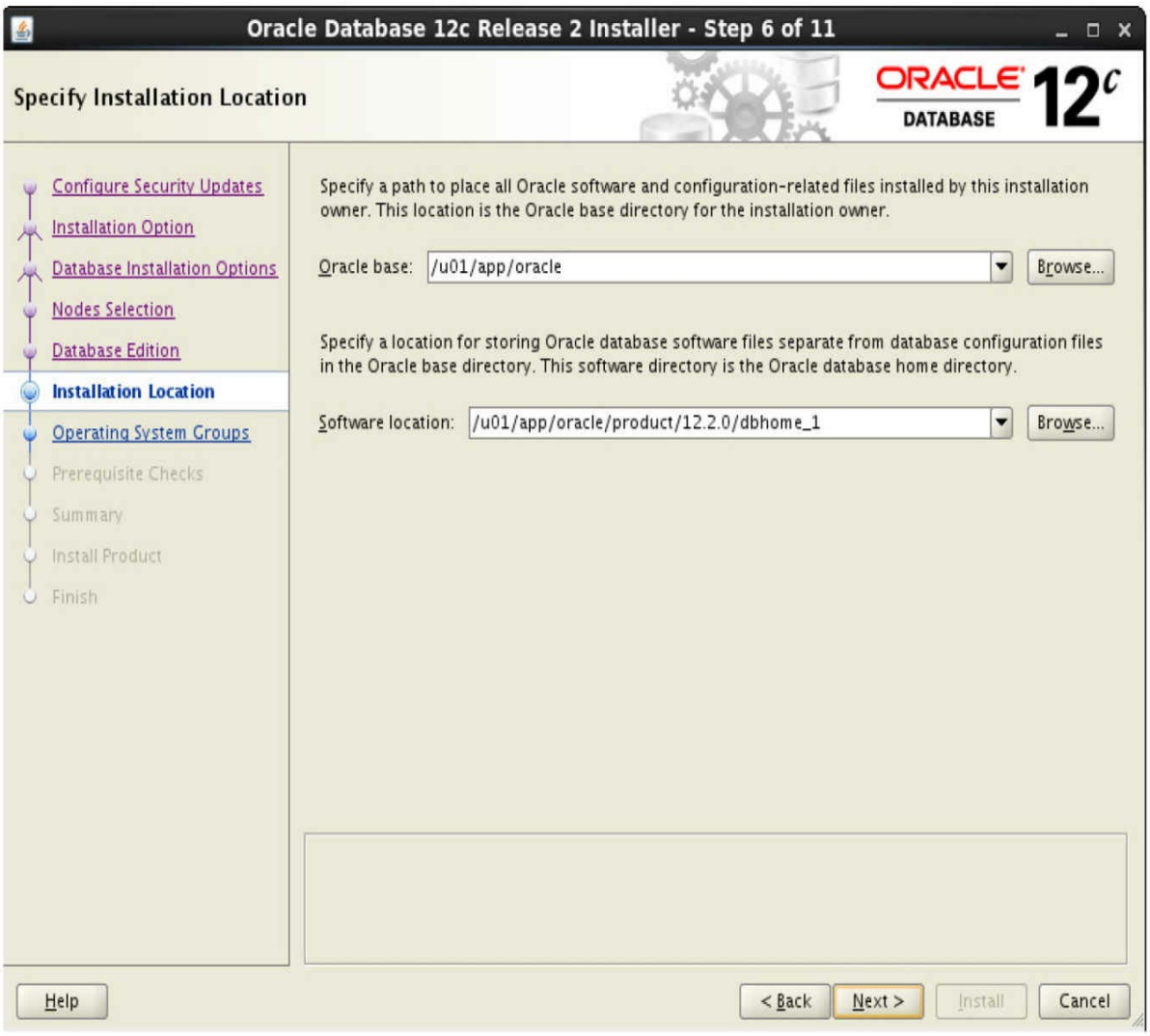


FIGURE 5-6. *The Specify Installation Location screen*

7. In the Privileged Operating System Groups screen, shown in Figure 5-7, select the database administrator and the database operator OS groups from the provided list of values. You should ensure that you are selecting the correct operating system group on this screen because a wrong selection may interfere with operation of the Oracle RAC database software. For simplicity, we'll choose to specify the same group name (dba) as the value for all the groups.

https://t.me/learningnets

Technet24

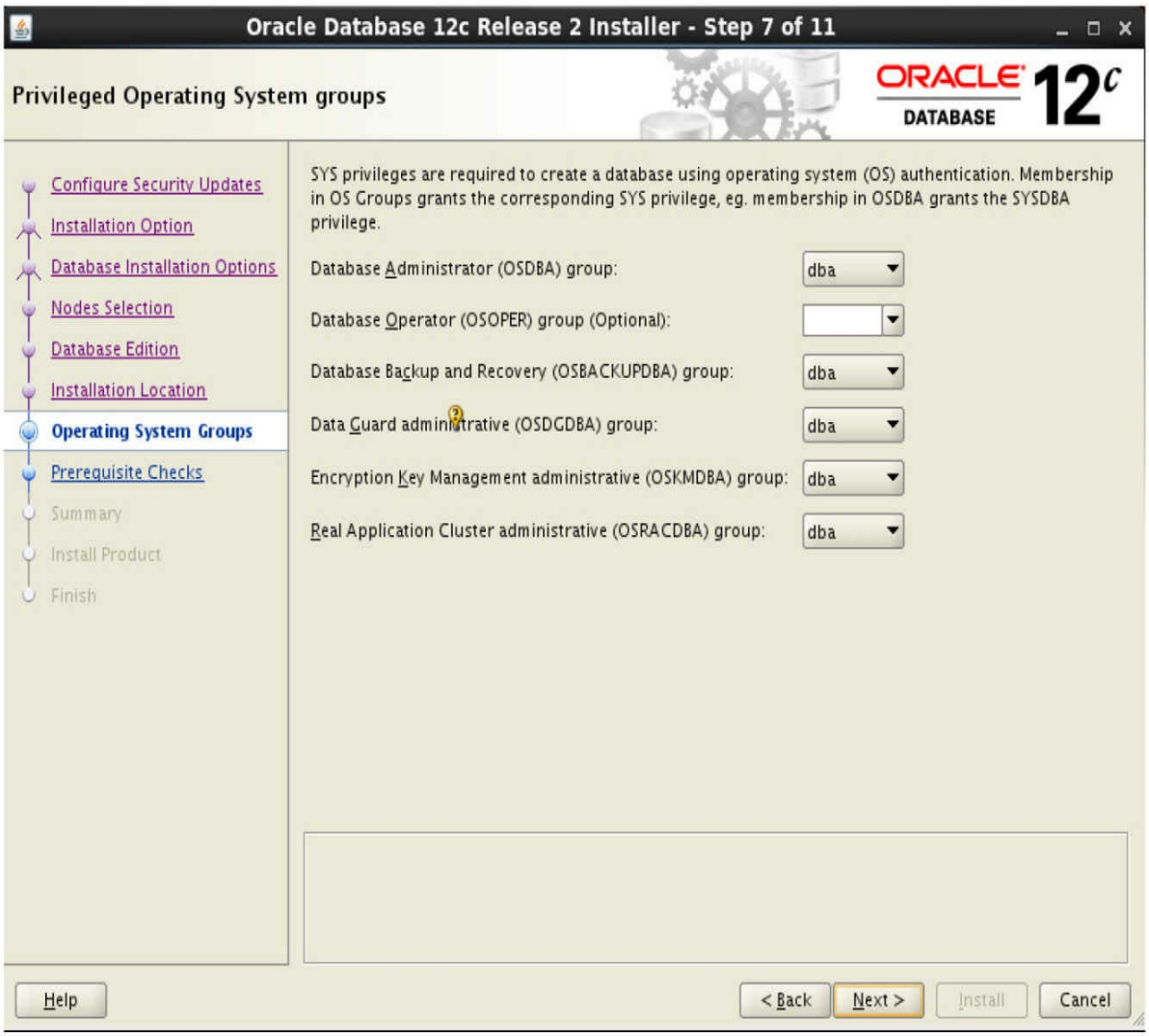


FIGURE 5-7. *The Privileged Operating System Groups screen*

8. In the Perform Prerequisite Checks screen, shown in Figure 5-8, the Oracle Universal Installer verifies the minimum requirements for installing the Oracle RAC database software. Oracle Universal Installer internally executes the Cluster Verification Utility to verify the operating system and hardware prerequisites. Based on the results of the verification tests performed by the utility, Oracle displays the failed prerequisites on this screen. Because we've run the Cluster Verification Utility before starting the Installer, we don't expect any surprises at this stage.

https://t.me/learningnets

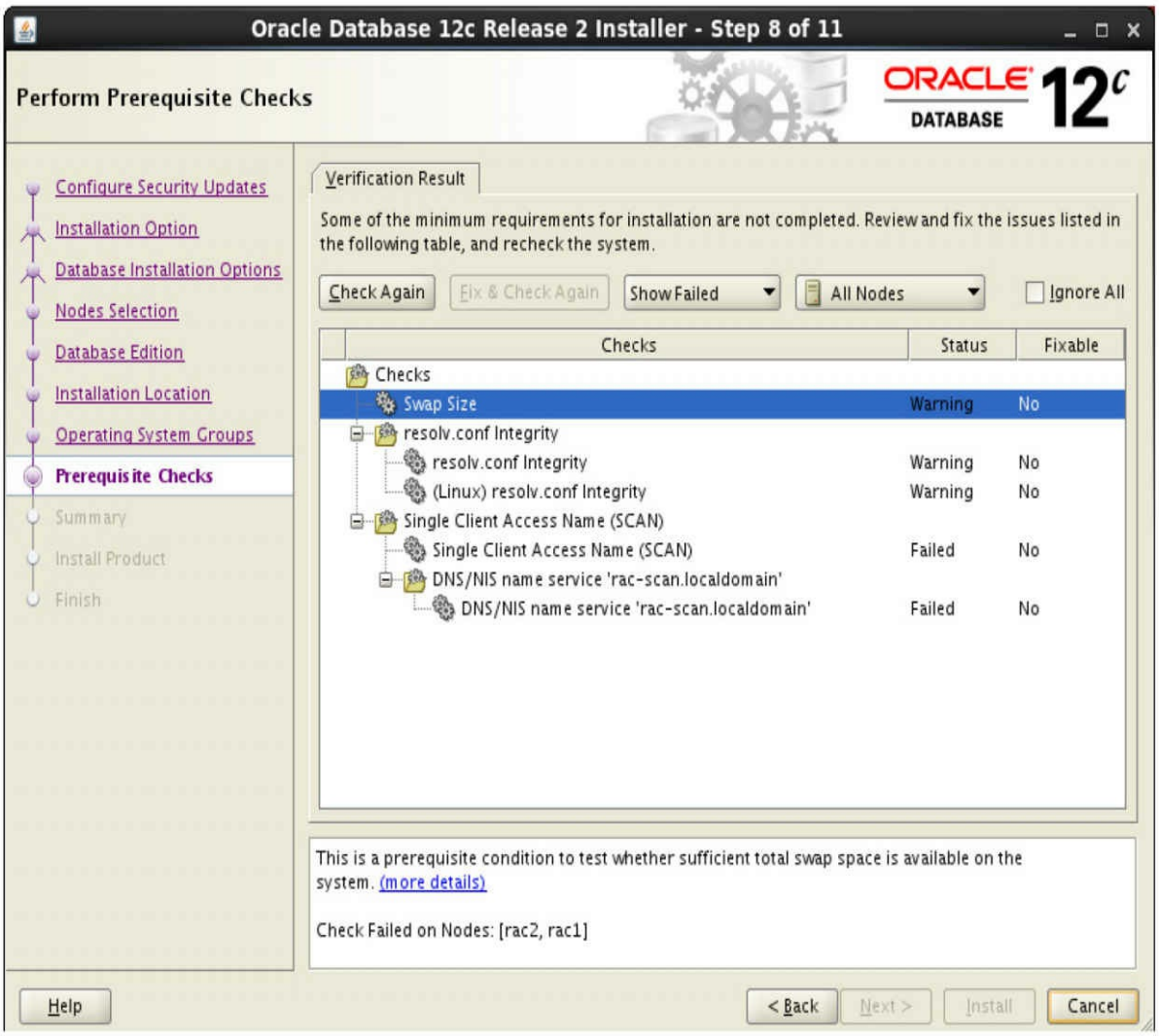


FIGURE 5-8. *The Perform Prerequisite Checks screen*

OUI provides what are called “fix-up” scripts to repair the failed but fixable prerequisite requirements. Oracle marks each failed prerequisite with a Fixable status of Yes if it can be repaired by a fix-up script, which is also displayed on the screen. You can select the failed prerequisite with a Fixable status Yes and click the Fix & Check Again button on this screen to fix and verify the prerequisites again. Once the Installer verifies the required prerequisites successfully, click Next to continue.

9. In the Summary screen, shown in Figure 5-9, Oracle displays the installation information. You should verify this information. You can also save this as a response file to use it for mass deployment with the silent install method. Refer to the Oracle Universal Installer Guide for running Oracle Universal Installer with a response file for automated deployment. Our step-by-step install process is known as an interactive installation process.

https://t.me/learningnets

Technet24

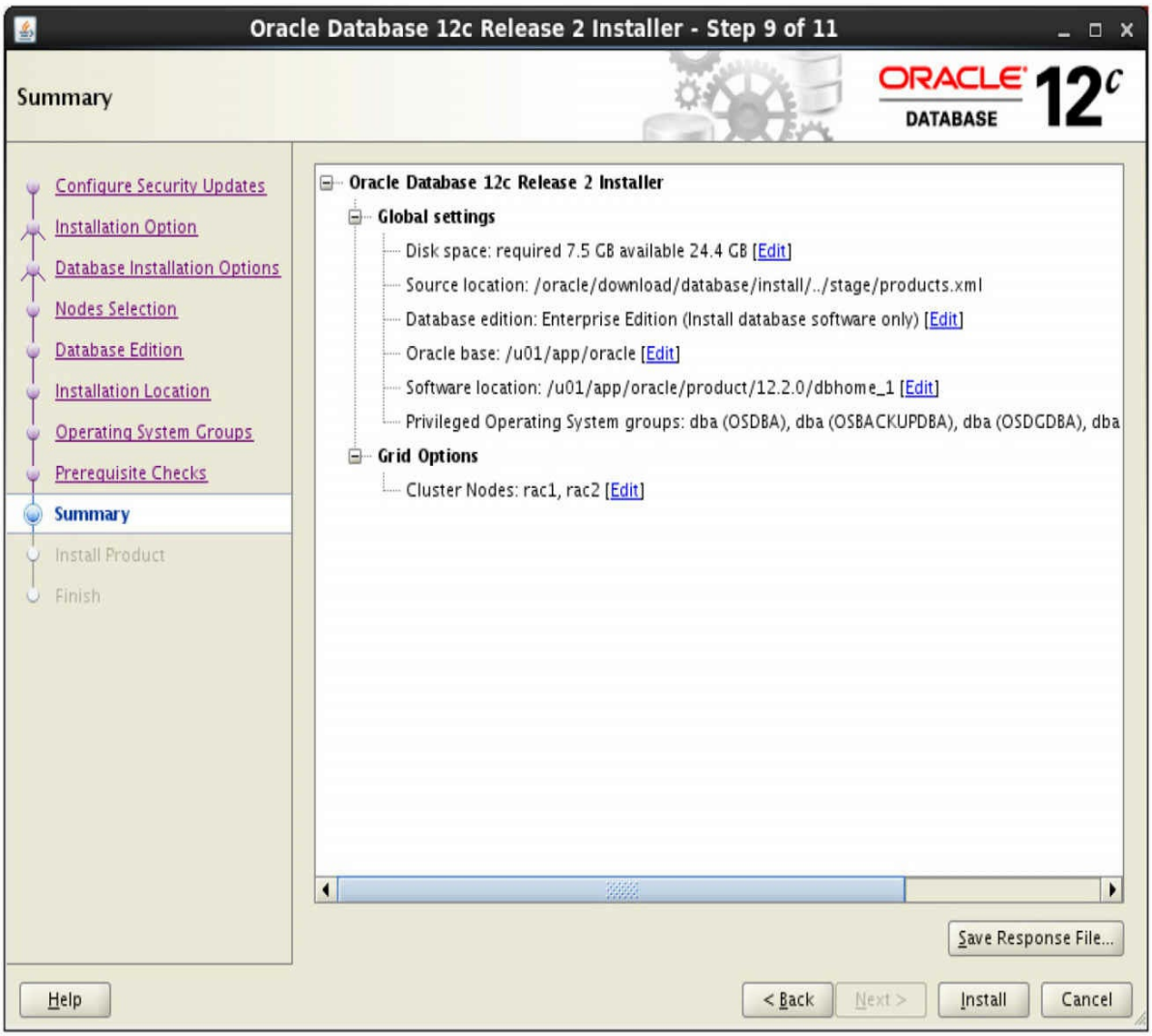


FIGURE 5-9. *The Summary screen*

Oracle Silent Installation

The silent installation method is used for mass deployment of Oracle products because using the interactive method to install Oracle software multiple times on multiple machines is time consuming and error prone. Also, silent install provides an option to have a uniform deployment pattern across the organization. This ensures multiple users in the organization use a standard installation option to install their Oracle products. This greatly helps the internal Oracle support teams because they already know what components and options are installed on each server and their environment settings, including locations of various trace files.

10. Once you verify that the information displayed in the Summary screen is correct, click Install to start the software installation.

Installation of Oracle RAC software may take up to 30 minutes, depending on

https://t.me/learningnets

the hardware configuration. The Installer will copy all the required files to the Oracle database home directory and link the files with the operating system libraries.

Once the installation and linking is done at the local node, the Installer will copy the files to the remote node. During the entire process, you can see the status in the progress bar of the Install Product screen (see Figure 5-10). This progress bar shows the completion percentage.

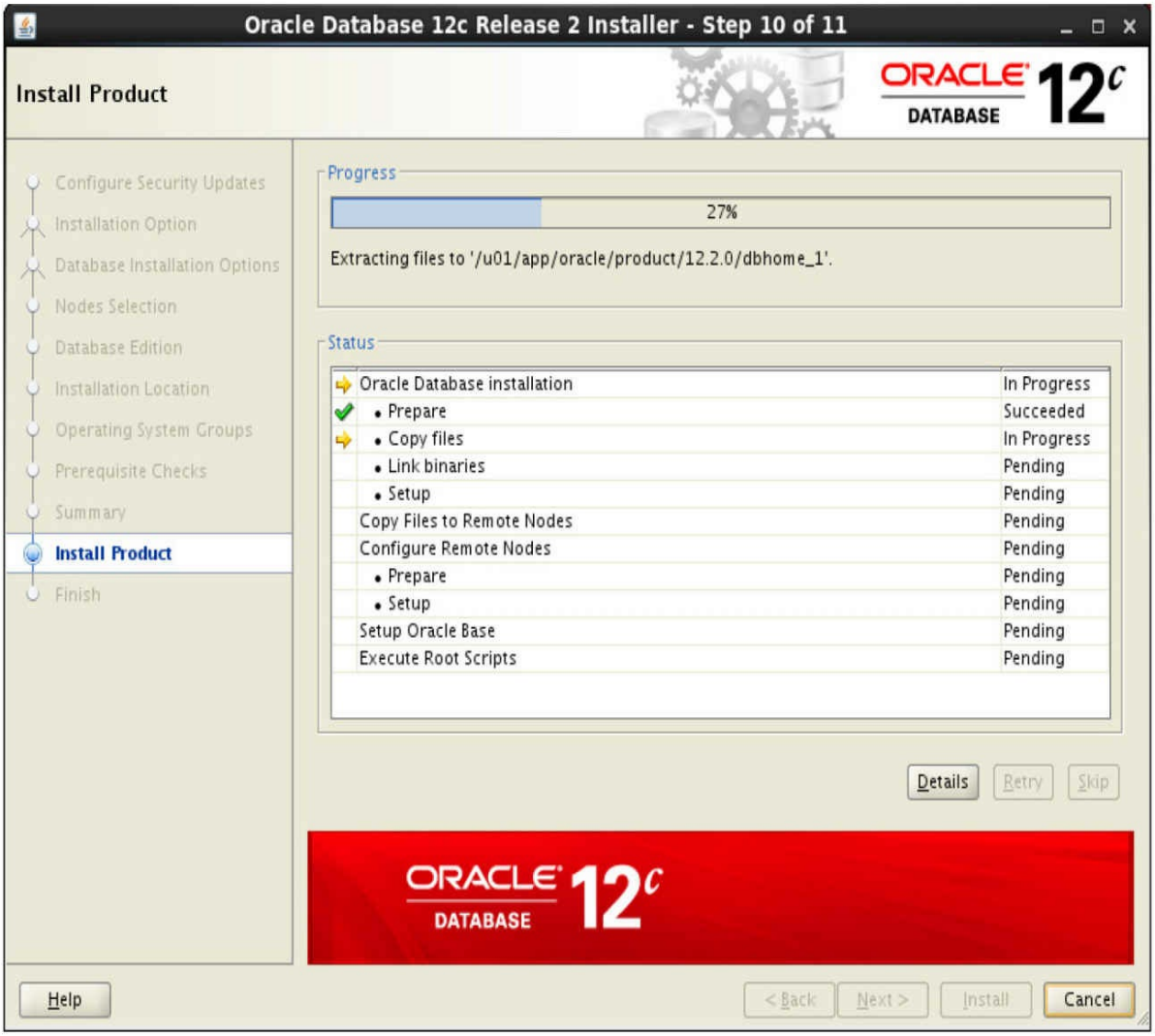


FIGURE 5-10. *The Install Product screen*

11. The OUI will next ask you to run the installation script as the superuser root. You need to open a new terminal window and run the root.sh script as the superuser on all nodes in the cluster (see Figure 5-11). This script creates the oraenv and oratab files under /etc and sets the Oracle executable permissions to the owner and group levels. The location of the file for the oratab entry is platform specific and normally found in either the /etc or /var/opt/oracle directory. This root.sh script must be run as the superuser root.

https://t.me/learningnets

Technet24

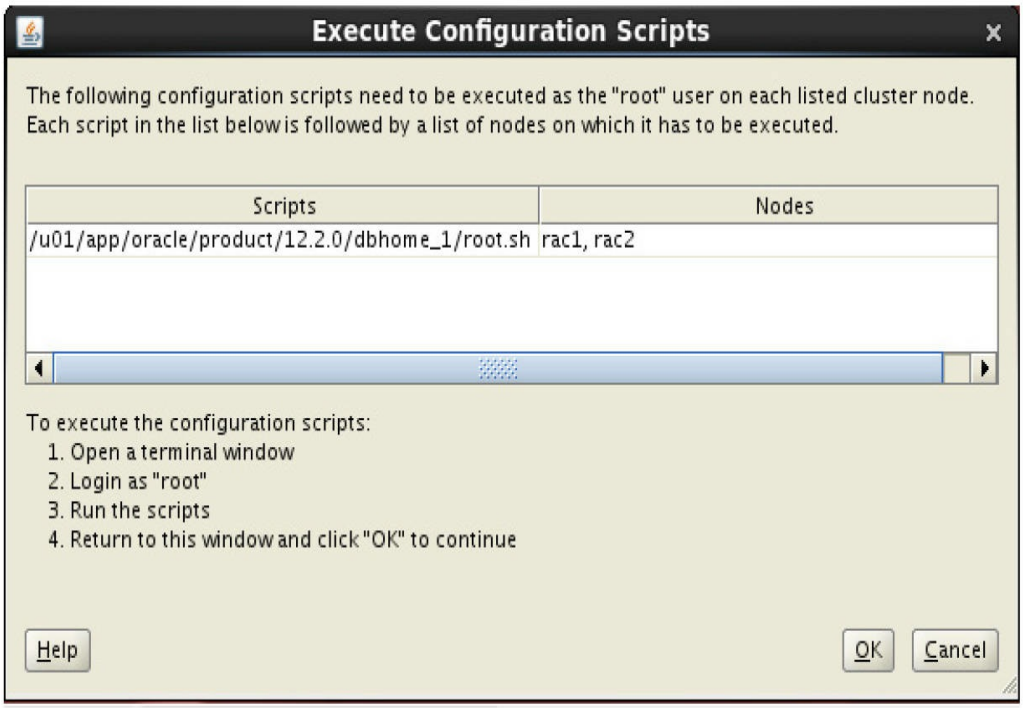


FIGURE 5-11. *The Execute Configuration Scripts dialog box*

12. After you execute the root.sh script, go back to the Installer and click the OK button to display the Finish screen (see Figure 5-12).

https://t.me/learningnets

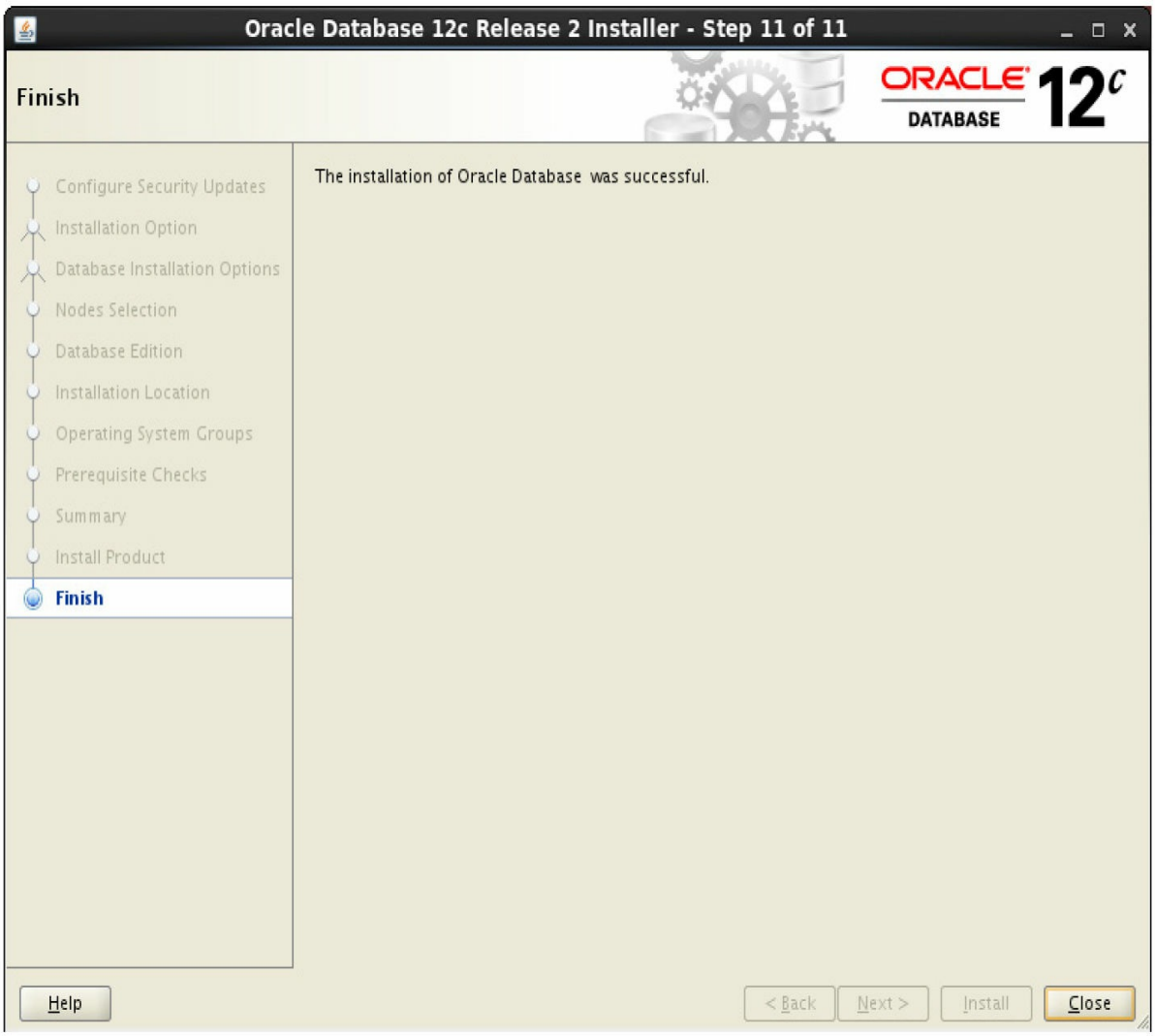


FIGURE 5-12. *The Finish screen*

13. Click the Close button in the Finish screen to close the Oracle Universal Installer.

Now that you've successfully installed the Oracle Database 12*c* software, you can create an Oracle RAC database, as we explain in the following section.

Creating the Oracle RAC Database

Creating the RAC database in an Oracle RAC environment is as simple as creating the database in a single-instance environment. The only change in the creation process using DBCA is to select the nodes in the Database Identification screen. Optionally, you can generate the scripts using DBCA and run them later.

To create a database using ASM-based storage for datafiles, walk through the following steps. (The number of steps may vary a bit based on the storage options you choose. However, the database-creation process is the same as creating the database in the ASM environment.)

https://t.me/learningnets

Technet24

1. Create the ASM disk groups that will be used to store the datafiles. Use the Automatic Storage Management Configuration Assistant (also known as ASMCA) to create and manage the ASM instances, volumes, cluster file system, and disk groups.

   You can launch ASMCA from the $GRID_HOME/bin directory as the grid user. We'll use the DATA disk group created as part of the Oracle Grid Infrastructure installation in the previous chapter, but the typical steps for creating an ASM disk group include marking disks to be used by ASM and then creating the ASM disk group using any ASM client, such as ASMCA, ASMCMD, or SQL*Pus. We'll discuss this in detail in the next chapter, where we cover Oracle ASM.

2. Start the Database Configuration Assistant (DBCA) as the user oracle. Navigate to the $ORACLE_HOME/bin directory. If you want to enable tracing, you need to customize the Java Runtime Environment to get the trace information.

   ```
   $./dbca
   ```

   DBCA enables you to create, configure, or delete a cluster database and manage database templates.

3. In the very first screen, which is named Select Database Operation, you can select from the following options, as shown in Figure 5-13:

https://t.me/learningnets

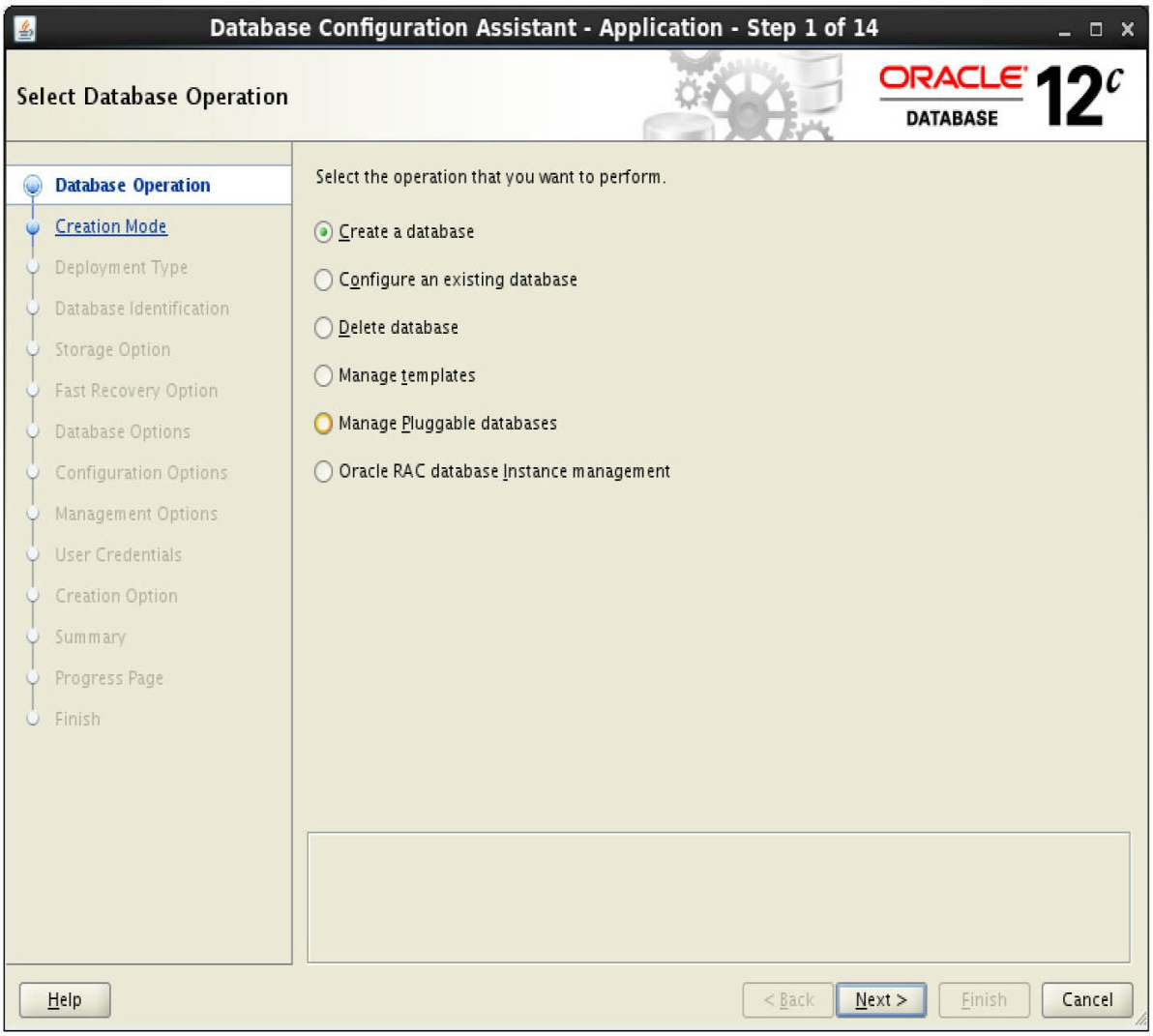


FIGURE 5-13. *The Select Database Operation screen*

- Create a database
- Configure an existing database
- Delete database
- Manage templates
- Manage Pluggable databases
- Oracle RAC database instance management

4. In the Select Database Creation Mode screen, shown in Figure 5-14, you're offered two choices:

https://t.me/learningnets

Technet24

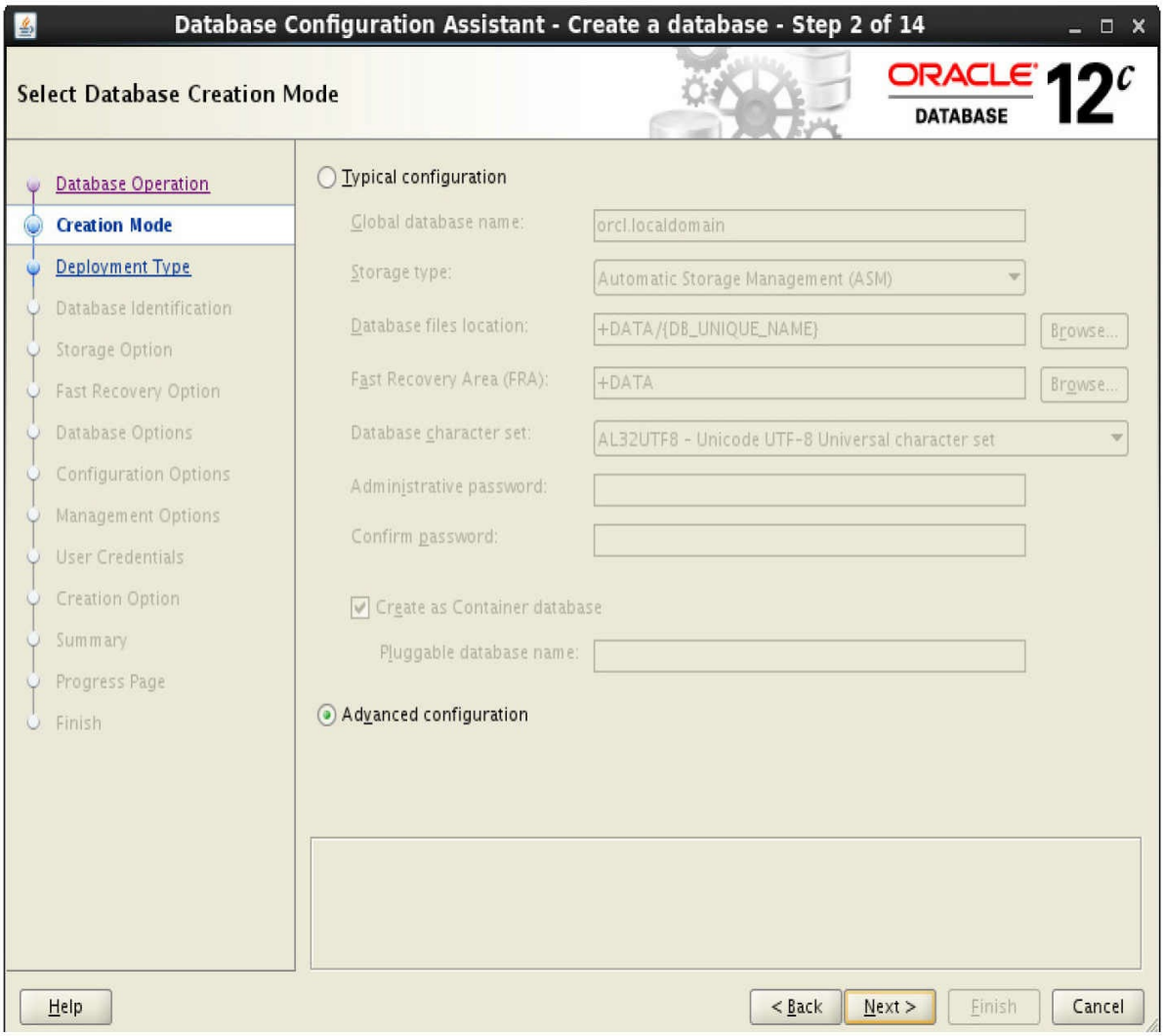


FIGURE 5-14. *The Select Database Creation Mode screen*

- **Typical configuration** This option requires minimal input from the user. You need to provide a global database name and storage details, if ASM is already configured (which could be the case if you've used ASM while configuring Grid Infrastructure, as we did in Chapter 4). The required fields are auto-generated. Enter a password and fill in the pluggable database name if the database is being configured as a container database.
- **Advanced configuration** You can customize the database configuration by choosing this mode. This mode choice offers various options to customize the database creation, such the database configuration type (either policy-managed, which is the default in Oracle 12*c*, or admin-managed).

  We'll choose the Advanced Configuration option. Click Next.

5. In the Select Database Deployment Type screen, shown in Figure 5-15, you choose whether you want an Oracle RAC database, a non-RAC database, or an Oracle RAC One Node database. We'll choose to create an Oracle RAC

https://t.me/learningnets

database, of course.

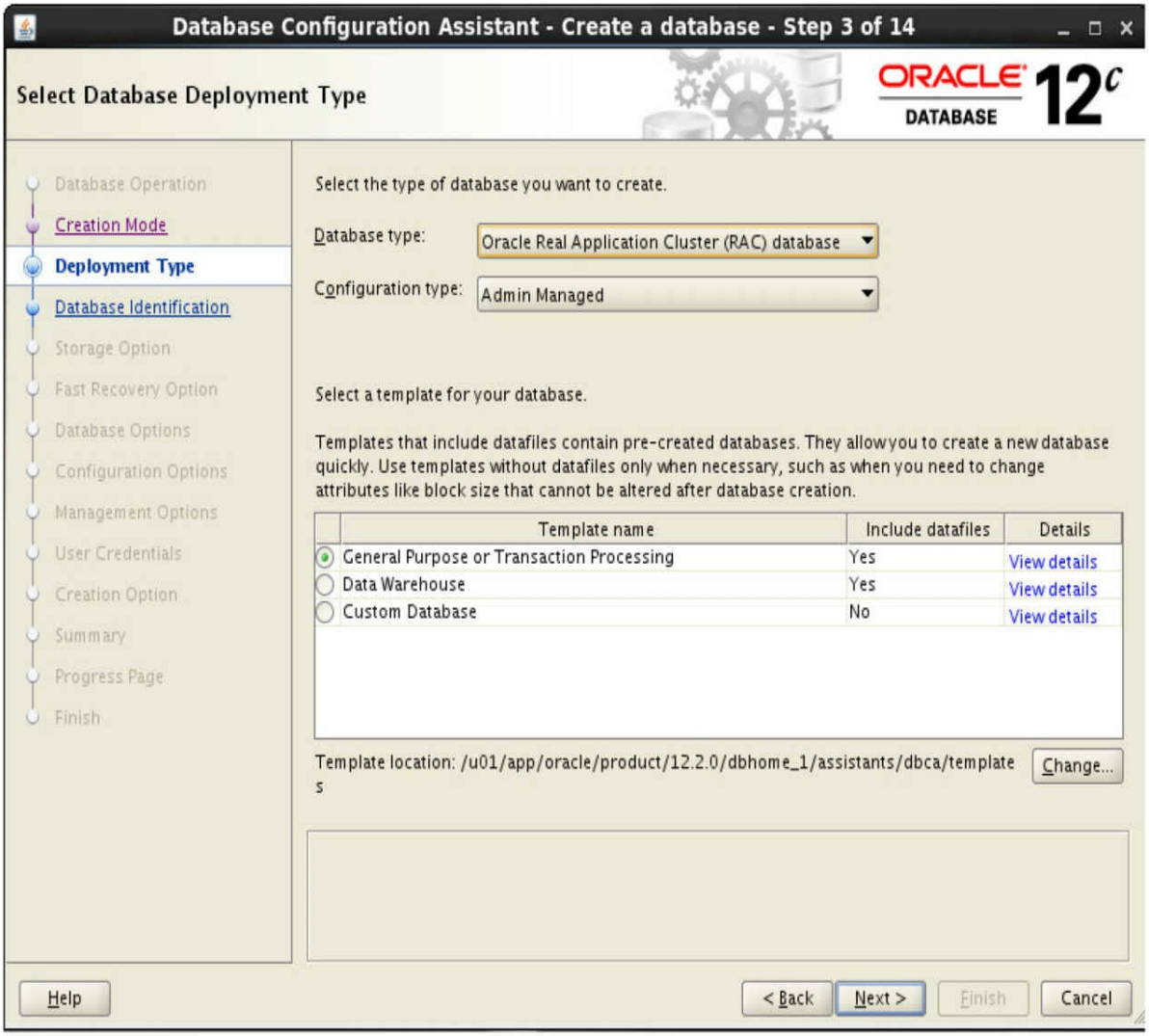


FIGURE 5-15. *The Select Database Deployment Type screen*

Once you select the database type (RAC, in our case) in this screen, you also need to select between two configuration types: an admin-managed or a policy-managed database. Choose the Admin Managed option for the configuration type.

Oracle offers several predefined database configuration templates to make the process of creating a database easier. In the Select Database Deployment Type screen, Oracle provides predefined templates that you can use to create a new database based on the expected type of workload.

The Oracle-provided templates come with datafiles, but you can use these templates without datafiles if you want to customize database attributes such as block size. This screen displays three options:

- General Purpose or Transaction Processing

https://t.me/learningnets

Technet24

- Data Warehouse
- Custom Database

Each of the options provides preconfigured parameters/attributes for the new database, and the Custom Database option allows you to choose among these attributes and parameters. Regardless of your choice on this screen, you can still customize the database in later steps. Choose the General Purpose or Transaction Processing option on this screen and click Next.

6. In the Select List of Nodes screen, select the nodes on which you want to create the cluster database and then click Next.
7. In the Specify Database Identification Details screen, shown in Figure 5-16, select the global name of this cluster database and a prefix that will serve as the database service identifier.

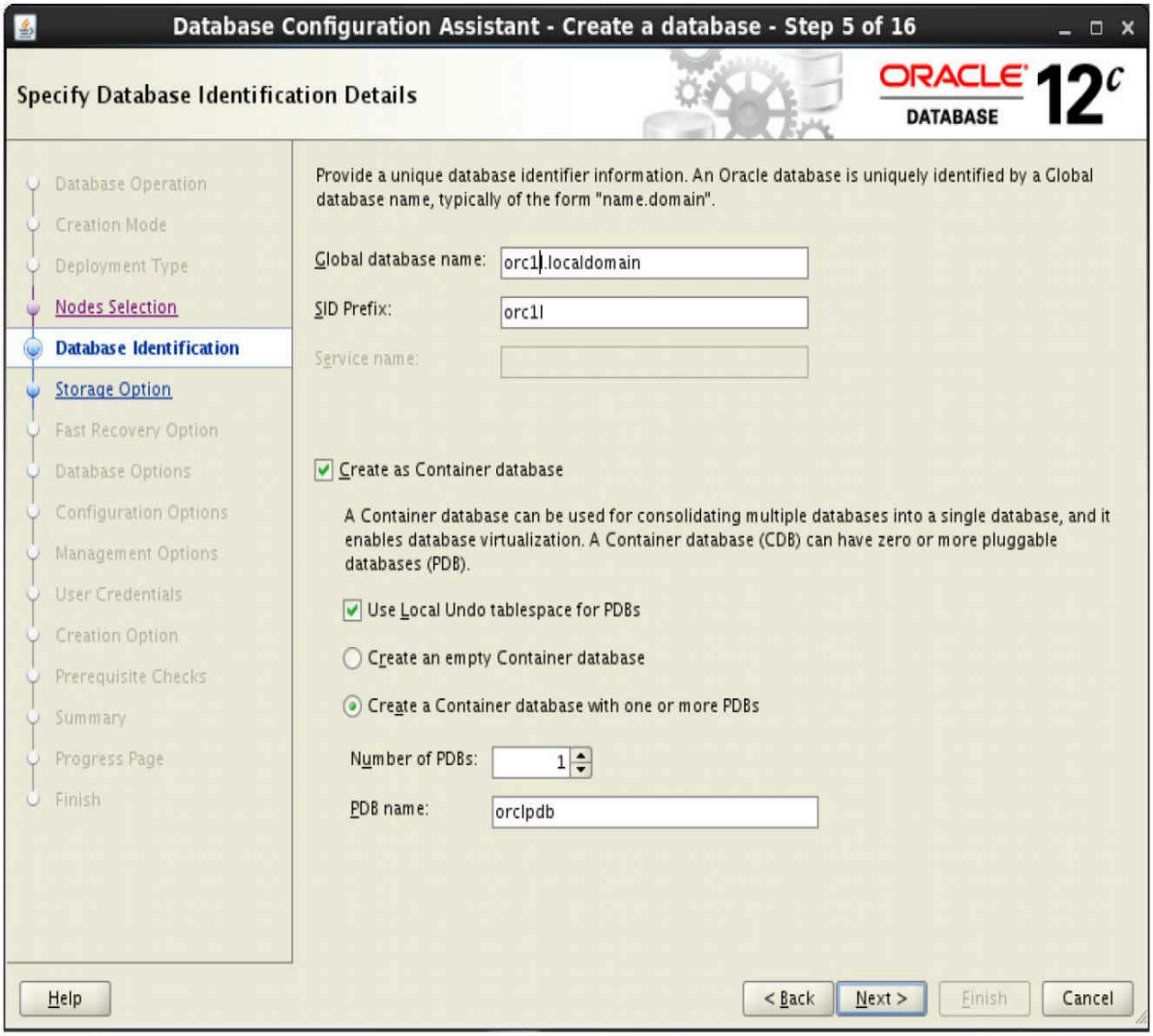


FIGURE 5-16. *The Specify Database Identification Details screen*

https://t.me/learningnets

NOTE

The Global Database Name and SID Prefix values must start with an alphabetical character but can be followed by alphanumerical characters.

On this screen, the Installer requires you to also specify whether you want to create the new database as a container database. If you choose to do so, you can also specify the name and number of the PDBs.

8. In the Select Database Storage Option screen (not shown), select ASM as the storage type for the database files and then choose the location for storing the database files. You can also specify whether to use the Oracle-Managed Files (OMF) for naming your datafiles.
9. In the Select Fast Recovery Option screen (not shown), select the Specify Fast Recovery Area option as well as the Enable Archiving option.
10. You can ignore the choices offered in the Select Oracle Data Vault Config Option screen for now.
11. Select Automatic Memory Management in the Specify Configuration Options screen (not shown).
12. In the Specify Configuration Options screen, you can specify various management options for the database, such as periodically running the Cluster Verification Utility checks.
13. On the Specify Database User Credentials screen (not shown), specify passwords for the user accounts such as SYS, SYSTEM, and PDBADMIN.
14. In the Select Database Creation Option screen, shown in Figure 5-17, you can choose between creating the database now or generating database-creation scripts for later use. It's always a good practice to generate scripts to be able to see what is being run and have these scripts available for future reference. You can choose the location of the scripts in any local directory. By default, they are stored under the $ORACLE_BASE/admin directory. When you click Next, OUI performs all the required prerequisite checks and validations.

https://t.me/learningnets

Technet24

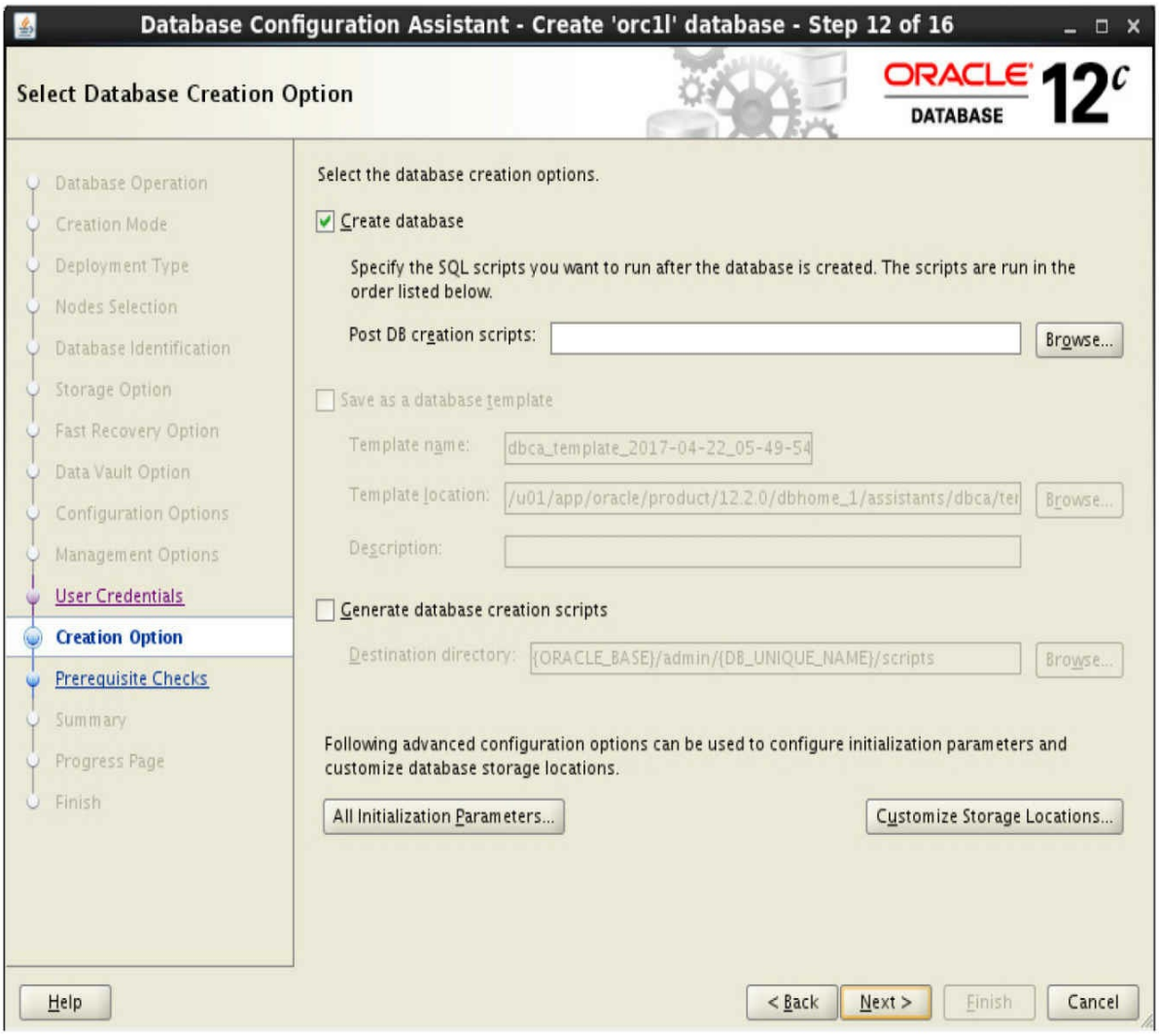


FIGURE 5-17. *The Select Database Creation Option screen*

By clicking the All Initialization Parameters button on this screen, you get the opportunity to modify any Oracle initialization parameters from their default values.

Admin-Managed or Policy-Managed Oracle RAC Database

An admin-managed Oracle RAC database is a traditional configuration approach where you can specify the servers in the cluster that will run the Oracle RAC database instances. You can also decide the preferred and available Oracle RAC database instances for a given database service. In a policy-managed Oracle RAC database, you don’t have flexibility to decide which server in the cluster will run the Oracle RAC database instance. Oracle will start the number of Oracle RAC database instances on servers in a server pool based on the cardinality of the Oracle RAC database.

https://t.me/learningnets

In an admin-managed RAC database, there is a relation/association between the database service and the Oracle RAC database instance, whereas in a policy-managed Oracle RAC database, there is a relation between a database service and the server pool, because Oracle will automatically decide the RAC database instance, which will serve a given database service. A server pool is a pool of different servers in the cluster, which Oracle uses to automatically host database instances. You should always have a number of servers in the server pool greater than the number of Oracle RAC database instances specified in cardinality. The CRSCTL and SRVCTL utilities can be used to create and manage server pools in an Oracle RAC database.

Fast Recovery Area

The Fast Recovery Area is a dedicated storage location Oracle uses to store all database backup-related files so that in the event of database recovery, Oracle can quickly restore backup pieces from this dedicated storage location rather than fetching from the slow tape libraries. This storage is configured using two database initialization parameters: DB_RECOVERY_FILE_DEST and DB_RECOVERY_FILE_DEST_SIZE. Oracle manages this storage area and automatically purges obsolete backups based on the backup retention policies.

15. You'll now be presented with the Summary screen (see Figure 5-18). Click Finish to start the script generation and database creation process.

https://t.me/learningnets

Technet24

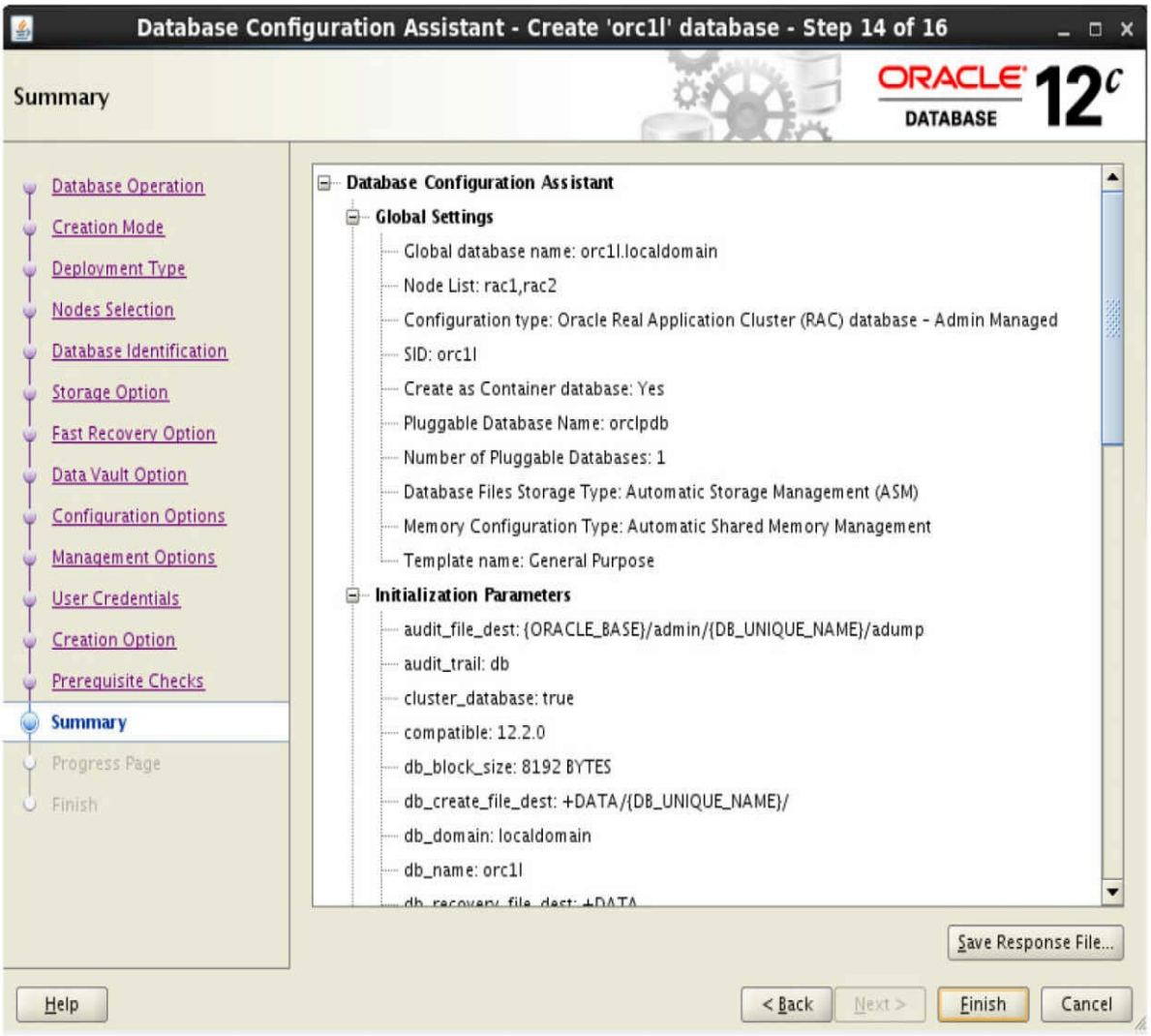


FIGURE 5-18. *The Summary screen*

16. Once the DBCA creates the database, it will provide a screen with a summary of the database it just created (see Figure 5-19). Click OK. The Installer creates the new RAC database and will automatically start it up. This database will be ready for its first use.

https://t.me/learningnets

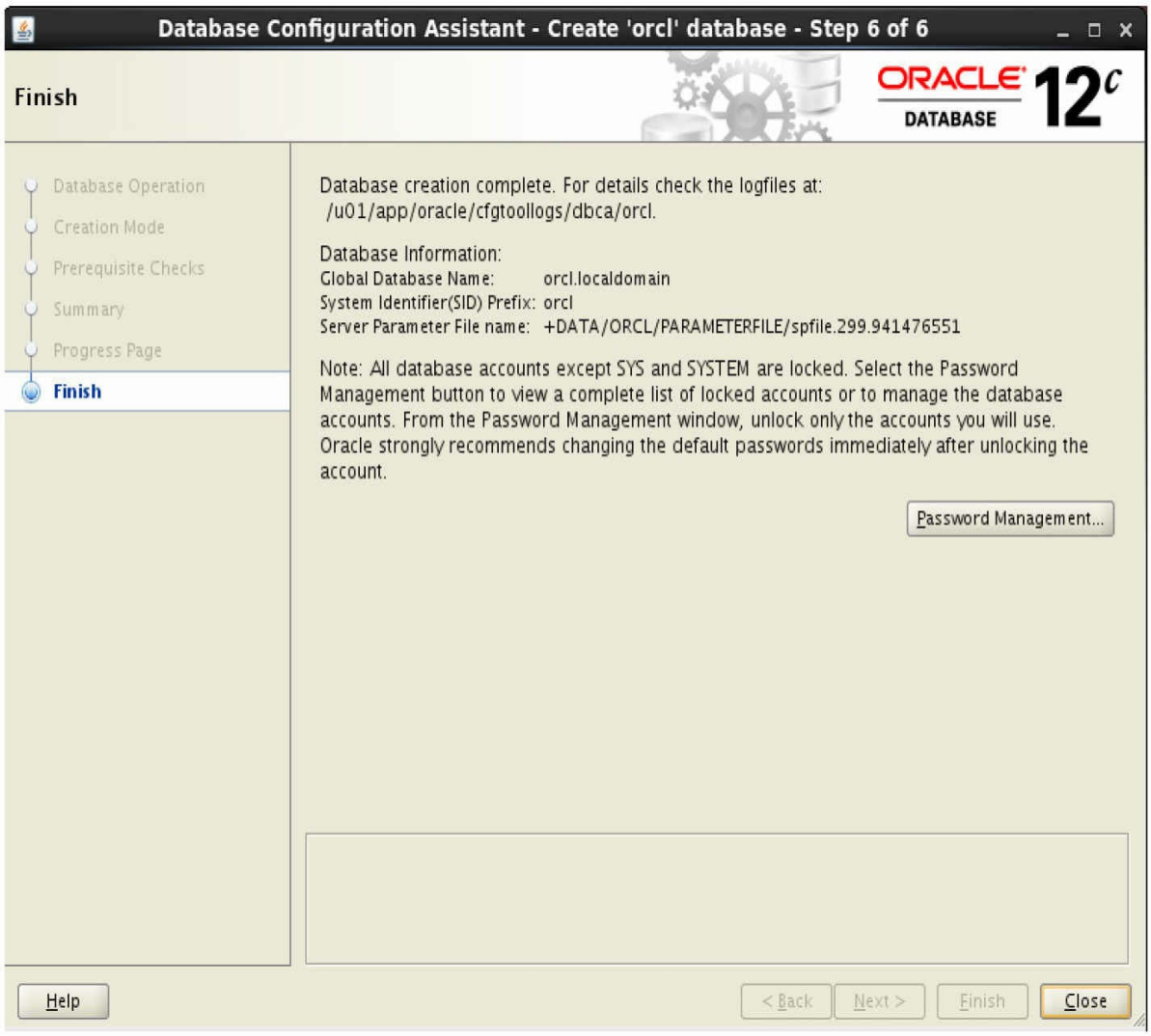


FIGURE 5-19. *The Finish screen*

You can check the status of the two instances of your new Oracle RAC database by running the following SRVCTL command:

```
[oracle@rac1 bin]$ pwd
/u01/app/oracle/product/12.2.0/dbhome_1/bin

[oracle@rac1 bin]$ ./srvctl status database -d orcl
Instance orcl1 is running on node rac1
Instance orcl2 is running on node rac2
[oracle@rac1 bin]$
```

It has been a bit of a tedious journey, but we now have our Oracle Database 12*c* Release 2 database running. Beginning with Chapter 6, we show you how to manage this database.

https://t.me/learningnets

Technet24

Summary

This chapter covered the installation process of Oracle Database 12*c* Release 2 RAC binaries, as well as the creation of an Oracle Database 12*c* database. As we explained, the process is pretty similar to that of an Oracle single-instance database. The number of steps in the installation in your case may vary slightly depending on the options and configurations you select.

Oracle RAC One Node is a newly introduced option that helps build a single-database infrastructure that's ready for the high availability features provided by Oracle Grid Infrastructure. Instance failures are detected by Oracle Clusterware and automatically restarted in another server in the server pool, which ensures failover protection for Oracle RAC One Node.

As you've probably noticed, we didn't have to install and configure ASM separately. Because ASM is part of Oracle Grid Infrastructure, it was already set up during the Oracle Grid Infrastructure installation and configuration process. In the next chapter, we discuss the ASM architecture and explore its functionality in detail.

https://t.me/learningnets

CHAPTER 6

Automatic Storage Management

Managing storage is one of the most complex and time-consuming tasks of a DBA. Data grows at an exponential pace due to the consolidation of databases, as well as the growth of a business over time. Business requirements demand the continuous availability of database storage systems, and the maintenance window for storage is shrinking from hours to minutes. Legal and compliance requirements add even more baggage because increasingly, organizations are required to retain data for extended periods of time. (We know some sites that store hundreds of terabytes of active data for seven years or more!)

Oracle Automatic Storage Management (Oracle ASM), Oracle ASM Cluster File System (Oracle ACFS), and Oracle ASM Dynamic Volume Manager (Oracle ADVM) are the key components of Oracle RAC storage management. Storage management has changed significantly over the past several years. You can now have remote ASM-based systems that use some form of a NFS (Network File System) for shared storage as well as appliances such as the Oracle Data Appliance, which has an NFS-based mechanism to expose ASM/ACFS to the virtual machines (VMs). In addition, some cloud-based storage solutions such as Amazon EC2/S3 and Microsoft Azure aren't capable of hosting Oracle RAC–based systems yet, but work is in progress to make this possible in these types of environments. Oracle's public cloud is also something

https://t.me/learningnets

Technet24

you can use in case you can't allocate on-premises storage.

Modern technologies help DBAs easily manage huge volumes of data without a considerable amount of administrative overhead. New tools are being developed that work closely with the RDBMS kernel to make data management and data consolidation easier. Automatic Storage Management (affectionately called *Awesome Storage Management)* is one of those revolutionary solutions from Oracle. After introducing Automatic Storage Management in Oracle Database 10*g*, Oracle has enhanced it over the following Oracle Database releases to provide a complete volume manager built into Oracle database software. The Oracle ASM file system is highly optimized to provide raw disk performance by avoiding various types of overhead associated with a conventional file system.

Standard Oracle ASM and Oracle Flex ASM

In order to use Oracle ASM to manage an Oracle database, you must first install ASM. However, there's no need for you to install ASM separately because you'll be installing the Oracle Grid Infrastructure before installing Oracle RAC databases. Oracle Grid Infrastructure consists of Oracle Clusterware and Oracle ASM. When you install Oracle Grid Infrastructure, the installer automatically installs Oracle ASM in the Oracle Grid Infrastructure home.

Up until the Oracle Database 12*c* release, the only way to use ASM was to run an Oracle ASM instance on each node where an Oracle RAC instance would run. Multiple database instances on the same server can share the same Oracle ASM instance, but database instances running on separate servers, as is the case with Oracle RAC, needed to have a separate ASM instance running on each server that was part of the cluster.

This traditional ASM configuration is now called the *Standard Oracle ASM* configuration, where each database instance depends on a separate ASM instance for managing its storage. Failure of an ASM instance means that any instance running on that server would also fail.

Oracle Database 12*c* has introduced the new Oracle Flex ASM configuration, where Oracle database instances can use Oracle ASM instances running on remote servers. This means that the failure of an Oracle ASM instance doesn't lead to the failure of the database instances being served by it, as the database instances will continue to be served by the remaining ASM instances in the Oracle Flex ASM cluster. We explain both the Standard ASM and Oracle Flex configurations in this chapter.

Introduction to Automatic Storage Management

Oracle ASM is the storage management solution that Oracle recommends as an alterative to traditional volume managers, file systems, and raw devices. ASM is a storage solution that provides volume manager and file system capabilities that are

https://t.me/learningnets

tightly integrated and optimized for Oracle databases. Oracle implements ASM via three key components:

- ASM instance
- ASM Dynamic Volume Manager (ADVM)
- ASM Cluster File System (ACFS)

The ASM Dynamic Volume Manager provides the functionality of a volume manager for the ASM Cluster File System. ASM Cluster File System first became available in Oracle Grid Infrastructure 11*g* Release 2.

ACFS helps customers in reducing overall IT budget costs because it eliminates the need for third-party cluster file systems, which is the most common requirement for clustered applications. Because ACFS is built as a true cluster file system, it can be used for non-Oracle enterprise applications as well.

ASM simplifies storage management by enabling you to do online reconfiguration and rebalancing of the ASM disks. ASM is highly optimized for the Oracle database because it distributes I/O operations onto all available disks in a disk group and provides raw disk performance.

In short, the ASM environment provides the performance of raw disk I/O with the easy management of a file system. It simplifies database administration by eliminating the need to manage potentially thousands of Oracle database files in a direct manner.

Physical Limits of ASM

The number of datafiles per database has consistently increased since Oracle version 7, in which only 1022 datafiles per database could be used. Current Oracle versions support 65,533 datafiles per tablespace, and managing thousands of files in a multi-database environment is a challenge. ASM simplifies storage management by enabling you to divide all available storage into disk groups. You can create separate disk groups for different performance requirements. For example, an ASM disk group can be created with low-performance hard disks to store archive data, whereas an ASM disk group with high-performance disks can be used for active data.

You manage a small set of disk groups, and ASM automates the placement of the database files within those disk groups. With ASM you can have 511 disk groups with 10,000 ASM disks in it, where each ASM disk can store up to 32 petabytes (PB) of data. A disk group can handle one million ASM files.

The maximum supported file size for a datafile in Oracle Database 12*c* is 128 terabytes (TB), whereas ASM supports up to a maximum of 320 exabytes (EB) for the entire storage system. Figure 6-1 compares the traditional data storage framework to that of ASM.

https://t.me/learningnets

Technet24

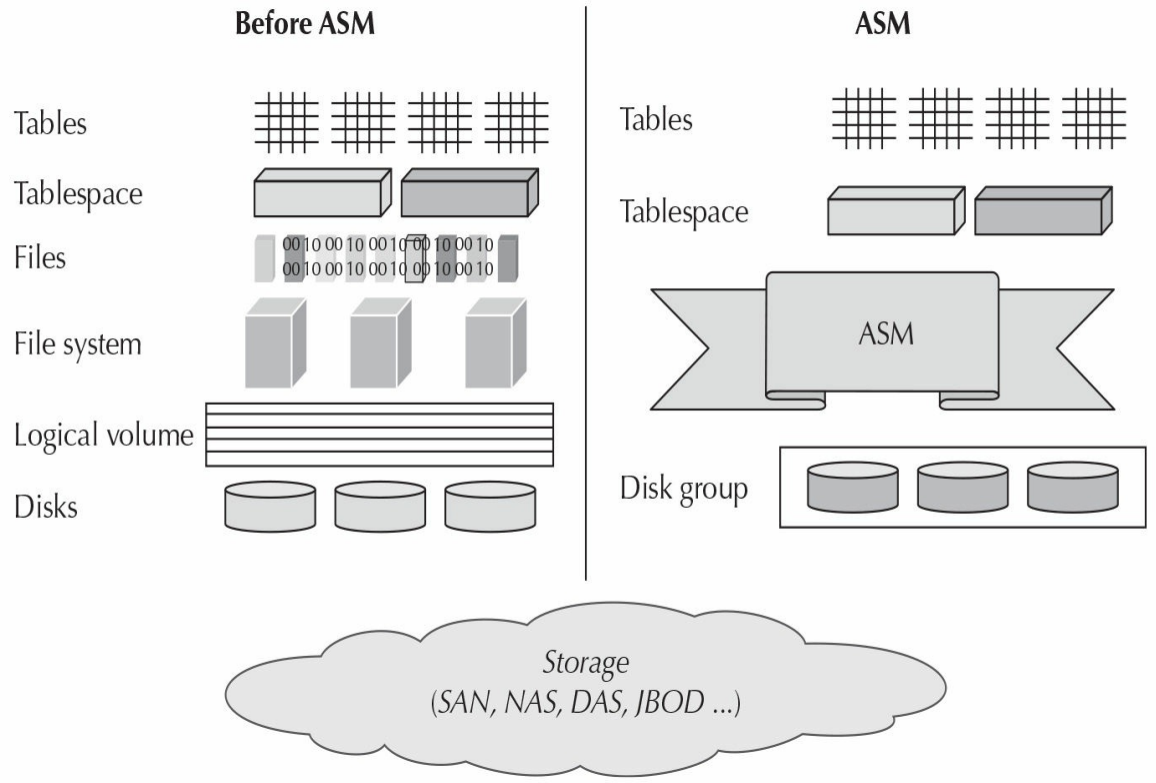


FIGURE 6-1. *Traditional framework vs. ASM*

ASM in Operation

Oracle ASM uses disk groups to store datafiles. Within each disk group, ASM exposes a file system interface for the database files. ASM divides a file into pieces and spreads these pieces evenly across all the disks, unlike other volume managers, which spread the whole volume onto the different disks. This is the key difference from the traditional striping techniques that use mathematical functions to stripe complete logical volumes independent of files or directories. Striping requires careful capacity planning at the beginning, because adding new volumes requires rebalancing and downtime.

With ASM, whenever you add or remove storage, ASM doesn't restripe all the data. It just moves an amount of data proportional to the amount of storage that you've added or removed, so as to redistribute the files evenly and maintain a balanced I/O load across the disks. This occurs while the database is active, and the process is transparent to the database and end-user applications.

The new *disk resync* enhancements in Oracle Database 12*c* let you recover very quickly from an instance failure. The disk resync checkpoint capability makes for fast instance recovery by letting the resync operation resume from where the resync process was stopped, instead of starting from the beginning.

ASM supports all files that you use with an Oracle database (with the exception of pfile and password files). Starting with Oracle 11*g* Release 2, ACFS can be used for non-Oracle application files. ASM also supports Oracle Real Application Clusters

https://t.me/learningnets

(Oracle RAC) and eliminates the need for a cluster Logical Volume Manager or a third-party cluster file system.

ASM is integrated with Oracle Clusterware and can read from a mirrored data copy in an extended cluster and improve I/O performance when used in an extended cluster. (An *extended cluster* is special-purpose architecture in Oracle RAC where the nodes are geographically separated.)

ASM provides the SQL interface for creating database structures such as tablespaces, controlfiles, redo logs, and archive log files. You specify the file location in terms of disk groups; ASM then creates and manages the associated underlying files for you. Other interfaces interact with ASM as well, such as ASMCMD, OEM, and ASMCA. SQL*Plus is the most commonly used tool to manage ASM among DBAs. However, system and storage administrators tend to like the ASMCMD utility for managing ASM.

ASM Striping

Striping is a technique used for spreading data among multiple disk drives. A big data segment is broken into smaller units, and these units are spread across the available devices. The size of the unit that the database uses to break up the data is called the *data unit size* or *stripe size*.

Stripe size is also sometimes called the *block size* and refers to the size of the stripes written to each disk. The number of parallel stripes that can be written to or read from simultaneously is known as the *stripe width*. Striping can speed up operations that retrieve data from disk storage as it extends the power of the total I/O bandwidth. This optimizes performance and disk utilization, making manual I/O performance tuning unnecessary.

ASM supports two levels of striping: fine striping and coarse striping. Fine striping uses 128KB as the stripe width, and coarse striping uses 1MB as the stripe width. Fine striping can be used for files that usually do smaller reads and writes. For example, online redo logs and controlfiles are the best candidates for fine striping when the reads or writes are small in nature.

Stripe and Mirror Everything

Oracle's Stripe and Mirror Everything (SAME) strategy is a simple and efficient technique used in managing a high volume of data. ASM implements the Oracle SAME methodology, in which the database stripes and mirrors all types of data across all available drives. This helps the database to evenly distribute and balance the I/O load across all disks within the disk group.

Mirroring provides the much-required fault tolerance for the database servers, and striping provides performance and scalability to the database. ASM implements the SAME methodology to stripe and mirror files inside the ASM dynamic volume.

Character Devices and Block Devices

You can classify any direct attached or networked storage device as a *character device*

https://t.me/learningnets

Technet24

or a *block device*. A character device holds only one file. Normally, raw files are placed on character devices. The location of the raw devices is platform dependent, and they are not visible in the file system directory. A block device is a type of character device, but it holds an entire file system.

If you use ASMLib (or the new ASM Filter Driver) as a disk API, ASM supports block devices. ASM presents an ASM dynamic volume device file as a block device to the operating system.

Oracle is obsoleting ASMLib and often doesn't recommend it for its customers. Oracle recommends that you instead use the new ASM Filter Driver, which we explain in detail in this chapter.

Storage Area Network

A *storage area network (SAN)* is the networked storage device connected via uniquely identified host bus adapters (HBAs). The storage is divided into logical unit numbers (LUNs), and each LUN is logically represented as a single disk to the operating system.

In Automatic Storage Management, the ASM disks are either LUNs or disk partitions. They are logically represented as a raw device to the ASM. The name and path of the raw device is dependent on the operating system. For example, in the Sun operating system, the raw device has the name *cNtNdNsN*:

- ***cN*** is the controller number.
- ***tN*** is the target ID (SCSI ID).
- ***dN*** is the disk number, which is the LUN descriptor.
- ***sN*** is the slice number or partition number.

So when you see a raw partition in the Sun operating system listed as *c0t0d2s3*, you'll know that the device is the third partition in the second disk connected to the first controller's first SCSI port. HP-UX does not expose the slice number in the raw device. Instead, HP uses the *cNtNdN* format for the raw partitions. Note that there is no concept of slice designation, because HP-UX does not support slices. (HP-UX Itanium does have slice support, but HP does not support the use of this feature.) The entire disk must be provided to ASM.

A typical Linux configuration uses *straight disks*. Raw functionality was an afterthought. However, Linux imposes a limit of 255 possible raw devices, and this limitation is one of the reasons for the development of Oracle Cluster File System (OCFS) and the use of ASMLib. Raw devices are typically stored in /dev/raw and are named raw1 to raw255.

ASM Building Blocks

ASM is implemented as a special kind of Oracle instance with the same structure and its own System Global Area (SGA) and background processes. Additional background

https://t.me/learningnets

processes in ASM manage storage and disk-rebalancing operations. You can consider the various components we discuss next to be the building blocks of ASM.

ASM Instance

An *ASM instance* is an Oracle instance that manages the metadata for disk groups, ADVM (ASM Dynamic Volume Manager), and ACFS (ASM Cluster File System). *Metadata* refers to information, such as the disks that belongs to a disk group and the space available in a disk group, that Oracle ASM uses to manage disk groups. ASM stores the metadata in the disk groups. All metadata modifications are done by an ASM instance to isolate failures. You can install and configure an ASM instance with either the Oracle Universal Installer (OUI) or the Oracle ASM Configuration Assistant (ASMCA).

You can use an spfile or a traditional pfile to configure an Oracle ASM instance. An Oracle ASM instance looks for its initialization parameter file in the following order:

1. The initialization parameter file location you specify in the Grid Plug and Play (GPnP) profile.
2. If you haven't set the location in the GPnP profile, then the Oracle ASM instance looks for the parameter file in the following locations:
   - The spfile in the Oracle ASM instance home
   - the pfile in the Oracle ASM instance home

NOTE

*You can administer Oracle ASM initialization parameter files with SQL*Plus, ASMCA, and ASMCMD commands*.

Database instances connect to an ASM instance to create, delete, resize, open, or close files, and database instances read/write directly to disks managed by the ASM instance. You can have only one ASM instance per node in an Oracle RAC cluster. Sometimes this can be a disadvantage for a large Oracle RAC cluster (such as an eight-node Oracle RAC, where you need to maintain eight separate ASM instances, in addition to eight "user" instances). An ASM instance failure kills the attached database instances in the local node.

An ASM instance is just like an Oracle database instance, which consists of System Global Area (SGA) and background processes. Just like the buffer cache in an Oracle database instance, an ASM instance has a special cache called *ASM cache* to read and write blocks during rebalancing operations.

Apart from ASM cache, an ASM instance's System Global Area has a shared pool, large pool, and free memory area. Oracle internally uses Automatic Memory Management, and you will hardly need to tune an Oracle ASM instance.

https://t.me/learningnets

Technet24

ASM instances use memory and background processes in the same way as a regular Oracle instance, but ASM's SGA allocation is minimal and doesn't impose a heavy overhead on the server running the ASM instance. The job of the ASM instance is to mount the disk groups and thus make the Oracle ASM files available to the database instances.

An Oracle ASM instance doesn't maintain a data dictionary, so you can only connect to an ASM instance through the system privileges SYSDBA, SYSADM, and SYSOPER, which are implemented by operating system groups for the Oracle Grid Infrastructure owner provided to the Oracle Universal Installer when you install Oracle Grid Infrastructure.

ASM Listener

The *ASM listener* is similar to the Oracle database listener, which is a process that's responsible for establishing a connection between database server processes and the ASM instance. The ASM listener process *tnslsnr* is started from the $GRID_HOME/bin directory and is similar to Oracle Net Listener.

The ASM listener also listens for database services running on the same machine, so there is no need to configure and run a separate Oracle Net Listener for the database instance. Oracle will, by default, install and configure an ASM listener on port 1521, which can be changed to a nondefault port while Oracle Grid Infrastructure is being installed, or later on.

Disk Groups

A *disk group* consists of disks that the database manages together as a single unit of storage, and it's the basic storage object managed by Oracle ASM. As the primary storage unit of ASM, this collection of ASM disks is self-describing, independent of the associated media names.

Oracle provides various ASM utilities such as ASMCA, SQL statements, and ASMCMD to help you create and manage disk groups, their contents, and their metadata.

Disk groups are integrated with Oracle Managed Files and support three types of redundancy: external, normal, and high. Figure 6-2 shows the architecture of the disk groups.

https://t.me/learningnets

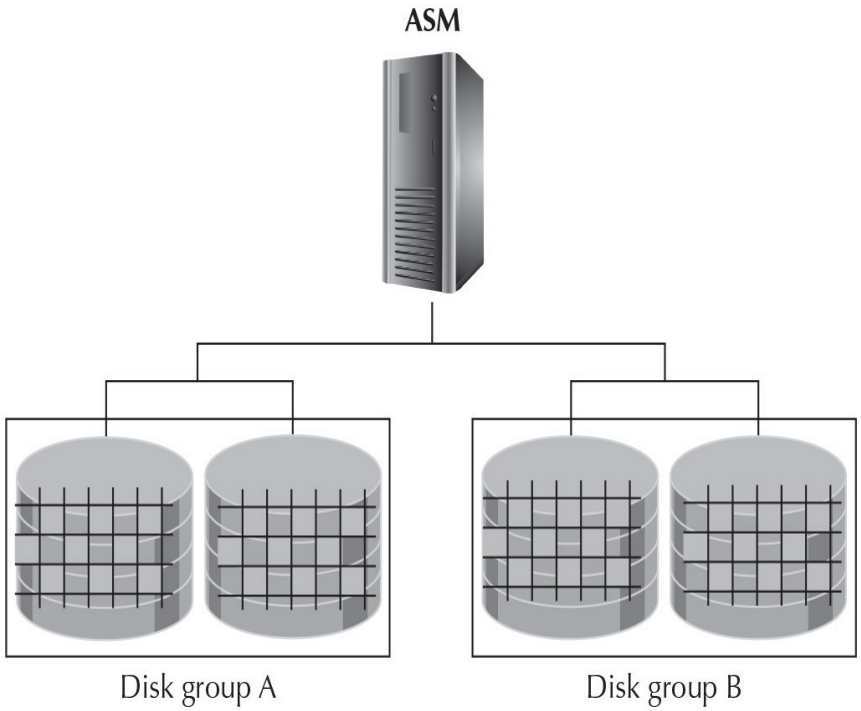


FIGURE 6-2. *Disk groups in ASM*

The disks in a disk group are called *ASM disks*. In ASM, a disk is the unit of persistent storage for a disk group and can be disks or partitions from a storage array or an entire disk, or partitions of a disk group. Disks can also be logical volumes or network-attached files that are part of a Network File System (NFS).

The disk group in a typical database cluster is part of a remote shared-disk subsystem, such as a SAN or network-attached storage (NAS). The storage is accessible via the normal operating system interface and must be accessible to all nodes.

Oracle must have read and write access to all the disks, even if one or more servers in the cluster fails. On Windows operating systems, an ASM disk is always a partition. On all other platforms, an ASM disk can be a partition of a logical unit number (LUN) or any NAS device.

NOTE

*Oracle doesn't support raw or block devices as shared storage for Oracle RAC in Oracle 11*g *Release 2 onward.*

Allocation Unit

ASM disks are divided into a number of units or storage blocks that are small enough not to be hot. The allocation unit of storage is large enough for efficient sequential

https://t.me/learningnets

Technet24

access. The allocation unit defaults to 1MB in size.

Oracle recommends 4MB as the allocation unit for most configurations. ASM allows you to change the allocation unit size, but you don't normally need to do this, unless ASM hosts a very large database (VLDB). You can't change the size of an allocation unit of an existing disk group.

Significance of 1MB in ASM

Various I/O clients and their usage models are discussed in this section to provide insight into I/O block size and parallelization at the Oracle level. The log writer writes the very important redo buffers into log files. These writes are sequential and synchronous by default. The maximum size of any I/O request is set at 1MB on most platforms. Redo log reads are sequential and issued either during recovery or by LogMiner or log dumps. The size of each I/O buffer is limited to 1MB in most platforms. There are two asynchronous I/Os pending at any time for parallelization.

DBWR (Database Writer) is the main server process that submits asynchronous I/Os in a big batch. Most of the I/O request sizes are equal to the database block size. DBWR also tries to coalesce the adjacent buffers in the disk up to a maximum size of 1MB whenever it can and submits them as one large I/O. The Kernel Sequential File I/O (ksfq) provides support for sequential disk/tape access and buffer management. The ksfq allocates four sequential buffers by default. The size of the buffers is determined by the DB_FILE_DIRECT_IO_COUNT parameter, which is set to 1MB by default. Some of the ksfq clients are Datafile, redo logs, RMAN, Archive log file, Data Pump, Oracle Data Guard, and the File Transfer package.

Volume Allocation Unit

Like the allocation unit of the ASM disk group, the *volume allocation unit* is the smallest storage unit that ASM allocates for space inside the ASM dynamic volume. ASM allocates space in multiples of the volume allocation units. The size of the volume allocation unit is related to the size of the allocation unit of the ASM disk group. By default, ASM creates a volume allocation unit of 64MB inside the ASM disk group created with a default allocation unit size of 1MB.

Failure Groups

Failure groups define ASM disks that share a common potential failure mechanism. A failure group is a subset of disks in a disk group dependent on a common hardware resource whose failure the database must tolerate. Failure groups are relevant only for normal or high redundancy configurations. Redundant copies of the same data are placed in different failure groups. An example might be a member in a stripe set or a set of SCSI disks sharing the same SCSI controller. Failure groups are used to

https://t.me/learningnets

determine which ASM disks should be used for storing redundant copies of data. By default, each disk is an individual failure group. Figure 6-3 illustrates the concept of a failure group.

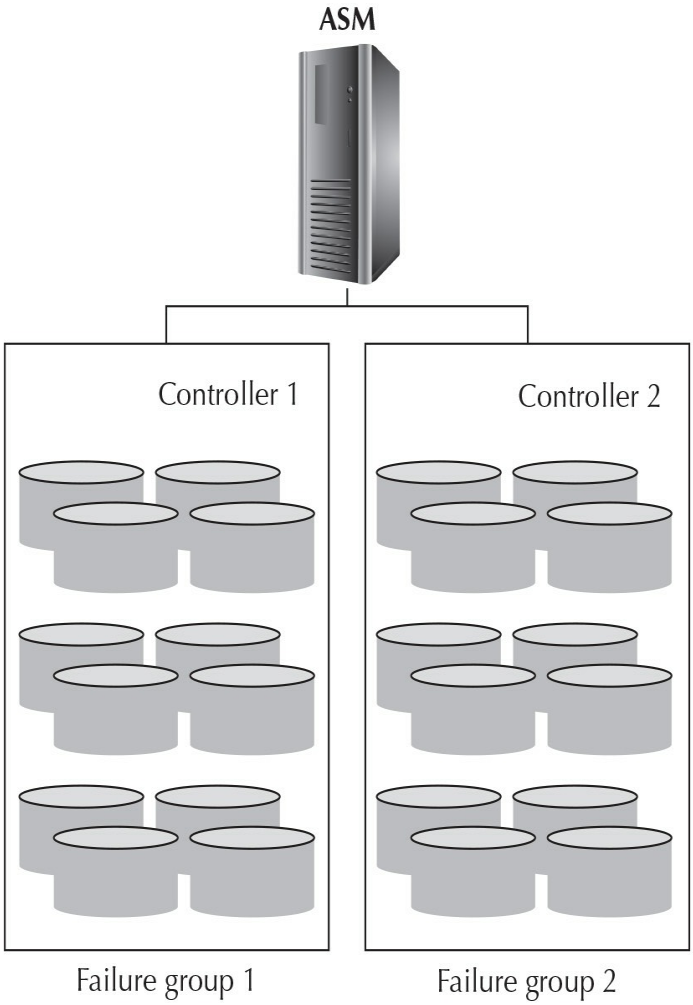


FIGURE 6-3. *Failure groups*

NOTE

Failure groups apply only to normal and high redundancy disk groups and are not applicable for external redundancy disk groups.

For example, if two-way mirroring is specified for a file, ASM automatically stores redundant copies of file extents in separate failure groups. You define the failure groups in a disk group when you create or alter the disk group.

Mirroring, Redundancy, and Failure Group Options

ASM mirroring is more flexible than operating system mirrored disks because ASM

https://t.me/learningnets

Technet24

mirroring enables you to specify the redundancy level on a per-file basis rather than on a volume basis. Internally, mirroring takes place at the extent level.

If a file is mirrored, depending on the redundancy level set for the file, each extent has one or more mirrored copies, and mirrored copies are always kept on different disks in separate failure groups.

You configure the failure group for a disk group when you create the disk group or by altering the disk group. The five types of disk groups based on their redundancy level are external, normal, high, flex, and extended redundancy. Of these, failure groups apply to the normal, high, flex, and extended redundancy disk groups.

The following table describes the available mirroring options that ASM supports on a per-file basis:

ASM Redundancy	Mirroring Level	Description
External	No mirroring (unprotected)	No mirroring from ASM. You can use this when mirroring is implemented at the storage or disk level.
Normal	Two-way mirroring	Each extent has one mirrored copy in different failure groups.
High	Three-way mirroring	Each extent has two mirrored copies in different failure groups.
Flex redundancy	Two-way, three-way, unprotected (none)	If you have five or more failure groups, the database tolerates two disk failures. For three or four failure groups, it tolerates one disk failure.
Extended redundancy	Two-way, three-way, unprotected (none)	If the site has five or more failure groups, the database tolerates two disk failures within a site. For three or four failure groups, the site will tolerate one disk failure.

ASM doesn’t require external volume managers or external cluster file systems for disk management; these are not recommended because the functionalities provided by the ASM will conflict with the external volume managers.

The number of failure groups you choose to create will determine how many failures the database tolerates without losing data.

Oracle provides Oracle Managed Files (OMF), which the database will create and manage for you. ASM extends the OMF capability. With ASM you get additional benefits such as mirroring and striping.

How ASM Deals with Disk Failures

Depending on the redundancy level you configure and how you define your

https://t.me/learningnets

failure groups, one of the following will happen when one or more disks fail in an ASM-managed file system:

- Oracle will take the offline and drop them, keeping the disk group mounted and usable. Because the data is mirrored, there's no data loss and the data continues to remain available. Once the disk drop is complete, Oracle ASM will rebalance the disk group to provide the original redundancy for the data that was on the failed disk.
- Oracle will dismount the disk group, which results in the unavailability of the data. Oracle recommends at least three failure groups for normal redundancy disk groups and five failure groups for high redundancy disk groups to maintain the necessary number of copies of the Partner Status Table (PST). This also offers protection against storage hardware failures.

Even Read for Disk Groups

Starting with the Oracle Database 12.1 release, the *even read* feature is enabled by default. As its name indicates, even read for a disk group distributes data evenly across all the disks in the group. The database directs each read request to the least-loaded disk.

ASM Files

Files that the Oracle database writes on ASM disks (which are part of an ASM disk group) are called *ASM files*. Each ASM file can belong to only a single Oracle ASM group. Oracle can store the following files in an ASM disk group:

- Datafiles
- Controlfiles
- Temporary files
- Online and archived redo log files
- Flashback logs
- RMAN backups
- Spfiles
- Data Pump dump sets

ASM filenames normally start with a plus sign (+). Although ASM automatically generates the names, you can specify a meaningful, user-friendly alias name (or alias) for ASM files. Each ASM file is completely contained within a single disk group and evenly divided throughout all ASM disks in the group.

Voting files and the Oracle Cluster Registry (OCR) are two key components of Oracle Clusterware. You can store both OCR and voting files in Oracle ASM disk groups.

https://t.me/learningnets

Technet24

Oracle divides every ASM disk into multiple allocation units. An *allocation unit (AU)* is the fundamental storage allocation unit in a disk group. A file extent consists of one or more allocation units, and each ASM file consists of one or more file extents. You set the AU size for a disk group by specifying the AU_SIZE attribute when you create a disk group. The value can be 1, 2, 4, 8, 16, 32, or 64MB. Larger AU sizes are beneficial for applications that sequentially read large chunks of data.

You can't change the data extent size because Oracle will automatically increase the size of the data extent (variable extent sizes) when an ASM file increases in size. The way Oracle allocates the extents is similar to how it manages locally managed tablespace extent sizes when an auto-allocate mode is used. Variable extent sizes depend on the AU size. If the disk group has an AU size of less than 4MB, the extent size is the same as the disk group AU size for the first 20,000 extent sets. The extent size goes up to 4*AU size for the next 20,000 extent sets; for anything over 40,000 extent sizes, the extent size will be 16*AU size. If the disk group AU size is at least 4MB, the extent sizing (that is, whether it is the same as the disk group AU size, 4*AU size, or 16*AU size) is figured using the application block size.

Disk Partners

Disk partners limit the possibility of two independent disk failures that result in losing both copies of a virtual extent. Each disk has a limited number of partners, and redundant copies are allocated on partners.

ASM automatically chooses the disk partner and places the partners in different failure groups. In both Oracle Exadata and the Oracle Database Appliance, there's a fixed partnership. In other cases, the partnering is not fixed and is decided based on how the disks are placed in their failure groups. Partner disks should be the same in size, capacity, and performance characteristics.

ASM Dynamic Volume Manager and ASM Cluster File System

ASM Cluster File System (ACFS) is a general-purpose cluster file system and supports non-Oracle application files. ACFS extends ASM functionality to support all customer files, including database files, application files, executables, alert logs, and configuration files. Oracle ASM Dynamic Volume Manager is the foundation for ACFS.

The ASM disk group is the basic element in creating ACFS because the disk group contains the ASM dynamic volume device files, which the ASM Dynamic Volume Manager (ADVM) presents to the operating system. Once these files are presented to the operating system, you can use the traditional mkfs utility to build and mount the ASM Cluster File System on the ASM dynamic volume device files, which are block devices.

ADVM supports ext3, ACFS, and NTFS file systems, and users can use the advmutil utility on Windows systems to build the file system on ADVM.

ADVM is installed in the Oracle Grid Infrastructure home as part of the Oracle Grid Infrastructure installation, and Oracle Clusterware loads the oracleadvm, oracleoks,

https://t.me/learningnets

and oracleacfs modules automatically on system reboot. These modules play a critical role in supporting the ADVM and ACFS functionalities of Oracle ASM. You can use ASM tools such as ASMCA, ASMCMD, Oracle Enterprise Manager, and SQL*Plus to create ASM dynamic volumes. You can use ASMCA for all ASM management operations.

There are some restrictions in terms of ACFS usage; for example, you can't use ACFS to create a root or boot directory because ACFS drivers are loaded by Oracle Clusterware. Similarly, you can't use ACFS to store Oracle Grid Infrastructure software. ACFS is a general file system, and you can perform I/O on an ACFS file system just as you do on any other third-party file system.

Oracle ACFS is integrated with Cluster Synchronization Services. In case of failure, Cluster Synchronization Services will fence the failed cluster node from the active cluster to avoid any possible data corruption.

Oracle uses various background processes to provide ADVM. These processes are explained in the section "ASM Background Processes," later in this chapter.

Once you create your disk groups and mount them on the ASM instance, you can use Oracle ASMCMD volume management commands to manage ADVM. For example, the **volcreate** command creates an ADVM volume in a disk group, and the **voldelete** command will delete the volume. You can use the **volinfo** command to display information about the ADVM volumes. Here's an example that shows how to use the **volcreate** command to create a volume group:

```
ASMCMD [+] > volcreate -G DGA -S 10g --WIDTH 1m --COLUMN 8 VOLUME1
```

The previous command creates an Oracle ADVM volume in a disk group. Once you create the volume, you can use the volume device associated with the dynamic volume to host an Oracle ACFS file system.

You use the ALTER DISKGROUP VOLUME statement to manage Oracle ADVM volumes, such as adding, modifying, disable, enabling, and dropping ADVM volumes. The following examples show how to issue the ALTER DISKGROUP VOLUME statement to manage Oracle ADVM volumes:

```
SQL> ALTER DISKGROUP data ADD VOLUME volume1 SIZE 10G;
Diskgroup altered.
SQL> ALTER DISKGROUP data RESIZE VOLUME volume1 SIZE 15G;
Diskgroup altered.
SQL> ALTER DISKGROUP data DISABLE VOLUME volume1;
Diskgroup altered.
SQL> ALTER DISKGROUP data ENABLE VOLUME volume1;
Diskgroup altered.
SQL> ALTER DISKGROUP ALL DISABLE VOLUME ALL;
Diskgroup altered.
SQL> ALTER DISKGROUP data DROP VOLUME volume1;
Diskgroup altered.
```

https://t.me/learningnets

Technet24

To resize a volume that hosts an Oracle ACFS file system, you must use the **acfsutil resize** command instead of the ALTER DISKGROUP VOLUME statement.

ACFS Snapshots

An exciting feature bundled with ACFS is the ability to create snapshots of the ASM dynamic volume; this allows users to recover from the deleted files or even to a point in time in the past.

ACFS snapshots are point-in-time copies of the ACFS file system, which is read-only and taken online. To perform point-in-time recovery or even to recover deleted files, you need to know the current data and changes made to the file. When you create a snapshot, ASM stores metadata such as the directory structure and the name of the files in the ASM dynamic volume. Along with the metadata, ASM stores the location information of all the data blocks of the files that have never had any data and the actual data blocks with data.

Once the snapshot is created, to maintain the consistency of the snapshot, ASM updates it by recording the file changes. Starting with Oracle Grid Infrastructure 12.1, you can create a snapshot from a previously created ACFS snapshot.

The ASM Cluster File System supports POSIX and X/Open file system APIs, and you can use traditional UNIX commands such as cp, cpio, ar, access, dir, diff, and so on. ACFS supports standard operating system backup tools, Oracle secure backup, and third-party tools such as storage array snapshot technologies.

Managing Oracle ASM Files and Directories

As we explained earlier, ASM supports most file types that an Oracle database needs, such as controlfiles, datafiles, and redo log files. In addition, you can store RAC-specific files such as the Oracle Cluster Registry files and voting files in ASM. You can also store Oracle ASM Dynamic Volume Manager volumes in ASM.

ASM Filenames

You can use a fully qualified filename or an alias to refer to Oracle ASM files.

Using a Fully Qualified Filename

Oracle gives new ASM files a filename generated by Oracle Managed Files (OMF) called the *fully qualified filename*. This filename represents the full path for the file in the ASM file system.

A fully qualified ASM file has the following form:

```
+diskgroup/dbname/filetype/filetypetag.file.incarnation
```

Here’s an example:

```
+daa/orcl/controlfile/current.123.653486242
```

https://t.me/learningnets

Oracle uses default templates for creating different types of files in ASM. These templates provide the attributes for creating a specific type of file. You thus have the DATAFILE template for creating datafiles, the CONTROLFILE template for creating controlfiles, and the ARCHIVELOG template for creating archive log files.

Alias Oracle ASM Filenames

Instead of using a fully qualified name, you can use an Alias Oracle ASM filename to refer to ASM files or for creating files. An alias name starts with the name of the disk group the file belongs to, preceded by a plus sign, as shown here:

```
+data/orcl/control_file1
+fra_recover.second.dbf
```

When you choose to create an ASM with an alias filename, internally, Oracle gives the file a fully qualified filename as well. You can add an alias name for an ASM file with a fully qualified system-generated filename by using the ALTER DISKGROUP command:

```
SQL> ALTER DISKGROUP data ADD ALIAS '+data/orcl/second.dbf'
                  FOR '+data/orcl/datafile/mytable.123.123456789';
```

Creating and Referencing ASM Files

You can create ASM tablespaces and files by specifying disk group locations using the Oracle Managed Files feature, or you can explicitly specify the filename in the file creation statement.

Using a Default File Location (OMF Based)

You can use the OMF feature to specify a disk group as the default location for creating ASM files. You can configure the following initialization parameters to specify the default disk group location in which to create various files:

- **DB_CREATE_FILE_DEST** Specifies the default disk group location for data and temp files
- **DB_CREATE_ONLINE_LOG_DEST_*n*** Specifies the default disk group location for redo log files and controlfiles
- **DB_RECOVERY_FILE_DEST** Specifies the default disk group for a fast recovery area
- **CONTROL_FILES** Specifies the disk group for creating controlfiles

You can configure a default disk group with one of the initialization parameters, as shown in the following example, which sets “data” as the default disk group for creating datafiles and temp files:

https://t.me/learningnets

Technet24

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST ='+data';
```

Once you configure the OMF-related initialization parameters, it's easy to create a tablespace that uses ASM files, as shown here:

```
SQL> CREATE TABLESPACE mytblspace;
```

Let's say you've set the DB_CREATE_FILE_DEST parameter to point to the ASM disk group "data." ASM will automatically create the datafile for the mytblspace tablespace on ASM disks in the disk group "data."

Explicitly Specifying Oracle ASM Filenames

The other way to create an ASM file is to specify the filename when you create the file. Here's an example:

```
SQL> CREATE TABLESPACE mytblspace DATAFILE '+data' SIZE 2000m AUTOEXTEND ON;
```

This statement will create the tablespace mytblspace with one datafile stored in the disk group "data."

Dropping an ASM File from a Disk Group

Use the ALTER DISKGROUP command to drop a file from a disk group. In the following example, we drop the file by referring to it with its alias name, but you can also use the fully qualified filename:

```
SQL> ALTER DISKGROUP data DROP FILE '+data/payroll/compensation.dbf'
```

Managing Disk Group Directories

ASM disk groups use a hierarchical directory structure (automatically generated by Oracle) to store ASM files, as shown here:

```
+DATA/ORCL/controlfile/Current.123.123456789
```

Here's an explanation of this fully qualified filename:

- The + sign refers to the root of the ASM file system.
- The DATA directory is the parent directory for files in the data disk group.
- The ORCL directory is the parent directory for all files in the orcl database.
- The controlfile directory stores all controlfiles in the orcl database.

Creating Directories

To store your aliases, you can create your own hierarchical directory structures. You create directories below the disk group level, and you must ensure that a parent directory exists before creating subdirectories or aliases. Here's an example:

https://t.me/learningnets

```
SQL> ALTER DISKGROUP data ADD directory '+DTA/ORCL';
```

Dropping Directories

You can delete a directory with the ALTER DISKGROUP statement:

```
SQL> ALTER DISKGROUP data DROP DIRECTORY '+DATA/MYDIR' force;
```

The **force** option deletes a directory that contains alias names. Note that you can't remove a system-generated directory.

TIP

The DBMS_FILE_TRANSFER package contains procedures that enable you to copy ASM files between databases or to transfer binary files between databases.

ASM Administration and Management

Administering an ASM instance is similar to managing a database instance, but it involves fewer tasks. An ASM instance doesn't require a database instance to be running for you to administer it. An ASM instance does not have a data dictionary because metadata is not stored in a dictionary. ASM metadata is small, and Oracle stores it in the ASM disk headers. ASM instances manage the disk group metadata and provide the layout of the datafiles in ASM to the Oracle instance.

The metadata that ASM stores in the disk groups includes the following types:

- Information relating to the disks that belong to each disk group
- Names of the files in each disk group
- Amount of space available in a disk group
- Location of the extents in each disk group
- Information pertaining to the Oracle ASM volume

You can use SQL*Plus to perform all ASM administration tasks in the same way you work with a normal RDBMS instance.

NOTE

*To administer ASM with SQL*Plus, you must set the ORACLE_SID environment variable to the ASM SID before you start SQL*Plus. The default ASM SID for a single-instance database is +ASM, and the default SID for ASM on Oracle RAC nodes is*

https://t.me/learningnets

Technet24

+ASMnode#. ASM instances do not have a data dictionary, so you must use operating system authentication and connect as SYSASM or SYSOPER. When connecting remotely through Oracle Net Services, you must use a password file for authentication.

You can check the integrity of Oracle ASM across the cluster using the Cluster Verification Utility (CVU), as shown here:

```
$ cluvfy comp asm all
```

This command checks all nodes in the cluster. You can check a single node or a set of nodes instead by replacing the **all** parameter with **[-n node_list]**.

Managing an ASM Instance

An ASM instance is designed and built as a logical extension of the database instances, and both it and a database instance share the same mechanism of instance management.

As with the database instance parameter file, an ASM instance has a parameter file called a *registry file* that's stored in the <cluster name>/ASMPARAMETERFILE directory of the ASM disk group specified to store the OCR and voting disk during installation of Oracle Grid Infrastructure.

The SID for an ASM instance defaults to +ASM for a single-instance database and +ASM*node#* for Oracle RAC nodes. The rules for filenames, default locations, and search orders that apply to the database initialization parameter files also apply to the ASM initialization parameter files. However, ASM instances come with a separate set of initialization parameters that you can't set for a database instance.

Administering the ASM Instance

You start an ASM instance similar to how you start an Oracle database instance. When connecting to an ASM instance with SQL*Plus, you must set the ORACLE_SID environment variable to the ASM SID.

The initialization parameter file for an ASM instance, which can be a server parameter file also known as an ASM parameter file, must contain the parameter INSTANCE_TYPE = ASM to signal to the Oracle executable that an ASM instance is starting and not a database instance.

Apart from the disk groups specified by the ASM_DISKGROUPS initialization parameter, ASM will automatically mount the disk groups used to store the voting disk, OCR, and the ASM parameter file.

Startup Modes The STARTUP command starts an ASM instance with the set of memory structures, and it mounts the disk groups specified by the initialization parameter ASM_DISKGROUPS. If you leave the value of the ASM_DISKGROUPS parameter blank, the ASM instance starts and warns you that no disk groups were mounted. You can then mount disk groups with the ALTER DISKGROUP MOUNT

https://t.me/learningnets

command (similar to the ALTER DATABASE MOUNT command).

The following table describes the various startup modes of the ASM instance:

Startup Mode	Description
NOMOUNT	Starts up the ASM instance without mounting any disk groups.
MOUNT	Starts up the ASM instance and mounts the disk groups.
OPEN	Mounts the disk groups and allows connections from the database. This is the default startup mode.
FORCE	Starts up MOUNT after a SHUTDOWN ABORT.

Other startup clauses have comparable interpretations for ASM instances as they do for database instances. For example, RESTRICT prevents database instances from connecting to this ASM instance. OPEN is invalid for an ASM instance. NOMOUNT starts up an ASM instance without mounting any disk group.

NOTE

An ASM instance doesn't have any data dictionary, and the only possible way to connect to an ASM instance is via operating system privileges such as SYSDBA, SYSASM, and SYSOPER.

You specify the operating system groups OSASM, OSDBA, and OSOPER during the installation of the Oracle Grid Infrastructure, and these operating system groups implement the SYSASM, SYSDBA, and SYSOPER privileges for the Oracle ASM instance. Oracle allows password-based authentication to connect to an ASM instance, which requires that the ASM initialization parameter REMOTE_LOGIN_PASSWORDFILE be set to a value other than NONE.

By default, the Oracle Universal Installer will create a password file for an ASM instance and new users will automatically be added to the password file. Following this, users can connect to an ASM instance over the network using Oracle Net Services.

Oracle connects as SYSDBA to an ASM instance when connecting via ASMCMD. You can use SQL*Plus to connect to the instance and run simple SQL commands such as **show sga** and **show parameter <parameter name>**. Here's an example:

```
SQL> show sga
Total System Global Area  283930624 bytes
Fixed Size                  2225792 bytes
Variable Size             256539008 bytes
ASM Cache                  25165824 bytes
```

You can query the V$PWFILE_USERS dynamic view to list users in the password

https://t.me/learningnets

Technet24

file. Another possible way to list users in the password file is to use the **lspwusr** command from the ASMCMD command prompt. ASMCMD can be used to create and manage a password file manually.

Shutting Down an ASM Instance Shutting down an ASM instance is similar to how you shut down any other Oracle database instance. You must, however, first shut down the database instances using the ASM instance before you can shut down the ASM instance.

When you issue a NORMAL, IMMEDIATE, or TRANSACTIONAL shutdown command, ASM waits for any in-progress SQL to complete. Once all ASM SQL is completed, all disk groups are dismounted and the ASM instance is shut down in an orderly fashion. If any database instances are connected to the ASM instance, the SHUTDOWN command returns an error and leaves the ASM instance running.

When you issue the SHUTDOWN ABORT command, the ASM instance is immediately terminated. The instance doesn't dismount the disk groups in an orderly fashion. The next startup requires recovery of ASM—similar to RDBMS recovery—to bring the disk groups into a consistent state. The ASM instance also has components similar to undo and redo (details are discussed later in the chapter) that support crash recovery and instance recovery.

If any database instance is connected to the ASM instance, the database instance aborts because it does not get access to the storage system that is managed by the ASM instance.

You can start and stop an ASM instance with the SQL*Plus, ASMCA, ASMCMD, and SRVCTL utilities. SRVCTL uses startup and shutdown options registered in the OCR to start or stop an ASM instance. The following are examples of starting and stopping the ASM instance +ASM1 on cluster node racnode01 using the SRVCTL utility.

NOTE

To manage ASM instances, use the SRVCTL executable in the Oracle Grid Infrastructure home. You can't use the SRVCTL executable located in the Oracle RAC or Oracle Database home for managing ASM.

Use the following command to start the ASM instance on all nodes in a cluster:

```
$srvctl start asm
```

Use the following command to start the ASM instance on a single node in the cluster (in this case, the racnode01 node):

```
$ srvctl start asm -node racnode01
```

https://t.me/learningnets

The following command shows how to stop the ASM instance on all active nodes in the cluster:

```
$ srvctl stop asm
```

You can stop the ASM instance on a specific node with the following command:

```
$ srvctl stop asm –node racnode1
```

In addition to starting up and shutting down an ASM instance, the SRVCTL utility also lets you perform other administrative tasks, as described in the following examples.

Use the following command to add an ASM instance:

```
$ srvctl add asm
```

Use the following command to remove an ASM instance:

```
$ srvctl remove asm
```

Use the following command to check the configuration of an ASM instance:

```
$ srvctl config asm –node node_name
```

Use the following command to display the status of an ASM instance:

```
$ srvctl status asm [-node node_name]
```

You can start and stop an ASM instance using SQL*Plus, just as you do for a regular Oracle Database instance. In order to connect to the ASM instance, make sure to set the ORACLE_SID environment variable to the Oracle ASM SID. The default SID for ASM on a RAC node is +ASM *node_number*, where *node_number* is the number of the node hosting the ASM instance. Here's how you start up an ASM instance with SQL*Plus:

```
$ sqlplus /nolog
$ startup
```

This command mounts the ASM disk groups. Therefore, MOUNT is the default option. You can also specify the FORCE, NOMOUNT, or RESTRICT parameter with the STARTUP command.

Once the instance starts up, you mount the disk groups with the ALTER DISKGROUP…MOUNT command. This command mounts all the disk groups specified by the ASM_DISKGROUPS initialization parameter.

To shut down an ASM instance with the SQL*Plus utility, use the following commands:

https://t.me/learningnets

Technet24

```
$ sqlplus /nolog
$ shutdown
```

Shutting down with just the SHUTDOWN command without any options is the same as a SHUTDOWN NORMAL command. You can also specify the IMMEDIATE, TRANSACTIONAL, or ABORT option with the SHUTDOWN command.

Be aware that you must first stop all the database instances connecting to the ASM instance, or else you'll receive an error, and the ASM instance will continue running. Similarly, if any Oracle ACFS file systems are mounted in Oracle ADVM volumes, you must first dismount those file systems before shutting down the ASM instance.

You can use the ASMCMD command-line utility to start and stop the ASM instance. Here are examples of starting and stopping an ASM instance using ASMCMD.

Use the following command to start the ASM instance in a mount state:

```
$ asmcmd
ASMCMD> startup --mount
```

Use the following command to shut down the ASM instance immediately:

```
$ asmcmd
ASMCMD> shutdown --immediate
```

NOTE

You can't start or shut down an ASM instance alone in the Oracle RAC database system if the OCR and voting disks are stored inside the ASM disk group. You must use the ***crsctl*** *command to start or stop the CRS, which will start/stop the ASM instance also.*

Oracle ASM File Access Control for Disk Groups

You can configure Oracle ASM File Access Control to restrict access to ASM disk groups to specific ASM clients that connect as SYSDBA. Clients in this instance are usually a database that uses ASM storage.

Using the access control, ASM determines the additional privileges it must give to databases that authenticate as SYSDBA on an Oracle ASM instance.

Setting Up ASM File Access Control You must set the ACCESS_CONTROL_ENABLED disk group attribute (default is false) and ACCESS_CONTROL_UMASK disk group attribute (default is 066) to set up Oracle ASM File Access Control for a disk group. The ACCESS_CONTROL_UMASK attribute determines which permissions are masked out when you create an ASM file;

https://t.me/learningnets

the permissions are for the owner of the file, users in the same user group as the owner, and others. The attribute applies to all files in a disk group.

The following example shows how to enable ASM File Access control for a disk group named data1, and how to set the umask permissions to 026, which means read-write to the owner, read to users in the group, and no access to others not in the same group as the owner:

```
ALTER DISKGROUP data1 SET ATTRIBUTE 'access_control.enabled' = 'true';
ALTER DISKGROUP data1 SET ATTRIBUTE 'access_control.umask' = '026';
```

Managing Oracle ASM File Access Control Managing ASM File Access Control involves adding and dropping user groups, adding and dropping members, adding and dropping users, and other related tasks. You use the ALTER DISKGROUP statement to perform all of these tasks. We show some typical access control management tasks here.

You can add an ASM user group to a disk group with the following ALTER DISKGROUP command. Before you run the command, make sure that the OS users you specify with the MEMBER clause are present in the disk group.

```
SQL> SELECT group_number, os_name FROM V$ASM_USER;
GROUP_NUMBER     OS_NAME
------------     ---------
           1     oracle1
           1     oracle2
...
SQL> ALTER DISKGROUP data ADD USERGROUP 'test_grp1'
     WITH MEMBER 'oracle1','oracle2';
```

You can add users to a user group with the following command:

```
SQL> ALTER DISKGROUP data MODIFY USERGROUP 'test_grp2' ADD MEMBER 'oracle2';
```

You can also add OS users to an ASM disk group so that the users have access privileges on the disk group:

```
SQL> ALTER DISKGROUP data ADD USER 'oracle1';
```

You can modify the permissions of an ASM file thus:

```
SQL> ALTER DISKGROUP data SET PERMISSION OWNER=read write, GROUP=read only,
     OTHER=none FOR FILE '+data/controlfile.f';
```

ASM-Related Dynamic Performance Views

Although ASM does not have a data dictionary, it provides dynamic performance views stored inside memory that can be used to extract metadata information from an ASM instance. Here are some short descriptions of the important dynamic

https://t.me/learningnets

Technet24

performance views. Refer to the Oracle documentation for a complete list of the ASM dynamic performance views.

- **V$ASM** This view displays the instance information of the ASM instance you are connected to.
- **V$ASM_DISK** This view shows every disk discovered by the Oracle ASM instance, by performing a disk discovery upon being queried.
- **V$ASM_DISKGROUP** This view lists the disk groups created inside the ASM along with metadata information such as free space, allocation unit size, and the state of the disk group.
- **V$CLIENT** This view shows the databases using the disk groups being managed by the ASM instance.
- **V$ASM_FILE** This view lists the files created within the disk groups listed in the V$ASM_DISKGROUP view.
- **V$ASM_ALIAS** This view lists the user-friendly name of the ASM files listed in the V$ASM_FILE view. This view is useful in identifying the exact name of the ASM file because the V$ASM_FILE view only lists the file number.
- **V$ASM_DISK_IOSTAT** This view lists the disk I/O performance statistics for each disk listed in the V$ASM_DISKGROUP view.
- **V$ASM_ACFSVOLUMES** This view lists the metadata information of the ASM dynamic volumes.
- **V$ASM_OPERATION** This view displays the current (long-running) operations, such as any rebalancing happening on the disk groups listed in the V$ASM_DISKGROUP view. This view is useful to monitor the rebalancing operation in ASM.
- **V$ASM_USER** This view shows the operating system user names of connected database instances and the names of the file owners.
- **V$ESTIMATE** This view shows an estimate of the work involved in performing ASM disk group rebalance and resync operations, without actually performing the operations.

ASM Background Processes

Because ASM is built using the RDBMS framework, the software architecture is similar to that of Oracle RDBMS processes. The ASM instance is built using various background processes, and a few of these processes specific to the ASM instance manage the disk groups, ASM Dynamic Volume Manager, and ASM Cluster File System in ASM. The following listing shows the background processes of the ASM instance having a SID of ASM:

https://t.me/learningnets

```
oracle    3419     1  0 19:36 ?        00:00:00 asm_pmon_+ASM1
oracle    3421     1  0 19:36 ?        00:00:00 asm_psp0_+ASM1
oracle    3435     1  4 19:36 ?        00:01:12 asm_vktm_+ASM1
oracle    3439     1  0 19:36 ?        00:00:00 asm_gen0_+ASM1
oracle    3441     1  0 19:36 ?        00:00:00 asm_mman_+ASM1
oracle    3445     1  0 19:36 ?        00:00:01 asm_diag_+ASM1
oracle    3447     1  0 19:36 ?        00:00:00 asm_ping_+ASM1
oracle    3461     1  0 19:36 ?        00:00:04 asm_dia0_+ASM1
oracle    3463     1  0 19:36 ?        00:00:04 asm_lmon_+ASM1
oracle    3465     1  0 19:36 ?        00:00:03 asm_lmd0_+ASM1
oracle    3467     1  0 19:36 ?        00:00:09 asm_lms0_+ASM1
oracle    3471     1  0 19:36 ?        00:00:01 asm_lmhb_+ASM1
oracle    3473     1  0 19:36 ?        00:00:00 asm_lck1_+ASM1
oracle    3475     1  0 19:36 ?        00:00:00 asm_dbw0_+ASM1
oracle    3477     1  0 19:36 ?        00:00:00 asm_lgwr_+ASM1
oracle    3484     1  0 19:36 ?        00:00:00 asm_ckpt_+ASM1
oracle    3492     1  0 19:36 ?        00:00:00 asm_smon_+ASM1
oracle    3500     1  0 19:36 ?        00:00:00 asm_lreg_+ASM1
oracle    3513     1  0 19:36 ?        00:00:00 asm_pxmn_+ASM1
oracle    3515     1  0 19:36 ?        00:00:00 asm_rbal_+ASM1
oracle    3522     1  0 19:36 ?        00:00:00 asm_gmon_+ASM1
oracle    3524     1  0 19:36 ?        00:00:00 asm_gcr0_+ASM1
oracle    3526     1  0 19:36 ?        00:00:00 asm_mmon_+ASM1
oracle    3528     1  0 19:36 ?        00:00:00 asm_mmnl_+ASM1
oracle    3552     1  0 19:36 ?        00:00:00 asm_lck0_+ASM1
oracle    3650     1  0 19:36 ?        00:00:00 asm_asmb_+ASM1
```

Look at the background processes closely, and you'll notice that the ones used in RDBMS instance management are similar to smon and pmon. However, additional processes, such as rbal and gmon, are specific to ASM instances.

Let's take a closer look at the ASM-specific processes. You will see additional background processes such as VDBG, VBG*n*, and VMB when ASM dynamic volumes are created in the ASM instance. Some important ASM background processes are explained here:

- **RBAL** This is the Rebalancing background process. It is responsible for the rebalancing operation and also coordinates the ASM disk discovery process.
- **GMON** This is the Group Monitor background process. It manages disk groups by marking a disk group "offline" or even by dropping the disk group.
- **ARB*n*** Whereas RBAL coordinates the rebalancing of disk groups, ARB*n* actually performs the rebalancing operations.
- **VMB** This is the Volume Membership Background process, which is responsible for cluster membership with the ASM instance. The ASM instance starts this background process when ASM dynamic volumes are created.
- **VDBG** This is the Volume Driver background process. It works with the

https://t.me/learningnets

Technet24

Dynamic Volume Driver to provide the locking and unlocking of volume extents. This is a vital process, and it will shut down the ASM instance if killed unexpectedly.

- **VBG*n*** This is the Volume Background process. VBG from the ASM instance communicates with operating system volume drivers. It handles messaging between ASM and the operating system.
- **XDMG** This is the Exadata Automation Manager. XDMG monitors all configured Exadata cells for state changes, such as a bad disk getting replaced. Its primary tasks are to watch for inaccessible disks and cells and, when they become accessible again, to initiate the ASM ONLINE operations.

ASM Processes in the Database Instance

Each database instance using ASM has two background processes: ASMB and RBAL. The ASMB background process runs in a database instance and connects to a foreground process in an ASM instance. Over this connection, periodic messages are exchanged to update statistics and to verify that both instances are healthy. All extent maps describing open files are sent to the database instance via ASMB. If an extent of an open file is relocated or the status of a disk is changed, messages are received by the ASMB process in the affected database instances.

During operations that require ASM intervention, such as the creation of a file by a database foreground process, the process connects directly to the ASM instance to perform the operation. Each database instance maintains a pool of connections to its ASM instance to avoid the overhead of reconnecting for every file operation.

A group of slave processes in the range 0001 to 0010 establishes a connection to the ASM instance, and these slave processes are used as a connection pool for database processes. Database processes can send messages to the ASM instance using the slave processes. For example, opening a file sends the open request to the ASM instance via a slave. However, the database doesn't use the slaves for long-running operations, such as those for creating a file. The slave connections eliminate the overhead of logging into the ASM instances for short requests. These slaves are automatically shut down when not in use.

Communication Between a Database and ASM Instance

ASM is like a volume manager for the Oracle database that provides the file system for Oracle database files. The ASM instance uses two data structures—Active Change Directory (ACD) and Continuing Operations Directory (COD)—to manage the metadata transactions it contains.

When you create a new tablespace or just add a new datafile to an existing tablespace, the Oracle database instance requests an ASM instance to create a new ASM file that the Oracle database instance can use. Upon receiving a new file-creation request, the ASM instance adds an entry into the Continuing Operation Directory and allocates space for the new file in the ASM disk group. ASM creates extents and then shares the extent map with the database instance; in fact, the ASMB background

https://t.me/learningnets

process in the database instance receives this extent map process.

Once the database instance opens the file successfully, the ASM instance commits the new file creation and removes the entry from the Continuing Operation Directory because new file information is now stored in the disk headers. Here is an important concept you need to understand: most people see database I/O being redirected to the ASM instance and the ASM instance performing I/O on behalf of the database instance; however, this is incorrect. The Oracle database instance performs I/O directly onto the ASM files, but it has to reopen the newly created ASM file once for when the ASM instance confirms with the database instance that the new file has been committed.

Active Change Directory (ACD) and Continuing Operations Directory (COD)

The Active Change Directory (ACD) is a journaling mechanism that provides functionalities similar to redo logs in an Oracle database. ACD records all the metadata changes in the ASM instance required to decide to roll forward in the event of unexpected failures—either due to operation failures or an instance crash. ACD is stored as a file in one of the ASM disks.

ASM metadata is triple-mirrored (high redundancy) and can grow within disk group when you add new instances. Oracle also mirrors ASM header metadata in different portions of the disk group. Transaction atomicity for ASM is guaranteed by ACD.

The Continuing Operations Directory (COD) is a memory structure in the ASM instance that maintains the state information of the active ASM operations and changes, such as rebalancing, new disk addition, and disk deletion. In addition, file creation requests from clients such as the RDBMS instance use COD to protect integrity. The COD records are either committed or roll backed upon success or failure of the ASM operation. COD is similar to the undo tablespace in an Oracle database; however, users can’t roll the data forward or back.

ASM Initialization Parameters

Just like a database instance, an ASM instance uses both mandatory and optional initialization parameters. You can set the ASM initialization parameters in both the database and the ASM instances, but some of the parameters are only valid for an ASM instance.

You can set following initialization parameters in an ASM instance. You can’t specify parameters that start with “ASM_” in database instances. You can use a pfile or an spfile as the ASM instance parameter file. If you use an spfile, you must place it in a disk group or in a cluster file system. Oracle recommends using an spfile and

https://t.me/learningnets

Technet24

storing it in a disk group. By default, the spfile for an ASM instance is stored in the Oracle Grid Infrastructure home ($ORACLE_HOME/dbs/spfile+ASM.ora).

- **INSTANCE_TYPE** This parameter instructs the Oracle executables about the instance type. For an ASM instance, you must set the value of the parameter to ASM (INSTANCE_TYPE=ASM). For an ASM instance in an Oracle Grid Infrastructure home, this parameter is optional. In Oracle Database 12*c*, this parameter has a third value: ASMPROXY. You must set the value of this parameter to ASMPROXY for an ASM proxy instance.
- **ASM_POWER_LIMIT** Sets the power limits for disk rebalancing. This parameter defaults to 1. Valid values are 1 through 1024 for disk groups whose compatibility is set to 11.2.0.2 or higher. Setting the value of this parameter to 0 will, of course, disable rebalancing, which isn't recommended! This parameter is dynamic, and you can use a SQL statement (**alter diskgroup data online disk dta_000 power 100**, for example) to modify the value on the fly. When you manually rebalance disks with the ALTER DISKGROUP…REBALANCE statement, the value for the POWER clause can't exceed the value of the ASM_POWER_LIMIT parameter. We discuss disk rebalancing in detail later in this chapter.
- **ASM_DISKSTRING** A comma-separated list of strings that limits the set of disks that ASM discovers. This parameter accepts wildcard characters. Only disks that match one of the strings are discovered. The string format depends on the ASM library in use and the operating system. The standard system library for ASM supports glob pattern matching. If you are using ASMLib to create ASM disks, the default path will be ORCL:*. For example, on a Linux server, where you aren't using ASMLib or the new Oracle ASM Filter Driver (Oracle ASMFD), you can set this parameter in this way: ASM_DISKSTRING = /dev/rdsk/mydisks/*. This will limit the discovery process to include just the disks in the /dev/rdks/mydisks directory.

NOTE

Many DBAs have issues properly configuring the ASM_DISKSTRING parameter. You must restrict this to the smaller set of the disks where ASM disk groups are located, and not /dev/, as the ASM instance will be forced to go through all disks in the location. Many of the devices here are not disks, and hence the disk discovery process will take much longer than necessary.*

- **ASM_DISKGROUPS** A list of the names of disk groups to be mounted by an ASM instance at startup or when the ALTER DISKGROUP ALL MOUNT statement is used. If this parameter is not specified, no disk groups are mounted except the ASM disk groups that store the spfile, OCR, and voting disk. This

https://t.me/learningnets

parameter is dynamic, and when a server parameter file (spfile) is used, altering this value isn't required. Here's an example: ASM_DISKGROUPS = DATA, FRA.

- **ASM_PREFERRED_READ_FAILURE_GROUPS** This parameter allows an ASM instance in extended cluster configuration, where each site has its own dedicated storage, to read data from the local disks rather than always reading data from the primary. Prior to Oracle 11*g*, ASM always read data from the primary copy regardless of the same extent being available on the local disks. This feature is very useful for performance in Oracle extended clusters. You should choose the number of failure groups carefully when configuring ASM for extended clusters because this will have a direct impact on ASM read performance.
- **DIAGNOSTIC_DEST** You set this parameter to the directory where you want the ASM instance diagnostics to be stored. The default value for this parameter is the $ORACLE_BASE directory. As with the database diagnostic directory setup, you'll find directories such as alert, trace, and incident under the main diagnostic directory.

By default, for all ASM instances, Automatic Memory Management is enabled. You don't need to set the MEMORY_TARGET parameter, however, because by default Oracle ASM uses a value for this parameter that works for most environments. If you want to explicitly control the memory management for ASM, you can set this parameter in the ASM spfile.

Managing ASM Disk Groups

Oracle provides different ASM tools such as ASMCA, ASMCMD, Oracle Grid Control, and SQL*Plus to create and manage disk groups inside the ASM instance.

It's important to understand the performance and availability requirements for an ASM disk group before creating the ASM disk group (for example, the redundancy level for the ASM disk group). If the underlying storage isn't protected by a RAID configuration, you should use ASM mirroring by choosing the correct redundancy for the ASM disk group.

ASMCA is a graphical tool that's self-explanatory and does not require much expertise, but you should click the Show Advanced Options button to change the different attributes of the ASM disk group. ASMCMD uses XML-style tags to specify the ASM disk group name, disk location, and attributes. You can specify the XML tags as inline XML, or you can create an XML file that you can use with the **mkdg** command inside the ASMCMD.

In SQL*Plus, use the CREATE DISKGROUP command to create a disk group in an ASM instance. Before creating a disk group, the ASM instance will check that the disk/raw partition being added in a disk group is addressable. If the disk/raw partition is addressable and is not being used by any other group, ASM writes specific

https://t.me/learningnets

Technet24

information in the first block of the disk or raw partition being used to create the disk group.

You can specify different attributes for an ASM disk group that impact its performance and availability. You specify disk group attributes with the ATTRIBUTE clause of the ALTER DISKGROUP or CREATE DISKGROUP statement, as shown here:

```
SQL> CREATE DISKGROUP data NORMAL REDUNDANCY
        FAILGROUP controller1 DISK
        '/devices/diska1' NAME diska1,
...
        ATTRIBUTE
        'compatible.asm' = '12.1.0.1',
        'content.type' = 'recovery',
...
SQL> ALTER DISKGROUP data SET ATTRIBUTE 'content.type' = 'data';
```

You can also set the attributes with the ASMCMD commands **setattr** and **mkdg** or through ASMCA. You can query the attributes of an ASM disk group from the V$ASM_ATTRIBUTE dynamic performance view. You can also do so with the ASMCMD **lsattr** command. Following is a list of important ASM disk group attributes you're most likely to use:

- **AU_SIZE** Used to specify the allocation unit size of the ASM disk group being created. You can't change the allocation unit of an existing disk group, so make sure you specify the correct allocation unit size while creating an ASM disk group. By default, Oracle uses 1MB for the allocation unit, but most Oracle users set this to 4MB.
- **DISK_REPAIR_TIME** This attribute is related to the performance and availability of the disk group. This attribute specifies the amount of time ASM will wait for an offline ASM disk to be dropped and the disk group to be rebalanced.
- **COMPATIBLE.ADVM** This attribute is required if the intended disk group will be used to create an ASM dynamic volume.
- **CELL_SMART_SCAN_CAPABLE** This attribute is only valid for Oracle Exadata Grid disks, and it enables Smart Scan predicate offload processing.
- **THIN_PROVISIONED** This attribute determines whether or not the unused storage space left after a disk group rebalancing operation is discarded. By default, the value of this parameter is **false**, meaning unused storage is retained. If you want the unused storage to be discarded, set the value of this parameter to **yes**.
- **PHYS_META_REPLICATED** This parameter enables Oracle to track a disk group's replication status. If the Oracle ASM disk group compatibility is 12.1 or higher, Oracle replicates the physical metadata of each piece of data—that is,

https://t.me/learningnets

it makes a (backup) copy of the metadata at the center point of the disk, such as the disk header and free space table blocks information. Oracle automatically sets this attribute's value to **true** if the entire physical metadata of all disks in a disk group has been replicated. The attribute has a value of **false** otherwise, and you can't set or change this attribute's value. ASM mounts the disk group automatically when the CREATE DISKGROUP command is executed, and a disk group name is also added in the ASM_DISKGROUPS parameter in the spfile so that whenever the ASM instance is restarted later, only this newly created disk group will be mounted.

If you want ASM to mirror files, define the redundancy level while creating the ASM disk group. Oracle provides two redundancy levels: normal redundancy and high redundancy. In normal redundancy, each extent has one mirrored copy; in high redundancy, each extent has two mirrored copies in different disk groups.

Disks in a disk group should be of a similar size with similar performance characteristics. It's always advisable to create different disk groups for different types of disks. All disks in disk groups must be of the same size to avoid wasting disk space in failure groups. All disks that you'll use to create disk groups must be in line with the ASM_DISKSTRING parameter to avoid disk discovery issues.

Creating a Disk Group

In the following example, we create a disk group named DGA with two failure groups, FLGRP1 and FLGRP2, using four raw partitions—namely, /dev/diska3, /dev/diska4, /dev/diska5, and /dev/diska6:

```
SQL> CREATE DISKGROUP DGA NORMAL REDUNDANCY
  2  FAILGROUP FLGRP1 DISK
  3 '/dev/diska3',
  4 '/dev/diska4',
  5  FAILGROUP FLGRP2 DISK
  6 '/dev/diska5',
  7 '/dev/diska6',
```

After creating a disk group, you may need to alter the disk group depending on your business requirements. Oracle allows you to perform create, drop/undrop, resize, rebalance, and mount/dismount operations on disk groups after they've been created.

You can optionally set the SECTOR_SIZE disk group attribute when creating a disk group. The value you set for SECTOR_SIZE determines the sector size for the disks in a disk group. You can set the sector size only when creating a disk group. If your disks support it, you may set the value of the SECTOR_SIZE parameter to 512 or 4096. The default value for sector size depends on the operating system platform.

Adding Disks to a Disk Group

https://t.me/learningnets

Technet24

Whenever you add disks to a disk group, Oracle internally rebalances the I/O load. The following example shows you how to add disks to an existing disk group. Oracle uses the ADD clause to add disks or a failure group to an existing disk group. In this example, the raw partition /dev/diska7 is being added to the existing group DGA:

```
ALTER DISKGROUP DGA ADD DISK
  '/dev/diska7' NAME disk5;
```

No failure group is defined in the statement, so the database assigns the disk to its own failure group. You can also use the ASMCA tool to add disks, as shown in the following example:

```
asmca -silent -addDisk -diskGroupName  mydg
              -disk '/devices/disk04'
              -disk  '/devices/disk05'
```

This command adds the two disks, disk04 and disk05, to the disk group named mydg.

Dropping, Undropping, Resizing, and Renaming Disks in a Disk Group

Oracle provides the DROP DISK clause in conjunction with the ALTER DISK GROUP command to drop a disk within a disk group. Oracle internally rebalances the files during this operation. Oracle fails the DROP operation if other disks in the disk group don’t have enough space.

If you are adding and dropping disks from a disk group, it is advisable that you add first and then drop, and both operations should be performed in a single ALTER DISKGROUP statement in order to reduce the time the database spends on rebalancing the data.

Oracle also provides force options to drop a disk within a disk group, even if ASM can’t read or write to those disks. You can’t specify the force option with external redundancy disk groups.

Undropping Disks in a Disk Group

If a disk dropping operation is pending, you can cancel the drop operation with the ALTER DISKGROUP...UNDROP DISK statement, as shown here:

```
SQL> ALTER DISKGROUP DGA UNDROP DISK;
```

Replacing Disks in a Disk Group

When a disk is damaged or missing, you normally perform the two-step operation of dropping the affected disk and then replacing it with a new disk. However, starting with the 12*c* release, you can consolidate the two steps into a single step of replacing the damaged or missing disk. The following example shows how you can replace the

https://t.me/learningnets

disk named disk5 with another disk named disk6:

```
SQL> ALTER DISKGROUP DGA REPLACE DISK
     DISK5 WITH DISK6;
```

You can include a POWER clause with the REPLACE DISK command, and the clause works the same way as when you use it in a disk-rebalance command.

Resizing the Disks

Oracle provides a RESIZE clause that you can specify with the ALTER DISKGROUP statement to resize a disk group, resize any specific disk in a disk group, or resize the disks within a specific failure group.

Resizing is useful for reclaiming disk space. For example, if the SIZE defined for the disks was less than the disk size when the disk group was created and later on you want to claim the full size of the disk, you can use this option without giving any SIZE so that Oracle will take SIZE as returned by the operating system. Here's an example:

```
ALTER DISKGROUP DGA RESIZE DISK '/dev/raw/raw6' SIZE 500M;
```

Renaming a Disk Group

You can rename an ASM disk group by first dismounting all disk groups on all the nodes in the cluster and then running the **renamedg** command on the disk group. Following is an example that shows how to rename a disk group:

1. Use the **renamedg** command to rename the disk group dgroup1 to the new disk group namedgroup2:

   ```
   $ renamedg dgname=dgroup1 newdgname=dgroup2
     asm_diskstring='/devices/disk'*' verbose=true
   ```

2. Because the **renamedg** command doesn't update the file references in the database, the original disk group isn't automatically deleted after you run this command. You can check the status of the old disk group with the following command:

   ```
   $ crsctl status res -t
   ```

3. Manually delete the old disk group resource with the following command:

   ```
   $ srvctl remove diskgroup dgroup1
   ```

4. Rename the disks in the new disk group with the following commands:

   ```
   SQL> ALTER DISKGROUP dgroup2 MOUNT RESTRICTED;
   SQL> ALTER DISKGROUP dgroup2 RENAME DISKS ALL;
   ```

Administering ACFS

You can't use ASM tools such as ASMCA, ASMCMD, OEM, and SQL*Plus to create and manage ACFS. Creating an ACFS file system is just like creating a file system

https://t.me/learningnets

Technet24

with any traditional volume manager. In a traditional volume manager, you first create a volume group and then the volumes. Finally, you create the file system and then mount the file system.

Just like an allocation unit in the ASM disk group, *volume allocation unit* is the smallest storage unit that can be allocated in an ASM dynamic volume. ASM allocates stripes in each volume allocation unit, where each stripe is the equivalent of the volume extent size, which is directly related to the allocation unit size of the underlying ASM disk group. By default, the size of one volume extent is 64MB for a 1MB allocation unit size disk group. When you create an ASM dynamic volume, you specify the number of stripes (also known as stripe columns) and the width of each stripe because Oracle internally uses the SAME (Stripe and Mirror Everything) approach to stripe and mirror data. Oracle will then distribute the file being stored inside the ACFS into chunks, with a stripe width size of 128KB (the default stripe width) on each volume extent.

For example, if you create an ASM volume of 400MB and store a 1MB file initially with default settings, Oracle will create a volume extent size of 64MB and each volume allocation unit will be of 256MB (the number of stripes multiplied by the volume extent size). Because Oracle allocates space in multiples of the volume allocation unit, it will allocate 512MB for the ASM dynamic volume although you requested 400MB. The ASM dynamic volume can be extended later on. ASM will distribute the file into eight chunks of 128KB and store them on volume extents. ASM stripes the volume extents in the same way on the available disks inside the ASM disk group; hence, it provides very efficient I/O operations.

You can use the V$ASM_ACFSVOLUMES and V$ASM_FILESYSTEM dynamic performance views of the ASM instance to display the ACFS file system information on the connected ASM instance.

Setting Up ACFS

Follow the steps given next to create the ASM Cluster File System:

1. Create an ASM disk group with the attribute COMPATIBLE.ADVM set at a minimum value of 11.2 or higher (Oracle introduced ACFS in Oracle 11*g* Release 2), because setting this attribute tells ASM that this disk group can store ASM dynamic volumes. If this attribute is not set, you can't create a dynamic volume in this ASM disk group. You can use either ASMCA, ASMCMD, OEM, or SQL*Plus to create the ASM disk group. If you're using ASMCA, you can set the disk group attribute by clicking the Show Advanced Options button. Other ASM tools allow you to specify the attribute at the command line.
2. Once the ASM disk group is created, you need to create the ASM dynamic volume in the ASM disk group created in the previous step. Use the ASMCMD **volcreate** command for this, as shown here:

   ```
   $ ASMCMD [+]> volcreate -G data -s10G volume1
   ```
3. Create the required OS directory structure to mount the newly created ASM

https://t.me/learningnets

dynamic volume.

4. Once you have created the directory, you need to create the ACFS file system using the operating system command **mkfs**. Make sure you use the following syntax to create the ACFS-type file system on the ASM dynamic volume:

```
$ /sbin/mkfs -t acfs /dev/asm/volume1-123
```

5. As an optional step, you can register the newly created file system with the **acfsutil registry** command:

```
$ /sbin/acfsutilregistry -a /dev/asm/volume1-123  /acfsmounts/acfs1
```

In this example, /acfsmounts/acfs1 is the name of the OS directory you created in step 3 in order to mount your new dynamic volume. Although purely optional, registering the file system in this manner offers a couple of benefits. First, you don't need to manually mount the new file system on each cluster member. Second, the new file system is automatically mounted when the Oracle Clusterware or the server restarts.

6. Now mount the newly created file system using the **mount** operating system command. Make sure you use the ACFS file system type to mount the file system. Here's an example:

```
$ /bin/mount -t acfs /dev/asm/volume1-123  /acfsmounts/acfs1
```

7. Once you mount the new file system, make sure you set the permissions on the file system appropriately so that users who need to access it can do so. For example, you can set the permissions for the user *oracle* in the following manner:

```
$ chown –R oracle:dba   /acfsmounts/acfs1
```

Creating an ACFS Snapshot

You use ASM tools to create and manage ACFS snapshots. You can use the ASMCMD command **acfsutil** to create and manage snapshots via a command-line interface. Here is an example of creating the ACFS snapshot:

```
acfsutil snap create acfs_snap_01 /app/oracle/myfirstacfs
```

The snapshot-creation process will create a hidden directory (.ACFS) and a directory structure of snaps/<snapshot name> inside the hidden directory. In our example, ASM will create the following directory structure:

```
.ACFS/snaps/acfs_snap_01
```

With all the exciting features of ACFS, it comes with some restrictions, which you must be aware of before using the ACFS. Although you can create other file systems on an ASM dynamic volume, Oracle only supports ACFS as a cluster file system on an ASM dynamic volume. Partitioning of an ASM dynamic volume is not supported, so you cannot use the **fdisk** command to partition an ASM dynamic volume device. Apart from this, you should not use ASMLib over an ASM dynamic volume because Oracle

https://t.me/learningnets

Technet24

does not support this configuration. You can use multipath devices to create an ASM disk group, but using multipathing on ASM dynamic volumes is prohibited.

ASM Fast Mirror Resync

Prior to Oracle 11g, whenever ASM was not able to write the data extent to the ASM disk, it took the ASM disk offline. In addition to this, further reads on this disk were prohibited and ASM would re-create the extents from the mirror copies stored on the other ASM disks. When you added the disk back to the disk group, ASM would perform this rebalancing operation again to reconstruct all the data extents. This rebalancing operation is very time consuming and also has an impact on the response time of the ASM disk group.

Starting with Oracle 11g, the database does not drop the offline ASM disk and waits and tracks the data extents modified up to the time specified by the ASM_DISK_REPAIR attribute of the associated disk group. When the failure is repaired within this timeframe, ASM will reconstruct the modified data extents only, rather than requiring a complete rebalancing of the whole disk group, thus resulting in an overall faster and more efficient rebalancing operation.

ASM Disk Rebalancing

ASM doesn't require any downtime during storage configuration and reconfiguration —that is, you can change the storage configuration without having to take the database offline. ASM automatically redistributes file data evenly across all the disks of the disk group after you add or drop disks from a group. This operation is called *disk rebalancing* and is transparent to the database.

A rebalancing operation evenly spreads the contents of every file across all available disks in that disk group. The operation is driven by space usage in the disks and not based on the I/O statistics on those disks. It is invoked automatically when needed, and no manual intervention is required during the operation. You can also choose to run the operation manually or change a running rebalancing operation.

Increasing the number of background slave processes responsible for the operation can speed up the rebalancing operation. The background process ARBx is responsible for disk rebalancing during storage reconfiguration. To increase the number of slave processes dynamically, you use the init.ora parameter ASM_POWER_LIMIT. It is recommended that you perform rebalancing using only one node when running in Oracle RAC. Shutting down any unused ASM instances can do this. Figure 6-4 shows the rebalancing functionality.

https://t.me/learningnets

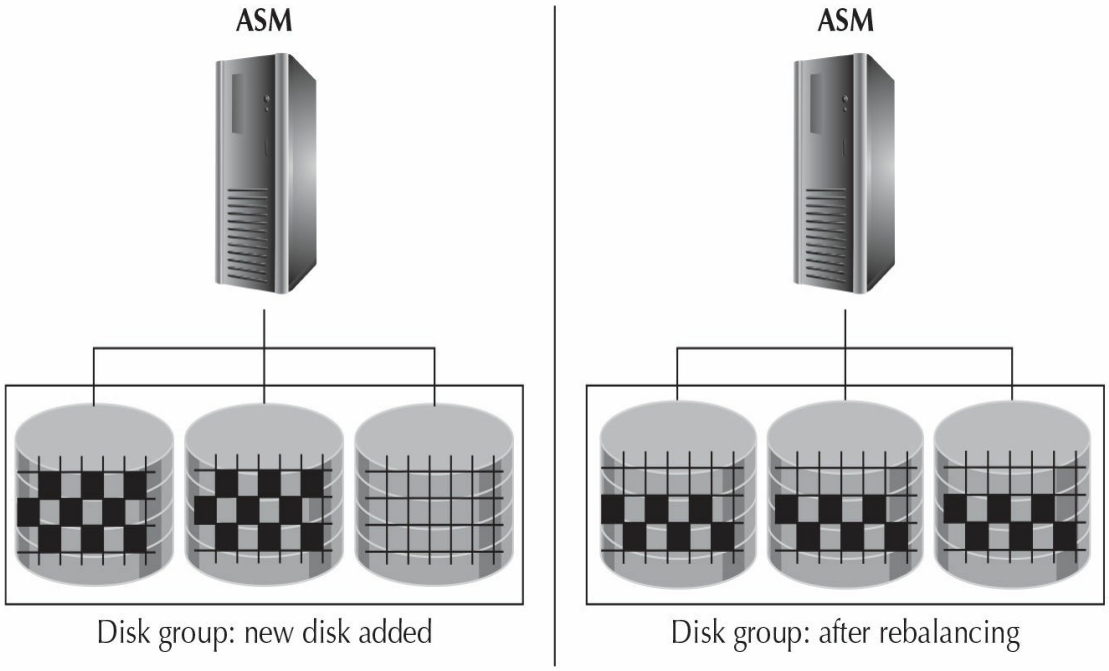


FIGURE 6-4. *ASM rebalancing*

If you don't specify the POWER clause in an ALTER DISKGROUP command, or when adding or dropping a disk implicitly invokes a rebalance, the rebalance power defaults to the value of the ASM_POWER_LIMIT initialization parameter. You can adjust this parameter dynamically.

The higher the value of the ASM_POWER_LIMIT parameter, the faster a rebalancing operation may complete. Lower values cause rebalancing to take longer but consume fewer processing and I/O resources, leaving these resources available for other applications, such as the database. The default value of 1 minimizes disruption to other applications. The appropriate value is dependent on your hardware configuration as well as the performance and availability requirements.

If a rebalance is in progress because a disk is manually or automatically dropped, increasing the power of the rebalance shortens the window during which redundant copies of that data on the dropped disk are reconstructed on other disks.

The V$ASM_OPERATION view provides information that can help you calibrate the value of the ASM_POWER_LIMIT parameter, and thus the power of rebalancing operations.

The V$ASM_OPERATION view also gives an estimate (EST_MINUTES column) of the amount of time remaining for the rebalancing operation to complete. You can see the effect of changing the rebalance power by observing the change in the time estimate. The column EST_WORK gives you an estimate of the number of allocation units the rebalance operation must perform. For each resync and rebalance operation, the database updates the PASS column, and the column can show a value of RESYNC, REBALANCE, or COMPACT.

https://t.me/learningnets

Technet24

Manually Rebalancing a Disk Group

You can manually rebalance the files in a disk group using the REBALANCE clause of the ALTER DISKGROUP statement. This would normally not be required, because ASM automatically rebalances disk groups when their composition changes. You might want to perform a manual rebalancing operation, however, if you want to control the speed of what would otherwise be an automatic rebalancing operation.

The POWER clause of the ALTER DISKGROUP...REBALANCE statement specifies the degree of parallelization, and thus the speed of the rebalancing operation. It can be set to a value from 0 to 1024 (the default value is 1). A value of 0 halts a rebalancing operation until the statement is either implicitly or explicitly invoked again. The default rebalance power is set by the ASM_POWER_LIMIT initialization parameter. You can also use the REBALANCE clause in an ALTER DISKGROUP command that adds, drops, or resizes ASM disks.

NOTE

ASM automatically disables the rebalancing feature when you create a disk group with REBALANCE POWER 0. If you add more disks to this disk group, ASM will not distribute data to the newly added disks. Also, when you remove disks from this disk group (DROP DISK), the status of the disk group remains DROPPING until you change REBALANCE POWER to greater than 0. We the authors don't think it's ever a good idea to set the rebalance power to 0.

Entering the REBALANCE statement with a new level can change the power level of an ongoing rebalancing operation. The ALTER DISKGROUP...REBALANCE command by default returns immediately so that you can issue other commands while the rebalancing operation takes place asynchronously in the background.

You can query the V$ASM_OPERATION view for the status of the rebalancing operation. You can also can use the **lsop** command on the ASMCMD command prompt to list the ASM operations. Here is an example showing how to use the **lsop** command:

```
ASMCMD> lsop
Group_Name  Pass  State  Power  EST_WORK  EST_RATE  EST_TIME
ASMCMD>
```

If you want the ALTER DISKGROUP...REBALANCE command to wait until the rebalancing operation is complete before returning, you can add the WAIT keyword to the REBALANCE clause. This is especially useful in scripts. The command also accepts a NOWAIT keyword, which invokes the default behavior of conducting the rebalancing operation asynchronously. You can interrupt a rebalance running in wait mode by pressing CTRL-C on most platforms. This causes the command to return

https://t.me/learningnets

immediately with the message “ORA-01013: user requested cancel of current operation” and to continue the rebalance operation asynchronously.

Here are some additional rules for the rebalancing operation:

- The ALTER DISKGROUP...REBALANCE statement runs on a single node and uses the resources of the single node on which you issue the statement.
- ASM can perform only one rebalance at a time on a given instance.
- Rebalancing continues across a failure of the ASM instance performing the rebalance.
- The REBALANCE clause (with its associated POWER and WAIT/NOWAIT keywords) can also be used in ALTER DISKGROUP commands that add, drop, or resize disks.

The following example shows how to rebalance a disk group named dgroup2:

```
ALTER DISKGROUP dgroup2 REBALANCE POWER 5;
```

This command returns immediately, and the rebalance operation takes place asynchronously in the background.

Specify the WAIT keyword with the REBALANCE clause so that the rebalancing command waits until the rebalance operation is complete before returning. The following example manually rebalances the disk group dgroup2. The command does not return until the rebalancing operation is complete:

```
ALTER DISKGROUP dgroup2 REBALANCE POWER 5 WAIT;
```

You can change the power level of an *ongoing* rebalance operation by issuing the rebalance statement with the MODIFY POWER clause, as shown here:

```
ALTER DISKGROUP data2 REBALANCE MODIFY POWER 10;
```

You can change the power setting back to the default value by doing the following:

```
ALTER DISKGROUP data2 REBALANCE MODIFY POWER;
```

Rebalancing Phase Options

When you rebalance disk groups, you can choose from various phase options by specifying the WITH or WITHOUT keyword. Following are the phase options you can choose from:

- **RESTORE** This phase is always run, and you can’t exclude it. The phase includes the following operations:
  - **RESYNC** This operation synchronizes the stale extents on the disks that the database is bringing online.

https://t.me/learningnets

Technet24

- **RESILVER** Applies only to Exadata systems. This is an operation where data is copied from a mirror to another mirror that has stale data.
- **REBULD** Restores the redundancy of forcing disks (a forcing disk is one that you've dropped with the FORCE option).

- **BALANCE** This phase restores the redundancy of all the disks in the disk group and balances extents on all disks.
- **PREPARE** This phase is applicable only to FLEX or EXTENDED redundancy disk groups. It completes the work relevant to the "prepare" SQL operation.
- **COMPACT** This phase defragments and compacts all extents.

If you don't specify any of the options (RESTORE, BALANCE, PREPARE, or COMPACT), Oracle will run all the rebalance phases by default.

Here are two examples that show how to specify the phase options when balancing a disk group:

```
ALTER DISKGROUP data2 REBALANCE WITH BALANCE COMPACT;
ALTER DISKGROUP data3 REBALANCE WITHOUT BALANCE;
```

Monitoring the Performance of Balancing Operations

You can query the V$ASM_DISK_STAT and V$ASM_DISKGROUP_STAT views to get at performance statistics. These views, along with V$FILESTAT, can provide a bunch of information about the performance of the disk groups and datafiles. You can execute the following query to get the performance statistics at the disk group level:

```
SELECT PATH, READS, WRITES, READ_TIME, WRITE_TIME,
READ_TIME/DECODE(READS,0,1,READS)"AVGRDTIME",
WRITE_TIME/DECODE(WRITES,0,1,WRITES) "AVGWRTIME"
FROM V$ASM_DISK_STAT;
PATH              READS   WRITES  READ_TIME WRITE_TIME  AVGRDTIME  AVGWRTIME
--------------- -------- -------- ---------- ---------- ---------- ----------
ORCL:DISK1        50477    67683      20.86 724.856718 .000413258 .010709583
ORCL:DISK2       418640   174842    100.259 802.975526 .000239487 .004592578
```

In Oracle Database 12*c*, you can rebalance multiple disk groups in a single rebalancing operation.

Tuning Disk Rebalancing Operations

You can query the V$ASM_OPERATION view to check the status of a rebalance operation. Just as you can use the EXPLAIN WORK statement to create an explain plan for an Oracle SQL statement, you can also determine the amount of work

https://t.me/learningnets

involved in a rebalancing operation before you issue the rebalancing commands.

Once you issue the EXPLAIN WORK statement, you can query the V$ASM_ESTIMATE view to check Oracle's estimates of the rebalancing operation. The following example shows how to use the EXPLAIN feature:

```
SQL> EXPLAIN WORK FOR ALTER DISKGROUP DISKGROUP DGA DROP DISK data_01;
Explained.
```

Once you execute the EXPLAIN WORK statement, query the V$ASM_ESTIMATE view, as shown here:

```
SQL> SELECT est_work FROM V$ASM_ESTIMATE;
EST_WORK
4211
```

The V$ASM_ESTIMATE view shows the effects of adjusting the ASM_POWER_LIMIT and thus the impact on the rebalancing operations. Although we issued a DROP DISK command here and not a rebalancing command, there's an implicit rebalancing operation when you add or drop ASM disks; hence, the V$ASM_ESTIMATE view shows the potential impact of the rebalancing operation.

The column EST_WORK in the V$ASM_ESTIMATE view estimates the number of ASM allocation units Oracle will move during a rebalancing operation. You can query the V$ASM_ESTIMATE view to get estimates of work involved in the execution plans for both disk group rebalance and resync operations.

Scrubbing Disk Groups

Starting with the Oracle Database 12c release, you can both check for as well as repair logical corruption in both normal and high redundancy disk groups. Because the scrubbing process uses the mirror disks to check and repair logical corruption, there's not a noticeable impact on your disk I/O.

Here's an example that shows how to specify the SCRUB option to check for logical corruption in the disk group DGA:

```
SQL> ALTER DISKGROUP DGA SCRUB POWER LOW;
```

The SCRUB clause checks and reports any existing logical corruption. The following example shows how to specify the REPAIR clause along with the SCRUB clause to automatically repair disk corruption:

```
SQL> ALTER DISKGROUP DGA SCRUB FILE
'+DGA/ORCL/DATAFILE/example.123.456789012' REPAIR POWER HIGH;
```

As with the rebalancing commands, you may optionally specify the WAIT and FORCE clauses as well when scrubbing a disk group, a specific disk, or a specific file of a disk group.

https://t.me/learningnets

Technet24

Backup and Recovery in ASM

An ASM instance is *not* backed up because an ASM instance itself does not contain any files but rather manages the metadata of the ASM disks. ASM metadata is *triple-mirrored*, which should protect the metadata from typical failures. If sufficient failures occur to cause the loss of metadata, the disk group must be re-created. Data on the ASM disks is backed up using RMAN. In case of failure, once the disk groups are created, the data (such as database files) can be restored using RMAN.

Each disk group is self-describing, containing its own file directory, disk directory, and other data such as metadata logging information. ASM automatically protects its metadata by using mirroring techniques, even with external redundancy disk groups. An ASM instance caches the information in its SGA. ASM metadata describes the disk group and files, and it is self-describing because it resides inside the disk group. Metadata is maintained in the blocks, and each metadata block is 4KB and triple-mirrored.

With multiple ASM instances mounting the same disk groups, if one ASM instance fails, another ASM instance automatically recovers transient ASM metadata changes caused by the failed instance. This situation is called *ASM instance recovery* and is automatically and immediately detected by the Global Cache Services.

With multiple ASM instances mounting different disk groups, or in the case of a single ASM instance configuration, if an ASM instance fails while ASM metadata is open for update, the disk groups that are not currently mounted by any other ASM instance are not recovered until they are mounted again. When an ASM instance mounts a failed disk group, it reads the disk group log and recovers all transient changes. This situation is called *ASM crash recovery*.

Therefore, when you're using ASM clustered instances, it is recommended that you have all ASM instances always mounting the same set of disk groups. However, it is possible to have a disk group on locally attached disks that are visible only to one node in a cluster, and have that disk group mounted only on the node where the disks are attached.

ASM supports standard operating system backup tools, Oracle Secure backup, and third-party backup solutions such as storage array snapshot technologies to back up the ACFS file system. Chapter 9 covers ASM metadata backups.

ASM Flex Clusters

Oracle Database 12*c* introduces Oracle Flex ASM clusters, which let an ASM instance run on a different server from the one where the database server runs. Up until this release, you had to run an ASM instance on each server where the database instance ran. Flex clusters remove the need to collocate ASM and Oracle database instances and thus allow a small set of ASM instances running in a cluster to service a large number of database instances. The default minimum number of instances in an ASM

https://t.me/learningnets

cluster is 3.

The key idea behind the introduction of Oracle Flex ASM is to increase database availability, because in the traditional ASM deployment model, where a separate ASM instance is required on every cluster node, the failure of a local ASM instance means that all database instances running on that server will also fail.

If an Oracle Flex ASM instance on a server fails, the database instances running on that server will continue to run without a problem. If an Oracle Flex ASM instance fails, the ASM instance simply fails over to another node in the cluster; meanwhile, the database instance(s) will continue to operate unhindered by using a non-local Oracle Flex ASM instance located elsewhere in the cluster. This way, you make the database instances independent from the local ASM instances, thus increasing database availability. In addition, the new ASM deployment model cuts back on the total amount of resources used for ASM instances.

A cluster performs the group membership services and consists of a set of nodes, which include at least one *hub* node. A hub node is a node that is connected to all nodes and one that directly accesses the shared disk system. Therefore, just the hub nodes have access to the ASM disks. A single set of ASM disk groups is managed by the ASM instances that are part of the cluster.

Using a single set of ASM disk groups under Oracle Flex ASM lets you consolidate all your storage into a more easily manageable set of disk groups. Each cluster has two networks—a private and a public network—and additionally a minimum of one ASM network. You can use the private network to perform double duty as the Oracle ASM network. These two networks need to be on separate subnets.

NOTE

*Oracle Flex ASM is a requirement for Oracle Flex clusters, which are new in the Oracle Database 12*c *release (see Chapter 7). Therefore, it is enabled by default when you install an Oracle Flex cluster*.

Configuration of Oracle ASM in Flex ASM

You can configure Oracle ASM in Flex ASM in the following ways:

- **Local ASM clients directly accessing ASM** Under this mode—which technically speaking isn't really an Oracle Flex mode because it doesn't use Flex ASM at all—the database clients run on the same server as the ASM instance. You can do this only on a hub node. The database instance runs as a local ASM client, directly accessing the ASM disks.
- **Flex ASM client directly accessing Oracle ASM** In this mode, database clients on the hub nodes directly access ASM storage, but also access ASM instances remotely for ASM metadata. The database instances don't run on the

https://t.me/learningnets

Technet24

same server hosting the ASM instance. Rather, they run on remote hub nodes.

- **ACFS access through an ASM proxy instance** An ASM proxy instance is one that runs on a hub node with a direct ASM client. ASM proxy instances can use both ACFS and ADVM. You set the ASM initialization parameter INSTANCE_TYPE to ASMPROXY to use this mode.

Setting Up Flex ASM

Setting up Oracle Flex ASM is easy. During a new installation, the Oracle Universal Installer (OUI) lets you choose between a regular Oracle ASM cluster and an Oracle ASM Flex deployment. When you choose Oracle Flex ASM, you must also choose the Oracle ASM networks. Each cluster has a minimum of one private and one public network. If you're going to use ASM for storage, the cluster will have at least one Oracle ASM network. The Oracle ASM listener is created for the new Oracle ASM network and automatically started on all nodes in the cluster.

If you're upgrading to Oracle Database, you can continue to use Oracle ASM as you've used it in earlier versions of the database. However, you must consider enabling Oracle Flex ASM to benefit from the new ASM capabilities in Oracle Database 12*c*. Oracle Flex ASM will protect you against unplanned outages (failures) as well as scheduled maintenance-related outages for work such as upgrading the OS and applying patches.

If you're currently using the Standard Oracle ASM configuration, you can enable Oracle Flex ASM by using the ASMCA to perform the conversion to Flex ASM. You must ensure that the OCR, the spfile, and the ORAPWD file for the current standard ASM configuration are all stored in a disk group before you can convert to Oracle Flex ASM configuration.

Click the Convert to Oracle Flex ASM button in the ASMCA Configure ASM: ASM Instances page. You'll then need to specify the listener, port, and network interface in the Convert to Oracle Flex ASM page.

Once you enter the network and port information on this page and click OK, the ASM Conversion dialog appears, showing the path to the script named convertToFlexASM.sh, which you must run as a privileged user on the node where ASMCA is running. You may also run the ASMCA utility in the silent mode to convert regular Oracle ASM to Oracle Flex ASM, as shown in the following example:

```
asmca -silent
      -convertToFlexASM
      -asmNetworks    eth1/10.10.10.0
      -asmListenerPort 1521
```

You use the standard utilities ASMCA, CRSCTL, SRVCTL and SQL*Plus to administer Oracle Flex ASM. You can find out if Oracle Flex ASM is enabled by issuing the **asmcmd showclustermode** command:

https://t.me/learningnets

```
$ asmcmd showclustermode
ASM cluster  : Flex mode enabled
```

In addition to the ASMCMD utility, you can also use ASMCA, CRSCTL, SQL*Plus, and SRVCTL to manage Oracle Flex ASM. The following command lets you determine the status of the instances in an Oracle Flex ASM configuration:

```
# srvctl status asm -detail
ASM is running on prod1, prod2
ASM is enabled.
```

The following example shows how to determine the number of ASM instances that are part of the Flex ASM configuration:

```
$ srvctl config asm
ASM instance count:  3
```

The **config asm** command shows that three ASM instances are part of the Flex ASM configuration. You can change the number of instances in the configuration with the **modify asm** command, as shown here:

```
$ srvctl modify asm -count 4
```

In this example, the term “count” is highly significant. It means that when one instance dies, Oracle brings up another instance on a different node, to maintain the number of instances at four. If an ASM instance fails, Oracle automatically switches the clients over to another ASM instance in the cluster. You can also manually switch a client, as shown here:

```
SQL> ALTER SYSTEM RELOCATE CLIENT 'client-id';
```

In this example, **client-id** is specified in the format *INSTANCE_NAME*:*DB_NAME*.

Managing ASM Flex Disk Groups

An ASM flex disk group is an ASM disk group type that supports ASM file groups and quota groups, which enable easy quota management. An ASM file group is a set of files that belong to a database, and it enables you to manage storage at the file group or database level.

NOTE

Flex disk groups let you manage storage at the granularity level of a database, in addition to enabling management at the disk group level.

https://t.me/learningnets

Technet24

Key Features

Here are the key features of an ASM flex disk group:

- Each database has its own file group.
- You can migrate to a flex disk group from a normal or high redundancy disk group.
- A flex disk group requires at least three failure groups.
- You set the redundancy for a flex disk group to FLEX REDUNDANCY. Each file group in the flex disk group has its own redundancy setting.
- File groups of flex disk groups describe database files.
- Flex disk groups have a flexible redundancy for files.
- Flex groups can tolerate two failures, the same as a high redundancy disk group.

Creating and Migrating to a Flex Disk Group

You create a flex disk group with the CREATE DISKGROUP statement:

```
SQL> CREATE DISKGROUP flex_data FLEX REDUNDANCY DISK my_disk_discovery_path;
```

You can migrate a normal disk group to a flex group with the ALTER DISKGROUP statement:

```
SQL> ALTER DISKGROUP data MOUNT RESTRICTED;
SQL> ALTER DISKGROUP data CONVERT REDUNDANCY TO FLEX;
```

Understanding ASM File Groups and ASM Quota Groups

An ASM flex disk group supports ASM file groups and quota groups. Flex groups enable storage management at the database level. An ASM file group consists of a set of ASM files. One or more file groups constitute a quota group. An ASM disk group can have multiple quota groups.

ASM Quota Groups

You use ASM quota groups to configure quotas that you allocate to a set of ASM file groups. A *quota* is a physical space limit that the database enforces when you create or resize ASM files. A *quota group* is the total storage space that is used by one or multiple file groups in the same disk group. Quota groups allow you to control the total storage used by different databases, especially in a multitenant environment.

If you have two disk groups, Disk Group 1 and Disk Group 2, you can create two quota groups, QGRP1 and QGRP2, in each of the two disk groups, and you can allocate a specific quota to each of the two quota groups. You can then have one or more file groups in each of the two quota groups.

https://t.me/learningnets

Following are the essential points to understand about quota groups:

- A file group can belong to only one quota group.
- A quota group can't span multiple disk groups.
- A quota has two values: the limit and the current used space.
- You can move a file group from one quota group to another.

You can add a quota group to a disk group in the following way:

```
SQL> ALTER DISKGROUP DiskGroup_2 ADD QUOTAGROUP Quotagroup_QGRP3
SET 'quota' = 100m;
```

Now that you've learned about quota groups, let's see how you create file groups within a quota group.

ASM File Groups

An ASM *file group* consists of a set of files that have the same properties and characteristics. The properties include redundancy, rebalance power limit, striping, quota group, and the access control list. You can use the file groups to set up different availability characteristics for databases that share the same disk group.

A *disk group* consists of one or more file groups, and it can store files belonging to multiple database. Each database has a separate file group.

Here are the key points to remember about disk groups and file groups:

- A disk group contains a minimum of one file group, called the default file group.
- For a disk group to contain a file group, it must have either flex or extended redundancy.
- A database can have only one file group in a disk group.
- A file group can describe a single database such as a pluggable database (PDB), container database (CDB), or cluster.
- A file group can belong to only one quota group.

The following example shows how to create a file group. Remember that you must first create the quota group before you can assign a file group to that quota group.

```
SQL> ALTER DISKGROUP DiskGroup1 ADD FILEGROUP FileGroup_PDB1
     DATABASE PDB1
     SET 'quota group' = 'QutoaGroup_QGRP1';
```

Using Quota Groups and File Groups in a Multitenant Environment

Oracle ASM file groups and quota groups are quite helpful when allocating storage

https://t.me/learningnets

Technet24

and enforcing limits on the space usage by multiple pluggable databases in a multitenant environment. Figure 6-5 illustrates how three pluggable databases in a multitenant environment make use of ASM files stored in multiple file groups, quota groups, and disk groups.

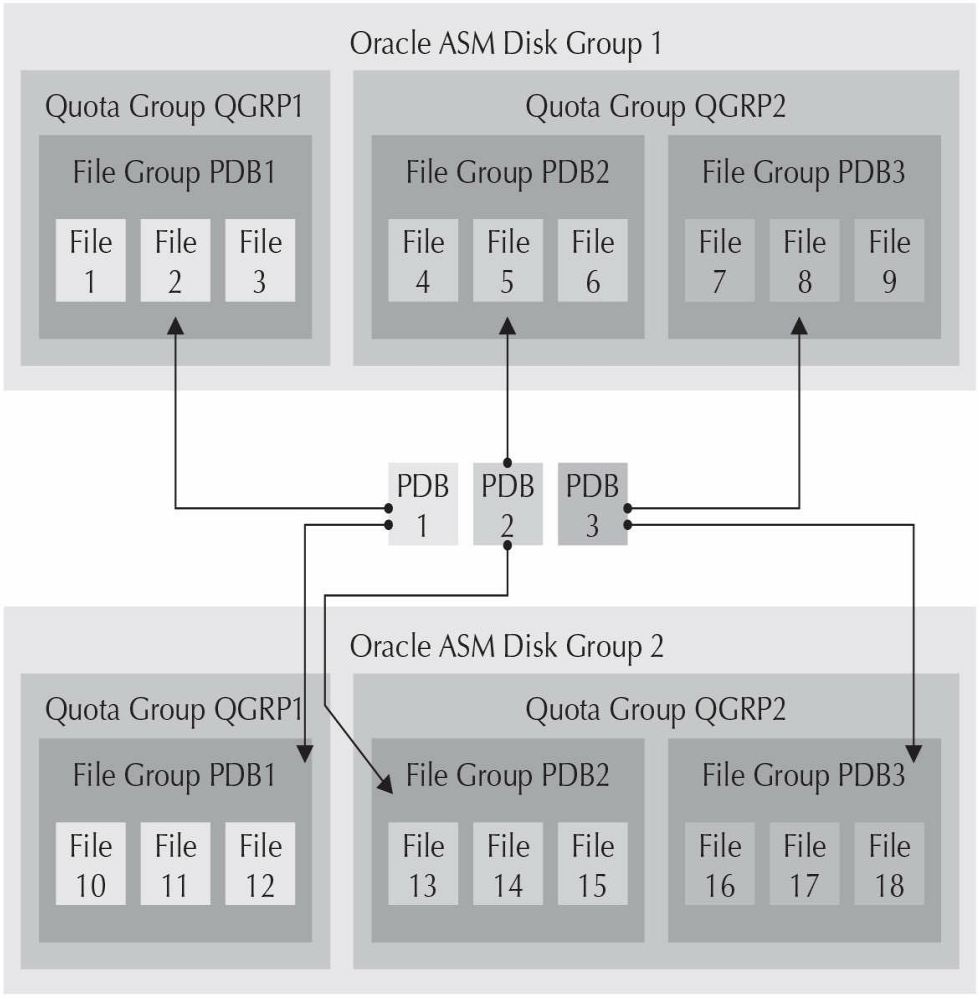


FIGURE 6-5. *Oracle ASM file groups, quota groups, and disk groups*

Figure 6-5 shows the architecture of Oracle ASM file groups and quota groups, as summarized here:

- There are two ASM disk groups: Disk Group 1 and Disk Group 2.
- Each of the two disk groups contains two quota groups: QGRP1 and QGRP2.
- The quota group QGRP1 contains one file group named PDB1. The quota group QGRP2 contains two file groups named PDB2.

This example has three pluggable databases: PDB1, PDB2, and PDB3. The following is how the three databases use the file groups in the two quota groups:

- The file groups named PDB1 (disk group 1 and disk group 2) are dedicated to

https://t.me/learningnets

the pluggable database PDB1.

- The file groups named PDB2 (disk group 1 and disk group 2) are dedicated to the pluggable database PDB2.
- The file groups named PDB3 (disk group 1 and disk group 3) are dedicated to the pluggable database PDB3.

Oracle Extended Disk Groups

Oracle ASM extended disk groups are similar to flex disk groups, and they are meant for use in an extended cluster, where the nodes are located in physically separate sites. Here are the key features of an extended disk group:

- You must set the redundancy of an extended disk group to EXTENDED REDUNDANCY.

NOTE

An extended disk group tolerates failures at the site level, in addition to failures at the failure group level.

- An extended disk group can withstand the loss of an entire site, in addition to the loss of up to two failure groups in another site.
- You configure the redundancy for each site but not for each disk group.
- The quota group space limit is the total space required for storing all copies across all the sites of an extended cluster. If a cluster has two sites, a 10MB file with a redundancy level of MIRROR uses 40MB of the quota limit.
- The minimum allocation unit (AU) size is 4MB.

The following example shows how to create an extended disk group:

```
SQL> CREATE DISKGROUP extended_site_data EXTENDED REDUNDANCY
            SITE TX FAILGROUP fg1 DISK '/devices/disks/disk01'
                    FAILGROUP fg2 DISK '/devices/disks/disk02'
                    FAILGROUP fg3 DISK '/devices/disks/disk03'
            SITE CA FAILGROUP fg4 DISK '/devices/disks/disk04'
                    FAILGROUP fg5 DISK '/devices/disks/disk05'
                    FAILGROUP fg6 DISK '/devices/disks/disk06'
            SITE QM QUORUM
                    FAILGROUP fg7 DISK '/devices/disks/disk07';
```

ASM Tools

https://t.me/learningnets

Technet24

ASM tools, such as the ASM command-line interface and the ASM file transfer utilities, emulate the UNIX environment within the ASM file system. Even though ASMCMD internally uses the SQL*Plus interface to query the ASM instance, it greatly helps system administrators and storage administrators because it provides the look and feel of a UNIX shell interface.

ASMCA: The ASM Configuration Assistant

ASM Configuration Assistant is a GUI tool that you use to install and configure an ASM instance, disk groups, volumes, and ACFS. Like the Database Configuration Assistant (DBCA), ASMCA can also be used in silent mode. Oracle Universal Installer internally uses ASMCA in silent mode to configure ASM disk groups to store OCR and voting files. ASMCA is capable of managing the complete ASM instance and associated ASM objects. Starting with Oracle 11*g*, DBCA does not allow creating and configuring ASM disk groups, and Oracle's future directions are to promote the use of ASMCA because it is a complete ASM management tool.

ASMCMD: The ASM Command-Line Utility

ASMCMD is a command-line tool that lets you access the ASM files and related information via a command-line interface, which makes ASM management easier and handy for DBAs. Oracle has enabled this tool with several management features that help users in managing an ASM instance as well as ASM objects such as disk groups, volumes, and an ASM cluster file system from the command line.

You can connect to either an ASM instance or an Oracle database instance when using ASMCMD, depending on what activities you want to perform. In order to log into ASMCMD, make sure that the ORACLE_HOME, PATH, and ORACLE_SID environment variables point to the ASM instance. Then, execute the **asmcmd** command located in the Oracle Grid Infrastructure home, which is the same as the Oracle ASM home. In order to connect to a database instance, run the ASMCMD utility from the bin directory of the Oracle Database home.

ASMCMD provides the DBA with a similar look and feel as well as the privileges of most UNIX-flavor systems, with commands such as **cd**, **ls**, **mkdir**, **pwd**, **lsop**, **dsget**, and so on. Starting with Oracle 11*g*, lots of new commands have been added to manage the complete ASM instance and its objects. It is now possible to completely manage an ASM instance from the command-line interface, including starting/stopping; managing the disk groups, volumes, and the ASM cluster file system; backing up and restoring metadata; and even accessing the GPnP profile and ASM parameter file within ASMCMD. You should refer to the command-line help for an explanation and sample usage of specific ASMCMD commands because the ASMCMD command-line help provides sufficient information for these commands. Following are a few examples that show how you can use ASMCMD to manage ASM.

Here's how to list all the disk groups:

https://t.me/learningnets

```
ASMCMD> ls
BACKUP/
DATA/
ASMCMD>

List all disks.
ASMCMD> lsdsk
Path
/dev/oracleasm/disks/DISK1
/dev/oracleasm/disks/DISK2
ASMCMD>
```

The following set of ASMCMD commands show how you can use familiar OS commands such as **cd**, **pwd**, and **ls** to find out the names of all the datafiles for a database that are stored in ASM:

```
ASMCMD> cd +DATA
ASMCMD> pwd
+DATA
ASMCMD> ls
ASM/
ORCL/
_MGMTDB/
orapwasm
rac-scan/
ASMCMD> cd ORCL
ASMCMD> ls
CONTROLFILE/
DATAFILE/
ONLINELOG/
PARAMETERFILE/
PASSWORD/
TEMPFILE/
ASMCMD> cd DATAFILE
ASMCMD> pwd
+DATA/orcl/DATAFILE
ASMCMD> ls
EXAMPLE.285.854663633
SYSAUX.277.854663311
SYSTEM.278.854663405
UNDOTBS1.280.854663501
UNDOTBS2.286.854664353
USERS.279.854663501
ASMCMD>
```

You can also run ASMCMD commands in a non-interactive manner by issuing a command from the command line itself without invoking ASMCMD. Here’s an

https://t.me/learningnets

Technet24

example that shows how to list the state and names of all disk groups:

```
$ asmcmd ls -l
State    Type    Rebal   Name
MOUNTED  EXTERN  N       BACKUP/
MOUNTED  EXTERN  N       DATA/
$
```

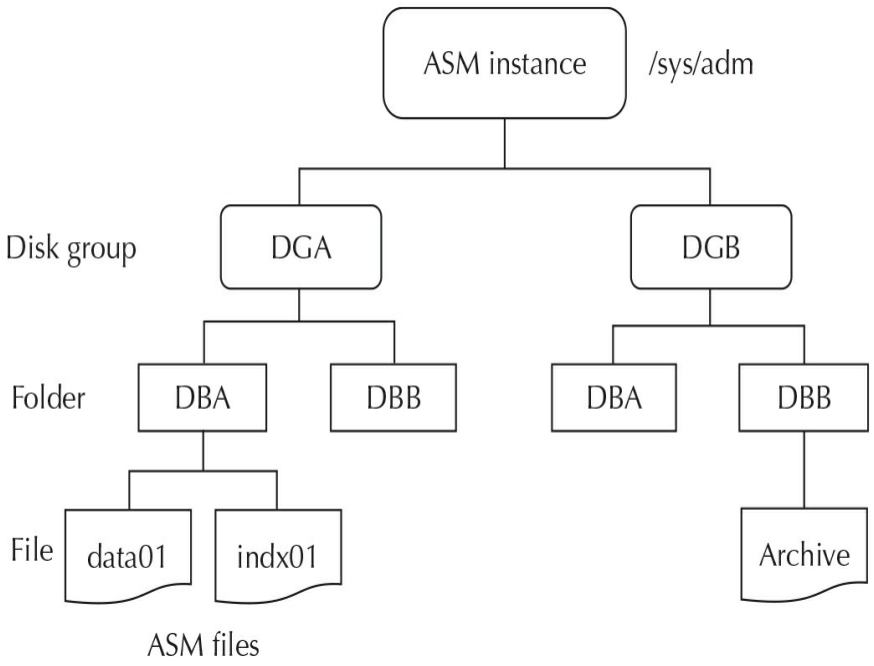
Because you can run ASMCMD interactively, it means you can also incorporate the commands in a shell script.

ASM FTP Utility

Oracle ASM supports ASM FTP, by which operations on ASM files and directories can be performed similar to conventional operations on normal files using conventional File Transfer Protocol (FTP). A typical use of such access to an ASM file can be to copy ASM files from one database to another.

The Oracle database leverages the virtual folder feature of XML DB, which provides a way to access the ASM files and directories through XML DB protocols such as FTP, Hypertext Transfer Protocol (HTTP), and programmatic APIs. An ASM virtual folder is mounted as /sys/asm within the XML DB hierarchy. The folder is called “virtual” because nothing is physically stored in XML DB. All operations are handled by underlying ASM components.

The virtual folder is created by default during the installation of XML DB. If the database is not configured to use automatic storage, this virtual folder will be empty and no operation will be permitted. The /sys/asm virtual folder contains folders and subfolders in line with the hierarchy of the ASM fully qualified naming structure. Figure 6-6 shows the hierarchy of an ASM virtual folder.



https://t.me/learningnets

FIGURE 6-6. *ASM virtual folder hierarchy*

As this figure shows, the virtual folder contains a subfolder for each mounted disk group in an ASM instance. Each disk group folder contains a subfolder for each database using that disk group subfolder, the database folder contains a file type subfolder, and the file type subfolder contains ASM files, which are binary in nature. Although we can access the ASM files as we access normal files in a conventional FTP application, some usage/access restrictions are in place. DBA privilege is a must to view the contents of the /sys/asm virtual folder.

The next example demonstrates accessing ASM files via the virtual folder. In this example, we assume that we are in the home directory of user *oracle*, which is /home/oracle, and we are connecting to the server, using FTP, where the ASM instance is hosted. The ASM instance is hosted on the racnode01 server. The disk group name is DGA, and the database name is *dba* using the DGA disk group.

First, we open the FTP connection and pass on the login information. Only users with the DBA privilege can access the /sys/asm folder. After connecting, we change our directory to the /sys/asm virtual folder and list the contents of the /sys/asm folder. A subfolder named DGA is used for the DGA disk group. Then we change to the DGA directory and see another subfolder with a database name, which is dba in our case. Then we list the contents of the dba directory that contains the ASM binary files related to the dba database. Finally, we download the file data01.dbf to our local directory, which is /home/oracle.

```
ftp> open racnode01 7777
ftp> use system
ftp> passwd manager
ftp> cd /sys/asm
ftp> ls
DGA
ftp> cd DGA
ftp> ls
dba
ftp> ls
data01.dbf
indx01.dbf
ftp> bin
ftp> get data01.dbf
```

ASMLib

ASMLib is the storage management interface that helps simplify the OS-to-database interface. The ASMLib API was developed and supported by Oracle to provide an alternative interface for the ASM-enabled kernel to identify and access block devices. The ASMLib interface serves as an alternative to the standard operating system

https://t.me/learningnets

Technet24

interface. It provides storage and operating system vendors the opportunity to supply extended storage-related features that provide benefits such as greater performance and integrity than are currently available on other database platforms.

Installing ASMLib

Oracle provides an ASM library driver for the Linux OS. You must install the ASM library driver prior to installing any Oracle Database software. In addition, it is recommended that any ASM disk devices required by the database be prepared and created before the OUI database installation. The ASMLib software is available on the Oracle Technology Network (OTN) for free download.

Three packages are available for each Linux platform. The two essential RPM packages are the oracleasmlib package, which provides the actual ASM library, and the oracleasm-support package, which provides the utilities to configure and enable the ASM driver. Both these packages need to be installed. The third package provides the kernel driver for the ASM library. Each package provides the driver for a different kernel. You must install the appropriate package for the kernel you are running.

Configuring ASMLib

After the packages are installed, the ASMLib can be loaded and configured using the configure option in the /etc/init.d/oracleasm utility. For Oracle RAC clusters, the oracleasm installation and configuration must be completed on all nodes of the cluster. Configuring ASMLib is as simple as executing the following command:

```
[root@racnode01/]# /etc/init.d/oracleasm configure
  Configuring the Oracle ASM library driver.

  This will configure the on-boot properties of the Oracle ASM library
  driver. The following questions will determine whether the driver is
  loaded on boot and what permissions it will have.  The current values
  will be shown in brackets ('[]'). Hitting  without typing an
  answer will keep that current value.  Ctrl-C will abort.

  Default user to own the driver interface []: oracle
  Default group to own the driver interface []: dba
  Start Oracle ASM library driver on boot (y/n) [n]: y
  Fix permissions of Oracle ASM disks on boot (y/n) [y]: y
  Writing Oracle ASM library driver configuration           [  OK  ]
  Creating /dev/oracleasm mount point                       [  OK  ]
  Loading module "oracleasm"                                [  OK  ]
  Mounting ASMlib driver filesystem                         [  OK  ]
  Scanning system for ASM disks                             [  OK  ]
```

Note that the ASMLib mount point is not a standard file system that can be accessed by operating system commands. Only the ASM library to communicate with

https://t.me/learningnets

the ASM driver uses it. The ASM library dynamically links with the Oracle kernel, and multiple ASMLib implementations can be simultaneously linked to the same Oracle kernel. Each library provides access to a different set of disks and different ASMLib capabilities.

The objective of ASMLib is to provide a more streamlined and efficient mechanism for managing disks and I/O processing of ASM storage. The ASM API provides a set of interdependent functions that need to be implemented in a layered fashion. These functions are dependent on the backend storage implementing the associated functions. From an implementation perspective, these functions are grouped into three collections of functions.

Each function group is dependent on the existence of the lower-level group. *Device discovery functions* are the lowest-layer functions and must be implemented in any ASMLib library. *I/O processing functions* provide an optimized asynchronous interface for scheduling I/O operations and managing I/O operation completion events. These functions, in effect, extend the operating system interface. Consequently, the I/O processing functions must be implemented as a device driver within the operating system kernel.

The *performance and reliability functions* are the highest-layer functions and depend on the existence of the I/O processing functions. These functions use the I/O processing control structures for passing metadata between the Oracle database and the backend storage devices. The performance and reliability functions enable additional intelligence on the part of backend storage. This is achieved when metadata transfer is passed through the ASMLib API.

Device Discovery

Device discovery provides the identification and naming of storage devices that are operated on by higher-level functions. Device discovery does not require any operating system code and can be implemented as a standalone library invoked and dynamically linked by the Oracle database. The discovery function makes the characteristics of the disk available to ASM. Disks discovered through ASMLib do not need to be available through normal operating system interfaces. For example, a storage vendor may provide a more efficient method of discovering and locating disks that its own interface driver manages.

I/O Processing

The current standard I/O model imposes a lot of OS overhead, due in part to mode and context switches. The deployment of ASMLib reduces the number of state transitions from kernel to user mode by employing a more efficient I/O schedule and call-processing mechanism. One call to ASMLib can submit and reap multiple I/Os. This dramatically reduces the number of calls to the OS when I/O is performed. Additionally, one I/O handle can be used by all the processes in an instance for accessing the same disk. This eliminates multiple open calls and multiple file descriptors.

https://t.me/learningnets

Technet24

One of the critical aspects of the ASMLib I/O interface is that it provides asynchronous I/O interface-enhancing performance and enables database-related intelligence on the backend storage devices. As for additional intelligence in the backend storage, the I/O interface enables passing metadata from the database to the storage devices. Future developments in the storage array firmware may allow the transport of database-related metadata to the backend storage devices and will enable new database-related intelligence in the storage devices.

Oracle ASM Filter Driver (Oracle ASMFD)

Oracle Database 12*c* offers Oracle ASM Filter Driver as an alternative to ASMLib, for eliminating the need to rebind ASM devices upon a system restart. ASMFD is a kernel module that validates I/O requests to the ASM disks by rejecting invalid I/O such as a non-Oracle I/O, which could overwrite ASM data.

You can choose to configure the new Oracle ASMFD when installing Oracle Grid Infrastructure. If you're already using ASMLib, you can also migrate to Oracle ASMFD.

The Oracle Installer automatically installs Oracle ASMFD with Oracle Database 12*c*. So, if you are migrating to Oracle Database 12*c*, you'll have Oracle ASMFD installed after the upgrade. If you prefer to keep using your current ASMLib setup, you don't need to do anything—the ASMFD executables will be installed but won't be automatically used by Oracle. If, on the other hand, you'd like to migrate from ASMLib to ASMFD, you must remove ASMLib and configure the ASM devices to use ASMFD instead. Here are the steps to follow:

1. Update the ASM disk discovery string so it can discover the ASM Filter Driver disks. You can update the ASM disk discovery string, as shown here, to add ASMFD disk label names to the string:

   ```
   ASM_DISKSTRING = 'AFD:DISK0',  'AFD:DISK2', /dev/rdsk/proddisks/*'
   ```

   You may also use the wildcard notation by specifying **'AFD:*'** instead of multiple disks. Another way to update the disk discovery string is by executing the **asmcmd dsset** command, as shown here:

   ```
   $ asmcmd dsset 'AFD:*', '/dev/rdsk/prodisks/*'
   ```

   You can use the **asmcmd dsget** command to view the value of the current ASM disk discovery string.
2. List all nodes and their roles by issuing the **olsnodes –a** command and run the following set of commands on each hub node.
3. Stop Oracle Clusterware with the **crsctl stop crs** command first and then configure ASMFD as follows:

   ```
   $ asmcmd afd_configure
   ```

   If the **afd_configure** command returns NOT AVAILABLE, it means that Oracle ASMCFD isn't configured. If ASMCFD was successfully configured, the command will show the status as "configured." At this point, the Oracle

https://t.me/learningnets

ASM instance can register with the ASM Filter Driver.

4. Verify the ASMFD status:

```
$ asmcmd afd_state
```

5. Start the Oracle Clusterware stack on the node (**crsctl start crs**) and set the Oracle ASMFD disk discovery string to its original ASM disk discovery string value:

```
$ asmcmd afd_dsset old_diskstring
```

From here on, ASMFD will manage the disks instead of ASMLib.

Summary

Automatic Storage Management is the best framework for managing data storage for the Oracle database. ASM implements the Stripe and Mirror Everything (SAME) methodology to manage the storage stack with an I/O size equal to the most common Oracle I/O clients, thus providing tremendous performance benefits.

Oracle has enhanced and improved ASM to a level that it is supported with non-Oracle enterprise applications to store application data. ASM provides the GUI tools ASMCA and OEM for management of the ASM instance and its associated objects. The ASM Cluster File System is a true cluster file system built on ASM foundations, and it eliminates the need for third-party cluster file systems in enterprise cluster applications. ACFS also comes with rich functionalities such as snapshots and encryption.

Command-line tools such as SQL*Plus, ASM FTP, and ASMCMD are also available to provide a command-line interface to design and build provisioning scripts. Added features of the ACFS and ACFS snapshot help keep managing a huge amount of storage from being a complex task that involves lots of planning and day-to-day administration issues.

https://t.me/learningnets

Technet24

PART III

Oracle RAC Administration and Management

https://t.me/learningnets

CHAPTER 7

Oracle RAC Basic Administration

Administering an Oracle Real Application Clusters database is similar to how you manage single-instance database administration. Accomplishing tasks such as managing users, datafiles, tablespaces, controlfiles, and undo tablespaces is similar to managing those tasks in a single-instance environment.

The information in this chapter assumes you are familiar with single-instance database administration and focuses on aspects of Oracle RAC that are different from those of a single instance. This chapter covers the common tasks an Oracle RAC DBA performs on a day-to-day basis.

Following are the key database management areas we focus on in this chapter:

- Oracle RAC initialization parameters
- Starting and stopping instances
- Common SRVCTL management commands
- Administering undo in an Oracle RAC database
- Administering a temporary tablespace
- Administering online redo logs
- Enabling the flashback area

https://t.me/learningnets

Technet24

- Managing database configuration with SRVCTL
- Killing sessions on specific instances of a cluster
- Managing database objects
- Administering server pools

Oracle RAC Initialization Parameters

Oracle is a highly flexible, function-rich RDBMS that can be used on a laptop by a single user or used on the world's most powerful computer systems and accessed by thousands of users. How does Oracle offer such flexibility? How does an end user choose which functions to use and optimize the database for the intended use? The answers lie in Oracle's parameter file, which contains a set of parameters that allow an end user to customize and optimize an Oracle database instance. When an Oracle instance starts, it reads these parameters, which in turn define the features the database uses, the amount of memory an instance uses, how users interact with the database, and other essential information.

In an Oracle RAC environment, a DBA can use a parameter file shared by all the instances, or each instance can have its own dedicated parameter file. Using a shared SPFILE for managing instances in an Oracle RAC environment is highly recommended.

NOTE

In an Oracle RAC environment, it's recommended that you use an SPFILE that multiple instances can share. You should therefore store the SPFILE on a shareable disk subsystem, such as a clustered file system or Automatic Storage Management (ASM).

In the SPFILE, the value before the period (.) tells you if the parameter value applies to a specific instance or to all database instances in the cluster. If an asterisk (*) is shown before the period, it means that the parameter value is identical across all instances. The following example means that the database-wide setting for the OPEN_CURSORS parameter is 500, with the exception of the prod2 instance, whose OPEN_CURSORS parameter value was explicitly set by the DBA to the value 1000:

```
*.OPEN_CURSORS=500
prod2.OPEN_CURSORS=1000
```

Running the following command won't raise the value of the OPEN_CURSORS parameter for the prod2 instance to 1500; it still remains at 1000 even though you specified **'*'** because earlier the DBA had explicitly set the value of this parameter for the prod2 instance to 1000. The new value of 1500 only applies to the rest of the

https://t.me/learningnets

instances in the cluster.

```
SQL> ALTER SYSTEM SET OPEN_CURSORS=1500 sid='*' SCOPE=SPFILE;
```

In order to reset the value of the OPEN_CURSORS parameter to 1500 for the prod2 instance as well, you must issue the following command:

```
SQL> ALTER SYSTEM SET OPEN_CURSORS=1500 SCOPE=SPFILE SID='prod2';
```

Oracle instance parameters for the Oracle RAC environment are grouped into three major categories:

- **Unique parameters** These parameters are unique to each instance. Parameters in this category identify resources to be used by an instance exclusively. Some examples of this type of parameter are INSTANCE_NAME, THREAD, and UNDO_TABLESPACE.
- **Identical parameters** Parameters in this category must to be the same on all the instances in an Oracle RAC environment. Parameters defining the database characteristics, inter-instance communication characteristics, and so on, fall into this category. The parameters MAX_COMMIT_PROPOGATION_DELAY, COMPATIBLE, DB_NAME, and CONTROL_FILE belong to this category. You can use the ISINSTANCE_MODIFIABLE column of the V$PARAMETER view to see a list of such parameters, like so:

```
select name, ISINSTANCE_MODIFIABLE
from v$parameter
where ISINSTANCE_MODIFIABLE='FALSE'
order by name;
```

- **Parameters that “should” be the same** This category contains initialization parameters that don’t fall into either of the preceding two categories. Those initialization parameters that define performance characteristics of an instance generally fall into this section. Some examples of this type of parameter are DB_CACHE_SIZE, LOCAL_LISTENER, and GCS_SERVER_PROCESSES.

Parameters That Are Unique to an Instance

In the previous section, you learned about the initialization parameters that are common to all instances in an Oracle RAC cluster. In addition to these, there are several initialization parameters that are unique to each instance. We explain these unique parameters in the following sections.

INSTANCE_NAME

This parameter defines the name of the Oracle instance. Oracle uses the INSTANCE_NAME parameter to assign redo log groups to an instance. Although not required, this parameter is chosen to be the same as the ORACLE_SID environment variable defined at the operating system level.

https://t.me/learningnets

Technet24

The default value for this parameter is also the value of the environment variable ORACLE_SID. In a single-instance environment, the instance name is usually the same as the database name. In an Oracle RAC environment, the instance name is constructed by appending a suffix to each instance of the database (for example, for a database named "prod," instance names could be prod1, prod2, prod3, and so on).

The value of this parameter isn't normally set in the parameter file, thus leaving it at its default value of SID. We strongly recommend *not* specifying this parameter and leaving it at its default setting.

Note that some of the dynamic performance views (more than 30 of them) that contain the column instance_name or inst_name derive the value from the environment variable ORACLE_SID and *not* from this parameter.

The following SQL session demonstrates the parameter's behavior:

```
$echo $ORACLE_SID
PROD1
$grep instance_name initPROD1.ora
instance_name=xyz
. . .
SQL>
SQL> select instance_name from v$instance ;
INSTANCE_NAME
----------------
PROD1
SQL> show parameter instance_name
NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------
instance_name                        string      xyz
```

INSTANCE_NUMBER

This is a unique number greater than zero and smaller than the MAX_INSTANCE parameter specified when the database was created. Oracle uses it internally to identify the instance. This way, it can distinguish among the multiple instances in a cluster at startup time. The INST_ID column in GV$ views (such as GV$LOGFILE) corresponds to the value of this parameter for that instance. Generally, the value of this parameter is kept the same as the value of the thread parameter, to simplify administration.

The following example sets the INSTANCE_NUMBER parameter value to 1 for the instance prod1:

```
prod1.instance_number=1
```

GCS_SERVER_PROCESSES

This parameter specifies the initial number of server processes for the Global Cache Services (GCS), which handle the inter-instance traffic among the instances. You can

https://t.me/learningnets

set this parameter to a different value across the instances in a cluster. The minimum setting for this parameter is 2, and the default number is based on the CPU count. If a system has two to eight CPUs, the parameter value is set to 2, and for systems with more than eight CPUs, the number of GCS processes is the number of CPUs divided by four.

This parameter specifies the number of Lock Manager Server (LMS) background processes used by the instance for the Cache Fusion traffic. Although you can change this parameter's value from its default value of 2, note that the number of LMS processes needs to be the same on the instances to allow for properly balanced communication across the cluster. This is a key parameter that has a vast bearing on performance.

THREAD

This parameter specifies the set of redo log files to be used by the instance. Each instance has its own set of redo. Each redo thread consists of at least two groups, each with one or more members (files). If you don't specify the THREAD parameter, all thread values default to 1. Therefore, only the first instance will start.

The value of the parameter INSTANCE_NUMBER should be the same as the value of the THREAD parameter. The following example assigns a THREAD value of 2 to instance prod2:

```
prod2.thread = 2
```

Oracle best practice is to use the INSTANCE_NUMBER parameter to specify the redo log groups for an instance.

UNDO_TABLESPACE

This parameter specifies the name of the undo tablespace to be used by an instance. An instance uses its own undo tablespace to write the undo data for its transactions; however, it can read other instances' undo tablespaces. The following line specifies undo tablespace UNDOTBS1 for the instance prod1:

```
prod1.undo_tablespace='UNDOTBS1'
```

For more information on managing the undo tablespace, refer to the "Administering Undo in an Oracle RAC Database" section of this chapter.

CLUSTER_INTERCONNECTS

This optional parameter is used only when the Oracle instance is not able to pick the correct interconnect for inter-instance communication automatically. This parameter can also be used to specify multiple network interconnects to be used for Oracle RAC traffic. Use this parameter to specify the IP address of the network to be used for Cache Fusion traffic. In the following example, 10.0.0.1 is the IP address of the private interconnect to be used for the cluster traffic:

https://t.me/learningnets

Technet24

```
prod1.cluster_interconnects = "10.0.0.1"
```

The following example specifies two IP addresses for Oracle RAC network traffic:

```
prod1.cluster_interconnects = "10.0.0.1:10.0.0.2"
```

When multiple networks cards/interconnects are to be used for Oracle RAC traffic, it is advisable that you use OS-level techniques to configure the desired failover and traffic-sharing features.

You can specify a private interconnect during installation and therefore do not need to use this parameter. The database should also automatically detect the network to be used for the private interconnect, so generally you do not need to use this parameter. Therefore, use this parameter as a last resort to specify network interconnects.

NOTE

Specifying multiple interconnects by using the CLUSTER_INTERCONNECTS parameter enables Cache Fusion traffic to be distributed on all the interconnects. However, if any one of the interconnects is down, Oracle will assume the network is down; thus, it does not provide redundancy and high availability.

ASM_PREFERRED_READ_FAILURE_GROUPS

This optional parameter specifies the name of the preferred failure disk groups so that disks in the failure groups become the preferred read disks in order for the instance to read from disks closer to it. This is an instance-specific parameter. The preferred read failure group is used in a special type of cluster configuration called *extended clusters* or *metro clusters*. Implementation consideration and installation strategies of extended clusters are discussed in Chapter 14.

Identical Parameters

Using the following query, you can determine that 131 parameters fall into the identical parameters category:

https://t.me/learningnets

```
select name, ISINSTANCE_MODIFIABLE
from v$parameter
where ISINSTANCE_MODIFIABLE='FALSE'
order by name;
NAME                                                    ISINSTANCE_
-------------------------------------------------- ---------------
DBFIPS_140                                                    FALSE
O7_DICTIONARY_ACCESSIBILITY                                   FALSE
active_instance_count                                         FALSE

131 rows selected.
SQL>
```

However, we will restrict our discussion to commonly used parameters. For information on other parameters, refer to the *Oracle Database Reference Manual*.

We divide the initialization parameters that must be identical across instances into two groups:

- Oracle RAC–specific parameters
- Database characteristic parameters (parameters that are alike across instances). Here's an example:

```
*.cluster_database = TRUE
```

Oracle RAC–Specific Initialization Parameters

Following are the Oracle RAC–specific initialization parameters.

CLUSTER_DATABASE The possible values of this Boolean parameter are TRUE and FALSE. The value of this parameter should be set to TRUE for all the instances. The TRUE value directs the instance during startup to mount the controlfile in shared mode. If an instance starting first has this parameter set to FALSE, it will be able to mount the controlfile in exclusive mode. However, this will prevent all subsequent instances from starting up.

During the following maintenance activities, you must start up an instance using **cluster_database = FALSE**:

- Converting from no archive log mode to archive log mode, and vice versa
- Enabling the Flashback Database feature
- During upgrades
- Performing a media recovery on a system tablespace
- Converting to a single-instance database from Oracle RAC database, and vice versa

The default value for this parameter is FALSE. To start up instances in an Oracle RAC environment, this parameter must be set to TRUE.

https://t.me/learningnets

Technet24

CLUSTER_DATABASE_INSTANCES This parameter specifies the number of instances that will be accessing the database. For administrator-managed cluster databases, the default value for this parameter is based on the number of database instances configured in the cluster database, whereas for policy-managed cluster databases, the default value for this parameter is 16 (you should set the value for this parameter higher than 16 if you are expecting more than 16 database instances in this cluster database). Oracle uses the value of this parameter to compute default values of some other parameters such as LARGE_POOL_SIZE.

The value of this parameter should be set equal to the maximum number of instances set to join the cluster.

Parameters Specifying Database Characteristics (Alike Across Instances)

The following parameters specify database characteristics that need to be alike across all instances. These parameters behave exactly the same as in a single-instance environment; refer to the Oracle Database 11*g* R2 reference manual for more information about these parameters.

- COMPATIBLE
- CLUSTER_DATABASE
- CONTROL_FILES
- DB_BLOCK_SIZE
- DB_DOMAIN
- DB_NAME
- DB_RECOVERY_FILE_SIZE
- DB _RECOVERY_FILE_DEST_SIZE
- DB_UNIQUE_NAME
- INSTANCE_TYPE
- PARALLEL_EXECUTION_MESSAGE_SIZE
- REMOTE_LOGIN_PASSWORDFILE
- UNDO_MANAGEMENT

Instance Parameters That “Should” Be the Same

In addition to the sets of unique and identical initialization parameters we just described, Oracle also recommends that you use identical settings for the following initialization parameters, even though it’s not required. This is for various reasons, such as minimizing overhead, improving performance, and enhancing the stability of the cluster.

https://t.me/learningnets

- ARCHIVE_LAG_TARGET
- CLUSTER_DATABASE_INSTANCES
- LICENSE_MAX_USERS
- LOG_ARCHIVE_FORMAT
- SPFILE
- TRACE_ENABLED
- UNDO_RETENTION

Managing the Parameter File

In the Oracle RAC environment, a DBA can use a parameter file shared by all instances, or each instance can have its own dedicated parameter file. Following is the syntax for a shared parameter file containing parameters for multiple instances:

```
<instance_name>.<parameter_name>=<parameter_value>
```

As mentioned earlier, an asterisk (*) or no value for <instance_name> implies that the parameter value is valid for all the instances. Here's an example:

```
inst1.db_cache_size = 1000000
*.undo_management=auto
```

If a parameter appears more than once in a parameter file, the last specified value is the effective value, unless the values are on consecutive lines, in which case values from consecutive lines are concatenated.

The ALTER SYSTEM SET command is available to change initialization parameters for an Oracle instance using SPFILE. Following is the complete syntax of the command:

```
alter system set <parameter>=<value>
  scope=<memory/spfile/both>
  comment=<'comments'>
  deferred
  sid=<sid, *>
```

- **scope=memory** Indicates that the parameter should be changed only for the current instance; these changes are lost upon instance restart. If a parameter can be modified only for the local instance, the following error is signaled: "ORA-32018: Parameter cannot be modified in memory on another instance."
- **scope=spfile** Indicates that the changes are to be made only in the file, and new values of the changed parameter will be effective upon instance restart. The current incarnation of the instance is not affected by this change. If an instance was not started using an SPFILE and you try to use this option, the following error is signaled: "ORA-32001: Write to SPFILE requested but no SPFILE

https://t.me/learningnets

Technet24

specified at startup.”

- **scope=both** Means that the intended parameter change will be effective for the current incarnation of the instance and is persistent through instance restarts.
- **comment** Allows you to specify comments to be associated with the parameter change.
- **deferred** Indicates that the changes are effective only for the sessions that are spawned after the command is issued. The sessions that are already active are not affected.
- **sid** Allows specifying the name of the instance the parameter change is intended for. An asterisk (*) implies that the change is to be made across all the instances. This is the default value as well. Generally, in an Oracle RAC environment, you can change parameter values for remote instances, with some restrictions depending on the status of the remote instance and the static/dynamic nature of the parameter. Here is an example:

```
alter system set db_2k_cache_size=10m scope=spfile sid='prod2';
```

Whenever a parameter is changed using the **alter system** command, Oracle logs the command in the alert.log file.

When the Database Configuration Assistant (DBCA) is used to create an Oracle RAC database, by default it creates an SPFILE on the shared disk subsystem used for the database. If for any reason SPFILE is not being used, the **create spfile...** command can be used to create an SPFILE from a parameter file.

Backing Up the Server Parameter File

You must make regular backups of the server parameter file to aid recovery. Simply run the **create pfile** command, as shown here, to back up the SPFILE:

```
CREATE PFILE='/u01/oracle/dbs/test_init.ora'
FROM SPFILE='/u01/oracle/dbs/test_spfile.ora';
```

You can use the Recovery Manager (RMAN) to back up a server parameter file. Add the CONTROLFILE AUTOBACKUP statement to a database backup command to ensure that RMAN automatically backs up the SPFILE.

Search Order for the Parameter Files in an Oracle RAC Database

You can find the location of the parameter file for an Oracle RAC database by issuing the **srvctl config database** command. Here’s an example:

https://t.me/learningnets

```
[oracle@rac1 bin]$ srvctl config database -db orcl
Database unique name: orcl
Database name: orcl
Oracle home: /u01/app/oracle/product/12.2.0/dbhome_1
Oracle user: oracle
Spfile: +DATA/ORCL/PARAMETERFILE/spfile.299.941476551
Password file: +DATA/ORCL/PASSWORD/pwdorcl.282.941475837
Domain: localdomain
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools:
Disk Groups: DATA
Mount point paths:
Services:
Type: RAC
Start concurrency:
Stop concurrency:
OSDBA group: dba
OSOPER group:
Database instances: orcl1,orcl2
Configured nodes: rac1,rac2
CSS critical: no
CPU count: 0
Memory target: 0
Maximum memory: 0
Default network number for database services:
Database is administrator managed
[oracle@rac1 bin]$
```

An Oracle RAC database searches for the parameter file in the following order on a Linux/UNIX platform:

```
$ORACLE_HOME/dbs/spfilesid.ora
$ORACLE_HOME/dbs/spfile.ora
$ORACLE_HOME/dbs/initsid.ora
```

Starting and Stopping Instances

An Oracle instance can start automatically at system boot time. However, many organizations choose not to start an instance automatically because it may have gone down (or been taken down) for a specific reason and taken the node down during node eviction.

In an Oracle RAC environment, you need to ensure that the cluster software is

https://t.me/learningnets

Technet24

started before database instances can be started. The DBCA configures database instances to be started automatically by Cluster Ready Services (CRS) at system boot time.

Using SRVCTL to Start/Stop Databases and Instance(s)

The SRVCTL utility provides a convenient and highly recommended way to start/stop multiple instances in an Oracle RAC environment.

Starting Databases, Instances, and Listeners

Use the following command to start all instances associated with a database:

```
srvctl start database -db prod
```

In this example, the database name is prod. You can execute this command from any of the nodes. The command also starts listeners on each node if they're not already running. This command also starts the services defined for the prod database.

If you want to just evaluate the impact of the **srvctl start** command without actually starting the instances, you can run the following **eval** command instead:

```
srvctl start database -db prod eval
```

You can also specify various startup options with the **-startoption** parameter, such as OPEN, MOUNT, and NOMOUNT. The following example shows how to mount all instances of an Oracle RAC database:

```
srvctl start database -db prod -startoption mount
```

The following example shows how to start up a database in the read-only mode:

```
srvctl start database -db prod -startoption 'read only'
You can start a listener with the srvctl start listener command:
$ srvctl start listener -listener LISTENER1
```

Checking the Status of a Service

You can check the status of the database instances by issuing the following command:

```
$ srvctl status database -db orcl
Instance orcl1 is running on node rac1
Instance orcl2 is running on node rac2
$
```

This command shows the status of the database named ORCL.

You can similarly check the status of an instance, a listener, or a disk group with

https://t.me/learningnets

the following commands.

The output of the **srvctl status instance** command reveals that instance *db00* running on the node *node10:*

```
$ srvctl status instance -db db00 -node node10 -verbose
```

The following command shows the status of the default listener on the node *node2*:

```
$ srvctl status listener -node node2
```

The **srvctl status diskgroup** command shows the status of a specific disk group on a set of nodes that you specify:

```
$ srvctl status diskgroup -diskgroup dgrp1 -node "mynode1, mynode2" -detail
```

You can check the status of all Oracle Clusterware–managed resources for an Oracle home with the following command:

```
$ srvctl status home -oraclehome /u01/app/oracle/product/12.1/dbhome_1 -statefile
 ~/state.txt -node stvm12
Database cdb1 is running on node stvm12
$
```

If you’re using pluggable databases (PDBs), you can check the status of those databases, not by checking the instances but by checking the availability of the services assigned to the PDBs, as shown here:

```
$ srvctl status service –db prod –service service1
```

In this example, service1 is the service that was assigned to the PDB named PDB1.

To start specific instances of an administrator-managed database, enter a comma-delimited list of instance names, like so:

```
$ srvctl start instance -db db_unique_name -instance instance_name_list
  [-startoption start_options]
```

To start a policy-managed database, enter a single node name, as shown here:

```
$ srvctl start instance -db db_unique_name -node node_name
  [-startoption start_options]
```

Stopping Databases, Instances, and Listeners

You can stop a database, or just one or more instances. You can execute the **srvctl stop** command to also stop the various listeners.

Stopping a Database To shut down all instances associated with the PROD database (that is, to shut down the PROD database completely), you can use the

https://t.me/learningnets

Technet24

following command (as with the START command, you can run this command with the added **eval** parameter, to evaluate the impact of the command and not actually shut down the database):

```
srvctl stop database  -db prod
```

This command shuts down only instances and services; it doesn't stop the listeners because they might be serving other database instances running on the same machines.

You can use the **-stopoption** option to specify startup/shutdown options. SRVCTL invokes CRS application programming interfaces (APIs) silently to perform start/stop operations. Options specified by the **-stopoption** parameter are directly passed on as command-line options for start/stop commands. For example, to stop all the instances with the **immediate** option, use the following command:

```
srvctl stop database -db prod -stopoption immediate
```

When stopping a database, you can use the **drain_timeout** option to specify the maximum time (in seconds) the service waits before stopping a database:

```
$ srvctl stop database -db db1 -drain_timeout 50 -verbose
Draining in progress on services svc1,svc2.
Drain complete on services svc1.
Draining in progress on services svc2.
Draining in progress on services svc2.
Drain complete on services svc2.
$
```

Stopping an Instance To perform a normal shutdown of instance prod3, use the **srvctl stop instance** command, as shown here:

```
srvctl stop instance  -db prod -instance "prod3"
```

When you shut down a single instance in a cluster, as shown in this example, the rest of the instances in the RAC cluster continue to function normally. Note that you could also stop a single instance running on a node by providing the **-node** parameter instead of the **-instance** parameter, as shown here:

```
$ srvctl stop instance  -db prod -node node1
```

The **-instance** parameter, however, lets you stop either one or a set of instances, as shown in the following example:

```
srvctl stop instance  -db prod -instance "prod1,prod2" -stopoption immediate
```

As with the case of the **stop database** command, you can add the **-stopoption** parameter to the **stop instance** command to specify various options for the shutdown command, such as NORMAL, TRANSACTIONAL, LOCAL, IMMEDIATE, and

https://t.me/learningnets

ABORT.

If you shut down an instance with the ABORT option, or if the instance terminates abnormally on its own, one of the instances that are running will perform instance recovery for the instance that was shut down or brought down. If no other instance is running at the time, the first instance to open the database will perform the instance recovery for all instances that require recovery.

It's important to understand that when you're stopping a service, or an instance, the drain timeout is the maximum time the service or instance waits before exiting. Here's an example showing how to specify the **drain_timeout** parameter while stopping an instance:

```
$ srvctl stop instance -db db1 -instance inst1 -drain_timeout 50 -verbose
Draining in progress on services svc1
Draining in progress on services svc1
Drain complete on services svc1
$
```

Stopping the Listener and Other Entities You can stop the listener as well as other entities such as disk groups, node apps, Oracle Notification Service (ONS), scans, services, virtual IP addresses (VIPs), and volumes with the **srvctl stop** command. Here's an example that shows how to stop all listeners running on a specific node:

```
$ srvctl stop listener -node mynode1
```

By running just the **srvctl stop listener** command, you can stop the listener on all nodes.

The following command stops the disk group named diskgroup1 on the nodes mynode1 and mynode2:

```
$ srvctl stop diskgroup -diskgroup diskgroup1 -node "mynode1,mynode2" -force
```

And the **srvctl stop home** command stops all Oracle Clusterware–managed resources that run from an Oracle home:

```
$ srvctl stop home -oraclehome /u01/app/oracle/product/12.1.0/db_1 -statefile
 ~/state.txt
```

Administering the Oracle ASM Instances

You can use the SRVCTL commands **start asm**, **stop asm**, and **status asm** to start, stop, and view the status of an Oracle ASM instance. Here's the syntax for these three commands:

https://t.me/learningnets

Technet24

```
srvctl start asm [-node node_name] [-startoption start_options]
srvctl stop asm [-node node_name] [-stopoption stop_options]
srvctl status asm [-node node_name]
```

The **config asm** command shows the configuration of an ASM instance:

```
srvctl config asm -node node_name
```

Using CRSCTL to Stop Databases and Instances

By running the **crsctl stop crs** command on each node or the **crsctl stop cluster -all** command, you can stop either a single node or the entire cluster, respectively. Both these commands will automatically shut down all database instances running on a node or in the entire cluster. However, when you restart the cluster, an instance recovery is required because running either of these CRSCTL commands is akin to issuing the SHUTDOWN ABORT command.

If you use SRVCTL instead of the CRSCTL commands to manually shut down a database or an instance, you can avoid the instance recovery process. However, this also means that you must first start the Oracle Clusterware and then manually restart the database instances.

Using SQL*Plus to Start/Stop Instances

As in the case of a single-instance environment, you can use SQL*Plus to start/stop instances individually. Make sure that you set the appropriate value for the environment variable ORACLE_SID for the local instance. Here's an example:

```
$sqlplus '/ as sysdba'
>shutdown immediate;
. . .
$ sqlplus '/ as sysdba'
. . .
>startup
```

If you'd like to use SQL*Plus to start or stop an instance running on a different node from the one where you logged into SQL*Plus, you can do so by connecting to the remote instance using the Oracle Net Services connection string for the remote instance, as shown here:

```
SQL> connect /#prod2 as sysdba
SQL> shutdown
```

Because starting and stopping an instance through SQL*Plus requires you to actually log into each instance, you can't stop or start multiple instances simultaneously through SQL*Plus.

https://t.me/learningnets

You can check the name of the instance to which you're connected by issuing the following command:

```
SQL> select instance_name from v$instance;
INSTANCE_NAME
-------------
orcl1
SQL>
```

Note that connecting as "/ as sysdba" requires OS authentication and needs the logged-in user to be a member of the OSDBA group.

From SQL*Plus, you can check which cluster instances are running by querying the V$ASM_INSTANCES view, as shown here:

```
SQL> select * from V$ACTIVE_INSTANCES;
INST_NUMBER     INST_NAME                    CON_ID
-----------     ----------------------       -------
 1              rac1.localdomain:orcl1       0
 2              rac2.localdomain:orcl2       0
SQL>
```

As you can see, the query identifies both the host name and the instance name, in addition to the CON_ID column, which is new in Oracle Database 12*c* and is relevant when you use a multitenant database with CDBs and PDBs.

Although you can administer the database instances through SQL*Plus, as shown here, Oracle recommends that you not use SQL*Plus to administer the Oracle ASM instances. Oracle Clusterware manages the ASM instances automatically, and if you do need to perform any tasks manually, use SRVCTL instead of SQL*Plus.

Common SRVCTL Management Commands

Administrators will use SRVCTL commands frequently to manage an Oracle RAC database. We summarize the most common of the SRVCTL commands pertaining to databases, instances, and the Oracle listener in this section.

You must log in as the software owner of the home you wish to manage. If you want to manage a database, log in as the database software owner. If you want to manage Oracle ASM instances, which are part of the Grid Infrastructure, log in as the Grid Infrastructure software owner.

NOTE

Some SRVCTL commands require you to log in as the root user in a Linux/UNIX system.

https://t.me/learningnets

Technet24

Members of the OSDBA OS group can start and stop the database; to start and stop an ASM instance, you must be part of the OSASM OS group.

To view help for all SRVCTL commands, type the following:

```
$ srvctl -help
```

To view the usage information for a specific command, issue a command such as this:

```
$ srvctl config database -help
```

This command shows the usage information for the **srvctl config database** command, with all the attributes you may configure.

Database-Related SRVCTL Commands

In this section, we summarize the most frequently used database-related SRVCTL commands.

Adding a Database

You can add a database configuration to Oracle Clusterware with the **srvctl add database** command. Here's an example:

```
$ srvctl add database -db crm -oraclehome /u01/oracle/product/12c/mydb
-domain example.com
```

This command shows how to add an administrator-managed database.

To add a policy-managed Oracle RAC database, you use a differently structured version of this command, as shown in the following example:

```
$ srvctl add database -db crm -oraclehome /u01/oracle/product/12c/mydb
    -domain example.com -spfile +diskgroup1/crm/spfilecrm.ora
    -role PHYSICAL_STANDBY -startoption MOUNT -dbtype RAC -dbname crm_psd
    -policy MANUAL -serverpool "svrpool1,svrpool2" -diskgroup "dgrp1,dgrp2"
```

Viewing the Configuration of a Database

You run the **srvctl config database** command to view the configuration of an Oracle RAC database. The command also lists all databases registered with Oracle Clusterware. Here's an example:

https://t.me/learningnets

```
$ srvctl config database -d main

Database unique name: main
Database name:
Oracle home: /u01/app/oracle
Oracle user: salapati
Spfile:
Password file:
Domain:
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools:
Disk Groups:
Mount point paths:
Services: test
Type: RAC
Start concurrency:
Stop concurrency:
OSDBA group: dba
OSOPER group: oper
Database instances: main41,main42
Configured nodes: salapati_main4_0,salapati_main4_1
CSS critical: no
CPU count: 0

Memory target : 0
Maximum memory: 0
CPU cap: 0
Database is administrator managed
```

Enabling/Disabling a Database

You can enable or disable a database. You enable a database with the **srvctl enable database** command. Here's an example:

```
$ srvctl enable database -db prod
```

This command enables all instances of the database named prod.

You disable a database with the **srvctl disable database** command, like so:

```
$ srvctl disable database -db prod
```

Enabling or disabling a database enables or disables all the instances as well.

https://t.me/learningnets

Technet24

Instance-Related SRVCTL Commands

This section covers several instance-level SRVCTL commands, such as **srvctl add instance**.

Adding an Instance

Run the **srvctl add instance** command to add the configuration for an instance to a cluster's database configuration, as shown here:

```
$ srvctl add instance -db crm -instance crm01 -node gm01
```

Enabling and Disabling Instances

Use the **srvctl enable instance** and **srvctl disable instance** commands to enable/disable an instance, respectively:

```
$ srvctl enable instance -db crm -instance "crm1,crm3"
$ srvctl disable instance -db crm -instance "crm1,crm3"
```

If you enable or disable all instances of a cluster, you'll be enabling or disabling the database as well.

Listener-Related SRVCTL Commands

You can create a listener as well as disable it with SRVCTL commands.

Creating a Listener

You can run the **srvctl add listener** command to create an Oracle Database listener, an Oracle listener, or a Leaf listener. The following example shows how to add a listener that runs on three different nodes of a cluster, on different ports:

```
$ srvctl add listener -listener listener112 -endpoints "1341,1342,1345"
-oraclehome /u01/app/oracle/product/12.2.0/db1
```

Enabling and Disabling a Listener

Use the following commands to enable/disable a listener running on a specific node (node2 in this example):

```
$ srvctl enable listener -listener listener_crm -node node2
$ srvctl disable listener -listener listener_crm -node node2
```

Displaying a Listener's Configuration

The **srvctl config listener** command shows a listener's configuration.

```
$ srvctl config listener listener_crm
```

https://t.me/learningnets

Setting, Unsetting, and Displaying the Environment Variables

Run the SRVCTL command with the **setenv**, **getenv**, and **unsetenv** options to set, display, and unset the values, respectively, for the environment variables in the configuration files.

You can run the **srvctl setenv**, **srvctl getenv**, and **srvctl unsetenv** commands for databases, instances, services, and node applications. Following are some examples showing how to administer the environment configuration of a database and a listener.

The following example shows how to set the language environment configuration for a database using the **setenv** command:

```
$ srvctl setenv database -db crm -env LANG=en
```

Similarly, the **srvctl unsetenv** command unsets the cluster configuration for a database or a listener. Here's how to unset the environment configuration for a database environment variable:

```
$ srvctl unsetenv database -db crm -envs "CLASSPATH,LANG"
```

The **srvctl getenv database** command displays the values of a database's environment variables:

```
$ srvctl getenv database -db crm
```

The **srvctl getenv listener** command shows the environment variables for a specific listener:

```
$ srvctl getenv listener
```

Changing the Configuration of Databases and Instances

When you need to change the configuration of a database instance, a listener, or any other Oracle Clusterware resource (such as the network, node app, ONS, scan, service, or server pool), you don't need to re-create the resource. Simply run the **srvctl modify <resource>** command to modify the service's configuration in the OCR. When you restart the modified service, the new configuration comes into effect.

Modifying a Database

You can modify a database's configuration, such as the disk groups assigned to it, or the role of the database. In the following examples, we modify the role of a database to a logical standby and also show how to specify ASM disk groups for a database:

https://t.me/learningnets

Technet24

```
$ srvctl modify database -db crm -role logical_standby
$ srvctl modify database -db racProd -diskgroup "SYSFILES,LOGS,OLTP"
```

Modifying an Instance

You can run the **srvctl modify instance** command to change an instance's configuration from its current node to a different node, as shown here:

```
$ srvctl modify instance -db amdb -instance amdb1 -node mynode
```

In the case of a policy-managed database, the command defines the instance name to use when the database runs on a specific node:

```
$ srvctl modify instance -db pmdb -instance pmdb1_1 -node mynode
```

The following example undoes the result of the previous command:

```
$ srvctl modify instance -db pmdb -instance pmdb1_1 -node ""
```

Relocating Services

You can relocate various services, such as an Oracle One Node database, instance, scan, or VIP with the **relocate** command. By running this command, you move the service to a different node.

The **srvctl relocate database** command will relocate an Oracle RAC One Node database to a different node. Thee **srvctl relocate instance** command relocates a database instance.

Removing the Configuration Information for Specific Targets

The **srvctl remove** command enables you to remove the associated resource from Oracle Clusterware. You can remove databases, instances, services, ONS, node apps, listeners, disk groups, services, and server pools, among other resources.

The **srvctl remove database** command, for example, removes a database configuration, as shown here:

```
$ srvctl remove database -db crm
```

The **srvctl remove instance** command will remove an instance's configuration:

```
$ srvctl remove instance -db crm -instance crm01
```

Predicting the Impact of Failures

You can use the **srvctl predict** command to see the potential impact of a resource failure. You can do this for a database, service, network, listener, disk group, scan, or

https://t.me/learningnets

VIP. Let's look at some examples of the **srvctl predict** command.

The **srvctl predict database** command shows what happens if a database fails:

```
$ srvctl predict database -db racdb
Database racdb will be stopped on nodes rac2,rac1
$
```

The **srvctl predict diskgroup** command shows the impact of an ASM disk group failure:

```
$ srvctl predict diskgroup -diskgroup data
Resource ora.DATA.dg will be stopped
$
```

Similarly, the **srvctl predict listener** command predicts the consequences of a listener failure:

```
$ srvctl predict listener -listener NODE3_CRMAPP_LISTENER
```

Managing Pluggable Databases in a RAC Environment

The Oracle Multitenant option allows a multitenant container database (CDB) to hold multiple pluggable databases (PDBs). Apart for a few minor differences, you administer Oracle RAC–based multitenant CDBs in the same way you manage non-CDBs. Some of the management tasks will apply to the entire CDB and some to the PDBs that are part of the CDB. And some tasks apply just to the root container.

When you use PDBs, the CDB is based on Oracle RAC, and you can make each of the PDBs available on all (or a subset) of the instances of the RAC-based CDB. To manage PDBs, you manage the Dynamic Database Services. Applications will use these services to connect to a specific PDB, just as they would connect to a regular single-instance Oracle database using Oracle Net Services for providing the connectivity.

You assign a database service to each PDB to help perform the start and stop operations on those databases and place the PDBs across instances in the RAC-based CDB.

The following example shows how you assign a service named myservice1 to a PDB named mypdb1. The PDB (mypdb1) is assumed to be policy-managed and belongs to the CDB name mycdb, and it is also in the server pool named mypool.

```
$ srvctl add service -db mycdb -pdb mypdb1 -service myservice1  -serverpool mypool
```

The database will manage the service myservice1 uniformly across all the nodes that belong to the server pool named mypool.

To open the PDB named mypdb1, in our example, you start the service to which the

https://t.me/learningnets

Technet24

PDB was assigned, which in this case will be the service named myservice1. Here's how you do this:

```
$ srvctl start service -db  mycdb -service myservice1
```

You can stop the service with the following command:

```
$ srvctl stop service -db  mycdb -service myservice1
```

Although you opened the PDB mypdb1 with the **start service** command, the **stop service** command doesn't close the PDB. To close the service, you must issue the following command:

```
$ alter pluggable database mypdb1 close immediate;
```

The command **srvctl status service** lets you check the status of a PDB.

Starting PDBs is automated in a RAC setup, as is the case with a non-CDB database. Once you restart Oracle Clusterware on a server, it automatically restarts the service hosting a PDB, provided that the service was in the ONLINE state when Oracle Clusterware was shut down.

Administering Undo in an Oracle RAC Database

Oracle stores the original values of the data called before an image in undo segments. The data stored in an undo segment is used to provide read consistency and to roll back uncommitted transactions. Oracle's Flashback feature also makes use of undo data.

Under automatic undo management, an Oracle instance uses a tablespace of type "undo" to store undo/rollback data. The instance creates the required number of undo segments in the dedicated tablespace and allocates them to transactions to store the undo data; these operations are totally transparent to the DBA and the end user.

The DBCA automatically configures the database to use automatic undo management. The following initialization parameters enable automatic undo management:

```
undo_management = "auto"
undo_tablespace = undo_tbs1
```

Another parameter related to automatic undo management is UNDO_RETENTION_TIME. Its default value is 900, which means that an Oracle instance will make the best effort not to overwrite (reuse) an undo block for 900 seconds after the transaction has been committed. This ensures that transactions/queries that are running for 15 minutes are not likely to get ORA-1555 (the "snapshot too old" error). If the instance is executing longer-running

https://t.me/learningnets

transactions/queries, the DBA needs to size the undo tablespace accordingly. You can use the average undo generation rate to calculate the required size of the undo tablespace.

In the Oracle RAC environment, each instance stores transaction undo data in its dedicated undo tablespace. The DBA can set the undo tablespace for each instance by setting the UNDO_TABLESPACE parameter. The following lines will set UNDO_TABLESPACE for instance prod1 to undo_tbs1 and for instance prod2 to undo_tbs2:

```
prod1.undo_tablespace= undo_tbs1
prod2.undo_tablespace= undo_tbs2
```

All the instances in an Oracle RAC environment use automatic undo management, and the parameter UNDO_MANAGEMENT needs to be the same across all the instances. However, note that the undo tablespace can't be shared among the instances, so the parameter UNDO_TABLESPACE must be unique for each instance.

Either of the following methods can be used to increase the size of an undo tablespace:

- Add another datafile to the undo tablespace.
- Increase the size of the existing datafile(s) belonging to the undo tablespace.

You can change the undo tablespace for an instance. Create the new undo tablespace as follows:

```
create undo tablespace undotbs_big
datafile '/ocfs2/prod/undotbsbig.dbf' size 2000MB;
```

Here's how to instruct the instance to use the newly created undo tablespace:

```
alter system set undo_tablespace=undotbs_big scope=both;
```

The instance will start using the new undo tablespace for newer transactions; however, transactions that were active before the new undo tablespace was assigned will continue to use the older undo tablespace until completed. The original undo tablespace can be dropped or taken offline when all the active transactions using it are committed and undo retention time has expired.

Administering a Temporary Tablespace

You can create traditional temporary tablespaces as well as the new (Oracle RAC Database 12.2) local temporary tablespaces in an Oracle RAC database. We discuss the management of traditional tablespaces (global temporary tablespaces) first, followed by a discussion of how to manage the new local temporary tablespaces.

https://t.me/learningnets

Technet24

Managing Traditional (Global) Tablespaces

Oracle uses a temporary tablespace as a “scratch pad” area to perform sort operations that cannot be done in memory. Oracle also uses the temporary tablespace to create TEMP tables.

Any user for whom you don’t assign a temporary tablespace explicitly will use the default temporary tablespace. DBCA automatically creates a default temporary tablespace. When you’re creating a database with a locally managed system tablespace, a default temporary tablespace must be specified.

You can also define a *temporary tablespace group* that can be used wherever a temporary tablespace is used. The use of a temporary tablespace group in an Oracle RAC environment is similar to that of a single-instance environment.

We recommend the use of a temporary tablespace group in an Oracle RAC environment, because different temporary tablespaces are assigned to the sessions when more than one user connects to the database using the same username. This allows different sessions to use different tablespaces for sort activities—a useful feature when you’re managing packaged applications such as Oracle applications.

In an Oracle RAC environment, a user will always use the same assigned temporary tablespace irrespective of the instance being used. Each instance creates a temporary segment in the temporary tablespace it is using. If an instance is running a big sort operation requiring a large temporary tablespace, it can reclaim the space used by other instances’ temporary segments in that tablespace. The default temporary tablespace cannot be dropped or taken offline; however, you can change the default temporary tablespace. Follow these steps to do this:

1. Create a new temporary tablespace:
```
create temporary tablespace temp2
tempfile '/ocfs2/prod/temp2.dbf' size 2000MB
autoextend on next 1M maxsize unlimited
extent management local uniform size 1M;
```
2. Make a new temporary tablespace as the default temporary tablespace:
```
alter database default temporary tablespace temp2;
```
3. Drop or offline the original default temporary tablespace.

The following v$ views contain information about temporary tablespaces:

- **GV$SORT_SEGMENT** Use this view to explore current and maximum sort segment usage statistics.
- **GV$TEMPSEG_USAGE** Use this view to explore temporary segment usage details such as username, SQL, and so on.
- **V$TEMPFILE** Use this view to identify temporary datafiles being used for a temporary tablespace.

https://t.me/learningnets

NOTE

You can use the GV$SORT_SEGMENT and GV$TEMPSEG_USAGE views to determine the temporary tablespace used by each instance. Use the column INST_ID to separate the data per instance. In the case of a multitenant database, use the column CON_ID in these two views.

In an Oracle RAC environment, all instances share the same temporary tablespace. The size of the tablespace should be at least equal to the concurrent maximum requirement of all the instances.

If an instance needs a larger sort space, it will ask other instances to release space. In turn, when other instances need more sort space, they will ask this instance to release the sort space. Frequent requests to release temporary space by other instances may impact performance. Pay attention to the columns FREED_EXTENTS and FREE_REQUESTS of the V$SORT_SEGMENT view; if the value of these columns is growing on a regular basis, consider increasing the size of the temporary tablespace.

Managing Local Temporary Tablespaces

You can create local temporary tablespaces in the Oracle Database 12*c* Release 2 (12.2) database. Oracle added this capability to improve I/O operations, so computations that spill over from memory to disk are written to the local disk on the reader nodes. You manage local temporary tablespaces in the same way as you manage regular temporary tablespaces.

The goal of local temporary tablespaces is to improve the management of temporary tablespaces in a read-only instance by taking advantage of I/O on local storage and avoiding cross-instance management of temporary tablespaces. It also helps performance by eliminating on-disk space metadata management. However, it’s possible for certain SQL operations, such as sorts, hash joins, and star transformations, to spill to the global temporary tablespace on shared disks.

Creating Local Temporary Tablespaces

Unlike in the case of shared global temporary tablespaces, where there’s a single temp file for a tablespace, creating a local temporary tablespace will create local temp files on every instance.

You can specify two key options when creating a local temporary tablespace—FOR RIM and FOR ALL—as explained here. Both tablespaces store their temp files on local disks.

- The LOCAL FOR_RIM option creates a local temporary tablespace for read-only instances. Here’s how you create one of these tablespaces:

https://t.me/learningnets

Technet24

```
CREATE LOCAL TEMPORARY TABLESPACE FOR RIM local_temp_ts_for_rim TEMPFILE\
   '/u01/app/oracle/database/12.2.0.1/dbs/temp_file_rim'\
   EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M AUTOEXTEND ON;
```

- Specify the LOCAL FOR ALL option to create a local temporary tablespace for both read-write and read-only instances. Here's an example:

```
CREATE LOCAL TEMPORARY TABLESPACE FOR ALL temp_ts_for_all  TEMPFILE\
  '/u01/app/oracle/database/12.2.0.1/dbs/temp_file_all'\
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 1M AUTOEXTEND ON;
```

NOTE

You can specify a local temporary tablespace as the default temporary tablespace for a user. You can also configure a user with two default tablespaces: a local temp tablespace to connect to read-only instances that run on reader nodes, and a shared temp tablespace when users connect to read-write instances running on a hub node.

Managing Local Temporary Tablespaces

As mentioned earlier, managing a local temporary tablespace is like managing a global temporary tablespace. Following are some examples of how to manage a local temporary tablespace:

- Run the following command to take a local temporary tablespace offline:

```
ALTER DATABASE TEMPFILE '/temp/temp_file_rim' OFFLINE;
```

- Use the SHRINK SPACE option to decrease the size of a local temporary tablespace:

```
ALTER TABLESPACE temp_ts_for_rim SHRINK SPACE KEEP 20M
```

- Use the AUTOEXTEND option to modify the auto-extension attributes of a local temporary file:

```
ALTER TABLESPACE temp_ts_for_rim AUTOEXTEND ON NEXT 20G
```

- Specify the RESIZE option to resize a local temporary file:

```
ALTER DATABASE TEMPFILE '/temp/temp_file_rim' RESIZE 1000M;
```

Regarding the data dictionary views and local temporary tablespaces, the DBA_TABLESPACES and USER_TABLESPACES views have a new column named SHARED to indicate if a temporary file is local or shared. Similarly, the DBA_TEMP_FILES and DBA_TEMP_FREE_SPACE views have new columns called SHARED and INST_ID.

Hierarchy of Temporary Tablespaces

When you create both a shared (global) temporary tablespace and a local temporary

https://t.me/learningnets

tablespace, the database uses them in a specific hierarchy.

In a read-only instance, following is the hierarchy for selecting which temporary tablespace to use for storing spills:

1. A user's local temporary tablespace
2. The default local temporary tablespace
3. A user's temporary tablespace
4. The database default temporary tablespace

For a read-write instance, the database allocates temp space in the following order of priority:

1. A user's shared temporary tablespace
2. A user's local temporary tablespace
3. The database default shared temporary tablespace
4. The database default local temporary tablespace

Administering Online Redo Logs

Oracle uses online redo log files to log any changes to data blocks. Each instance has its own set of online redo log files. A set of redo logs used in a circular manner by an instance is known as a *thread*. A thread contains at least two online redo logs. The size of an online redo log is independent of other instances' redo log sizes and is determined by the local instance's workload and backup and recovery considerations.

Each instance has exclusive write access to its own online redo log files. An instance can read another instance's current online redo log file to perform instance recovery if that instance has terminated abnormally. Thus, an online redo log needs to be located on a shared storage device and cannot be on a local disk.

Operations such as add, delete, and mirror performed on redo log files are similar to those operations performed on a single-instance environment. Use the dynamic views V$LOG and V$LOGFILE to explore which log files are allocated to each thread, their sizes, names, and other characteristics.

Enabling Archive Logs in the Oracle RAC Environment

As mentioned, online redo log files are reused by Oracle in a circular manner. To facilitate media recovery, Oracle allows you to make a copy of the online redo log files before they are reused. This process is called *archiving*.

DBCA allows you to enable archiving at the time of database creation. If the database has been created in "no archive log" mode, use the following process to change to archive log mode:

https://t.me/learningnets

Technet24

1. Set CLUSTER_DATABASE to **false** for the instance:
```
alter system set cluster_database=false scope=spfile sid= 'prod1';
```
2. Shut down all the instances accessing the database:
```
srvctl stop database -db prod
```
3. Mount the database using the local instance:
```
SQL> startup mount
```
4. Enable archiving:
```
SQL> alter database archivelog;
```
5. Set the parameter CLUSTER_DATABASE to **true** for the instance prod1:
```
alter system set cluster_database=true scope=spfile sid='prod1'
```
6. Shut down the local instance:
```
SQL> shutdown;
```
7. Bring up all the instances:
```
srvctl start database -db prod
```

Once in archive log mode, each instance can archive redo logs automatically.

Enabling the Flashback Area

Oracle's flashback area allows you to roll back the database to a time in the recent past. A flashback log is different from an archive log and is kept in a separate location specified by the flashback log location.

The procedure for enabling the flashback is similar to the procedure for enabling the archiving log mode—it requires the database to be mounted in the exclusive mode. If you attempt to enable flashback while mounted/open in shared mode, Oracle will signal an error.

The following steps enable the database's flashback mode. This can be done from any node.

1. Verify that the database is running in archive log mode (if not already enabled, use the preceding procedure to enable archive log mode):
```
SQL> archive log list
Database log mode    Archive Mode
Automatic archival   Enabled
Archive destination  /u01/app/oracle/11g/dbs/arch
Oldest online log sequence 59
Next log sequence to archive    60
Current log sequence 60
```
2. Set the CLUSTER_DATABASE parameter to **false** for the instance to perform this operation:
```
alter system set cluster_database=false scope=spfile sid= 'prod1';
```
3. If not already done, set the parameters DB_RECOVERY_FILE_DEST_SIZE and DB_RECOVERY_FILE_DEST. The DB_RECOVERY_FILE_DEST

https://t.me/learningnets

parameter should point to a shareable disk subsystem because it needs to be accessible to all the instances. The value of the DB_RECOVERY_FILE_DEST_SIZE parameter should be the same for all the instances.

```
alter system set DB_RECOVERY_FILE_DEST_SIZE=200M scope=SPFILE;
alter system set DB_RECOVERY_FILE_DEST='/ocfs2/flashback' scope=SPFILE;
```

4. Shut down all instances accessing the database:
```
srvctl stop database -db prod
```
5. Mount the database using the local instance:
```
SQL> startup mount
```
6. Enable the flashback by issuing the following command:
```
SQL> alter database flashback on;
```
7. Set the parameter CLUSTER_DATABASE back to **true** for the instance prod1:
```
alter system set cluster_database=true scope=spfile sid='prod1'
```
8. Shut down the instance:
```
SQL> shutdown;
```
9. Start all the instances:
```
$srvctl start database -db prod
```

NOTE

You can optionally turn on or off the flashback logging for a particular tablespace as shown here:

SQL> alter tablespace user_data flashback on;

SQL> alter tablespace user_data flashback off;

Managing Database Configuration with SRVCTL

SRVCTL, which stands for *server control utility*, is a command-line utility. You can divide SRVCTL's functions into two major groups:

- Database configuration tasks
- Database instance control tasks

Oracle stores database configuration information in a repository. The file itself should be located on a shared storage device so that it can be accessed from all the nodes. The repository information is stored in the Oracle Cluster Registry (OCR) that is created while CRS is being installed and must be located on shared storage. In

https://t.me/learningnets

Technet24

addition, another repository, known as the Oracle Local Repository (OLR), stores a local copy of OCR data on each node in the cluster.

Type **srvctl** without any command-line options to display the usage options:

```
$ srvctl
Usage: srvctl <command> <object> [<options>]
    commands: enable|disable|export|import|start|stop|relocate|status|add|remove|modify|getenv|
setenv|unsetenv|config|convert|update|upgrade|downgrade|predict
    objects: database|instance|service|nodeapps|vip|network|asm|diskgroup|listener|srvpool|
server|scan|scan_listener|oc4j|home|filesystem|gns|cvu|havip|exportfs|rhpserver|
rhpclient|mgmtdb|mgmtlsnr|volume|mountfs
For detailed help on each command and object and its options use:
  srvctl <command> -help [-compatible] or
  srvctl <command> <object> -help [-compatible]
[oracle@rac2 dbs]$[oracle@rac2 dbs]$ srvctl
Usage: srvctl <command> <object> [<options>]
    commands: enable|disable|export|import|start|stop|relocate|status|add|remove|modify|getenv|
setenv|unsetenv|config|convert|update|upgrade|downgrade|predict
    objects: database|instance|service|nodeapps|vip|network|asm|diskgroup|listener|srvpool|
server|scan|scan_listener|oc4j|home|filesystem|gns|cvu|havip|exportfs|rhpserver|
rhpclient|mgmtdb|mgmtlsnr|volume|mountfs
For detailed help on each command and object and its options use:
  srvctl <command> -help [-compatible] or
  srvctl <command> <object> -help [-compatible]
$
```

As indicated in this output, you can use the **-help** option of the **srvctl** command to display detailed help. Let's have a look at some of these commands. To display the databases registered in the repository, you can run the following command:

```
$srvctl config database
prod
test
```

The output of this command shows all configured databases that are part of Oracle Clusterware in your system. The **srvctl config database** command can also display the configuration of an Oracle RAC database. Here's how to display the configuration details of the database named ORCL:

```
$ [oracle@rac1 ~]$ srvctl config database -db orcl
Database unique name: orcl
Database name: orcl
```

https://t.me/learningnets

```
Oracle home: /u01/app/oracle/product/12.1.0/dbhome_1
Oracle user: oracle
Spfile: +DATA/ORCL/PARAMETERFILE/spfile.289.854664583
Password file: +DATA/ORCL/PASSWORD/pwdorcl.276.854663243
Domain: localdomain
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools:
Disk Groups: DATA
Mount point paths:
Services:
Type: RAC
Start concurrency:
Stop concurrency:
OSDBA group: dba
OSOPER group: dba
Database instances: orcl1,orcl2
Configured nodes: rac1,rac2
Database is administrator managed
$
```

To change the policy of database prod from automatic to manual, use the following command:

```
$ srvctl modify database -db PROD -policy MANUAL
```

Here's the command to change the startup option of the database prod from open to mount:

```
$ srvctl modify database -db PROD -s mount
```

You can also use the **srvctl modify database** command to convert an administrator-managed database to a policy-managed database. As with some of the other SRVCTL commands we discussed earlier in this chapter, you can add the **-eval** option with the **srvctl modify database** command to find out the potential impact of the command, without actually running it.

To display the status of the database prod, use the following command:

```
$ srvctl status database -db prod
Instance prod1 is running on node node_a
Instance prod2 is running on node node_b
Instance prod3 is running on node node_c
```

You can also check the status of an instance by running the following command:

https://t.me/learningnets

Technet24

```
$ srvctl status instance -db prod -instance "prod1, prod2" verbose
```

The optional parameter **verbose** displays more output.

As an aside, you can also use SQL*Plus to check the status of the instances, as shown here:

```
SQL> CONNECT SYS/as SYSRAC
Enter password: password

SQL> SELECT * FROM V$ACTIVE_INSTANCES;
INST_NUMBER INST_NAME
-----------  ----------------
1            db1-sun:db1
2            db2-sun:db2
3            db3-sun:db3
```

Here's how to display the status of services running in the database prod:

```
$ srvctl status service -db prod
Service prod_batch is running on instance(s) PROD1
Service prod_oltp is running on instance(s) PROD1
```

And here's how to relocate the PROD_BATCH service of the database prod from instance PROD1 to PROD2:

```
$ srvctl relocate service -db PROD -service prod_batch -oldinst prod1 -newinst prod2
```

You can determine the status of the listener with the following command:

```
$ srvctl status listener -node prod1
```

This command will show you if the listener is running on the node PROD1. To find the status of the listener on all nodes in the cluster, issue the following command:

```
$ srvctl  status listener
Listener LISTENER is enabled
Listener LISTENER is running on node(s): prod1,prod2
$
```

Finally, use the following command to check the node apps running on a node:

https://t.me/learningnets

```
$ srvctl status nodeapps
VIP rac1-vip.localdomain is enabled
VIP rac1-vip.localdomain is running on node: rac1
VIP rac2-vip.localdomain is enabled
VIP rac2-vip.localdomain is running on node: rac2
Network is enabled
Network is running on node: rac1
Network is running on node: rac2
ONS is enabled
ONS daemon is running on node: rac1
ONS daemon is running on node: rac2

$
```

You can also check the status of the node apps running on a specific node by issuing the previous command and adding the **-node*node_name*** options, as shown here:

```
$ srvctl status  nodeapps -node rac1
VIP rac1-vip.localdomain is enabled
VIP rac1-vip.localdomain is running on node: rac1
Network is enabled
Network is running on node: rac1
ONS is enabled
ONS daemon is running on node: rac1
$
```

Although mostly transparent to the end user and the DBA, CRS (introduced in Oracle 10*g*) has changed the mechanism SRVCTL uses to interface with the database. In Oracle 10*g* onward, OCR is used to store all the information, including the database configuration information used by the SRVCTL utility. SRVCTL uses CRS to communicate and perform startup and shutdown functions on other nodes.

SRVCTL uses the Global Services Daemon (GSD) to perform instance startup and shutdown operations. GSD should be started before SRVCTL can be used. SRVCTL stores database information in a repository known as the Server Control Repository, which must reside on shared storage because it is accessed from all the nodes.

Killing Sessions on Specific Instances of a Cluster

You use the familiar **alter system kill session** statement to terminate sessions in a specific instance of an Oracle RAC database. However, you must specify the ID of an instance, along with the SID and SERIAL# of the session.

Follow these steps to terminate a session running in a specific instance of an Oracle

https://t.me/learningnets

Technet24

RAC database:

1. Find out the value of the INST_ID column from the GV$SESSION dynamic performance view. The ISNT_ID column identifies the session you want to kill.

```
SQL> SELECT SID, SERIAL#, INST_ID FROM GV$SESSION WHERE USERNAME='SAM';
       SID    SERIAL#    INST_ID
---------- ---------- ----------
       120          8          2
```

2. Issue the **alter system kill session** command along with the SID, serial number, and instance ID you've obtained in step 1:

```
SQL> ALTER SYSTEM KILL SESSION '80, 6, @2';
System altered.
SQL>
```

Note that you prefix the @ character to the INST_ID attribute.

If the session is running on the same instance where you're executing the **alter system kill session** command, you can omit the INST_ID attribute.

The previous command kills the session after waiting for all outstanding activity to complete. If instead you want to immediately terminate the session, you can do so by adding the IMMEDIATE parameter to the **alter system kill session** statement:

```
SQL> ALTER SYSTEM KILL SESSION '80, 8, @2' IMMEDIATE;
System altered.
SQL>
```

Managing Database Objects

Managing database objects in an Oracle RAC environment is similar to managing them in a single-instance environment. In this section, we point out particular items to which you should pay special attention in the Oracle RAC environment.

Managing Sequences

It is good practice to increase the cache values for the sequences that are used by multiple concurrent sessions in multiple instances. This will eliminate the possibility of contention on sequences. The default value for a cache sequence is 20. However, depending on the frequency of use, cache values as high as 2000 could be acceptable.

The disadvantage of using a very high cache value is that the unused sequence numbers are lost in the event of an instance's crash or when a shared pool is flushed. In a RAC environment, the use of NOORDER (versus ORDER) in a sequence is critical. Application logic may prevent the use of NOORDER, but by default, the best strategy would be to use NOORDER.

Managing Tables

The Oracle RAC environment does not require specific treatment for tables. Best

https://t.me/learningnets

practices, such as partitioning very large tables that are followed in a single-instance environment, are sufficient for Oracle RAC as well.

Managing Indexes

Avoiding contention on B-tree index blocks is a huge challenge in designing highly scalable systems. A good application design—whether it is for a single-instance environment or for Oracle RAC—should use all available techniques for avoiding index block contention. Using partitioned indexes and reverse key indexes are the two most commonly used techniques available to avoid index block contention. The use of global indexes that need to be kept valid during partition maintenance operations is somewhat better in Oracle Database 12*c*, with the ability to maintain them after partition maintenance activity completes. However, local indexes are still preferable wherever possible.

Scope of SQL Commands

Although administration of an Oracle RAC database is similar to a single-instance database, you must be cautious about the scope of the commands being executed. Some commands affect the specific instance, whereas others may affect the whole database. For example, the RECOVER command affects the whole database, whereas the STARTUP and SHUTDOWN commands affect only the connected instance.

On the same note, some ALTER SYSTEM commands only affect the connected instance; for example, ALTER SYSTEM SWITCH LOGFILE will switch the log file on the connected or local instance, whereas ALTER SYSTEM SWITCH LOG CURRENT will switch the current log file on all instances.

Database Connections

Oracle 11*g* introduced Single Client Access Name (SCAN) to further simplify the database connection in an Oracle RAC database because all databases in the same cluster can now connect to the Oracle RAC database by using the SCAN in the connection string instead of listing all virtual IPs of the cluster nodes running the cluster database instance. In Oracle Database 12*c*, you can specify a list of nodes and subnets from which the SCAN listener will accept registration. You can do this through SRVCTL, which will store the information in the SCAN listener resource profile.

Oracle 12*c* provides different ways of connecting with an Oracle RAC database. Following are examples of connection strings for the database prod.

Here’s how to connect to the database prod using a conventional TNS alias:

https://t.me/learningnets

Technet24

```
prod =
(DESCRIPTION =
(ADDRESS = (PROTOCOL = TCP)(HOST = prod-scan)(PORT = 1701))
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = PROD_OLTP)
)
)
```

In the preceding connection string, users can connect to the database prod using the SCAN prod-scan on port 1701, listening and providing service as PROD_OLTP.

Here's a sample connection JDBC string for database prod:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=prod-scan)(PORT=1701))(CONNECT_DATA=(SERVICE_NAME=PROD_OLTP)))
```

EZConnect is another easy way to connect a database without looking for service names in the conventional tnsnames.ora file. EZConnect does not use any naming or directory system. A sample EZConnect connection string for the database prod is shown here:

```
$ sqlplus username/password@prod-scan:1701/PROD_OLTP
```

Administering Server Pools

You can create Oracle RAC databases in one of the following two deployment models:

- An *administrator-managed deployment* is where you configure each database instance to run on a specific node. You configure database services to run on specific instances of a database by specifying the *preferred* and *available* designations.
- A *policy-managed deployment* uses server pools. Database services run in a server pool as *singleton* or *uniform* across all servers in a server pool. Databases are deployed in one or more server pools.

A policy-managed database runs in one or more database server pools that you create in the cluster, and it can run on different servers at different times. Every server pool must have a minimum of one database service. A database instance runs on a server that's in the server pool you define for that database.

A server pool is nothing but a logical grouping of servers, and its main purpose is to provide enhanced scalability and availability for the applications it supports. While Oracle Clusterware is responsible for the availability of the server pool (or pools) you create, you can enhance availability by configuring the server pools, as we explain later.

Server pools host both database and non-database applications. You use the

https://t.me/learningnets

SRVCTL utility to create and manage server pools supporting Oracle RAC databases. You use the CRSCTL utility to create and manage server pools supporting non-database applications.

Because servers are assigned to server pools by number and not by name, you must configure all the servers to run any database. If you can't do so, you can restrict servers through server category definitions that determine server pool membership eligibility.

Server pools are highly configurable through the setting of various server pool attributes when you create or modify a server pool. When you use CRSCTL to configure a server pool for non-database applications, you can configure all server pool attributes. When you configure a server pool for servers hosting an Oracle RAC database, on the other hand, you can configure only a subset of the attributes.

Configuring Server Pools

When configuring a server pool for servers hosting an Oracle database, you can set the following server pool attributes:

- **-category** The category of servers to use for the server pool. By default, this parameter is empty and you can define a custom category. In an Oracle Flex ASM cluster, the possible values are **hub** and **leaf**.
- **-importance** The importance of the server pool, which can range from 1 to 1000, with 0 being the default. The **importance** attribute lets you rank services and makes it unnecessary to run a service on all nodes to ensure its availability. The **importance** server pool attribute is used at the time you start a cluster, when nodes fail, or when nodes are evicted. This attribute determines which databases are started first and which remain online when there's a multinode outage of servers.
- **-min_size** The minimum size of the server pool. Default is 0.
- **-max_size** The maximum size of the server pool. Default is -1, meaning it's unlimited. If the value of the **-max_size** parameter is greater than that of the **-min_size** parameter, then the remaining servers are deployed into the server pool with the higher importance value.
- **-serverpool** The name of the server pool. SRVCTL will automatically add "ora." in front of the server name you specify.
- **-servers** A comma-delimited list of candidate node names. Note that all nodes in this list won't necessarily be in a server pool. The value of the **-category** parameter determines which servers are assigned to a server pool.

In addition to these parameters, you can optionally add the **-eval** (to see the potential impact of executing the SRVCTL command), **-force**, and **-verbose** parameters to the SRVCTL command when creating a server pool. The following example shows what might happen if you relocate a server:

https://t.me/learningnets

Technet24

```
$ srvctl relocate server -servers "rac1" -eval -serverpool pool2
Database db1
     will stop on node rod1
     will start on node prod5
     Service prodSrv1
          will stop on node rac1, it will not run on any node
     Service prodServ2
          will stop on node prod1
          will start on node prod4
Server rac1
     will be moved from pool prodPoolX to pool pool2
$
```

Consolidation of Databases

A policy-managed database makes it easy to consolidate your databases. Because server pools are dynamically scalable, you can increase or decrease database instances to suit your business requirements.

Policy-managed databases let you run multiple databases using the same set of servers. This is possible because the databases share the same server pool through *instance caging*, which is the ability to limit the number of CPUs an instance can use simultaneously for foreground processes. This enables you to dynamically increase or decrease databases by using the server pool size and the CPU_COUNT initialization parameter to meet your workload requirements.

Policy-managed databases remove the need to reserve capacity on each server to meet workload requirements when a database instance or server fails.

You rank server pools based on the importance of the workloads they run, through setting the MIN_SIZE, MAX_SIZE, and IMPORTANCE server pool attributes.

Creating a Server Pool

The **srvctl add srvpool** command enables you to add a server pool, as shown in the following example:

```
srvctl add srvpool -serverpool SP1 -importance 1 -min 3 -max 6
```

This example creates a server pool named SP1, whose importance is set to 1, and the minimum and maximum number of nodes in the server pool are set to 3 and 6, respectively.

Displaying a Server Pool's Configuration

The **srvctl config srvpool** command shows the name, the minimum and maximum sizes, the importance, and a list of server names for a specific server pool in an Oracle RAC cluster. Here's an example:

https://t.me/learningnets

```
srvctl config srvpool -serverpool dbpool
```

Modifying a Server Pool

Execute the **srvctl modify database** command to modify the configuration of a server pool in a cluster. If you increase the minimum and maximum sizes, or the **importance** attributes of the server pool, the CRS daemon may try to reassign servers to the server pool to satisfy the new sizes you configured for this server pool.

The following example shows how to change the **importance** attribute to 0, the minimum size to 2, and the maximum size to 6 for a server pool named srvpool1:

```
$ srvctl modify srvpool -serverpool srvpool1 -importance 0 -min 2 -max 6
```

It's a good idea to run the **srvctl modify srvpool** command in the evaluate mode because the command involves many changes that could, in a consolidated setup, affect many databases, services, and server pools. The following example evaluates the impact of modifying a server pool:

```
$ srvctl modify srvpool -l 3 -g online -eval
Service erp1 will be stopped on node test3
Service reports will be stopped on node test3
Service inventory will be stopped on node test3
Service shipping will be stopped on node test3
Database dbsales will be started on node test3
Service orderentry will be started on node test3
Service billing will be started on node test3
Service browse will be started on node test3
Service search will be started on node test3
Service salescart will be started on node test3
Server test3 will be moved from pool backoffice to pool online
```

Relocating a Server to a Server Pool

You can relocate servers to a server pool in the cluster by running the **srvctl relocate server** command, as shown here:

```
$ srvctl relocate server -servers "server1, server2" -serverpool sp3
```

This command relocates server1 and server2 from their present location to server pool sp3.

Removing a Server Pool

You can remove a server pool with the **srvctl remove srvpool** statement, as shown in the following example, which removes the server pool named srvpool1:

```
$ srvctl remove srvpool -serverpool srvpool1
```

https://t.me/learningnets

Technet24

When you run this command, the server pool servpool1 is removed, and the database assigns its servers to other servers, provided the minimum/maximum sizes and the **importance** attributes permit it. Alternatively, the database could return the servers to the database's Free Server Pool.

Checking the Status of a Server Pool

You can check the status of a server pool and view the server pool names, number of servers in that server pool, and the names of these servers with the **srvctl status srvpool** command:

```
$ srvctl status srvpool -serverpool srvpool1 -detail
```

The **detail** attribute is optional and shows the names of the servers in this server pool.

Converting an Administrator-Managed Database to a Policy-Managed Database

It's straightforward to migrate from an administrator-managed database to a policy-managed database. Following are the steps you need to perform to move to a policy-managed database:

1. Create a server pool for the database:

   ```
   $ srvctl add srvpool -serverpool mysrvpool1_pool -min 2 -max 6
   ```

   In this example, the minimum number of servers in the server pool is 2 and the maximum is 6.
2. Stop the database:

   ```
   srvctl stop database -db prod
   ```
3. Run the **srvctl modify database** command so that the database is in the new server pool:

   ```
   srvctl modify database -db prod -serverpool mysrvpool1
   ```
4. Run the following commands to make sure that your changes were correct:

   ```
   srvctl config database -db db_unique_name
   srvctl config service -db db_unique_name
   ```

Summary

Administering the Oracle RAC database is similar to managing a single-instance Oracle database. Because more than one instance is accessing the same set of database objects, a little caution is required while you are managing the Oracle RAC database.

Managing the initialization parameter file (SPFILE) requires some special handling in Oracle RAC databases because the parameters have a database-wide effect. A few database parameters should be set the same across Oracle RAC instances, and

https://t.me/learningnets

instance-specific parameters can be set to different values for Oracle RAC instances. Using a common SPFILE is highly recommended with SID-prefixed instance-specific parameters. This is the best practice in managing the parameter file.

The server control utility (SRVCTL) helps manage the Oracle RAC database efficiently, and you can use this as a central management tool to manage the Oracle RAC instances. An Oracle RAC instance depends on Oracle Cluster Ready Services, and we'll discuss the administration of Oracle Clusterware in the next chapter.

https://t.me/learningnets

Technet24

CHAPTER 8

Managing Oracle Clusterware

In Chapters 6 and 7, we discussed the management of both Oracle ASM and Oracle RAC databases. This chapter delves deep into the management of Oracle Clusterware, the foundation of Oracle RAC. Specifically, we discuss the following areas in detail:

- Configuring and administering Oracle Clusterware
- Role-separated management
- Weight-based server node eviction
- Administering SCAN and GNS
- Using the CLUVFY utility for managing Oracle RAC
- The Oracle Clusterware startup process
- Using CRSCTL to manage the clusterware
- Managing Oracle Clusterware with CRSCTL and other utilities such as OCRCONFIG, OLSNODES, CLSCFG, and IFCFG
- Administering the Oracle Cluster Registry (OCR)
- Administering the Oracle Local Registry (OLR)
- Administering the voting disks

https://t.me/learningnets

In addition to these topics, Oracle RAC DBAs must also be proficient in a critical set of tasks related to workload management. This area is so important that Chapter 13 is solely devoted to RAC-related workload management.

Configuring and Administering Oracle Clusterware

Administering Oracle Clusterware involves the management of not only Oracle databases but also the applications that run in a cluster as well as the networking within the cluster.

You can configure and administer clusters in two different ways:

- **The administrator-managed cluster** This is the traditional method, where you manually allocate resources and manage the workloads. You determine which instances run on which nodes as well as on which nodes(s) the instances restart when one of them fails. Thus, you determine which nodes manage the workload of the cluster.
- **The policy-managed cluster** This method enables you to automate the administration of a cluster by configuring workloads to run in server pools. You configure *policy sets* that dictate how the workloads are managed in each server pool. You focus on managing the server pools and policy sets, and you don't have to worry about the nodes on which the instances run or the workload on the instances. The policies you configure take care of both instance location and workload placement.

If you want, you can automate things further by adopting Oracle Quality of Service (QoS) Management.

Benefits of Server Pools

Policy-based management with server pools provides the following benefits:

- Enables online server reallocation based on policies to satisfy workload requirements
- Guarantees resource allocation for critical workloads based on policy
- Ensures isolation so you can provide dedicated servers for critical applications and databases
- Configures policies to change server pools according to application demand and workloads, so the pools can provide the required capacity when needed

Server Pools and Policy-Based Management

Server pools are logical grouping of servers. The resources that Oracle Clusterware

https://t.me/learningnets

Technet24

manages, such as database instances, database services, application VIPs, and application components, are contained within the server pools.

NOTE
A server pool is a logical grouping of servers that hosts applications, databases, or both.

Server pools enable you to run specific types of workloads on the cluster's nodes, and they provide an easy way to administer the workloads. You can use a *cluster configuration policy set* to dynamically manage the cluster policies. This policy set is a document that defines the names of all the server pools in the cluster, along with one or more configuration policies.

You can create multiple cluster configuration policies, but only one policy can be in effect at any one time. You can, however, choose to set different policies to be in effect at various times of the day.

Server Pools and Categorization

Server categorization is a way to group servers into categories based on a set of server attributes, such as processor type and memory. You can configure server pools to restrict which server categories the server pools will accept. You create a server category by providing attributes for the parameter SERVER_CATEGORY.

Server categorization enables you to create clusters made up of heterogeneous nodes.

How Server Pools Work

Server pools logically divide a cluster, and a server can be in any one server pool at a given point in time. Server pools host applications (such as database services) or non-database applications. Server pools are logical groups of servers that host both uniform and singleton applications. Here's the difference between the two:

- A *uniform* application is one whose workload is distributed over all servers in a pool.
- A *singleton* application runs on a single server in the server pool.

In Chapter 7, we showed you how to create server pools with the SRVCTL utility for hosting databases or database servers. Clusterware administrators use the CRSCTL utility to create and manage all other server pools for non-database use, such as those that host an application. Only a cluster administrator can create top-level server pools that logically divide a cluster.

https://t.me/learningnets

Types of Server Pools

You have two basic types of server pools to work with:

- Default server pools (that is, the Generic and Free server pools)
- User-created server pools

The following sections briefly describe these two types of server pools.

Default Server Pools

When you install Oracle Clusterware, two default pools, called the Generic server pool and the Free server pool, are created internally. All servers are initially assigned to the Free server pool, but they can be assigned to a new server pool when you create one, and assign servers to that server pool. The Free server pool simply contains the servers you haven't assigned to any server pool. You can't edit the SERVER_NAMES, MIN_SIZE, and MAX_SIZE properties of the Free server pool.

The Generic server pool is for servers that host non-policy-managed applications, such as administrator-managed databases. You can't modify the configuration attributes of this pool because all the attributes are read-only.

User-Created Server Pools

User-created server pools are named server pools that you create and assign databases and other applications to. Any servers that you don't place into a (user-created) server pool go into the Free server pool.

Creating Server Pools

You can use either SRVCTL or CRSCTL to create server pools for databases and other applications. In Chapter 7, we showed how to create server pools with the SRVCTL utility. However, SRVCTL lets you configure only a subset of the server pool attributes, including the following:

- -category
- -importance
- -min
- -max
- -serverpool
- -servers

Also, SRVCTL prefixes "ora." to the name of the server pool.

By default, each server pool has the following attributes:

- **MIN_SIZE** The minimum number of servers in a server pool. If the number

https://t.me/learningnets

Technet24

of servers in a server pool falls below this value, Oracle Clusterware moves servers into the server pool.

- **MAX_SIZE** The maximum number of servers in this server pool.
- **IMPORTANCE** A number (from 0 to 1000, with 0 being least important) that ranks this server pool among all other server pools in this cluster.

In addition to these three mandatory attributes, you can add other attributes according to your cluster configuration policy. You typically add attributes such as SERVER_CATEGORY and EXCLUSIVE _POOLS to enhance cluster performance.

As the root user or the Oracle Clusterware installation owner, run the **crsctl add serverpool** command to create a server pool. The following example shows how to create a server pool named “testsp” with a maximum of five servers in the pool:

```
# crsctl add serverpool testsp -attr "MAX_SIZE=5"
```

You can also use a file to specify a bunch of attribute values. In the following example, we create the file sp1_attr to list a set of attribute values for a server pool named “sp1”:

```
IMPORTANCE=1
MIN_SIZE=1
MAX_SIZE=2
SERVER_NAMES=node3 node4 node5
PARENT_POOLS=Generic
EXCLUSIVE_POOLS=testsp
ACL=owner:oracle:rwx,pgrp:oinstall:rwx,other::r--
```

Following this, we can create the sp1 server pool by pointing to the sp1_atr file for the attribute values:

```
$ crsctl add serverpool sp1 -file /tmp/sp1_attr
```

Evaluating the Addition of a Server Pool

You can review the potential impact of adding a server pool, before making any changes to your cluster, by running the **crsctl eval add serverpool** command:

```
[oracle@rac1 bin]$ crsctl eval add serverpool sp1 -explain
Stage Group 1:
--------------------------------------------------------------------------------
Stage  Required  Action
--------------------------------------------------------------------------------
    1      E     Starting to look for server pools that will donate servers to be
          reallocated to the new pool 'sp1'.
--------------------------------------------------------------------------------
[oracle@rac1 bin]$
```

https://t.me/learningnets

Deleting a Server Pool

You delete a server pool with the **crsctl delete serverpool** command:

```
$# crsctl delete serverpool sp1
```

Role-Separated Management

By default, the user who installed Oracle Clusterware and the root user are permanent cluster administrators. In addition, primary group privileges (such as privileges on the oinstall group) enable DBAs to create databases.

Role-separated management is a more recent approach to managing cluster resources and workloads. Under role-separated management, you create OS users and groups to administer resources such as databases, based on roles that you assign the users and groups. You can add permissions on resources and server pools according to access control lists (ACLs). Oracle ASM also allows for role separation to manage storage.

By default, role-separated management isn't in force, and you must explicitly set it up. This is especially useful in a consolidated environment where multiple users and groups are competing for a cluster's resources, and where you also need to enforce isolation between the groups.

Managing Cluster Administrators

In the case of a policy-managed database, you enable role separation before creating the server pools. The database stores the access permissions for each server pool in the ACL attribute. Use access control lists (ACLs) to define the administrators for a cluster. Only cluster administrators can create server pools. To have the permission to create a server pool (or other cluster resources), the OS user or OS group of which a user is a member must have the read, write, and execute permissions set in the ACL attribute.

Configuring Role Separation

In order to configure role separation, you must first determine the following:

- The roles you need
- The resources and server pools each role will administer
- The access privileges necessary for each role

Following this, you can create (or modify) user accounts for group privileges, such as the oinstall and grid groups, using ACLs.

You first create the cluster resources or server pools as the root or grid user. Next, you enable specific OS users to administer the resources or server pools by granting

https://t.me/learningnets

Technet24

them the correct permissions. For example, let's say you create two OS users and assign them to the oinstall group. You can then create two server pools and assign each of the users as the administrator for one of the two server pools you've created. That'll be all the users who are allowed to administer the two server pools because all the other users are excluded from administrative privileges on the server pools.

Using the crsctl setperm Command

You can create ACLs to assign to cluster resources or server pools to enforce role separation. Use the **crsctl setperm** command to create the ACLs. Here's the syntax for the command:

```
crsctl setperm {resource | type | serverpool} name {-u acl_string |
-x acl_string | -o user_name | -g group_name}
```

The two key things you need to know are the flag options and the ACL strings. The flag options are as follows:

- **-u** Updates the ACL
- **-x** Deletes the ACL
- **-o** Changes the entity owner
- **-g** Changes the primary group of the entity

And the following are the ACL strings:

```
{user:user_name[:readPermwritePermexecPerm] |
group:group_name[:readPermwritePermexecPerm] |
other[::readPermwritePermexecPerm] }
```

In this syntax statement, *user* is the user ACL, *group* is the group ACL, and *other* is the ACL denoting the access granted to users and groups not explicitly granted any access permissions. The *readPerm*, *writePerm*, and *execPerm* attributes denote the location of the read, write, and execute permissions, respectively.

The following example shows how you can set ACLs on a resource called MyProgram (an application administered by user Sam) for the group "crsadmin":

```
# crsctl setperm resource MyProgram -u user:Sam:rwx,group:crsadmin:r-x,other:---:r--
```

In this example, the privileges are as follows:

- The administrative user (Sam) has read, write, and execute privileges.
- Members of the crsadmin group have read and execute privileges.
- Other users have only read access, which means that those users can check the status and configuration but will be unable to modify any configuration settings.

https://t.me/learningnets

Weight-Based Server Node Eviction

You can configure database instances and nodes with *weighting functions* so specific nodes can continue running when Oracle Clusterware needs to terminate or evict cluster nodes due to a cluster interconnect failure. This is a way for you to keep Oracle Clusterware from evicting nodes hosting critical, singleton resources.

NOTE

You can assign weighting to specific nodes, resources, or services, but only in an administrator-managed cluster. You cannot do this in a policy-managed cluster.

The following is how you assign weighting to database instances, services, servers, and non-ora.* resources:

- For database instances and services, specify the **-css_crtical yes** parameter when running the **srvctl add database/service** commands. You can do the same when running the **srvctl modify database/service** commands as well.
- To assign weighting to servers, specify the **-css_critical yes** parameter when running the **crsctl set server** command.
- To assign weighting to non-ora.* resources, specify the **-attr "CSS_CRITICAL=yes"** parameter with the **crsctl add resource** and the **crsctl modify resource** commands.

You must restart the Oracle Clusterware stack on the node once you make any changes. Some resources don't require a restart of the stack.

Oracle Database Quality of Service Management

Oracle Database Quality of Service (QoS) Management is an automated policy-based mechanism that helps you monitor and manage shared resources across applications. QoS Management automatically adjusts the system configuration to keep applications running at the required performance levels and avoid fluctuations in application performance.

QoS Management recommends actions to restore performance when it identifies bottlenecks. For administrator-managed databases, QoS Management adjusts the CPU shares of performance classes running in the database, as well as adjusts CPU counts for databases running on the same servers.

For policy-managed databases, QoS Management involves expanding and shrinking the instances by changing the server pool sizes.

https://t.me/learningnets

Technet24

Administering SCAN

We explained SCAN (Single Client Access Name) in Chapter 3. SCAN is simply a domain name registered for one to three IP addresses, using either DNS (Domain Naming Service) or GNS (Grid Naming Service). The GNS handles requests sent by external DNS servers and performs name resolution for the hostnames you define in a RAC configuration. Using GNS and the Dynamic Host Configuration Protocol (DHCP), Oracle Clusterware can configure the VIP addresses for the SCAN you provide during the configuration of the cluster.

In this scenario, the DHCP server provides the node VIP and the three SCAN VIPs. Oracle Clusterware dynamically gets the VIP address of the new server joining the cluster and makes that server accessible through the GNS.

As an alternative to GNS, you can manually configure addresses. In this scenario, you configure the following set of addresses:

- A single public address and hostname for each node
- A VIP address for each node
- A maximum of three SCAN addresses for the cluster

SCAN simplifies the administration and management of client network connection files via changes in cluster database configuration. You can manage many SCAN-related operations with the SRVCTL utility, as explained in this section.

Starting and Stopping SCAN

You can stop a running SCAN VIP with the following command:

```
$ srvctl stop scan
```

This command will stop all running SCAN VIPs. If you'd like to stop a specific SCAN VIP, you may do so with the following command:

```
$ srvctl stop scan -scannumber
```

In this command, you assign a value between 1 and 3 for the **scannumber** parameter to denote the specific SCAN VIP you want to stop.

Displaying the SCAN Status

To display the status of SCAN VIPs, use the following command:

https://t.me/learningnets

```
[root@alpha2 bin]# $GRID_HOME/bin/srvctl status scan
SCAN VIP scan1 is enabled
SCAN VIP scan1 is running on node alpha3
SCAN VIP scan2 is enabled
SCAN VIP scan2 is running on node alpha2
SCAN VIP scan3 is enabled
SCAN VIP scan3 is running on node alpha1
[root@alpha2 bin]#
```

This command shows the status of all SCAN VIPs. To check the status of a specific SCAN VIP, specify the **scannumber** parameter with the **srvctl status scan** command. The value for the **scannumber** parameter must be between 1 and 3.

To display the status of SCAN listeners:

```
[root@alpha2 bin]# $GRID_HOME/bin/srvctl status scan_listener
SCAN Listener LISTENER_SCAN1 is enabled
SCAN listener LISTENER_SCAN1 is running on node alpha3
SCAN Listener LISTENER_SCAN2 is enabled
SCAN listener LISTENER_SCAN2 is running on node alpha2
SCAN Listener LISTENER_SCAN3 is enabled
SCAN listener LISTENER_SCAN3 is running on node alpha1
[root@alpha2 bin]#
```

Again, you can include the **scannumber** parameter with this command to check the status of a specific SCAN listener.

Here's how to display the existing configuration of SCAN:

```
[oracle@rac1 grid]$ $GRID_HOME/bin/srvctl config scan
SCAN name: rac-scan.localdomain, Network: 1
Subnet IPv4: 192.168.56.0/255.255.255.0/eth0, static
Subnet IPv6:
SCAN 0 IPv4 VIP: 192.168.56.91
SCAN VIP is enabled.
SCAN VIP is individually enabled on nodes:
SCAN VIP is individually disabled on nodes:
SCAN 1 IPv4 VIP: 192.168.56.92
SCAN VIP is enabled.
SCAN VIP is individually enabled on nodes:
SCAN VIP is individually disabled on nodes:
SCAN 2 IPv4 VIP: 192.168.56.93
SCAN VIP is enabled.
SCAN VIP is individually enabled on nodes:
SCAN VIP is individually disabled on nodes:
[oracle@rac1 grid]$
```

To check the status of a specific SCAN VIP out of the three SCAN VIPs, you must

https://t.me/learningnets

Technet24

include the **-scannumber** parameter, with a value between 1 and 3, as shown here:

```
$ srvctl config scan -scannumber 1
```

The **add scan** command adds Oracle Clusterware resources for the SCAN you want. It creates the same number of SCAN VIP resources as the number of IP addresses the SCAN resolves to.

Here's how to add a SCAN:

```
[root@alpha3 bin]# $GRID_HOME/bin/srvctl add scan -scanname prod-scan
```

Here's how to remove a SCAN:

```
[root@alpha3 bin]# $GRID_HOME/bin/srvctl remove scan
```

The **remove scan** command removes all Oracle Clusterware resources from all the SCAN VIPs. Users can use the **-force** option with the preceding command to remove the SCAN forcefully, without first stopping it.

Here's how to modify a SCAN:

```
[root@alpha3 bin]# $GRID_HOME/bin/srvctl modify scan -scanname new_scan
```

You modify a scan when the DNS is modified to add/change/remove IP addresses. This command will match the SCAN VIPs to the number of IP addresses returned by the *scan_name* specified in DNS. For example, your current SCAN may resolve to a single IP address in DNS. You can modify the SCAN to resolve to three IP addresses and then run the **srvctl modify scan** command to create the additional SCAN VIP resources.

Here's how to add a SCAN listener:

```
[root@alpha3 bin]# $GRID_HOME/bin/srvctl add scan_listener
```

This command will add a listener with a default port, but users can use the **-endpoints** option to specify a nondefault port for the listener.

Here's how to remove a SCAN listener:

```
[root@alpha3 bin]# $GRID_HOME/bin/srvctl remove scan_listener
```

This command removes Oracle Clusterware resources from the SCAN listener. Users can use the **-force** option to remove the SCAN listener forcefully.

Finally, here's how to modify a SCAN listener port:

```
[root@alpha3 bin]# $GRID_HOME/bin/srvctl modify scan_listener
                    -endpoints <port_number>
```

Users can use the **-u** option to reflect changes to the current SCAN listener only. This command helps you match the SCAN listener to the SCAN VIPs or to modify the

https://t.me/learningnets

SCAN listener endpoints.

Administering the Grid Naming Service (GNS)

Use SRVCTL to start and stop GNS on the server cluster, as shown here (run the commands as the root user):

```
# srvctl start gns
# srvctl stop gns
```

Using the CLUVFY Utility for Managing Oracle RAC

In earlier chapters, you've learned to use the Cluster Verification Utility (CLUVFY) to perform various pre- and post-installation checks of both Oracle Clusterware and the Oracle RAC database. Oracle Universal Installer (OUI) runs the **cluvfy** command internally to check the prerequisites during the installation of the Oracle software.

You can also use CLUVFY to manage several aspects of a running Oracle RAC system. In this section, we show you how to use CLUVFY to perform critical integrity checks of the clusterware and databases, as well as components such as the amount of free space available in a RAC system. You can check the integrity of an Oracle ASM Cluster File System (OCFS) with the following command:

```
$ [oracle@rac1 grid]$ cluvfy comp acfs
Verifying ACFS Integrity
Task ASM Integrity check started...
Starting check to see if ASM is running on all cluster nodes...
ASM Running check passed. ASM is running on all specified nodes
Disk Group Check passed. At least one Disk Group configured
Task ASM Integrity check passed...
Task ACFS Integrity check started...
Task ACFS Integrity check passed
UDev attributes check for ACFS started...
UDev attributes check passed for ACFS
Verification of ACFS Integrity was successful.
[oracle@rac1 grid]$
```

This command verifies the integrity of the Oracle ACFS on all the cluster nodes. Note that the parameter **comp** in this and the other **cluvfy** commands refers to a cluster component (or subcomponent), which in this case happens to be OCFS.

Use the following command to check the integrity of Oracle ASM on all the node's clusters:

https://t.me/learningnets

Technet24

```
$ [oracle@rac2 ~]$ cluvfy comp asm
Verifying ASM Integrity
Task ASM Integrity check started...
Starting check to see if ASM is running on all cluster nodes...
ASM Running check passed. ASM is running on all specified nodes
Disk Group Check passed. At least one Disk Group configured
Task ASM Integrity check passed...
Verification of ASM Integrity was successful.
[oracle@rac2 ~]$
```

As you can see, this command checks the status of ASM on all the cluster's nodes. To check the status of one or more nodes, specify the nodes with the **-n** parameter (default is **all**).

You can create a baseline of the system and cluster configuration with the **cluvfy comp baseline** command, which allows you to collect and compare baselines. You can keep a baseline following an upgrade of Oracle Clusterware, or even as part of a standard DBA task to maintain a standard reference of configuration information. If you capture multiple baselines at different points in time, you can compare them as well. To create a baseline of your cluster configuration, you can use the following key parameters with the **cluvfy comp baseline** command:

- **-collect** Collects a baseline.
- **-compare** Specify this parameter to compare two baselines.
- **-bestpractice** Collects a baseline for only best-practice recommendations.
- **-mandatory** Collects a baseline for only mandatory requirements. The **-bestpractice** and **-mandatory** parameters are mutually exclusive. By omitting both these options, you will be specifying that you want to collect both best-practice recommendations and mandatory requirements.
- **<baseline1, baseline2>** You can specify one or more baselines to compare.
- **-deviations** You can specify that you want only the deviations from best-practice recommendations and/or mandatory requirements (depends on the specification of the **-bestpractice** and **-mandatory** options) to be reported.
- **<all|cluster|database>** You can collect baselines related to Clusterware or databases or both.
- **<node_list>** A comma-separated list of non-domain-qualified node names on which the test should be conducted. If **all** is specified, then all the nodes in the cluster will be used for verification.
- **<report_name>** Baseline report name.
- **<save_dir>** The reports of validations can be saved in a location of your choice.

The following examples show how to specify the **-collect**, **-compare**, and **-bestpractice** parameters:

https://t.me/learningnets

```
$ cluvfy comp baseline -collect all -n all -db prod -bestpractice -report bp_1
$ cluvfy comp baseline -compare bp_1, bp2
```

Clusterware Startup

It’s important to understand the behind-the-scenes operations of Oracle Clusterware. In this section, we discuss the Oracle Clusterware startup process and the steps it goes through during this process.

The Clusterware Startup Process

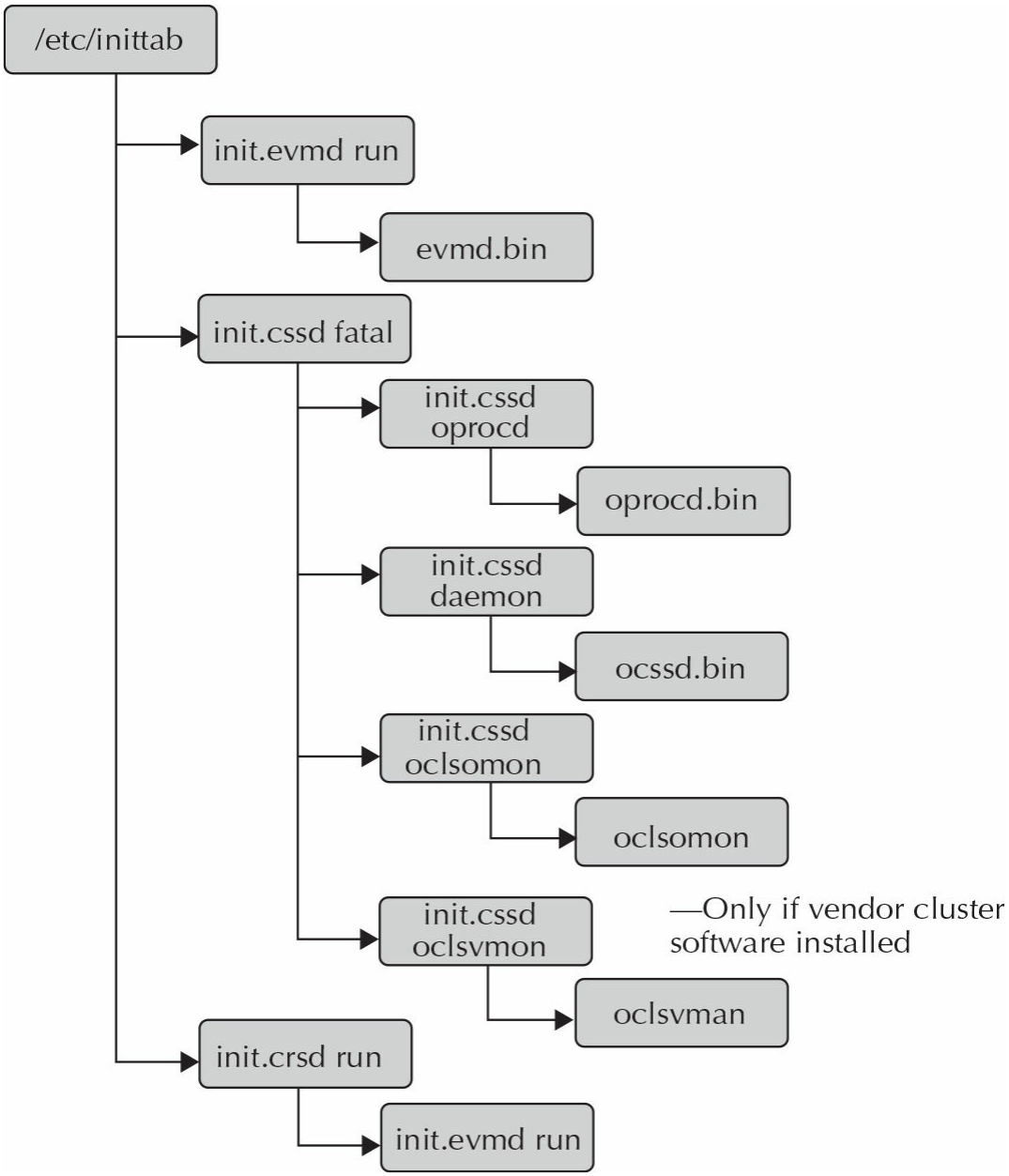
During the installation of Oracle Clusterware (Oracle Grid Infrastructure), the OUI installs three wrapper scripts—init.crsd, init.cssd, and init.evmd—on each cluster node. These wrapper scripts set the required environment variables and start the daemons, depending on the user action and auto-startup configuration of Oracle Clusterware. Oracle places one entry each for these wrapper scripts in the /etc/inittab file on UNIX and Linux systems, and the operating system init daemon starts these wrapper scripts:

```
h1:35:respawn:/etc/init.d/init.evmd run >/dev/null 2>&1 </dev/null
h2:35:respawn:/etc/init.d/init.cssd fatal >/dev/null 2>&1 </dev/null
h3:35:respawn:/etc/init.d/init.crsd run >/dev/null 2>&1 </dev/null
```

These wrapper scripts are very important for Oracle Clusterware startup and must always be running on the cluster node. Therefore, these wrapper scripts are configured with the **respawn** action so they will be restarted whenever they die. The init.cssd wrapper script is passed with **fatal** arguments, causing a cluster node to reboot on failure. The following diagram shows the startup flow from the inittab file.

https://t.me/learningnets

Technet24



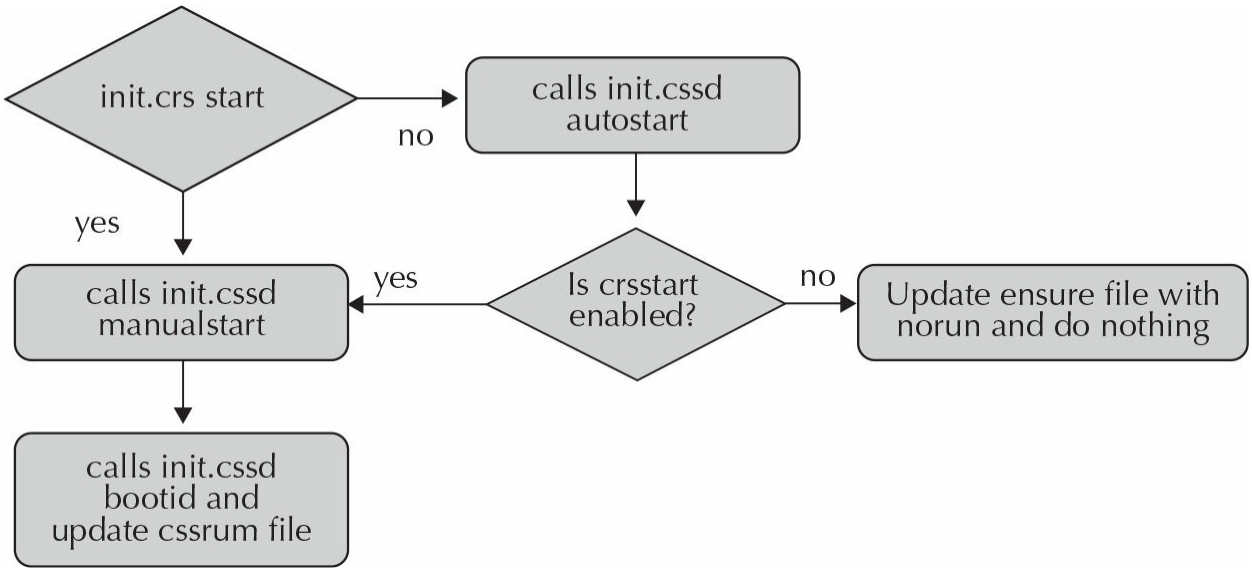
You can configure Oracle Clusterware to start automatically when a cluster node starts, or you can start it up yourself. Oracle Clusterware uses cluster control files to maintain the state and runtime information of different Oracle Clusterware daemons. Oracle stores these cluster control files under the <scls_dir>/root directory on each cluster node. The <scls_dir> directory is OS dependent.

Clusterware Starting Sequence

As just discussed, the wrapper scripts run continuously and are responsible for starting the Oracle Clusterware daemons. Therefore, it is very important to understand the flow of these wrapper scripts. Oracle calls the init.cssd wrapper script with **startcheck** and **runcheck** in all wrapper scripts. During **startcheck**, Oracle checks the run status of the wrapper scripts, the presence of vendor clusters, and the dependencies of file systems. On the other hand, the check of the run status **(runcheck)** is intended to check the run status for any of the run or fatal scripts (evm/crs/css) as initialized by the automatic startup or manual startup routine. This check is supposed to be very fast and silent because it will be invoked regularly while the system is up. The following

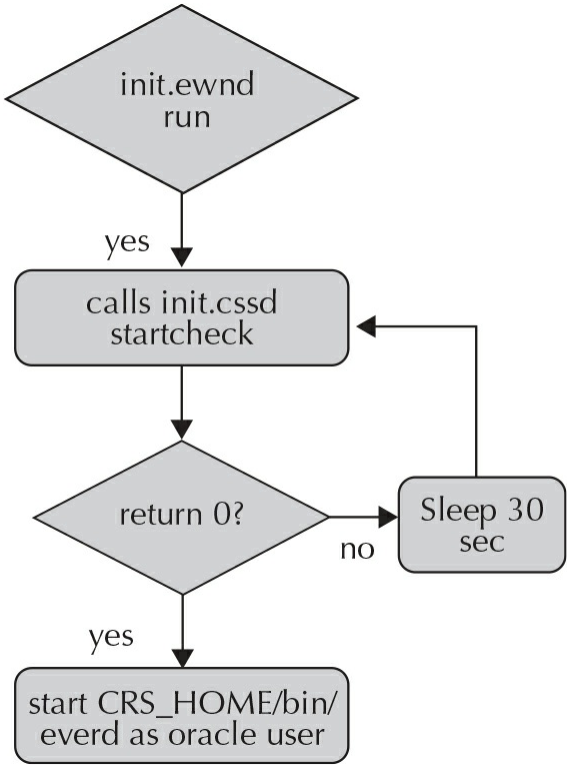
https://t.me/learningnets

diagram illustrates the startup sequence of the clusterware.



EVMd Process Startup

The operating system's init daemon executes the init.evmd wrapper script with the **run** parameter, which internally calls the init.cssd wrapper script with the **startcheck** parameter. Upon success of the init.cssd **startcheck**, Oracle starts the EVMd daemon. Oracle disables the respawning of the init.evmd wrapper script if the boot timestamp or the hostname of the cluster node isn't the same before and after starting the EVMd daemon. The following diagram illustrates the startup sequence of the EVMd process.



CRSd Process Startup

The operating system's init daemon executes the init.crsd wrapper script with the **run** parameter, which internally calls the init.cssd wrapper script with the **startcheck**

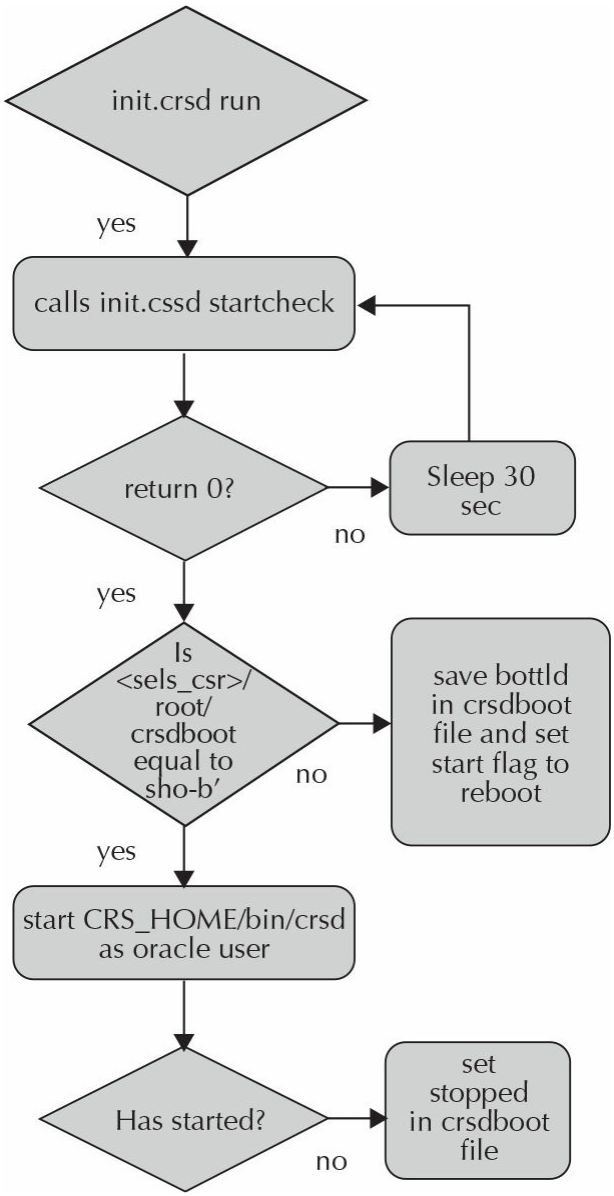
https://t.me/learningnets

Technet24

parameter. Upon success of the init.cssd **startcheck**, Oracle compares the actual boot time of the cluster node with the boot timestamp stored in the crsdboot cluster controlfile.

If the boot timestamps are not the same, Oracle stores the current boot timestamp in the crsdboot cluster controlfile and starts the CRSd daemon with the **reboot** start flag; otherwise, the CRSd daemon is started with the **restart** start flag. Oracle sets "stopped" in the crsdboot cluster controlfile if the CRSd daemon does not start successfully.

The following diagram illustrates the startup sequence and the associated validations during the CRSd startup.



CSSd Process Startup

The operating system's init daemon executes the init.cssd wrapper script with the **fatal** parameter, which internally calls the init.cssd wrapper script with the **startcheck** parameter. This script is responsible for starting four different daemons: CSSd, OPROCd, OCLSOMON, and OCLSVMON.

Upon success of the init.cssd **startcheck**, Oracle creates three cluster controlfiles—

https://t.me/learningnets

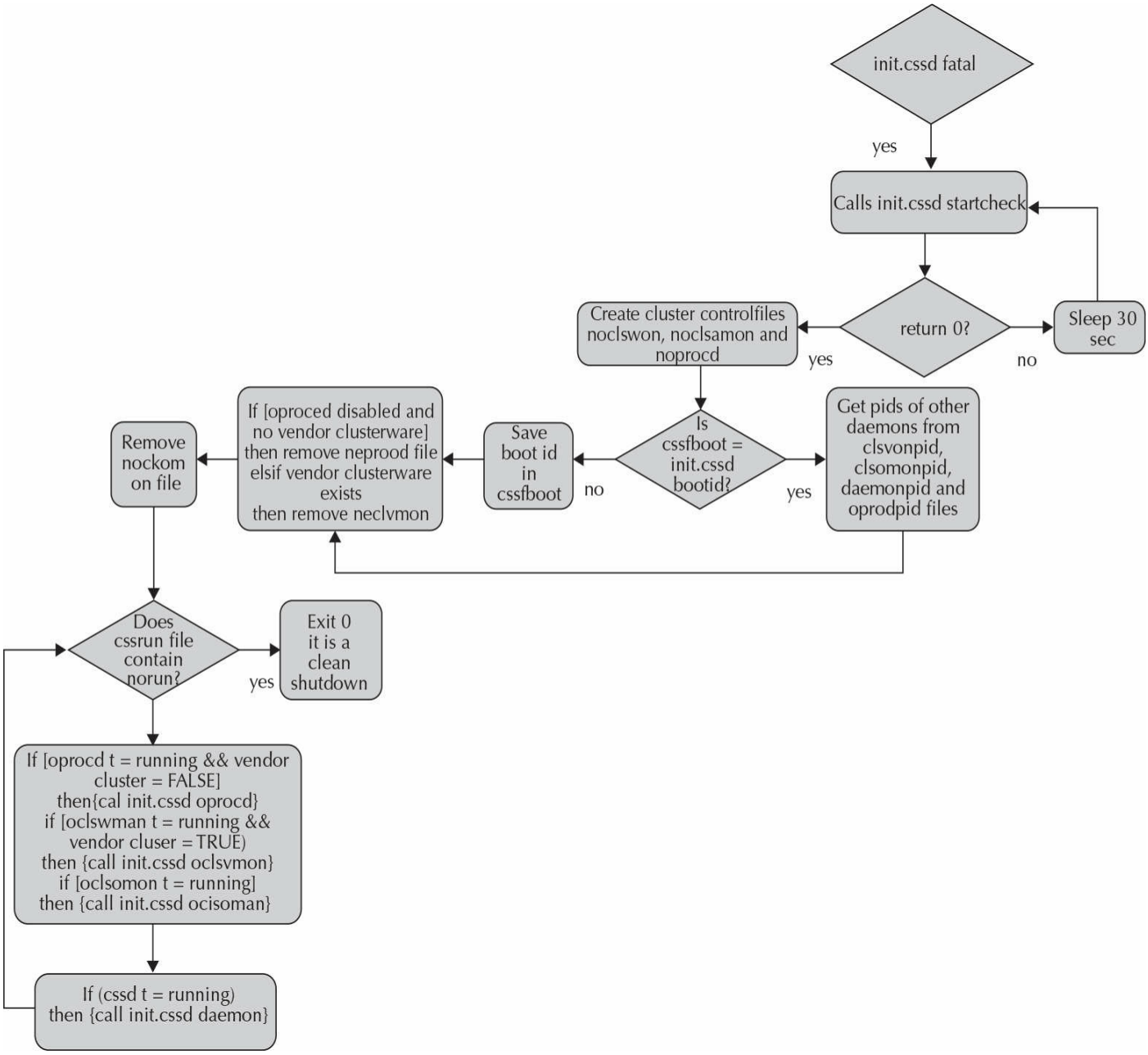
noclsvmon, noclsomon, and noprocd—under the <scls_dir>/root directory to disable the OCLSVMON, OCLSOMON, and OPROCd daemons, respectively. Later, Oracle removes the noclsomon file because the OCLSOMON daemon runs regardless of whether or not the vendor cluster is used.

During the clusterware startup, the startup process also compares the actual boot time of the cluster node with the boot timestamp stored in the cssfboot cluster controlfile. If the boot timestamps are the same, it is highly possible that the other daemons are already running because the operating system's init daemon has respawned this wrapper script. Oracle extracts the PID of the running daemons from the clsvmonid, clsomonpid, daemonpid, and oprocpid files and stores them in memory (environment variables). If the boot timestamps are not the same, Oracle stores the new boot timestamp in the cssfboot cluster controlfile and removes the noprocd file if OPROCd is enabled and no vendor cluster is installed.

Also during the clusterware startup, Oracle checks the content of the cssrun cluster controlfile; if this file contains "norun," the init.cssd fatal script completes successfully without starting the daemons, assuming that this is a clean shutdown of the Oracle Clusterware. If the cssrun file does not contain "norun," Oracle starts the OPROCd daemon if it is enabled, starts OCLSOMON if it's not running, and finally starts the CSSd daemon by calling the init.cssd daemon wrapper script again. The following diagram illustrates the sequence of actions in CSSd.

https://t.me/learningnets

Technet24



Regardless of the method used to start Oracle Clusterware, Oracle uses the wrapper scripts described. Different Oracle Clusterware startup methods set the cluster controlfiles differently to queue the Oracle Clusterware startup and wrapper scripts, to read these cluster controlfiles, and to take the appropriate action to start different daemons.

Oracle Clusterware Auto-Startup

The operating system init process calls init.crs **start** upon operating system startup, which internally calls the init.cssd wrapper script with the **autostart** parameter. Oracle first checks the content of the crsstart cluster controlfile to determine whether auto-startup for Oracle Clusterware is enabled. If the crsstart file contains the “enable” string, Oracle will call the init.cssd wrapper script with the **manualstart** parameter, identify the full timestamp of the “/proc” file system, and set this information in the cssrun file and queue the startup for Oracle Clusterware.

https://t.me/learningnets

Oracle Clusterware Manual Startup

Oracle calls the init.cssd wrapper script with the **manualstart** parameter upon manual startup of Oracle Clusterware. Oracle identifies the full timestamp of the "/proc" file system and sets this information in the cssrun file stored under the <scls_dir>/run directory.

The operating system's init daemon starts the clusterware processes by using the Oracle Clusterware wrapper scripts, which are installed by the OUI during the installation of Oracle Clusterware. Oracle installs three wrapper scripts—init.crsd, init.evmd, and init.cssd—that are configured with the **respawn** action so that they will be restarted whenever they fail (except the init.cssd wrapper script, which is configured with the **fatal** parameter, causing the cluster node to reboot to avoid any possible data corruption, as explained later in this chapter).

Using CRSCTL to Manage the Clusterware

CRSCTL is the CRS control utility you use to check, start, stop, get the status of, enable, and disable CRS components and resources controlled by CRS. You also use CRSCTL for managing OCR and voting disks, as well as debugging various CRS components.

In the following sections, we review how you can use the CRSCTL utility to perform critical management tasks such as stopping and stopping CRS, verifying the status of the clusterware, and more.

Starting and Stopping CRS

Oracle CRS is started automatically at system boot time. Manually starting and stopping CRS is generally not required, except in the following rare situations:

- While applying a patch set to Oracle Grid Infrastructure
- During OS/system maintenance
- While debugging CRS problems

In Oracle Database 12*c*, you can start CRS (Oracle High Availability Services) using the following command:

```
#$GRID_HOME/bin/crsctl start crs
[root@alpha2 bin]# ./crsctl start crs
CRS-4123: Oracle High Availability Services has been started.
[root@alpha2 bin]#
```

This command starts the Oracle High Availability Services only on the local node where you execute the command. You must run this command as root.

To stop CRS, you use the following command:

https://t.me/learningnets

Technet24

```
 # $GRID_HOME/bin/crsctl stop crs
[root@alpha2 bin]# ./crsctl stop crs
CRS-2791: Starting shutdown of Oracle High Availability Services-managed
 resources on 'alpha2'
<<OUTPUT TRIMMED>>

     CRS-2673: Attempting to stop 'ora.gipcd' on 'alpha2'
     CRS-2677: Stop of 'ora.diskmon' on 'alpha2' succeeded
     CRS-2677: Stop of 'ora.gipcd' on 'alpha2' succeeded
     CRS-2793: Shutdown of Oracle High Availability Services-managed resources on
     'alpha2' has completed
     CRS-4133: Oracle High Availability Services has been stopped.
     [root@alpha2 bin]#
```

This command stops the Oracle High Availability Services on the local server, and you must run this as the root user. You can optionally specify the **–f** parameter to unconditionally stop all resources and to stop the Oracle High Availability Services, if you find that of some of the clusterware-managed services are still running after you execute this command.

Clusterized (Cluster-Aware) CRSCTL Commands

You don’t have to log into each node to individually start or stop the cluster resources running on that node. You can run what are called *clusterized* commands on one node to perform Oracle Clusterware operations on one or all the nodes in a cluster. These operations are deemed *cluster-aware* operations and are also called *remote* operations. Remote operations depend on the OHASD (Oracle High Availability Services daemon). Following are the three clusterized commands available to you.

crsctl check cluster

The **crsctl check cluster** command enables you to check the status of the Oracle Clusterware stack from any of a cluster’s nodes:

https://t.me/learningnets

```
$ crsctl check cluster -all
**************************************************************
node1:
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
**************************************************************
node2:
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
**************************************************************
$
```

crsctl start cluster

You can run the **crsctl start cluster** command on any of the nodes in an Oracle RAC cluster to start up the Oracle Clusterware stack, as shown here:

```
# $GRID_HOME/bin/crsctl start cluster
```

You can run this from any node in a cluster. If you want to start the Oracle Clusterware stack on a specific node, you can issue the following command instead:

```
# $GRID_HOME/bin/crsctl start cluster -n prod1
```

Here, we specified the **-n** parameter to start the stack on just the single node, prod1.

You can also specify a list of servers by providing the server names separated by a space, as shown here:

```
# crsctl start cluster -n prod1 prod2
```

crsctl stop cluster

To stop the Oracle Clusterware stack on all servers in the cluster, issue the following command:

```
# $GRID_HOME/bin/crsctl stop cluster -all
```

Again, you can run this command from any node in your cluster. If you don't specify either **–all** or one or more services names with the **–n** parameter, only the Oracle Clusterware stack running on the local node is stopped. If you want to stop the Oracle Clusterware stack on a specific server, run the following command instead:

```
# $GRID_HOME/bin/crsctl stop cluster -n prod1
```

Verifying the Status of CRS

https://t.me/learningnets

Technet24

You can execute CRSCTL commands to verify the status of the cluster and related CRS daemon processes. If you want to check the status of the Oracle Clusterware stack on all nodes in your cluster, you can issue the following command:

```
# $GRID_HOME/bin/crsctl check cluster -all
```

Because this is a clusterized command, it shows the status of the Oracle Clusterware stack on all nodes in the cluster. To check the status of Oracle Clusterware on just the local node, issue the command **crsctl check cluster**. You can also check the status of the Oracle Clusterware stack on a specific node by specifying the **-n** parameter, as shown here:

```
# $GRID_HOME/bin/crsctl check cluster -n prod1
```

Next, here's how to check the current status of both the Oracle High Availability Services and the Oracle Clusterware stack on a local node:

```
[oracle@rac1 grid]$ $GRID_HOME/bin/crsctl check crs
CRS-4638: Oracle High Availability Services is online
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
[oracle@rac1 grid]$
```

You can check the current status of just the OHASD daemon in the following way:

```
[root@alpha2 bin]# $GRID_HOME/bin/crsctl check has
CRS-4638: Oracle High Availability Services is online
[root@alpha2 bin]#
```

You can display the cluster configuration information with the following command:

```
$ crsctl get cluster configuration
```

Disabling and Enabling CRS

By default, CRS will restart on system reboot. If system maintenance is being done, and CRS needs to be prevented from starting on system reboot, you can disable it. Also, the **enable** and **disable** CRS commands are effective only for future reboots of the node and do not impact the availability of the currently running CRS and its components. You can use the following commands to enable/disable CRS and its daemons.

To disable all CRS daemons:

```
#$GRID_HOME/bin/crsctl disable crs
```

This command prevents the Oracle High Availability Services from starting up

https://t.me/learningnets

automatically upon a server reboot, and it affects the local server only. You must use the command **crsctl start crs** consequently to start up the Oracle High Availability Services.

To enable all CRS daemons:

```
#$GRID_HOME/bin/crsctl enable crs
```

This command doesn't start up the Oracle High Availability Services. Rather, it ensures that the Oracle High Availability Services start up automatically following a server reboot.

NOTE

Oracle used to offer a number of CRS utilities for helping you manage CRS, such as crs_stat, which shows the status of resources controlled by CRS. These utilities have been deprecated in the Oracle 11g and 12c releases. Over time, Oracle has consolidated the functionality of these deprecated utilities into the CRSCTL utility. Some of the functionalities for managing resources (such as crs_profile and crs_register) are merged with the SRVCTL utility. You may still find all the deprecated CRS utilities under $GRID_HOME/bin. All CRS utilities are located in the $GRID_HOME/bin directory, and some of them are also available in the ORACLE_HOME directory. However, you should always execute the CRS utilities from the $GRID_HOME/bin directory.

You can find the version of Oracle Clusterware using the following command:

```
[oracle@rac1 grid]$ $GRID_HOME/bin/crsctl query  crs softwareversion
Oracle Clusterware version on node [rac1] is [12.1.0.2.0]
[oracle@rac1 grid]$
```

Similarly, users can query the active version and the release version of Oracle Clusterware by specifying the parameters **activeversion** and **releaseversion** with the **crsctl query crs** command.

The CRSCTL EVAL Commands

In the previous section, we discussed how to use the CRSCTL utility to perform various clusterware management tasks. Oracle Database 12*c* offers a brand-new set of Oracle Clusterware commands, known as the **eval** commands, that lets you simulate commands without actually running them. This offers you a great opportunity to view potential changes before running key clusterware commands.

For example, the **crsctl eval start resource** command shows you the effects of running the **crsctl start resource** command. Similarly, the **crsctl eval modify**

https://t.me/learningnets

Technet24

resource command shows you the expected effect of running the **crsctl modify resource** command. There are only about a dozen or so CRSCTL **eval** commands in this release. Here's a list of some of the CRSCTL **eval** commands you can use in Oracle Database 12*c:*

- **crsctl eval add serve** Simulates the addition of a server
- **crsctl eval modify serverpool** Predicts the impact of modifying a server pool
- **crsctl eval add serverpool** Predicts the effects of adding a specific server pool
- **crsctl eval relocate server** Predicts the effects of relocating a server
- **crsctl eval delete server** Predicts the impact of deleting a server

Note that you can also specify the **-eval** option with the SRVCTL commands to evaluate the potential impact of a command. This is especially useful in a production environment, where you want to ensure that you don't run any command without knowing its potential impact on the system. Throughout this book, we show how to specify the new **-eval** option with several CRSCTL and SRVCTL commands.

Using Other Utilities to Manage Oracle Clusterware

The previous section provided several examples of how you can use the CRSCTL utility to manage Oracle Clusterware. In addition to the main CRS utility, CRSCTL, there are other utilities you need to be familiar with. This section briefly describes those utilities.

Using the olsnodes Command

The **olsnodes** command provides information about all nodes that are part of a RAC cluster. You use this utility to check that all the nodes have joined the cluster. The **olsnodes** command syntax is shown here:

```
Usage: olsnodes [ [-n] [-i] [-s] [-t] [<node_name> | -l [-p]] | [-c] ] [-a] [-g] [-v]
-n print node number with the node name
-p print private interconnect address for the local node
-i print virtual IP address with the node name
<node> print information for the specified node
-l print information for the local node
-s print node status - active or inactive
-t print node type - pinned or unpinned
-g turn on logging
-v Run in debug mode; use at direction of Oracle Support only.
-c print clusterware name
-a displays only active nodes in the cluster with no duplicates
```

https://t.me/learningnets

You can check the status of Oracle CRS by executing this command without any command-line options. The output should be a listing of the nodes, confirming that CRS is up and running and that all the nodes can communicate with each other via the CRS:

```
[oracle@rac1 grid]$ $GRID_HOME/bin/olsnodes
rac1
rac2
[oracle@rac1 grid]$
```

You can also use the OLSNODES utility to print the name of the cluster, as shown next:

```
[oracle@rac1 grid]$ olsnodes -c
prod
[oracle@rac1 grid]$

The -i option will show information pertaining to each of the cluster nodes,
including the VIP address assigned to each node.
[oracle@rac1 grid]$ $GRID_HOME/bin/olsnodes -i
rac1  rac1-vip.localdomain
rac2  rac2-vip.localdomain
[oracle@rac1 grid]$
```

The **–n** command option shows the node numbers for each node along with their names. Finally, the **olsnodes –a** command will show you the role of each node, as demonstrated here:

```
$ [oracle@rac1 grid] $ $GRID_HOME/bin/olsnodes -s
rac1  Active
rac2  Active
[oracle@rac1 grid]$
```

You can list the node role of each node in a cluster with the **olsnodes -a** command:

```
[root@node1]# olsnodes -a
node1    Hub
node2    Hub
node3    Leaf
node4    Leaf
```

The GPnP Tool

The Oracle Grid Plug and Play profile management tool (GPNPTOOL), shown next, is used to manage the Grid Plug and Play profile:

https://t.me/learningnets

Technet24

```
[root@alpha1 bin]# ./gpnptool
Oracle GPnP Tool

Usage:
 "gpnptool <verb> <switches>", where verbs are:

     create   Create a new GPnP Profile
     edit     Edit existing GPnP Profile
     getpval  Get value(s) from GPnP Profile
     get      Get profile in effect on local node
     rget     Get profile in effect on remote GPnP node
     put      Put profile as a current best
     find     Find all RD-discoverable resources of given type
     lfind    Find local gpnpd server
     check    Perform basic profile sanity checks
     c14n     Canonicalize, format profile text (XML C14N)
     sign     Sign/re-sign profile with wallet's private key
     unsign   Remove profile signature, if any
     verify   Verify profile signature against wallet certificate
     help     Print detailed tool help
     ver      Show tool version
```

Here's how to display the current GPnP profile on the local node alpha1:

```
[root@alpha1 bin]# ./gpnptool get
```

The Cluster Health Monitor

The Cluster Health Monitor (CHM) is a new diagnostic tool in Oracle Database 12*c* that detects and analyzes OS- and cluster-level resource failures and symptoms. The data that CHM collects is stored in the new Oracle Grid Infrastructure Management Repository, for potential analysis by Oracle Support.

Three key services support the capabilities offered by the Cluster Health Monitor, and we discuss them briefly in the following sections.

The Cluster Logger Service

The Cluster Logger Service (OLOGGERD) manages data from up to 32 nodes in an Oracle RAC cluster. The OLOGGERD service automatically restarts on an alternative node if it goes down. The service receives the OS metrics from all the nodes in the cluster and saves them in the Oracle Grid Infrastructure Management Repository (MGMD) database, besides also managing the database.

The System Monitor Service

The System Monitor Service (OSYSMOND) exists on every node in the cluster and monitors the OS as well as collects the data to be sent to the Cluster Logger Service.

https://t.me/learningnets

The Oracle Grid Infrastructure Management Repository

The MGMTDB database is a brand-new Oracle Database 12*c* database that Oracle uses to store the real-time OS system metrics the CHM collects. The repository stores historical metric data, which makes it very easy to view past performance and diagnostic events.

The MGMTDB database is configured during the installation of or upgrade to Oracle Database 12*c* and is automatically installed for you. Oracle recommends that you run the MGMTDB database on the same node as the OLOGGERD service, for performance reasons. Note the following features of the MGMTDB database:

- The database runs only on a single node in a cluster. In case of a node failure, you must enable failover to a different node.
- The database files are stored along with the OCR and voting files in the same ASM disk group.
- Besides the CHM-related data, the Grid Infrastructure Management Repository also enables other features such as Oracle Database QoS Management and Rapid Home Provisioning.

The OCLUMON Tool

The OCLUMON tool is new to Oracle Database 12*c*, and you use it to manage the Cluster Health Monitor. You not only can use this tool to query the CHM database, but you can also employ it to perform other duties such as changing the size of the MGMTDB database. Here's an explanation of the OCLUMON commands:

- **oclumon debug** This command sets the debug level for CHM services.
- **oclumon dumpnodeview** This command allows you to view all metrics collected by CHM for a point in time. Here's an example that shows how to display node view from all nodes in the cluster:

https://t.me/learningnets

Technet24

```
oracle@rac1 grid]$ /u01/app/12.1.0/grid/bin/oclumon
query> dumpnodeview
dumpnodeview: Node name not given. Querying for the local host
----------------------------------------
Node: rac1 Clock: '17-06-29 15.26.14 ' SerialNo:1124
----------------------------------------
SYSTEM:
#pcpus: 1 #vcpus: 1 cpuht: N chipname: Intel(R) cpu: 26.19 cpuq: 0
swaptotal: 3096572 hugepagetotal: 0 hugepagefree: 0 hugepagesize: 2048 ior: 119
iow: 82 ios: 23 swpin: 0 swpout: 0 pgin: 119 pgout: 56 netr: 36.274 netw:
37.222
procs: 302 procsoncpu: 1 rtprocs: 12 rtprocsoncpu: N/A #fds: 32704
#sysfdlimit: 6815744 #disks: 5 #nics: 4 nicErrors: 0
TOP CONSUMERS:
topcpu: 'mdb_vktm_-mgmtd(4467) 2.79' topprivmem: 'java(1874) 117372' topshm:
'ora_mman_orcl1(6203) 246500' topfd: 'crsd.bin(3772) 216'
topthread: 'console-kit-dae(1953) 64'
…
[oracle@rac1 grid]
```

- **oclumon manage** This command enables you to change the Management Repository database retention time and size. The following example shows how to change the size of the repository:

```
$ oclumon manage -repos changereposize 6000
```

- **oclumon version** This command gets you the version of the CHM in use, as shown here:

```
$ oclumon version
Cluster Health Monitor (OS), Version 12.2.0.1.0 - Production Copyright 2007,
2016 Oracle. All rights reserved.
$
```

At this point, we should point out that Oracle's new diagnostic collection tool called Oracle Trace File Analyzer (TFA), which we discuss in detail later in this chapter, automatically collects all the data you can collect with the OCLUMON utility.

Oracle Interface Configuration: oifcfg

The Oracle interface configuration tool helps you to define network interface card usage in an Oracle RAC environment. This utility is useful for managing the hostname, IP address changes during datacenter migrations, and network reconfigurations that require changing IP addresses to cluster members. Here are some usage examples for this tool.

The **oifcfg getif** command should return values for **global public** and **global cluster_interconnect**. Here's an example:

```
[oracle@rac1 grid]$ $GRID_HOME/bin/oifcfg getif
eth0  192.168.56.0  global  public
eth1  192.168.10.0  global  cluster_interconnect
[oracle@rac1 grid]$
```

https://t.me/learningnets

If the command does not return a value for **global cluster_interconnect**, enter the following commands to delete and set the desired interface:

```
# oifcfg delif -global
# oifcfg setif -global <interface name>/<subnet>:public
# oifcfg setif -global <interface name>/<subnet>:cluster_interconnect
```

You can list the interface names and subnets of all available interfaces on the local node with the **iflist** keyword:

```
oifcfg iflist
eth0     172.19.141.0
eth1     172.21.65.0
```

Cluster Configuration Utility: clscfg

This utility is used during CRS installation. However, you shouldn't use it unless instructed to do so under guidance from Oracle Support Services (OSS). The command **clscfg -concepts** provides a good description of some CRS concepts, including private interconnect, hostnames, node names, voting disk, OCSSd, EVMd, CRSd, and OCR.

The following options are available with the Cluster Configuration utility:

```
[root@rac1 oracle]# /u01/app/12.2.0/grid/bin/clscfg
clscfg: EXISTING configuration version 5 detected.
clscfg: version 5 is 12c Release 1.
clscfg -- Oracle cluster configuration tool

  This tool is typically invoked as part of the Oracle Cluster Ready
  Services install process. It configures cluster topology and other
  settings. Use -help for information on any of these modes.
  Use one of the following modes of operation.
  -install    - creates a new configuration
  -add        - adds node specific configuration on the newly added node.
  -upgrade    - upgrades an existing configuration
  -localupgrade - upgrades an existing configuration in OLR
  -downgrade  - downgrades an existing configuration
  -nodedowngrade - downgrades the node specific configuration
  -local      - creates a special single-node configuration for ASM
  -localadd   - creates keys in OLR for HASD
  -patch      - patches an existing configuration
  -localpatch - patches an existing configuration in OLR
  -concepts   - brief listing of terminology used in the other modes
  -trace      - may be used in conjunction with any mode above for tracing
WARNING: Using this tool may corrupt your cluster configuration. Do not
         use unless you positively know what you are doing.
[root@rac1 oracle]#
```

https://t.me/learningnets

Technet24

The Cluster Name Check Utility: cemutlo

This utility prints the cluster name information. Following is the usage syntax for this utility:

```
#$ORA_CRS_HOME/bin/cemutlo [-n] [-w]
        where:
  -n prints the cluster name
  -w prints the clusterware version in the following format:
  <major_version>:<minor_version>:<vendor_info>
#$ORA_CRS_HOME/bin/cemutlo -n
prod
#$ORA_CRS_HOME/bin/cemutlo -w
2:1:
```

Note that you can also use the OLSNODES utility to print the name of the cluster.

Oracle Trace File Analyzer

The Oracle Trace File Analyzer (TFA) is a brand-new diagnostic tool introduced in the Oracle Database 12*c* release. It is used to simplify diagnostic data collection for Oracle Clusterware and Oracle RAC.

TFA can automate the collection of diagnostic data, and it also collects, packages, and centrally stores the data. Using a simple command and specifying the approximate time when a problem occurred, you can now gather all the necessary files from all nodes in the TFA configuration, including the diagnostic data from Oracle Clusterware, ASM, and the database. The following key business drivers play a role in deciding whether to use the Trace File Analyzer:

- You can reduce the length of service interruption through compressing the length of the problem-resolution cycle. This lets you reduce the time and effort involved in collecting and uploading diagnostic data.
- You can decrease the complexity of handling the process of diagnostic data collection over an entire cluster.
- You can enable Oracle to provide higher quality support service by minimizing the need for supplementing the transmission of the initial diagnostic data by Oracle customers.

TFA stores its metadata about the configuration, as well as the directories and files it monitors, in its own dedicated Berkeley database, which runs on every node in the cluster. TFA is implemented by a Java virtual machine (JVM) that runs as a daemon on each of a node's clusters. You don't need to install TFA—it's automatically installed when you install or upgrade to Oracle Database 12*c*. You use the **tfactl** command-line utility to execute the TFA-supported commands. On our servers, the **tfactl** utility is located here:

https://t.me/learningnets

```
/u01/app/12.2.0/grid/tfa/rac2/tfa_home/bin
```

The TFA daemon starts automatically upon node startup, and you can use the **init.tfs** command to start, stop, and restart the daemon, as shown here:

```
$ /etc/init.d.init.tfa start
$ /etc/init.d.init.tfa stop
$ /etc/init.d.init.tfa restart
```

You can confirm that the TFA daemon is running by checking the running processes in the following way:

```
[root@rac2 oracle]# ps -ef|grep tfa
root       887     1  0 16:10 ?        00:00:00 /bin/sh /etc/init.d/init.tfa run
root      1691     1  0 16:11 ?        00:00:45 /u01/app/12.1.0/grid/jdk/jre/bin/java
      -Xms128m -Xmx512m -classpath
      /u01/app/12.1.0/grid/tfa/rac2/tfa_home/jlib/RATFA.
      jar:/u01/app/12.1.0/grid/tfa/rac2/tfa_home/jlib/je-
      5.0.84.jar:/u01/app/12.1.0/grid/tfa/rac2/tfa_home/jlib/ojdbc6.jar:/u01/app/12.1.0/
      grid/tfa/rac2/tfa_home/jlib/commons-io-2.2.jar oracle.rat.tfa.TFAMain
      /u01/app/12.1.0/grid/tfa/rac2/tfa_home
```

You can use the **tfactl** command-line utility to start a diagnostics collection, purge diagnostic data, and print data from the Berkeley database. You can issue the command **tfactl** to view the available options, like so:

```
 [root@rac2 bin]# /u01/app/12.1.0/grid/tfa/rac2/tfa_home/bin/tfactl
Usage : /u01/app/12.1.0/grid/bin/tfactl <command> [options]
<command> =
          start        Starts TFA
          stop         Stops TFA
          enable       Enable TFA Auto restart
          disable      Disable TFA Auto restart
          print        Print requested details
          access       Add or Remove or List TFA Users and Groups
          purge        Delete collections from TFA repository
          directory    Add or Remove or Modify directory in TFA
          host         Add or Remove host in TFA
          diagcollect  Collect logs from across nodes in cluster
          analyze      List events summary and search strings in alert logs.
          set          Turn ON/OFF or Modify various TFA features
          uninstall    Uninstall TFA from this node
For help with a command: /u01/app/12.1.0/grid/bin/tfactl <command> -help
[root@rac2 bin]#
```

Here are some examples of how you can use the **tfactl** command:

https://t.me/learningnets

Technet24

```
$ tfactl print config
$ tfactl print hosts
$ tfactl print status
$ tfactl purge -older 30d
```

By default, **tfactl** always collects diagnostic information from the database, ASM, and Clusterware alert logs, based on specific patterns in the logs. You may also collect on-demand diagnostic data by specifying various parameters pertaining to a specific incident or time range. Here's an example:

```
$ tfactl diagcollect -all since 8h
```

This command collects and zips up all files updated in the past eight hours from the entire cluster.

The next example shows how to collect all diagnostic datafiles updated on a specific data and create a .zip file from those logs:

```
$ tfactl diagcollect -for May/29/2017
```

The most commonly specified option, time-wise, is the **-since 1d** flag. Here's an example:

```
[root@rac2 bin]# /u01/app/12.1.0/grid/tfa/rac2/tfa_home/bin/tfactl
                 diagcollect -since 1d
```

https://t.me/learningnets

```
Collecting data for all nodes
Repository Location in rac2 : /u01/app/oracle/tfa/repository
2017/05/29 18:58:23 EST : Running an inventory clusterwide ...
2017/05/29 18:58:23 EST : Collection Name : tfa_Sat_Nov_29_18_58_16_EST_2014.zip
2017/05/29 18:58:23 EST : Sending diagcollect request to host : rac1
2017/05/29 18:58:23 EST : Waiting for inventory to complete ...
2017/05/29 18:58:43 EST : Run inventory completed locally ...
2017/05/29 18:58:43 EST : Getting list of files satisfying time range [05/28/2017
 18:58:23 EST, 05/29/2017 18:58:43 EST]
2017/05/29 18:58:43 EST : Starting Thread to identify stored files to collect

2017/05/29 18:58:43 EST : Collecting extra files...
2017/05/29 18:58:43 EST : Getting List of Files to Collect
2017/05/29 18:58:44 EST : Trimming file :
/u01/app/oracle/diag/crs/rac2/crs/trace/crsd_1.trc with original file size : 10MB
2017/05/29 18:58:44 EST : Trimming file :
/u01/app/oracle/diag/crs/rac2/crs/trace/crsd_oraagent_oracle.trc with original
file size : 8.7MB

2017/05/29 18:58:45 EST : Trimming file :
/u01/app/oracle/diag/crs/rac2/crs/trace/crsd_orarootagent_root.trc with original
file size : 3.5MB

2017/05/29 18:58:45 EST : Finished Getting List of Files to Collect
2017/05/29 18:58:45 EST : Trimming file :
/u01/app/oracle/diag/crs/rac2/crs/trace/crsd_scriptagent_oracle.trc with original
 file size : 3.7MB

2017/05/29 18:58:46 EST : Trimming file :
/u01/app/oracle/diag/crs/rac2/crs/trace/evmd.trc with original file size : 2.4MB

2017/05/29 18:58:46 EST : Trimming file :
/u01/app/oracle/diag/crs/rac2/crs/trace/gipcd.trc with original file size : 1.6MB

2017/05/29 18:58:46 EST : Trimming file :
/u01/app/oracle/diag/crs/rac2/crs/trace/ocssd.trc with original file size : 21MB

2017/05/29 18:58:47 EST : Trimming file :
/u01/app/oracle/diag/crs/rac2/crs/trace/octssd.trc with original file size : 6.4MB
2017/05/29 18:58:48 EST : Trimming file : /u01/app/oracle/diag/crs/rac2/crs/trace/
ohasd.trc with original file size : 5.1MB
2017/05/29 18:58:48 EST : Trimming file : /u01/app/oracle/diag/crs/rac2/crs/trace/
ohasd_cssdagent_root.trc with original file size : 4.2MB
2017/05/29 18:58:49 EST : Trimming file : /u01/app/oracle/diag/crs/rac2/crs/trace/
ohasd_cssdmonitor_root.trc with original file size : 6.5MB
2017/05/29 18:58:49 EST : Trimming file :
/u01/app/oracle/diag/crs/rac2/crs/trace/ohasd_oraagent_oracle.trc with original
file size : 3MB
2017/05/29 18:58:49 EST : Trimming file : /u01/app/oracle/diag/crs/rac2/crs/trace/
ohasd_orarootagent_root.trc with original
file size : 7MB
2017/05/29 18:58:51 EST : Trimming file :
/u01/app/oracle/diag/tnslsnr/rac2/listener/trace/listener.log with original file
size : 1MB
2017/05/29 18:58:52 EST : rac2: Zipped 100 Files so Far
2017/05/29 18:58:52 EST : Trimming file : /u01/app/oracle/diag/tnslsnr/rac2/listener_
```

https://t.me/learningnets

```
scan1/trace/listener_scan1.log with
original file size : 1MB
2017/05/29 18:58:53 EST : rac2: Zipped 200 Files so Far
2017/05/29 18:59:01 EST : Collecting ADR incident files...
2017/05/29 18:59:01 EST : Waiting for collection of extra files
```

A review of the output reveals the extraordinarily comprehensive range of diagnostic collection initiated by just this one simple command!

If you don't want the TFA to collect every single bit of diagnostic data it's capable of, you can certainly narrow the range of data collection to a single component, such as the database, ASM, or CRS, for example. You can view the complete TFA usage syntax and all its available options by issuing the following command:

```
[root@rac2 bin]# /u01/app/12.1.0/grid/tfa/rac2/tfa_home/bin/tfactl diagcollect -h
```

Administering the OCR

The Oracle Cluster Registry (OCR) is the repository for Oracle RAC configuration information. It manages information about the cluster node list and instance-to-node mapping. Processes that make up the CRS and other cluster-aware applications use this repository to share information among themselves. Its contents include but are not limited to the following:

- Node membership information
- Database instance, node, and other mapping information
- Service characteristics
- Characteristics of any third-party applications controlled by CRS (10*g* R2 and later)

OCR's location is specified during CRS installation. The file pointer indicating the OCR device location is in the file ocr.loc, whose location is somewhat platform dependent. For example, on Linux systems it is located in /etc/oracle, and on Solaris it is located at /var/opt/oracle. The contents of ocr.loc follow:

```
#ocrconfig_loc=+ASMCCF1
local_only=FALSE
```

The first line offers information about the last operation performed on OCR impacting the contents of the ocr.loc file. You can mirror OCR at the Oracle level or at the OS level to provide high availability.

Little day-to-day maintenance is required for OCR. However, OCR is a critical component of an HA framework, so if anything happens to the OCR, you should be prepared to take corrective actions immediately. The following Oracle utilities are used to administer OCR. Practice these commands on a test system to be ready for any

https://t.me/learningnets

eventuality with the OCR:

- **ocrcheck** Performs a quick health check on OCR and prints space-usage statistics
- **ocrdump** Dumps the contents of OCR to an OS file
- **ocrconfig** Performs export, import, add, replace, remove, restore, and show backup operations on OCR

Checking OCR Integrity

You can use the OCRCHECK utility to perform a quick health check on OCR, as shown next. The command returns the OCR version, total space allocated, space used, free space, location of each device, and the result of the integrity check.

```
[root@alpha2 bin]# $GRID_HOME/bin/ocrcheck
Status of Oracle Cluster Registry is as follows :
         Version                  :          4
         Total space (kbytes)     :     262120
         Used space (kbytes)      :       3400
         Available space (kbytes) :     258720
         ID                       :  238319528
         Device/File Name         : +ASMCCF1
Device/File integrity check succeeded
Cluster registry integrity check succeeded
Logical corruption check succeeded
```

The command also creates a log file in the directory $GRID_HOME/log/<hostname>/client; the contents of this log file reflect what is displayed in the output.

Dumping OCR Information

The Oracle-supplied utility OCRDUMP writes the contents of OCR to an OS file. By default, it dumps the contents into a file named ocrdump in the current directory. Alternatively, you can specify a destination file and also dump information in XML format. The command-line options available for this command can be viewed by using the command with the **-help** option.

The contents of the dump file are generally used by Oracle Support Services (OSS) to check configuration information in the OCR. The dump file is an ASCII file that you can open using any text editor. The file contains a set of key name, value type, and key value information.

Here’s how to dump content of the OCR file into an XML file:

https://t.me/learningnets

Technet24

```
# >/u01/app/12.1.0/grid/bin/ocrdump.bin -local -stdout -xml

<OCRDUMP>
<TIMESTAMP>11/29/2014 15:50:05</TIMESTAMP>
<COMMAND>/u01/app/12.1.0/grid/bin/ocrdump.bin -local -stdout -xml </COMMAND>
<KEY>
<NAME>SYSTEM</NAME>
<VALUE_TYPE>UNDEF</VALUE_TYPE>
<VALUE><![CDATA[]]></VALUE>
<USER_PERMISSION>PROCR_ALL_ACCESS</USER_PERMISSION>
<GROUP_PERMISSION>PROCR_READ</GROUP_PERMISSION>
<OTHER_PERMISSION>PROCR_READ</OTHER_PERMISSION>
<USER_NAME>root</USER_NAME>
…
$
```

Managing the OCR with the OCRCONFIG Utility

You can use the OCRCONFIG utility to perform numerous OCR management tasks. In order to see all the options available to you with this command, type the following at the command line:

```
$ ocrconfig –help
```

The OCRCONFIG utility creates its log files in the $GRID_HOME/log/<*host_name*>/client directory.

Maintaining a Mirror OCR

Oracle allows you to create a mirror copy of the OCR, thereby eliminating OCR as a single point of failure. This also eliminates the need for mirroring OCR using methods external to Oracle, such as storage- or array-level mirroring.

The following command adds/relocates the ocrmirror file to the specified location:

```
ocrconfig -replace ocrmirror '+ASMCCF1'
```

NOTE

*Using **ocrconfig -replace** is the only way to add/relocate OCR files. Copying the existing OCR file to a new location and manually adding/changing the file pointer in the ocr.loc file is not supported and will not work.*

https://t.me/learningnets

You can use the following command to relocate an existing OCR file:

```
ocrconfig -replace ocr '+ASMCCF1'
```

You can relocate OCR only when OCR is mirrored. To relocate OCR or a mirror to a new location, another copy of OCR should be fully functional.

NOTE
OCR mirror add/relocate operations can be performed while CRS is running and hence do not require any system downtime.

The following command can be used to remove the OCR or the ocrmirror file:

```
ocrconfig –replace ocr
```

or

```
ocrconfig –replace ocrmirror
```

There’s no need to specify the file location as an option on the command line. The command will retrieve the file location and remove the intended file.

Migrating the OCR to ASM

In Oracle Database 12*c*, by default, OCR is designed to be stored in an ASM disk group. If you’re upgrading to Oracle Database 12*c* from an earlier release, you can migrate OCR to ASM storage, thus enhancing the manageability of the OCR files. Use the OCRCONFIG utility to migrate non-ASM-based OCR to ASM. Following are the steps you must follow to migrate OCR to ASM storage after upgrading your database to Oracle Database 12*c:*

1. Ensure that the ASM Compatibility initialization parameter file is set to at least 11.2.0.2.

2. Start Oracle ASM Configuration Assistant (ASMCA) on all the cluster nodes.

3. Create a disk group with at least normal redundancy to hold the OCR files.
4. Execute the following command as the root user:
   `$ ocrconfig –add _+new_disk_group`
   You must execute this command as many times as the number of OCR locations. That is, if you choose to have five OCR locations, you must execute this command five times, specifying a different disk group each time.

https://t.me/learningnets

Technet24

5. Remove the old OCR locations:

```
$ ocrconfig  -delete /ocrdata/ocr_1
$ ocrconfig  -delete /ocrdata/ocr_2
```

You can now store the OCR backups in the ASM disk groups you created. This simplifies OCR management because you can access these backups from any node in the cluster, as the backups are located on common ASM storage and not on specific nodes.

Administering the Oracle Local Registry

The Oracle Local Registry (OLR) is the local registry for node-specific resources; therefore, there's an OLR for each node in an Oracle RAC cluster. The default location of the OLR is $GRID_HOME/cdata/<*host_name*>.olr.

Although all the commands available to manage OCR aren't available to OLR, Oracle Local Registry is still managed in similar ways to how we manage the Oracle Cluster Registry. You therefore can use the OCRCHECK, OCRDUMP, and OCRCONFIG utilities, but you must add the **–local** option with each of these commands.

To check the status of the OLR on the node where you're working, issue the following command:

```
#[root@rac2 oracle]# /u01/app/12.1.0/grid/bin/ocrcheck -local
Status of Oracle Local Registry is as follows :
       Version                  :          4
       Total space (kbytes)     :     409568
       Used space (kbytes)      :        972
       Available space (kbytes) :     408596
       ID                       : 2049431744
       Device/File Name         :      /u01/app/12.1.0/grid/cdata/rac2.olr
       Device/File integrity check succeeded
       Local registry integrity check succeeded
       Logical corruption check succeeded
[root@rac2 oracle]#
```

If you want to dump the OLR contents on to the text terminal, issue the **ocrdump** command as shown here:

```
# ocrdump -local
```

You can issue the following command to manually back up the OLR:

```
# ocrconfig -local-manualbackup
```

To restore the OLR on a node, issue the following set of commands in sequence:

https://t.me/learningnets

```
# crsctl stop crs
# ocrconfig -local -restore <file_name>
# ocrcheck -local
# crsctl start crs
# cluvfy comp clr
```

Administering the Voting Disks

Oracle Clusterware uses a voting disk to resolve cluster membership issues in the event of partitioned clusters. Consider an eight-node cluster experiencing a breakdown in communication between the nodes—four nodes cannot communicate with the other four nodes. Situations like this can cause serious data integrity issues. A voting disk or a quorum disk provides a mechanism to resolve such issues. In case of a break in communication and a partitioned cluster, a voting disk helps in deciding which set of nodes should survive if another set of nodes should go down.

NOTE

Before you add, delete, replace, or restore a voting file, run the ***ocrcheck*** *command first because you need a valid and working OCR before you can manage voting files.*

All voting disks must be placed on shared storage to be accessible by all the nodes. A voting disk is a small file, usually a few hundred megabytes in size.

You can have a maximum of 15 voting files. Having multiple voting disks removes the voting disk as a single point of failure and eliminates the need to mirror them outside Oracle.

Oracle Universal Installer (OUI) allows you to specify up to three voting disks during Oracle Grid Infrastructure installation. Having three voting disks allows CRS operation to continue uninterrupted when any of the three voting disks fails. For the cluster to survive failure of *x* number of voting disks, you need to configure (2*x*+1) voting disks.

Using ASM Storage for Voting Files

Starting with Oracle 11*g*, voting disks can reside in Oracle ASM. Users can add, remove, or migrate voting disks in the Oracle ASM or between ASM and non-ASM file systems. Oracle will back up the voting disk in the Oracle Cluster Registry as a result of any configuration changes. Oracle will automatically restore the voting disk data into the newly added voting disk.

When you choose ASM storage for the voting files. Oracle ASM stores the cluster’s voting files in the disk group you select. Oracle stores each voting file in its own failure group within the disk group you assign.

https://t.me/learningnets

Technet24

The redundancy level of the disk group you select, which can be external, normal, or high, will determine the number of voting disks. Other than this, you have no way to select the number of voting files in a disk group. Because Oracle itself determines the number of voting disks based on the disk group's redundancy level, you can't use the traditional **crsctl add/delete vote disk** commands with the voting disks stored on ASM.

Backing Up Voting Disks

You don't need to worry about making backups of the voting disks because Oracle automatically backs up the voting file data in the OCR whenever you make any configuration changes. It also restores the voting disk data to any voting files you add.

Restoring Voting Disks

You must restore a voting disk when all voting disks have been corrupted. You do so by undertaking the following steps:

1. If the OCR has also been corrupted along with all the voting disks, first restore the OCR.
2. Because you don't have any usable voting disks, you must start the Oracle Clusterware stack in exclusive mode, as shown here:

   ```
   # crsctl start crs -excl
   ```

   Starting the CRS stack in exclusive mode doesn't require the voting files.
3. Get the list of the voting files:

   ```
   $ crsctl query css votedisk
   ##  STATE    File Universal Id                File Name Disk group
   --  -----    -----------------------------    --------------------
   1. ONLINE   296641fd201f4f3fbf3452156d3b5881  (/ocfs2/host09_vd3) []
   2. ONLINE   8c4a552bdd9a4fd9bf93e444223146f2  (/netapp/ocrvf/newvd) []
   3. ONLINE   8afeee6ae3ed4fe6bfbb556996ca4da5 (/ocfs2/host09_vd1) []
   Located 3 voting file(s).
   ```

   The **crsctl query css votedisk** command shows the state of the voting disk (online/offline), the File Universal ID (FUID), the filename, and the disk group, if the voting disk is stored on ASM.
4. If you stored all voting disks in ASM, migrate them to a new ASM disk group:

   ```
   $ crsctl replace votedisk  +DGROUP2
   ```
5. If the voting disks weren't stored on ASM, first delete the voting disks by specifying the File Universal Identifier (FUID) shown in the output of the command issued in step 3:

   ```
   $ crsctl delete css votedisk FUID
   ```
6. Add the file for the new voting disk:

   ```
   $ crsctl add css votedisk <path_to_voting_disk>
   ```
7. Stop and then start the Oracle Clusterware stack:

https://t.me/learningnets

```
# crsctl stop crs
# crsctl start crs
```

Adding and Deleting Voting Disks

This section shows how to add and delete voting disks. In order to add a voting disk, use the following command:

```
$ crsctl add css votedisk path_to_voting_disk
```

You can delete a voting disk with the following command:

```
$ crsctl delete css votedisk (FUID|path_to_voting_disk)
```

If you'd like to replace a voting disk, you must first add the new voting disk and then delete the disk you want to replace. If you'd like to add a new disk and remove the existing voting disk, run the following command:

```
$ crsctl add votedisk path_to_voting_diskNew –purge
```

In this example, *voting_diskNew* stands for your new voting disk, and you must replace *path_to_voting_diskNew* with the actual pathname for this new voting file. Note that the **-purge** option deletes the existing voting files.

Migrating Voting Disks

To migrate a voting disk to ASM-based storage, run the following command:

```
$ crsctl replace votedisk +DGROUP1
```

In this example, DGROUP1 is the ASM group that will store the voting disk.

Summary

You can administer Oracle Clusterware in one of two ways: using the traditional administrator management or the newer policy-based management. Policy-managed administration is new to most administrators, so we explained that approach in detail in this chapter.

Role separation for administrators using ACLs is a new concept to facilitate administration, and we introduced ACLs and showed how to create them with the help of the **crsctl setperm** command.

Administering SCAN is a complex area in Oracle Clusterware administration, and this chapter explained the most important management commands for administering SCAN (and GNS).

As an Oracle Clusterware administrator, you'll be using the CRSCTL utility often, so we explained many CRSCTL commands. The chapter also shows how to use other

https://t.me/learningnets

Technet24

utilities such as OCRCONFIG, OLSNODES, CLSCFG, and IFCFG to manage your Oracle Clusterware environment.

Finally, this chapter showed how to administer the two important Oracle Clusterware entities—the Oracle Cluster Registry (OCR) and the Oracle Local Registry (OLR)—as well as how to perform critical tasks pertaining to the management of the voting disks.

https://t.me/learningnets

CHAPTER 9

Oracle RAC Backup and Recovery

In this chapter, we discuss the key concepts and internals of backup and recovery operations in Oracle Real Application Clusters (Oracle RAC) databases. Basic backup and recovery methods for the Oracle RAC database are similar to those for single-instance databases, except in Oracle RAC we have redo records from archived redo logs from multiple instances. No special considerations for media recovery or any other advanced recovery mechanisms are required for Oracle RAC, so this chapter does *not* cover generic backup (and recovery) mechanisms.

However, in this chapter, we discuss the recovery issues that are specific to Oracle RAC, starting from the basics and then delving deeper into Oracle RAC–based backup and recovery as well as into other information that is not easily available or accessible. You can use the Oracle documentation—specifically, "Database Backup and Recovery User's Guide, 12*c*" and "Database Backup and Recovery Reference, 12*c*"—as supplemental references for basic information of backup and recovery operations.

We discuss the following topics pertaining to backups, recovery, and instance failures in the context of an Oracle RAC database:

- Introduction to backups
- Oracle backup basics

https://t.me/learningnets

Technet24

- Instance recovery in Oracle RAC
- Crash recovery
- Instance recovery
- Dynamic reconfiguration and affinity remastering
- Backup and recovery of the voting disk and OCR

Introduction to Backups

One of the key features of the Oracle RDBMS engine is its ability to recover from various kinds of faults, disasters, and human errors. The Oracle backup and recovery mechanism has evolved into a near "fault-tolerant" database engine that can guarantee data protection as well as recoverability more than any other database vendor's product. Transaction recovery is possible mainly due to the redo log.

Oracle has developed its own tools/utilities (Recovery Manager, or RMAN) to perform backups and restore/recover a database in the simplest or easiest way possible. Oracle also provides the user with the flexibility to choose between RMAN and other tools to make backups and restore data. Recovery of restored data can be done only through an Oracle interface such as SQL*Plus or RMAN.

Oracle Backup Options

Generally, Oracle's backup options can be classified based on the tools or technologies used to implement them. The following serves as a rough guide to understanding the various tools used for Oracle backups in the industry today.

Small, Noncritical Databases

These databases use operating system (OS) built-in utilities such as tar and cpio that are available to Windows users. The backup media can be a tape or disk. Usually, Windows systems do not have a separate backup tool or framework. This type of backup is almost exclusively a closed-database backup, where the business operation has the luxury of performing a *cold backup*—that is, stopping and shutting down the databases for backup.

Medium-to-Large Databases

These databases may also use common utilities such as tar and cpio. DBAs generally write or borrow automated scripts to perform these backups. Servers have a fast tape drive or two connected to the database host. Automated tape libraries are used in large organizations. It is common to find backup tools such as Veritas NetBackup, HP OmniBack or Storage Data Protector, IBM Tivoli, and so on, implemented at the database level. RMAN is also widely used, but with a lesser degree of integration with the third-party backup tools.

Oracle Data Guard is standard choice for disaster recovery in many types of

https://t.me/learningnets

organizations. Since Oracle 8*i*, Oracle Data Guard has presented a “near real-time switchover capability” that was not possible before with an Oracle database. Additional server hardware and disk storage (equivalent to your primary system) is required to implement Oracle Data Guard.

Enterprise Databases

These databases belong to big-spending mega corporations that use sophisticated technologies such as disk mirroring and splitting and remote geographical mirroring, which could be in conjunction with cluster technologies such as Oracle RAC. These companies may also deploy large, automated tape libraries or robotic libraries with scores of tape drives and redundancies. It has become increasing difficult for tape libraries to keep up with the rapid pace of Oracle data growth. Backup and recovery time is critical here. To overcome this time constraint, large enterprises employ disk mirroring as a method of quick data backup (sometimes known as *snapshots*) and also use the backup copy for reporting purposes or as a standby database. This comes at an extra cost, which can be millions of dollars.

NOTE

If you’re using snapshot and split-mirror for database backups, the backup set should be required to be “crash consistent,” which is the state of disks equivalent to what would be found following a catastrophic failure that abruptly shuts down the system. A restore from such a shadow copy set would be equivalent to a reboot following an abrupt shutdown.

Oracle Backup Basics

Oracle backups can be taken at the physical level (hot or cold) or at the logical level. A physical backup is composed of the following:

- Datafiles
- Controlfiles
- Archived redo log files (if the database is in ARCHIVELOG mode)
- Parameter files (init.ora and SPFILE)

A physical backup is performed by taking an operating system copy of these files. Alternatively, RMAN can be used to accomplish the backup of the database files, with the exception of the init.ora file. It is a well-known fact that online redo logs need not be backed up, irrespective of the fact that a database is in ARCHIVELOG mode or NOARCHIVELOG mode. For detailed information on this topic and other basics mentioned in this chapter, refer to the “Database Backup and Recovery User’s Guide,

https://t.me/learningnets

Technet24

12*c*."

Performing Backups in Oracle

Oracle databases exist in every size and shape. With terabyte-sized databases becoming common, large organizations and corporations no longer depend on tape libraries for backups; sophisticated disk-mirroring mechanisms haven taken their place. Oracle as a database company has come up with excellent solutions such as RMAN and Oracle Data Guard that allow even the largest databases to be backed up. The backup scheme needs to be chosen carefully. Some like to back up to disk and then push data to tape, whereas others like to back up directly to tape.

You can perform full, incremental, and incremental merge backups using RMAN by itself, or in combination with other third-party backup tools such as IBM Tivoli, Veritas NetBackup, and HP Storage Data Protector. If you use a third-party tool without RMAN integration, you need to follow the traditional method of putting a database in backup mode and backing it up.

RMAN Backup for Oracle RAC Databases

RMAN is capable of handling Oracle RAC databases, and it can be configured to perform Oracle RAC database backups. Most of the backup configuration you perform is persistent, so you need to perform that task just once, during the setup phase.

As mentioned earlier, you should consider RMAN as the backup solution even for Oracle RAC. The only consideration would be the time taken to perform the backup for large databases (larger than 500GB) when tape drives or tape libraries are used, which would be the same whether Oracle RAC is used or not. In such cases, disk-mirroring technology can be used if backup and restore time is of prime importance and/or the database size exceeds a terabyte.

Just about all RMAN commands and configuration items work the same way for an Oracle RAC database as they do for a regular single-instance database. In the following sections, we focus on a few important areas where you need to do things a bit differently for RAC database backups.

Configuring RMAN Channels for RAC Databases

You can set up automatic load balancing of backups by configuring RMAN channels in the following way:

```
CONFIGURE DEVICE TYPE [disk | sbt] PARALLELISM number_of_channels;
```

If you want each Oracle RAC database instance to use a separate RMAN channel, do the following:

```
CONFIGURE CHANNEL DEVICE TYPE sbt CONNECT '@racinst1'
CONFIGURE CHANNEL DEVICE TYPE sbt CONNECT '@racinst2'
```

The instance names "racinst1" and "racinst2" are connect strings that you've placed in the tnsnames.ora file.

https://t.me/learningnets

Choosing the Archive Log Destination in an Oracle RAC Database Configuring the destination of the archived redo logs in an Oracle RAC database is a crucial decision for a DBA. Because archive logs can be placed on ASM and a Cluster File System drive (shared) or a local drive (non-shared), this decision has serious implications on your backup and recovery strategies.

The primary consideration is to ensure that all archived redo logs can be read from every node during recovery and, if possible, even during backups, irrespective of whether RMAN is used.

When you use a cluster file system archiving schema, each node writes to a single location on that system when archiving redo log files. Each node can also read archived redo log files sent by the other nodes.

RMAN can back up the archive logs from any nodes in the cluster, as the files written by a node can be read by any node. Your RMAN backup and recovery scripts are thus greatly simplified. Because the file system is clustered, the archive log directory for each node in the cluster has an identical name, as shown by the following example:

```
*.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest"
```

When the database archives the redo logs to a non-clustered file system, each of the nodes in the cluster will send the archived redo logs to a unique local directory. Each node writes to a directory on the server where it's running, but can read archived redo log files in remote destinations of the other nodes. During a recovery, you must configure the node where you're performing recovery so it can access the remote archived red log destinations, using a Network Files System (NFS), for example.

You must configure the node performing recovery for remote access to all the other nodes, if you're using a non-cluster file system for local archiving. This enables the recovery process to read all the archived redo log files.

The key point to understand is that when a node generates an archived redo log, Oracle always records the filename of the archived log in the controlfile of the database. If you are using RMAN with a recovery catalog, RMAN also records the archived redo log filenames in the recovery catalog when a resynchronization occurs. However, archived log file pathnames do not include the node name, so RMAN expects to find the files it needs on the nodes where the channels are allocated.

The archived redo log naming scheme that you use is important because when a node writes to a log with a specific filename on its file system (or CFS), the file must be readable by any node that needs to access this archived redo log for backup and/or recovery purposes.

NOTE

When you configure the Fast Recovery Area (FRA), the location and the size should be

https://t.me/learningnets

Technet24

the same across the databases. The FRA simplifies the backup management and is highly recommended for Oracle RAC databases. The Oracle-recommended configuration for Oracle RAC is to use Automatic Storage Management (ASM) for storing the FRA, using a different disk group for your recovery set than for the datafiles.

The backup and recovery strategy you choose depends on how you configure the archiving destinations for each node. It doesn't matter if only one or all nodes perform archived redo log backups; you need to ensure that all archived redo logs are backed up.

If a cluster file system is used, all instances can write to a single archive log destination. Backup and recovery of the archive logs are easy because all logs are centrally located. If a cluster file system is not available, Oracle generally recommends that local archive log destinations be used for each instance with Network File System (NFS) read mount points to all other instances. This is known as the *local archive with NFS scheme*. During backup, you can either back up the archive logs from each host or select one host to perform the backup for all archive logs. During recovery, one instance may access the logs from any host without having to copy them first to the local destination. It is still crucial to provide a second archive destination to avoid single points of failure, irrespective of the scheme being used. ASM is always the recommended mechanism for backups. NFS can be flaky at times and is an alternative only if there's no shared storage.

Although all nodes can restore the archived redo logs in parallel, the recovery process works differently. During recovery, only one node will apply the archived logs. If RMAN parallelism is used during recovery, the node that performs the recovery must have access to all the archived redo logs needed for the recovery.

You can configure the number of parallel threads during a recovery by setting a value for the PARALLEL clause when you issue the RECOVER command. If you don't specify the clause, Oracle Database will figure out the optimum number of parallel threads to use. Multiple nodes can restore archived logs in parallel. However, during recovery, only one node applies the archived logs. Therefore, the node that is performing the recovery must be able to access all of the archived logs needed for recovery.

Oracle Data Guard and Oracle RAC

Oracle Data Guard fully supports Oracle Real Application Clusters. By this we mean that the primary database can be Oracle RAC and/or the standby database can be Oracle RAC. This provides for and extends the maximum availability demands that 24×7 shops place on databases and other systems today.

Oracle Data Guard also works seamlessly with RMAN, and this integration is helpful for creating standby databases using the DUPLICATE DATABASE command. Refer to the *Oracle Recovery Manager Reference* for more details.

https://t.me/learningnets

Instance Recovery in Oracle RAC

Instance recovery in Oracle RAC is not radically different from the way it is performed on a single instance. Some additional steps are required because redo entries for a data block can now exist in any redo threads. Hence, global (cluster-wide) coordination and synchronization are required to ensure that the consistency and reliability of data blocks are maintained, just like the Oracle recovery mechanism would in the case of a single-instance recovery.

At this juncture, it is important that you understand the subtle differences in the terms *instance recovery* and *crash recovery*. Although they generally mean the same thing in terms of operations, some minor contextual differences exist.

In a single-instance environment, the two terms are one and the same. In RAC, *crash recovery* means that all instances in the Oracle RAC database have failed and hence all instances have to be recovered or all instances may need recovery, depending on the operations they were performing. During a crash recovery, it isn't necessary for all instances to participate. At least one of the instances needs to participate.

Instance recovery means that one or more instances in the cluster database have failed and need to be recovered by the other surviving instance. Thread recovery is applicable in both situations because a single instance is being recovered, and the term generally describes the recovery of a single thread (or instance) in a cluster database and has more relevance in an Oracle RAC database than in a single-instance scenario.

The following section looks at the key concepts of recovery operations in the Oracle RAC database. Instance recovery and media recovery involve additional steps in the Oracle RAC environment as multiple redo streams record database changes, and they should be seamlessly applied to the database during recovery in a chronological order. The following structures ensure the recovery operations in Oracle RAC databases.

Redo Threads and Streams

Redo information generated by an instance is called a *thread of redo*. A thread is a sequence of changes occurring in an instance. All log files for that instance belong to this thread.

An online redo log file belongs to a group, and the group belongs to a thread.

If the database is running in ARCHIVELOG mode, the thread is nothing but all the archived redo log files. A record in the controlfile describes each and every online redo log file. The log file group and thread association details are stored in the controlfile.

A *redo stream* consists of all the threads of redo information ever recorded. The stream forms a timeline of changes performed to the database. In a single-instance database, the terms *thread* and *stream* refer to the same thing because a single instance has only one thread. Oracle RAC databases have multiple threads of redo—that is, each active instance has one active thread. In Oracle RAC, the threads are parallel timelines and together form the stream.

https://t.me/learningnets

Technet24

Redo Records and Change Vectors

When users make changes to the database, Oracle records them in the redo log file as *change vectors*. Each change vector is a description of a single change, usually to a single block. A set of change vector structures makes up the content of each redo record. A redo record contains one or more change vectors, is located by its *redo byte address (RBA)*, and points to a specific location in a redo log file (or a thread). It consists of three components: a log sequence number, a block number within the log, and a byte number within the block.

Checkpoints

Checkpoints are important database events that synchronize the database buffer cache and the datafiles. Without the various checkpoint mechanisms, recovery of data would be impossible. Checkpoints are used to determine the location or point from where recovery should start. This is the most important use of checkpoints and is indispensable in instance recovery (single and Oracle RAC).

In simple but crude terms, a *checkpoint* is a framework that enables the writing of dirty blocks to disk based on a system commit number (SCN) and RBA validation algorithm. More importantly, a checkpoint limits the number of blocks required for recovery.

Checkpoints ensure that data blocks that have redo generated up to a point in the redo log are written to disk. Checkpoint information is stored in a data structure called the *checkpoint structure*, which defines (points to) a specific location in a given redo log file. Checkpoint structures are stored in datafile headers and in the controlfile, and they are usually made up of the checkpoint SCN, checkpoint RBA, thread ID, timestamp, and some other control information. Like an RBA, these structures are used to find the starting point for reading the redo log for a redo application.

Checkpoints are triggered by a number of events (such as log switches, hot backups, and shutdowns) that in turn produce different types of checkpoints. The most important ones are briefly explained next.

Thread Checkpoint or Local Checkpoint

A thread checkpoint collects all dirty buffers in an instance that contains changes to any online datafile before a designated SCN—the thread checkpoint SCN—and writes them to disk. The SCN is associated with a specific RBA in the log, which is used to determine when all the buffers have been written. A thread checkpoint can occur at a log switch or if any of the thread checkpoint conditions are satisfied.

All blocks dirtied prior to a thread checkpoint SCN are written to disk. In an Oracle RAC database, a thread checkpoint occurs independently and separately for each instance, because each has its own thread of redo. This information is recorded in a structure called the *thread checkpoint structure* and in multiple controlfile records and all online datafile headers.

https://t.me/learningnets

Database Checkpoint or Global Checkpoint

When a database checkpoint needs to be triggered, Oracle looks for the thread checkpoint that has the lowest checkpoint SCN for all open and enabled threads (highest checkpoint SCN for all closed threads), and that itself becomes the database checkpoint. All blocks in memory that contain changes made prior to this SCN across all instances must be written out to disk. For a single-instance database, a database checkpoint is the same as a thread checkpoint. The information is recorded in several controlfile records and all online datafile headers.

Incremental Checkpoint

When data blocks are modified in the buffer cache, they are placed in a queue called the Checkpoint Queue (CKPTQ) for background writing by the DBWR (Database Writer) process. The CKPTQ was introduced in Oracle 8. This queue is ordered by the RBA of the first log record to modify the block, which is nothing but the earliest modification record for that block. The oldest dirty blocks are the first in the queue and are waiting to be written out. Since Oracle 8*i*, when the touch count mechanism was introduced, some blocks may still remain in the buffer cache if it is an active buffer or a hot buffer.

The idea behind incremental checkpoints is to reduce the amount of redo that has to be read during instance recovery. This allows instance recovery time to be bounded by a DBA. To accomplish this, an “in-memory” checkpoint record is updated approximately every 3 seconds and action is also taken to ensure that the number and age of dirty data blocks in the cache are limited.

A CKPTQ enables the incremental checkpoint to avoid having a cache filled with many dirty blocks, which all must be written at once when a checkpoint occurs. Because the dirty block count is kept low, the number of blocks that need to be recovered in case of a crash is fewer, resulting in faster database recovery. The length of this list is an indicator of the amount of blocks that need recovery if the block is lost in a crash. Information on this is visible in the V$INSTANCE_ RECOVERY view.

Log History Considerations

The controlfile always maintains a record of the archived log files generated for each thread. This enables an instance that is performing media recovery to identify the archived log files it needs regardless of which instance generated them. The number of entries maintained in the controlfile is determined by the value assigned with the MAXLOGHISTORY setting in the CREATE DATABASE command.

Usually, DBAs overlook or sometimes do not pay much attention to the log history until the following occurs:

- Oracle complains that more online redo log files cannot be created during an attempt to create them.

https://t.me/learningnets

Technet24

- Standby redo log files cannot be created in a physical Oracle Data Guard setup.
- The DBA is asked to generate metrics about the rate at which redo is generated.

Generally, it is recommended that MAXLOGHISTORY not be less than the total number of archived log files generated across all your instances between each complete database backup. This enables a recovering instance to identify the archived log files that it requires after you restore the most recent backup.

If there are insufficient archived log entries in your controlfile, you are prompted for the required filenames when you initiate the recovery process. This can be hard to do when hundreds or maybe thousands of archived logs need to be applied. To increase the size of the MAXLOGHISTORY setting, the controlfile must be re-created.

Crash Recovery

The internal mechanics of recovery are well beyond the discussion context of this chapter, yet it is important to remind you of some important characteristics of recovering a single instance (non-OPS or Oracle RAC). As mentioned earlier, the terms *instance recovery* and *crash recovery* refer to the same recovery aspect. The only notable difference is that in Oracle RAC, crash recovery involves *all* the instances. For a single instance, crash recovery and instance recovery are the same. Some of these points are also applicable to Oracle RAC databases because the mechanisms are basically the same.

Steps in Crash Recovery (Single Instance)

When an instance failure occurs (which leads to a database crash in a single-instance database), the following is the recovery process:

1. The on-disk version of a block is the starting point for recovery. Oracle will need to consider only the block on disk, and the recovery is straightforward. Crash recovery is automatic, using the online redo logs that are current or active.
2. The starting point of a thread recovery is at most the last full checkpoint. The starting point is provided in the controlfile and compared against the same information in all datafile headers. Only the changes from a single redo thread need to be applied.
3. The block specified in the redo log is read into the cache. If the block has the timestamp earlier than the redo record (SCN match is satisfied), the redo is applied. The block is then written out by a recovery checkpoint at a log switch or when aged out of the cache.

https://t.me/learningnets

Crash Recovery in Oracle RAC

Oracle performs instance recovery automatically upon detecting that an instance has died. Instance/crash recovery is performed automatically when the database is opened for the first time after a crash or when one of the instances of an Oracle RAC database fails. In the case of Oracle RAC, a surviving instance detects the need to perform instance recovery for one or more failed instances by the following methods:

- A foreground process in a surviving instance detects an “invalid block lock” condition when an attempt is made to read a block into the buffer cache. This is an indication that another instance died while a block covered by that lock was in a potentially dirty state in its buffer cache.
- The invalid block condition is more of a suspect lock that needs to be resolved because it’s in a dubious state before being accessed or its master died in the instance crash and hence a new master needs to be constructed. Therefore, the foreground process sends a notification to its instance’s System Monitor (SMON) process, which begins a search for dead instances.
- The death of another instance is detected if the current instance is able to acquire that instance’s redo thread locks, which is usually held by an open and active instance.

NOTE

In Oracle RAC, the SMON process in the surviving instance obtains a reliable list of dead instances together with a list of “invalid” block locks. These locks are invalid because the instance that had locked these blocks has crashed and their status remains “fuzzy” and/or unknown. The background process SMON also performs the recovery. The instance performing the recovery would then clean up these locks and make them available for normal use as soon as recovery and cleanup are complete.

Instance Recovery

The procedure used to recover from single-instance crash failures as well as failures of Oracle RAC instances is called *instance recovery*. In the case of Oracle RAC, a surviving instance recovers all the failed instances. Instance recovery aims to restore the data block changes that were in the cache of the failed instance and to close the thread that was left open. Instance recovery uses only online redo log files and current online datafiles (not restored backups).

With Oracle 9*i* and Cache Fusion Phase II, disk ping is avoided in many situations because the holding instance downgrades its lock (EXL → SHR), keeps a “past image” of the block (which cannot be modified from this point onward by this instance or any other instance), and sends the block across the interconnect to the requesting instance,

https://t.me/learningnets

Technet24

which can then make the required changes to the block after being granted the required and compatible locks. This topic is discussed in “Internals of Cache Fusion Recovery,” later in this chapter.

Oracle RAC introduced a few good optimization techniques in recovery, and one of them is the *thread merge* mechanism. Using this formula, redo from multiple threads is merged when crash recovery or instance recovery is performed in Oracle RAC. This is because changes from multiple instances may not have made it to the datafiles. This is termed *thread merge recovery*. Media recovery also uses the thread merge mechanism where redos from all the threads are merged and applied simultaneously. The redo records are merged in increasing SCN order. More on the thread merge mechanism is presented in “Two-Pass Recovery,” later in this chapter.

Crash Recovery and Media Recovery

Some of the basic differences between crash and media recovery are presented in the following table for completeness in understanding recovery structures:

Number	Crash Recovery	Media Recovery
1	Automatic. Uses the online redo logs that are CURRENT or ACTIVE.	Manual process. Can be performed even without ARCHIVELOG mode, if there's enough information in the online redo logs.
2	No concept of incomplete recovery unless block corruption (data or redo) is identified. This is a complicated situation and is not discussed here.	Can be complete or incomplete. Incomplete recovery requires RESETLOGS to open the database.

Bounded Recovery

Oracle’s two-pass recovery mechanism has a lot to do with the concept of *bounded time recovery*. Bounded time recovery is a feature that enables you to control the amount of time taken for crash recovery with some predetermined limits.

Block-Written Record (BWR)

One of the optimization recovery mechanisms used by Oracle is the writing of additional (yet critical) information into the redo log about checkpoints. Normally, the cache aging and incremental checkpoint system would write a number of blocks to disk. When the Dirty Buffer Writer (DBWR) completes a data block write operation, it also adds a redo record (in the redo log file) that states that the block has been written. It basically writes the data block address along with the SCN information. DBWR can write block-written records (BWRs) in batches, although in a “lazy” fashion.

In Oracle RAC, a BWR is written when an instance writes a block covered by a

https://t.me/learningnets

global resource or when it is told that the past image (PI) buffer it is holding is no longer necessary. Recovery processes that indicate redo information for the block is not needed prior to this point use this record. The basic use of BWR is to make recovery more efficient; hence, the instance does not force a flush of the log buffer after creating it because it is not essential for ensuring the accuracy of recovery.

Past Image (PI)

PI is what makes Oracle RAC Cache Fusion click, and it's also one big reason that block cache sizes need to be much larger in Oracle RAC, especially for active, write-heavy RAC databases that also have a heavy read workload.

Cache Fusion eliminates potential write/write contention that could be caused by block transfers from one node's buffer cache to another's because the transfers don't require an intermediate disk write (a ping), with the attendant I/O and network performance deterioration.

In the simplest of terms, a PI is a copy of a globally dirty block and is maintained in the database buffer cache. It can be created and saved when a dirty block is shipped across to another instance after the resource role is set to global (if it wasn't already).

A PI must be maintained by an instance until it or a later version of the block is written to disk. The Global Cache Service (GCS) is responsible for informing an instance that its PI is no longer needed after another instance writes a newer (current) version of the same block. PIs are discarded when GCS posts to all the holding instances that a new and consistent version of that particular block is now on disk.

In simpler terms, PI is a buffer modified in an instance but is shipped out to another instance before being written to the disk. Since Oracle needs the copy of the buffer to perform recovery, the buffer is marked as PI. Once written to the disk, the buffer is no longer needed for recovery and hence it is no longer marked as PI.

In Cache Fusion, when an instance needs to write a block to satisfy a checkpoint request, the instance must check the role of the resource covering the block. If the role is global, the instance must inform GCS that it needs to write that block to disk. GCS is responsible for finding the most current block image and informing the instance holding that image to perform the block write. GCS then informs all holders of the global resource that they can release their PI copies of the block, hence allowing the global resources to be released.

The async log writer feature in Oracle Database RAC 12*c* uses a mechanism where the LGWR forces sessions to queue on its resource instead of being polled on, especially if it has too many clients and they get posted when their redo request is complete. This is very handy because it handles the heavy-loading scenarios where too much polling wouldn't be desirable. Under the flash log feature of Exadata, the writes are aligned to the LGWR's write size to make the flash log writes much faster, like 15–50 ms instead of 600 ms.

Two-Pass Recovery

https://t.me/learningnets

Technet24

Oracle RAC introduced the concept of two-pass recovery, where the recovery process (SMON or the foreground process) performs a two-step "read" procedure. This limits the number of I/O reads required for recovery by recording more information in the logs (BWR). The first read builds a list of blocks that are mentioned in the redo log (all data blocks that have redo information). Some of these redo records could be BWR entries, denoting that the block mentioned is up to date until that point in the redo. Therefore, the recovery process need not "recover" this block, and it is removed from the list being built. The resultant list of this first pass is a list of blocks that have redo information but were not written to disk (because there was an instance failure). This list is called the *recovery set*.

The second read now processes only blocks from this list or set, which is smaller than the blocks touched in the redo stream in the first pass. Redo is applied in this phase, and fewer data blocks are read and written in the second pass, thus offsetting the cost of reading the online redo log files twice. If the system is unable to perform two-pass recovery, it will fall back to the single pass. The alert file states the result from two-pass recovery.

Two-pass recovery in Oracle RAC has some additional steps to be performed because multiple instances (threads) may have failed or crashed. This involves reading and merging all the redo information for a particular block from all the threads, and is called a *log merge* or a*thread merge* operation.

One of the bigger challenges in Oracle RAC recovery is that a block could have been modified in any of the instances (dead or alive). This was not the case in Oracle Parallel Server (OPS). Hence, in Oracle RAC, getting ahold of the latest version of a dirty block needs an intelligent and efficient mechanism that completes the identification of the latest version of that block and processes it for recovery. The introduction of PI images and BWRs makes it possible to significantly reduce recovery time and efficiently recover from an instance failure or crash.

First Pass

This pass does not perform the actual recovery but rather merges and reads redo threads to create a hash table of blocks that need recovery and that are not known to have been written back to the datafiles. This is where incremental checkpoint SCN is crucial because the redo byte address (RBA) denotes a starting point for recovery. All modified blocks are added to the recovery set. As BWRs are encountered, the file, RBA, and SCN of each change vector are processed to limit the number of blocks to recover in the next pass. A block need not be recovered if its BWR version is greater than the latest PI present in any of the buffer caches.

Redo threads from all failed instances are read and merged by SCN, beginning at the RBA of the last incremental checkpoint for each thread. When the first change to a data block is encountered in the merged redo stream, a block entry is added in the recovery set data structure. Entries in the recovery set are organized in a hash table.

Second Pass

https://t.me/learningnets

In this stage, SMON rereads the merged redo stream (by SCN) from all threads needing recovery. The redo log entries are again compared against the recovery set built in the first pass, and any matches are applied to the in-memory buffers as in single-pass recovery. The buffer cache is flushed, and the checkpoint SCN for each thread is updated upon successful completion. This is also the single-pass thread recovery if only one pass is to be done.

Cache Fusion Recovery

Cache Fusion recovery is applicable only in Oracle RAC, and it requires additional steps—GRD (Global Resource Directory) (re)configuration, internode communication, and so on—on top of the existing recovery steps. The SMON from a surviving instance recovers the failed instance. If a foreground process detects instance recovery, it posts SMON.

Crash recovery is a unique case of instance recovery in which all instances have failed. Yet, in either case, the threads from failed instances need to be merged. The only distinction being that in instance recovery, SMON performs the recovery. In crash recovery, a foreground process performs the recovery.

Let's now examine the main steps involved in Cache Fusion recovery or instance recovery. The main advantages or features of Cache Fusion recovery are as follows:

- Recovery cost is proportional to the number of failures, not the total number of nodes.
- It eliminates disk reads of blocks that are present in a surviving instance's cache.
- It prunes recovery sets based on the global resource lock state.
- The cluster is available after an initial log scan, even before recovery reads are complete.

In Cache Fusion recovery, the starting point for recovery of a block is its most recent PI version. A PI could be located in any of the surviving instances, and multiple PI blocks of a particular buffer can exist. An on-disk version is used for recovery only if no PI is available. This feature is called Cache Fusion recovery because, from Oracle 9*i* onward, the on-disk version of a block might not be the latest copy, given that Cache Fusion allows the shipping of copies of CURRENT blocks across the interconnect by using the PI concept.

Dynamic Reconfiguration and Affinity Remastering

Remastering is the term used to describe the operation whereby the node attempting the recovery tries to own or master the resource(s) once mastered by another instance prior to a failure. Hence, the term *remaster* is used for the operation performed during

https://t.me/learningnets

Technet24

instance recovery or when a node joins or leaves a cluster.

When one instance departs the cluster, the GRD component of that instance needs to be redistributed to the surviving nodes. Similarly, when a new instance enters the cluster, the GRD portions of the existing instances need to be redistributed to create the GRD portion of the new instance.

During reconfiguration, remastering freeze occurs and all access to blocks is temporarily frozen as the cluster reconfigures the GRD. A brownout ensues as a result, impacting online applications. Depending on the size of the System Global Area (SGA), number of active blocks, and so on, it can take up to a minute or more for this freeze to be lifted. This occurs during both planned and unplanned node restarts/instance events.

As an optimization feature during instance recovery, remastering of all resources does not happen across all nodes. Oracle RAC uses an algorithm called *lazy remastering* to remaster only a minimal number of resources during a reconfiguration. A minimum subset of resources is remastered to maintain the consistency of the lock database. This occurs in parallel during the first pass log read where the recovery set is built. The entire Parallel Cache Management (PCM) lock space remains invalid while the DLM (Distributed Lock Manager) and SMON complete the following two crucial steps:

1. The Integrated Distributed Lock Manager (IDLM) master node discards locks that are held by dead instances; the space reclaimed by this operation is used to remaster locks that are held by the surviving instance for which a dead instance was mastered.
2. SMON issues a message saying that it has acquired the necessary buffer locks to perform recovery.

While the lock domain is invalid, most PCM lock operations are frozen, making the database unavailable for users requesting a new or incompatible lock. Operations that do not require interaction with the DLM can proceed without affecting the remastering operations. If a second instance fails, its resources are remastered on the other surviving instances evenly. As resources are remastered, they are cleared of any reference to the failed instance.

In addition, the database can automatically adapt and migrate resources in the GRD, based on the affinity of a resource to a particular instance. If a single instance is identified as the sole user of a tablespace, the block resource masters for files of that tablespace are lazily and dynamically moved to that instance.

In both cases, the use of dynamic resource remastering provides the key benefit of greatly increasing the likelihood of local cache access without additional interconnect traffic. Another great advantage of the lazy remastering scheme is that instances keep many of their locks/resources during a reconfiguration. Because of this concept, many processes can resume active work during a reconfiguration because their locks/resources do not have to be moved away or deleted.

https://t.me/learningnets

Fast Reconfiguration in Oracle RAC

Fast reconfiguration is designed to increase the availability time for Oracle RAC instances during instance reconfiguration. Reconfiguration of open DLM locks/resources takes place under the following conditions:

- An instance joins the cluster.
- An instance fails or leaves the cluster.
- A node is halted.

Oracle RAC uses optimization techniques where the thrust is on decreasing the amount of time it takes to complete reconfiguration and allowing some processes to continue work (in parallel) during reconfiguration.

Oracle uses the *lazy remastering* algorithm (introduced in the preceding section) to remaster only a minimal number of locks/resources during a reconfiguration. For a departing instance (expected or unexpected), Oracle RAC tries to determine how best to distribute only the locks/resources from the departing instance and a minimal number from the surviving instances.

Let's consider a simple example of this phenomenon in Figure 9-1, although the actual resource-mastering process is quite different from this simplified one:

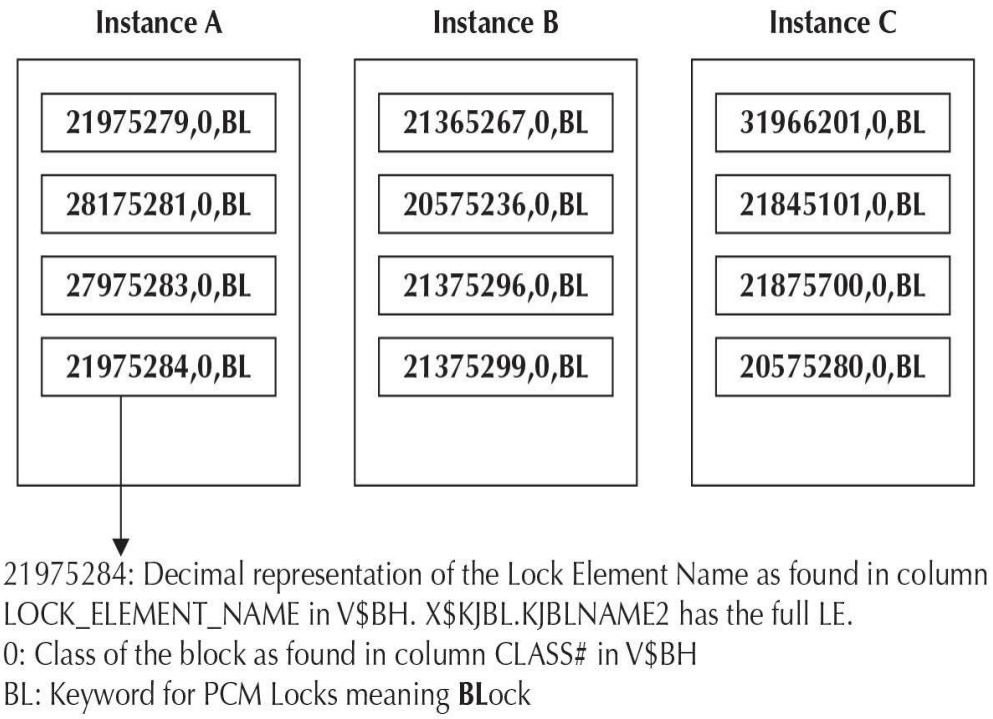


FIGURE 9-1. *Resource mastering example*

- Instance A masters resources 1, 3, 5, and 7.
- Instance B masters resources 2, 4, 6, and 8.

https://t.me/learningnets

Technet24

- Instance C masters resources 9, 10, 11, and 12.

Now, assume that instance B crashes (see Figure 9-2). In this case, resources 2, 4, 6, and 8 will be affected by the instance failure. The resources mastered in instance A and instance C are not affected by this failure.

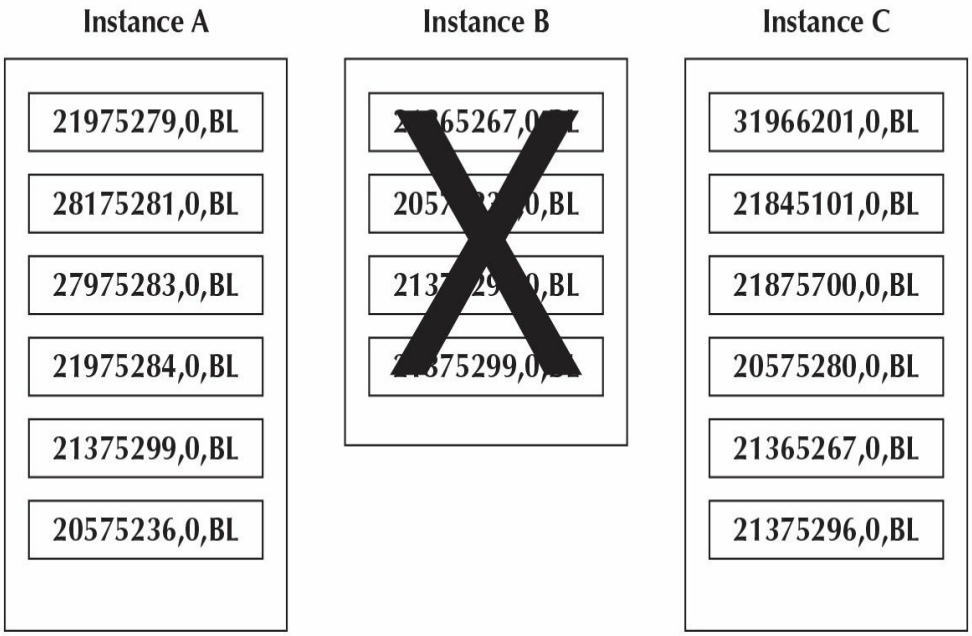
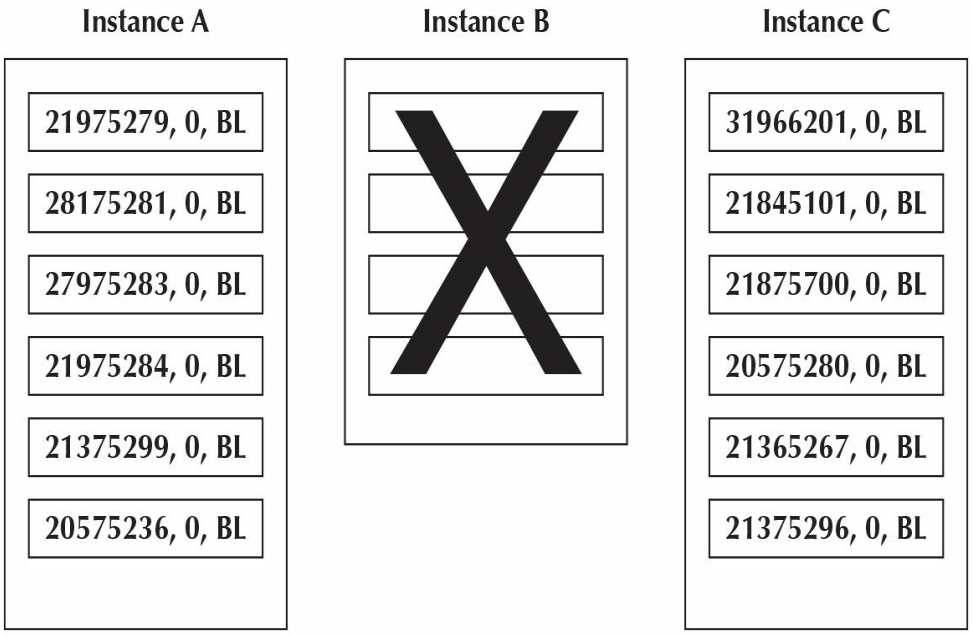


FIGURE 9-2. *Instance crash*

Now instances A and C remaster their resources. After remastering, the DLM database (GRD) could look like this (see Figure 9-3):



https://t.me/learningnets

FIGURE 9-3. *Resource remastering*

- Instance A masters resources 1, 3, 5, 7, 4, and 8.
- Instance C masters resources 9, 10, 11, 12, 2, and 6.

So, instead of removing all resources and remastering them evenly across instances, Oracle RAC will remaster only the resources necessary (in this case, those owned by the departing instance), thus using a more efficient means of reconfiguration. With an instance joining the cluster, Oracle RAC will remaster a limited number of resources from the other instances to the new instance.

Assume that instance B now rejoins the cluster; a minimal number of resources are remastered to instance B from the other instances. This is much faster than the 8*i* behavior of redistributing all resources. Here is the configuration after instance B rejoins the GRD:

- Instance A masters resources 1, 3, 5, and 7.
- Instance B masters resources 2, 4, 6, and 8.
- Instance C masters resources 9, 10, 11, and 12.

Fast reconfiguration is controlled by the Oracle undocumented initialization parameter __GCS_FAST_RECONFIG. Another parameter, _LM_MASTER_WEIGHT, controls which instance will hold or (re)master more resources than others. Similarly, the _GCS_RESOURCES parameter is used to control the number of resources an instance will master at a time. Each instance can have a different value.

With the concept of lazy remastering, instances retain many of their locks/resources during a reconfiguration process, whereas in previous versions, all locks were deleted from all instances. Because of this concept, many processes can resume active work during a reconfiguration because their locks/resources do not have to be moved or remastered.

Internals of Cache Fusion Recovery

As mentioned, when you recover an Oracle RAC database, additional steps are involved due to the presence of a cluster and multiple instances. When an instance fails, for example, in a two-node Oracle RAC database, the failure is detected by the surviving instance, which then performs the following recovery steps, as explained in Figure 9-4. Note that GRD reconfiguration (remastering) and single-pass recovery can be done in parallel.

https://t.me/learningnets

Technet24

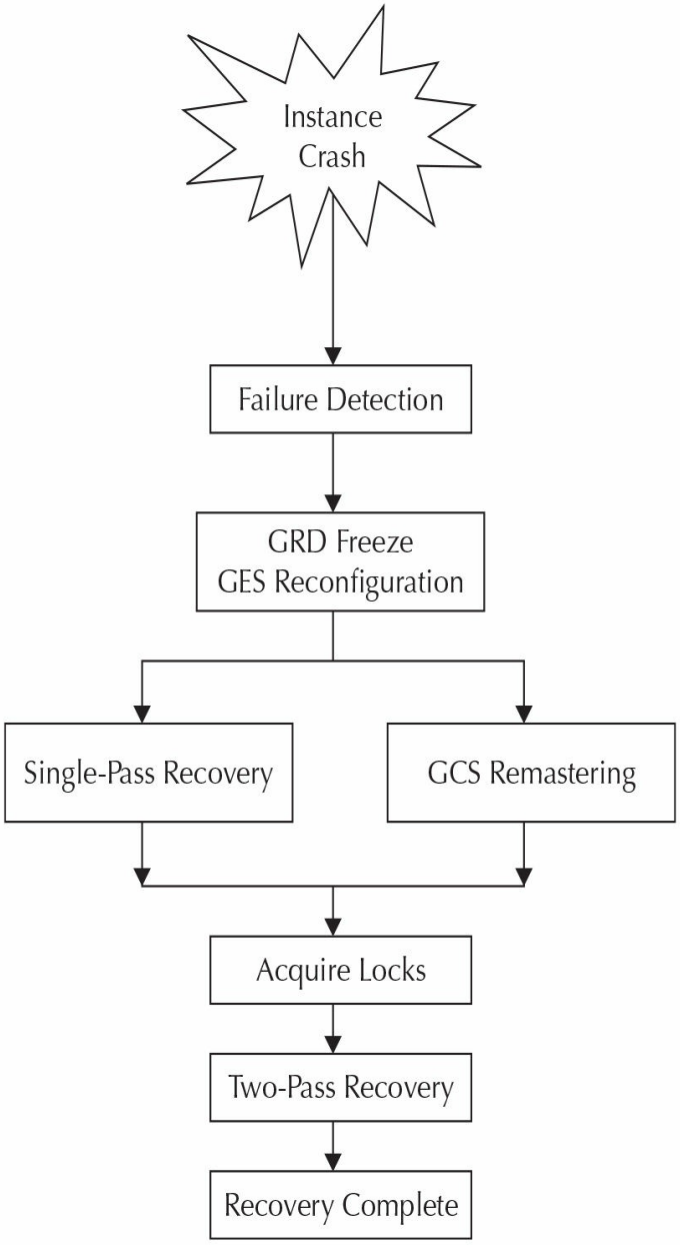


FIGURE 9-4. *Two-pass recovery in Oracle RAC*

The steps for GRD reconfiguration are as follows:

1. Instance death is detected by the cluster manager process.
2. Requests for GRD locks are frozen.
3. Enqueues are reconfigured and made available.
4. DLM recovery is performed.
5. GCS (PCM lock) is remastered.
6. Pending writes and notifications are processed.

The steps for single-pass recovery are as follows:

https://t.me/learningnets

1. The instance recovery (IR) lock is acquired by SMON.
2. The recovery set is prepared and built. Memory space is allocated in the SMON Program Global Area (PGA).
3. SMON acquires locks on buffers that need recovery.

The two-pass recovery steps are as follows:

1. The second pass is initiated. The database is partially available.
2. Blocks are made available as they are recovered.
3. The IR lock is released by SMON. Recovery is complete.
4. The system is available.

Global Resource Directory Reconfiguration

An instance failure is detected by the cluster manager (Cluster Group Services). The DLM reconfiguration process is started and all locks owned by the failed instance are remastered. This is when the PCM lock database is frozen and no instance can acquire a PCM lock. The reconfiguration phase (in lazy mode) can continue in parallel along with the first-pass read process. During this phase, existing PIs in any instance buffer caches are identified as potential candidates for recovery. DLM reconfiguration or recovery is performed by LMON (Lock Monitor).

First-Pass Redo Log Read

SMON acquires the IR (Instance Recovery) enqueue. By doing that, Oracle keeps multiple surviving instances from trying to recover a failed instance simultaneously and causing severe inconsistencies and more failures.

The SMON process of a surviving instance that has acquired the IR enqueue starts the first-pass log read of the failed instance's redo thread. SMON then merges the redo thread ordered by SCN to ensure that changes are written in a sequential order. The recovery set is built during this phase and contains the first dirty and last dirty version information (SCN, Seq#) of each block. SMON trims the recovery set (removes blocks no longer needed for recovery) based on the RBAs found in the redo stream because these blocks are nothing but PIs already written to disk. The BWR helps in this trimming.

The result of the first-pass log read is a recovery set (built into the PGA of SMON) that contains only blocks modified by the failed instance with no subsequent BWR to indicate that the blocks were later written. Each entry in the recovery list is ordered by the first dirty SCN to specify the order in which to acquire instance recovery locks.

The recovering SMON process will then inform each lock element's master node for each block in the recovery list that it will be taking ownership of the block and lock for recovery. Logically speaking, only the part of the database requiring recovery is locked; the rest of the database is still available for normal operations. Note that the actual locking is done during the second-pass read.

Acquiring block buffers (and their locks) for recovery is a complex process and

https://t.me/learningnets

Technet24

needs DLM (GES) messaging and coordination. Depending on the lock status of each buffer during the time of recovery, the recovery process has to perform a series of steps before an exclusive lock on the buffer can be held. A detailed description of the various lock states and eventual outcome is discussed in the Oracle Support Note 144152.1.

Recovery Lock Claim It is in this phase that SMON indicates to DLM (GES) that it needs IR locks on all the buffers identified in the recovery set. SMON continues to acquire locks on all buffers until it runs through the recovery set. If the master node for a resource has failed and the DLM remastering has not completed, SMON waits until the resource is made available. This also implies that DLM recovery is occurring in parallel. Now, SMON sends a cluster-wide message to indicate that all required locks on buffers have been acquired. The DLM lock database is unfrozen and is available for normal operations.

Validate Global Resource Directory After the DLM completes reconfiguration, only the resources that are locked for recovery are unavailable to foreground lock requests. Once the PCM lock database is released, other foreground processes can continue to acquire buffer locks as long as they are not requesting an “in-recovery” buffer. The GRD is globally resynchronized and flagged as available.

Second-Pass Log Read and Redo Application

It is in this phase that redo threads of failed instances are again read and merged by SCN. Buffer space for recovery is allocated in the database buffer cache and the resources that were identified in the previous reading of the redo logs are claimed as recovery resources. This is done to prevent other instances from accessing those resources. Then, assuming that there are PIs or current images of blocks to be recovered in other caches in the cluster, the most recent PI is the starting point of recovery.

If neither the PI buffers nor the current buffer for a data block are in any of the surviving instances’ caches, SMON performs a log merge of the failed instances. Then redo is applied to each buffer identified in the first pass until its SCN matches the last dirty SCN that was identified in the first pass. SMON recovers and posts DBWR to write the recovery buffer and clear the “in-recovery” state of the buffer. Soon after the write, SMON releases the recovery resources so that more blocks become available as recovery proceeds. Recovery time is determined by this phase.

After all blocks have been recovered and the recovery resources have been released, the system is again fully available. In summary, the recovered database or recovered portions of the database become available earlier and before the completion of the entire recovery sequence. This makes the system available sooner and it makes recovery more scalable.

https://t.me/learningnets

NOTE

The performance overhead of a log merge is proportional to the number of failed instances and to the size of the redo logs for each instance.

Online Block Recovery for Cache Fusion When a data buffer becomes corrupt in an instance's cache, the instance will initiate online block recovery. Block recovery will also occur if either a foreground process dies while changes are being applied or an error is generated during redo application. In the first case, SMON initiates block recovery, and in the second case the foreground process initiates block recovery. Online block recovery consists of finding the block's predecessor and applying redo changes from the online logs of the thread in which corruption occurred. The predecessor of a fusion block is its most recent past image. If there is no past image, the block on disk is the predecessor. For non-Fusion blocks, the disk copy is always the predecessor.

If the lock element (LE) of the block needing recovery is held in XL0 (Exclusive, Local, no past images) status, the predecessor will be located on disk. If the LE of the block needing recovery is held in XG# (Global Exclusive) status, the predecessor will exist in another instance's buffer cache. The instance with the highest SCN PI image of the block will send a consistent read copy of the block to the recovering instance.

Backup and Recovery of the Voting Disk and OCR

The voting disk and OCR are the two critical components in Oracle Clusterware. In this section, we discuss the backup and recovery of the voting disk and OCR.

Oracle Clusterware backups should include the voting disk and OCR components. The voting disk is a disk partition that stores information about the status of the nodes and membership details. The OCR is a file that manages the configuration and the details about the services. We recommended mirroring the voting disk and the OCR using Oracle-provided mirroring techniques. Mirroring the voting disk and OCR can be configured during installation or at later stages dynamically.

Backup and Recovery of Voting Disks

The voting disk data is automatically included with OCR backups. Upon a successful OCR restore, you can simply run the **crsctl add css votedisk** command to create new voting disks in the desired location. Manual backup of voting disks using **dd** or any copy command is unsupported since Oracle 11*g* Release 2.

Backup and Recovery of OCR

https://t.me/learningnets

Technet24

Because Oracle Cluster Registry (OCR) is very critical to the functionality of Oracle Grid Infrastructure, the clusterware automatically creates OCR backups every four hours. At any one time, Oracle always retains the last three backup copies of the OCR in the master node.

Also, the CRS daemon (CRSd) process that creates the backups also creates and retains an OCR backup for each full day and at the end of each week. At any point in time you will have five successful OCR backups created automatically by the clusterware.

You can find the backup location by issuing the following command:

```
$ ocrconfig –showbackuploc
```

You can see which backups are available with the following command:

```
$ ocrconfig –showbackup
```

The default location for generating backups on UNIX-based systems is GRID_HOME/cdata/<cluster_name>, where <cluster_name> is the name of your cluster. The Windows-based default location for generating backups uses the same path structure. By default, the second node is selected as the master node, and OCR is backed up and stored in the master node.

Apart from automatic backup, you can also manually back up the OCR using the **ocrconfig** command. This command backs up the OCR without waiting for the scheduled automatic backups:

```
ocrconfig  -manualbackup
```

Oracle Local Registry (OLR) is excluded in the automatic backup mechanism, and manual backup is the only option for OLR. In order to back up the OLR, use the same command as before, but add the **-local** option, as shown here:

```
ocrconfig -local -manualbackup
```

The **ocrconfig -showbackup** command shows the backup location, timestamp, and the originating node name. The command lists both the manual and automatic backups. However, you can query just the manual or the automatic backups by specifying the **-showbackup manual** or **-showbackup auto** options.

```
ocrconfig –showbackup
```

You can change the location where OCR stores the backups by using the following command:

```
$ ocrconfig –backuploc Grid_home/cdata/cluster1
```

If you’re using ASM for storing the OCR files, you can specify the ASM disk group name for the *file_name* parameter in this command, as shown in the next

https://t.me/learningnets

example:

```
$ ocrconfig –backuploc +bkupdg
```

An OCR recovery is warranted when there is a corruption in the OCR or loss of an OCR device. Restarting the resources or removing the resources from and then adding the resources back to OCR should be considered before attempting an OCR restore, because OCR requires downtime for the Oracle Grid Infrastructure. You can use the **ocrcheck** command to check the status of the OCR before attempting the OCR restore.

To recover the OCR, you should stop Oracle Clusterware from all the nodes. This can be done by executing **crsctl stop crs** in all the nodes. If the command fails due to OCR corruption or if it returns any errors, you can use **crsctl stop crs -f** to force the CRS to stop.

Once the CRS daemons are stopped in all the nodes, the following command restores the OCR:

```
ocrconfig -restore <file_name>
```

To restore a local OCR file, just add the **-local** parameter to the previous command. You can restart Oracle Clusterware on all the nodes in your cluster by restarting each node or running the following command:

```
crsctl start crs
```

Once you’re done with the restore, you can check the integrity of OCR across all nodes by running the following **cluvfy** command:

```
$ cluvfy comp ocr -n all -verbose
```

OCR can also be exported and imported. This can be an alternative recovery mechanism for OCR recovery. The following command can be used to export the contents of OCR:

```
ocrconfig -export <file_name>
```

This creates the binary copy of the OCR. This file cannot be edited by text editors. Human-readable information about OCR can be obtained by using the **ocrdump** command. This command dumps the contents of the OCR to a text file in the current directory.

To import the OCR, stop Oracle CRS by executing **crsctl stop crs**. Once the CRS daemon is stopped in all the nodes, you can use the following command to import the OCR:

```
ocrconfig -import <file_name>
```

Optionally, you can run the Cluster Verification utility to verify the integrity of all the cluster nodes configured as a part of your cluster:

https://t.me/learningnets

Technet24

```
$ cluvfy comp ocr -n all -verbose
```

NOTE

The file formats and compatibility of the backup restore operations are totally different in ***ocrconfig -backup*** *and* ***ocrconfig-export****. The first one is a byte-by-byte copy of the contents of the OCR, and the second is the logical copy of the OCR. If you want to recover from manual or automatic backup, you need to use the* ***ocrconfig -restore*** *option. If you want to import the OCR from a previously exported OCR, use the* ***ocrconfig -import*** *option. Extreme care should be taken while recovering the OCR.*

Validating OCR Backups

OCR backups can be validated by the **ocrdump** utility. This utility is normally used to dump the contents of OCR to a text file. The same can be used to verify the contents and structural integrity of the OCR backups:

```
ocrdump -backupfile <backup_file_name>
```

NOTE

It is recommended that you manually back up the OCR from other nodes because the clusterware-initiated automatic OCR backup process backs up the OCR only for the master node. If the master node is down and cannot be rebooted, OCR backups are lost. Hence, it is important to integrate the OCR backup process with the regular backup schedules.

Summary

Oracle RAC backup and recovery processes are the same as single-instance backup and recovery, except for the thread merging and two-pass recovery concepts.

We discussed the concepts and inner workings of Oracle RAC recovery operations. Oracle RAC introduced two-pass recovery to optimize media recovery. Fast reconfiguration helps make instance recovery efficient by distributing the resources from failed nodes to surviving nodes. This also helps to reduce the reconfiguration times while members are joining or leaving the clusters.

We also discussed the backup and recovery operations of the two key components of the clusterware: Oracle Clusterware Repository and the voting disk. These are automatically backed up when clusterware is running. We also discussed the manual backup and restore operation of these critical components.

https://t.me/learningnets

CHAPTER 10

Oracle RAC Performance Management

In this chapter, we look at considerations and procedures for optimizing performance in the Oracle RAC environment, which consists of a database opened by multiple instances. Inter-instance coordination and reliable communication are necessary to preserve data integrity and other database characteristics. This requires CPU, network bandwidth, and memory resources. However, this need for additional system resources should not impact the end user's transaction processing time. Techniques mentioned in this chapter should help you to identify and address user response time problems and optimize performance in an Oracle RAC environment.

Traditionally, you can increase the capacity of a database tier by adding hardware or by replacing the existing system with a higher capacity system. Oracle RAC offers an alternative to this approach by allowing you to scale the database tier horizontally, by adding one or more database servers to the existing machine(s). This also lets you make use of low-cost servers, including those running Linux operating systems, thereby reducing the overall cost of ownership.

Overall, the process for Oracle RAC database instance tuning is similar to that of single-instance database tuning. The same principles of SQL tuning—identifying hot spots and tuning contentions—are applicable to the Oracle RAC environment. Tools used for monitoring Oracle instance and system usage are also the same.

https://t.me/learningnets

Technet24

You also need to consider tuning the cluster interconnects and cache synchronization delays while tuning Oracle RAC.

Because the buffer caching and shared pool components are global in the Oracle RAC environment, another key consideration is that any issue with one instance may potentially affect the other instances. However, the good news is the enhanced wait event model and the extensive framework data collection mechanism at the global cache layer provide a great amount of information on how they are working under the hood.

The Automatic Workload Repository (AWR) and STATSPACK reports contain additional information required for tuning the Oracle RAC instance and database.

Oracle RAC Design Considerations

The best applications, database designs, and coding practices applicable to a single-instance Oracle database are also applicable to Oracle RAC. A well-tuned application in a single-instance environment on a symmetric multiprocessing (SMP) machine should run well in an Oracle RAC environment.

If an application has capacity issues, moving to the Oracle RAC environment will not solve them. At best, it'll provide some temporary relief until the basic design or code issues are fixed; at worst, it could actually degrade performance because Oracle RAC might amplify an existing performance flaw. So, is there anything you can do to make an application run better in an Oracle RAC environment? The following sections address this question.

Oracle Design Best Practices

Let's quickly recall the Oracle design best practices for application and database layout:

- Optimize connection management. Ensure that the middle tier and programs that connect to the database are efficient in connection management and do not log on or off repeatedly. Not retaining database connections has an impact in a single-instance environment. However, the impact is a lot higher in an Oracle RAC environment.
- Ensure that the SQL execution is well tuned. Optimize execution plans and data access paths. Many textbooks are devoted to this subject, and lots of tools are available to tune SQL automatically. Most database performance problems are due to inefficient SQL execution plans. Tools such as Automatic Database Diagnostic and Monitor (ADDM) and SQL Tuning Advisor go a long way in pointing out which SQL statements to optimize, and how.
- Ensure that the application is using bind variables to minimize parsing. The initialization parameter **cursor_sharing** helps quite a bit in this area. Using bind variables is quite critical for online transaction processing (OLTP)

https://t.me/learningnets

applications. For data warehouse applications, using bind variables might not be desirable.

- Use packages and procedures in place of anonymous PL/SQL blocks and big SQL statements. Packages and procedures are compiled and stored in the database and can be reused without runtime compilation overhead.
- If a sequence is being used frequently and by multiple concurrent sessions, ensure that you configure a large cache for the sequence. The default value for a cache sequence is 20.
- Avoid use of Data Definition Language (DDL) statements in a production environment that operates during normal business hours. Use of DDLs increases invalidations of the already parsed SQL statements, and they need to be recompiled before reuse. DDLs also increase the invalidations of the buffers in the cache, which is another source of performance issues.
- Index leaf block contention is the single largest cause of buffer busy waits in a single-instance environment and can cause buffer cache contention (gc buffer busy) in an Oracle RAC environment. Indexes are useful because they help you retrieve data faster in most cases. However, they become a drag on Data Manipulation Language (DML) statements because the database needs to update the indexes during insert, update, and delete operations. Consider using reverse key index and index-only tables. The use of effective partitioning tables and indexes is another effective way to minimize contention.
- Optimize contention on data blocks by avoiding small tables with too many rows in a block. You can use the **minimize_records_per_block** clause of the **alter table** command to restrict the number of rows per block.

Oracle RAC–Specific Design Best Practices

Following are some application design considerations that can help you optimize performance specifically in an Oracle RAC environment:

- Do not reread the recently modified/inserted data just to validate it. There is no need to read and compare the data to confirm what has been inserted or committed. Built-in checks and balances help ensure data integrity and avoid corruptions.
- If business requirements do not dictate otherwise, consider using application partitioning. We discuss this in detail in the following section.
- Consider restricting DML-intensive users to just one in the cluster. If a few users are performing DML operations on certain sets of data, allow those users to access the same instance; although not necessary, this strategy helps in reducing cache contention.
- Group read-only data together and put it in a read-only tablespace (or tablespaces). Read-only tablespaces require little resource and lock

https://t.me/learningnets

Technet24

coordination. Keeping read-only and read-intensive data separate from DML-intensive data will help optimize Oracle RAC Cache Fusion performance.

- Avoid unnecessary auditing in an Oracle RAC environment. Auditing can result in a lot of negative side effects because it creates more shared library cache locks. Also, if required, implement auditing at the application layer.
- Use full table scans sparingly because they cause the Global Cache Service to service lots of block requests. Tuning the inefficient SQL queries and collecting system statistics for the optimizer to use CPU costing will also help. The "table scans (long tables)" statistics in V$SYSSTAT provide the number of full table scans done by the instance.
- If the application makes a large number of new session "logon storms," increase the cache value of the **sys.audses$** sequence. Here's an example:

```
alter sequence sys.audses$ cache 10000;
```

Partitioning the Workload

In workload partitioning, the database executes a certain type of workload on an instance—that is, partitioning allows users who access the same set of data to log on from the same instance. This limits the amount of data shared among the instances and therefore saves resources used for messaging and Cache Fusion data block transfer.

To partition or not to partition? The decision is a fine balance between business needs and system resources conservation. You can save system resources by implementing workload partitioning in an Oracle RAC environment. If the system's processing resources are being used to their full capacity, partitioning might help. However, if the system has enough spare resources, partitioning probably isn't necessary.

Consider the following before implementing workload partitioning:

- Oracle RAC can handle buffer cache contention, and if the system CPU resources and private interconnect bandwidth are sufficient, you do not need to partition.
- At times, it might be much easier and more economical to add extra CPU power or interconnect bandwidth rather than implementing a workload partitioning scheme.
- You don't need to consider partitioning the user workload unless evidence indicates that contention is impacting performance and partitioning the workload will improve performance. Chapter 14 discusses this topic in greater detail.
- Establish baseline statistics with and without workload partitioning. If the comparison shows significant performance improvements with partitioning, consider implementing workload partitioning.

https://t.me/learningnets

Scalability and Performance

Scalability and high availability are the two most significant advantages that Oracle RAC offers. A system is said to be “fully scalable” if its response time remains unchanged, regardless of how many users it serves.

Consider a real-life example: If the freeways are 100 percent scalable, travel time from point A to point B will always remain the same, whether one car is on the highway or millions of cars. Are highways fully scalable? Probably not!

A four-lane freeway with properly designed exit and entrance ramps might support hundreds or even thousands of cars, but if the number of cars increases beyond a limit, the drivers will have to slow down to yield and to avoid accidents. A freeway that is scalable for thousands of cars and allows them to travel at a speed of 65 mph may not be able to support the same number of trucks at that speed. Therefore, the scalability of a system not only depends on the number of users but also on the type of workload. It is reasonable to expect that the same freeway cannot support 10,000 mixed vehicles (cars, trucks, bikes, and so on).

Here are some points to note:

- No real-life system is fully scalable all the time.
- Every system has a limit beyond which it may not be able to support additional users without compromising the response time.
- A system that is scalable for a certain number of users of some type may not be able to support the same number of a different type of user.

However, in practice, a system is assumed to be fully scalable as long as its response time does not degrade (or remains within service-level agreements) for an intended maximum number of users. Regarding Oracle RAC, scalability is generally discussed in terms of the ability to support additional user workload when a node, identical to the existing node, is added. For example, consider the following data:

3000 = the number of user transactions per second processed with three nodes

4000 = the number of user transactions per second processed with four nodes

Scalability = ((4000 – 3000) / 3000 / 3) × 100 = 100%

Here are the assumptions for this example:

- All nodes have the same hardware configuration.
- Response time and the user workload characteristics remain unchanged. Only the number of user transactions is increased.

A real-life system consists of multiple components. For a system to be scalable, each of its components needs to support the increased user workload—that is, all its components need to be scalable. For an Oracle RAC system to be fully scalable, all the following components need to be scalable:

https://t.me/learningnets

Technet24

- Application
- Network
- Database
- Operating system
- Hardware

If any of the components cannot perform for the intended number of users, the system will not be scalable.

A specific application's scalability may vary depending on several factors. In general, the scalability of an application is built during the design process by avoiding any point of serialization. An application's scalability in the Oracle RAC environment might be limited if the same data blocks are frequently needed by all the instances, thereby increasing the Cache Fusion activity. For example, if each transaction is updating a running total maintained in a row, the block containing the row needs to be updated by each transaction, thereby causing the block to travel among the instances very rapidly. This behavior would limit the application's scalability.

Choosing the Block Size for an Oracle RAC Database

Most operating systems have a buffer cache that keeps frequently used data in memory to minimize disk access. To optimize I/O efficiency, the first choice is to make the Oracle block size equal to the file system buffer size. If that is not acceptable, the Oracle block size should be set to a multiple of the OS file system cache block size.

For example, if the file system buffer is 4K, then the preferred Oracle block sizes will be 4K, 8K, 16K, and so on.

If the OS buffer cache is not being used or the OS being used does not have a buffer cache (as in Windows), you can choose any block size without compromising the I/O efficiency. You achieve this by using a shared file system, or Automatic Storage Management (ASM). Therefore, you don't need to consider the file system buffer size while choosing a block size for Oracle RAC.

Oracle supports multiple block sizes inside a database; therefore, different tablespaces can have different block sizes. The parameter **db_block_size** represents the standard block size for the database and is used by the system tablespace, temporary tablespace, auxiliary tablespace, and any other tablespace for which a different block size is not specified while it is being created.

Most of the applications use the 8K block size for the database. If you are designing a decision support system (DSS)/data warehouse system, consider larger block sizes. For data that is mostly read-only or retrieved using full table scans, a large block size is helpful.

A large block size is also helpful when row size is larger—for example, while using large objects (LOBs) to store documents, images, and so on. Using a small block size

https://t.me/learningnets

for an index is sometimes helpful because it reduces contention on index branch blocks. Similarly, for tables with smaller row sizes, using smaller block sizes will reduce contention.

Wait events—buffer busy waits, global cache busy, buffer busy global cr, and so on—are indications of block-level contention, and moving the index/table with high contention to a tablespace with a smaller block size will help reduce contention.

Introduction to the V$ and GV$ Views

Dynamic performance views (V$ views) contain database statistics, and you commonly use them for performance analysis and tuning. In the Oracle RAC environment, a global (GV$) view corresponds to each V$ view. V$ views contain statistics for one instance, whereas GV$ views contain information from all the active instances. Each GV$ view contains an INST_ID column of type NUMBER that identifies the instance associated with the row data.

Parallel Query Slaves

When you issue a query against a GV$ view, the instance gets statistics from remote instances using the parallel query mechanism. The database dedicates special parallel query slaves to this task. The tasks are named pz98, pz99, and so on, to distinguish them from general parallel query slaves. Here's an example:

```
$ ps -ef | grep PROD
oracle    6810     1  0 Jun07 ?        00:00:59 ora_pmon_PROD2
oracle    6812     1  0 Jun07 ?        00:00:00 ora_diag_PROD2
oracle    6824     1  0 Jun07 ?        00:01:17 ora_dbw0_PROD2
<<<unwanted lines deleted from here>>>
oracle    7168     1  0 Jun07 ?        00:01:41 ora_pz99_PROD2
oracle   25130     1  0 Jun29 ?        00:00:07 ora_pz98_PROD2
```

Most of the performance-tuning tools (including Oracle Enterprise Manager, AWR, and ADDM) and scripts make use of the GV$ views.

V$ Views Containing Cache Fusion Statistics

Generally, an AWR report should suffice to analyze the performance in an Oracle RAC environment. An AWR report includes seven or eight Oracle RAC–specific segments (described later in this chapter) and provides statistical information about Cache Fusion performance. Refer to Appendix A for a discussion of some of the important V$ views that can be used to manage performance in an Oracle RAC environment.

Oracle RAC Wait Events

https://t.me/learningnets

Technet24

An *event* can be defined as an operation or a particular function that the Oracle kernel performs on behalf of the user session or its own background process.

Tasks such as writing and reading data blocks to and from datafiles, receiving data from other instances' memory, and waiting for the permission to read and acquire the block from the resource master are known as *database events*, and they have specific names. But why are these events called *wait events?*

All sessions accessing Oracle database instances need resources to perform their tasks concurrently and independently. A resource may be a data buffer, a latch, an enqueue (lock), or a piece of code to be executed exclusively.

Some of these resources are serialized by access, and at any point in time only one process can have exclusive access; the others wanting the resource must wait in the queue. Whenever a session has to wait for something, the wait time is tracked and charged to the event that is associated with that wait.

For example, a session that needs a block that's not in the current instance's cache makes a read call to the operating system to be delivered from another instance's cache and waits for the block. The wait time is charged to the waiting session's account.

If the block is read from the disk, the wait time is charged to the requesting instance's "db file sequential read" wait event. If the block is coming from the other instance's cache, the wait time is charged against "global cache" wait events.

Another session may have completed the last instruction and is now waiting for user input. This is generally termed an *idle wait*, and the wait time is charged to the "SQL*Net message from client" event. In short, when a session is not using the CPU, it may be waiting for a resource, for an action to complete, or simply for more work. Events that are associated with all such waits are known as *wait events*.

You can classify wait events by type—Cluster, User I/O, System I/O, and Network, for example. Table 10-1 lists different wait classes and the number of wait events in each class. Even though more than a thousand wait events are available in Oracle Database 12*c* R2, ranging over 13 categories, you would normally deal with fewer than 50 events while analyzing Oracle RAC performance.

https://t.me/learningnets

Wait Class	Number of Wait Events in the Class
Commit	2
Scheduler	7
Queuing	9
Application	17
Configuration	24
System I/O	30
Concurrency	32
Network	35
User I/O	45
Cluster	50
Administrative	54
Idle	94
Other	719

TABLE 10-1. *Wait Classes in Oracle Database 12cR2*

Understanding Cluster Waits

When a session requests access to a data block in CR/CUR (Consistent Read/Current) mode, it sends a request to the lockmaster for proper authorization. However, whether the session will receive the block via the Cache Fusion mechanism or a permission to read from the disk is not known until the request is satisfied.

Two placeholder events—"gc cr request" and "gc current request"—keep track of the time a session spends in this state. For example, in a two-node cluster, when a session sends a request for a CR block, its wait time is counted against "gc cr request." Let's assume that the lockmaster grants the lock to the requestor, thereby authorizing it to read the block from the disk so all the wait times are now logged against the wait event "gc cr grant 2-way."

Similarly, the wait time would be logged against "gc cr block 2-way" if the requesting instance receives the CR copy of the block from the master. In the end, there should not be any wait times logged against these two placeholder wait events.

Two-Way and Three-Way Wait Events

The Oracle RAC Database tracks whether a lock request was satisfied with just two-way communication between requestor and master/holder. In two-way events, the master instance is also the current holder of that block and is able to send the needed

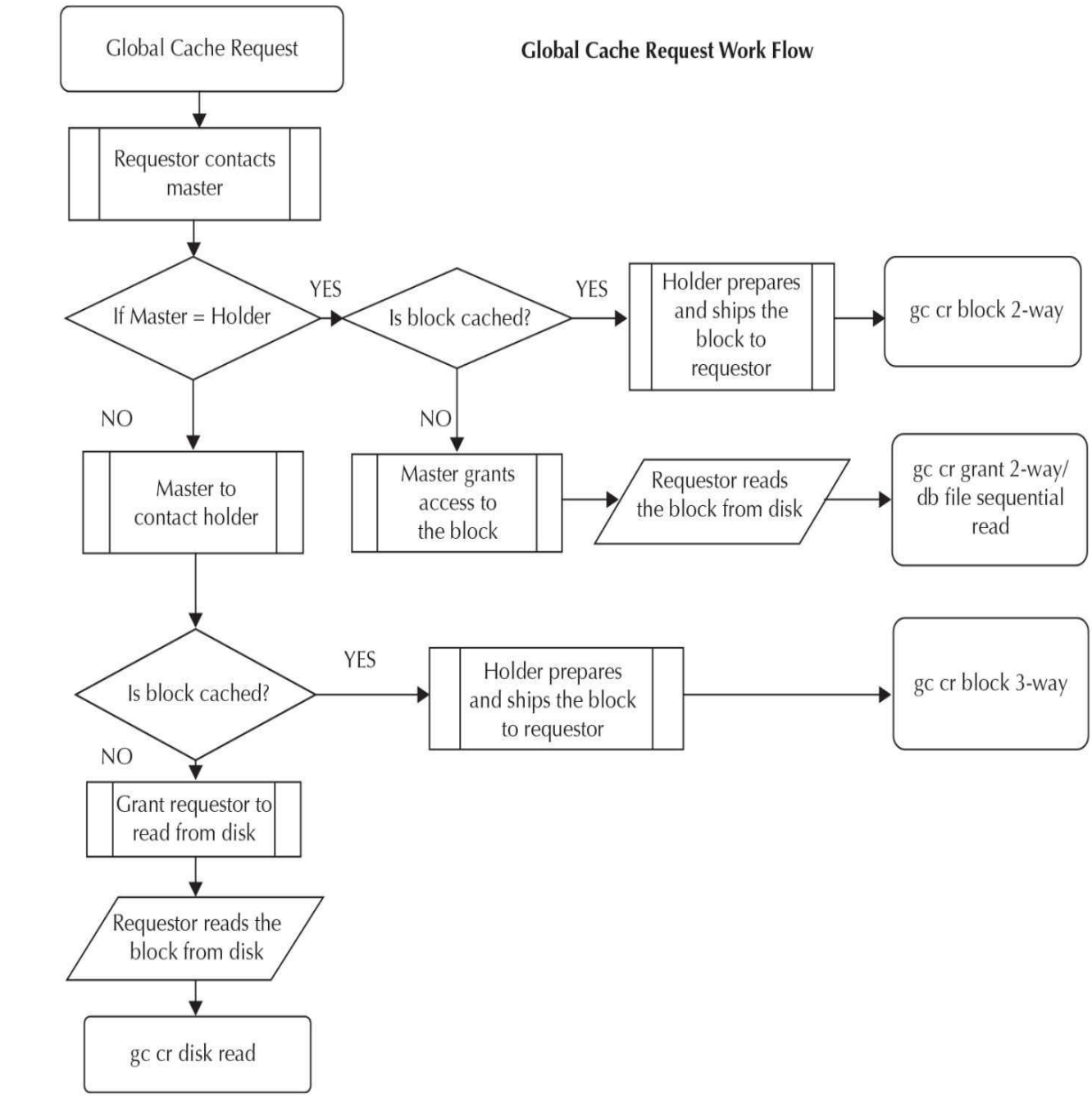
https://t.me/learningnets

Technet24

CR/CUR copy of the block via Cache Fusion to the requestor. This is the typical case in two-node cluster environments.

In the case of three-way events, the master forwards the request to the holder. The holder in turn sends the CR/CUR copy of the block as desired by the requestor. Three-way waits and messages are seen in clusters with more than two nodes. Any kind of messaging and block transfer will be completed in at most three hops, irrespective of the number of nodes in the cluster.

Figure 10-1 explains the typical processing of a global cache wait and its potential outcome based on the status of the block in any other instance's cache. Depending on the caching status/availability, the outcome of the wait will be a block transfer or message grant.



https://t.me/learningnets

FIGURE 10-1. *Global cache processing workflow*

The cluster wait events belong to one of the following categories:

- Block-oriented waits
- Message-oriented waits
- Contention-oriented waits
- Load-oriented waits

Block-Oriented Waits

Block-oriented waits are the most common wait events in the cluster wait events. The block-oriented wait event statistics indicate that the requested block was served from the other instance. In a two-node cluster environment, a message is transferred to the current holder of the block, and the holder ships the block to the requestor. In a cluster environment with more than two nodes, the request for the block is sent to the holder of the block through the resource master and includes an additional message.

These events are usually the most frequent in the absence of block contention, and the length of the wait is determined by the time it takes on the physical network, the time to process the request in the serving instances, and the time it takes for the requesting process to wake up after the block arrives.

You should consider the average wait time and the total wait time when being alerted to performance issues where these particular waits have a high impact. Usually, either interconnect or load issues or SQL execution against a large shared working set can be found to be the root cause.

The following are the most common block-oriented waits:

- gc current block 2-way
- gc current block 3-way
- gc cr block 2-way
- gc cr block 3-way

Message-Oriented Waits

The message-oriented wait event statistics indicate that no block was received because it was not cached in any instance. Instead, a global grant was given, allowing the requesting instance to read the block from disk or to modify it. These waits are normally followed with a disk read event such as “db file sequential read” or “db file scattered read.”

If the time consumed by these events is high, it may be assumed that the frequently executed SQL causes a lot of disk I/O (in the event of a “cr grant”) or that the workload inserts a lot of data and needs to find and format new blocks frequently (in the event of a “current grant”).

https://t.me/learningnets

Technet24

The following are the most common message-oriented waits:

- gc current grant 2-way
- gc current grant 3-way
- gc cr grant 2-way
- gc cr grant 3-way

Contention-Oriented Waits

The contention-oriented wait event statistics indicate that a block was received that was pinned by a session on another node, was deferred because a change had not yet been flushed to disk, or was deferred because of high concurrency and therefore could not be shipped immediately. This typically happens with concurrent DMLs on index blocks where the other instance is waiting for the current block to add the index entry while the index block is busy in the holder. In this case, the contention-oriented wait "gc current block busy" will be followed by the "gc current split" or "gc buffer busy" event.

A buffer may also be busy locally when a session has already initiated a Cache Fusion operation and is waiting for its completion when another session on the same node is trying to read or modify the same data. These waits can occur when a foreground process is waiting for an antilock broadcast to update a read-mostly object. Read-mostly locks and reader-bypass optimizations are discussed in the next chapter.

High service times for blocks exchanged in the global cache may exacerbate the contention, which can be caused by frequent concurrent read and write accesses to the same data.

The following are the most common contention-oriented waits:

- gc current block busy
- gc cr block busy
- gc current buffer busy

The wait event "block busy" suggests that a *single* session required access to a specific block that is deemed "busy" on a remote node (for various reasons stated earlier). A "buffer busy" indicates that *multiple* sessions are waiting for a block that is deemed "busy" on a remote node. In other words, dealing with a "buffer busy" situation involves multiple sessions contending for the same block and requires tuning that is similar to dealing with "buffer busy waits" or "read by other sessions," albeit in an Oracle RAC environment.

Load-Oriented Waits

The load-oriented wait events indicate that a delay in processing has occurred in the GCS, which is usually caused by high load or by CPU saturation and would have to be solved by either adding CPUs, using load balancing, or offloading processing at

https://t.me/learningnets

different times or to a new cluster node. For the events mentioned, the wait time encompasses the entire round trip from the time a session starts to wait after initiating a block request until the block arrives.

Whenever a request for any block transfer or message transfer waits for more than 1 ms in the internal queue after the inter-process communication (IPC) layer, the system is considered "overloaded." This could typically happen when an LMS process (the Global Cache Service Process, which maintains a lock database for Global Cache Services and buffer cache resources) is not able to catch up with the arrival rate and the messages are queued for LMS processing.

You should seldom see the congested waits in a well-balanced cluster environment. The following are the most common load-oriented waits; the existence of these waits indicates that the system is overloaded or underpowered:

- gc current block congested
- gc cr block congested
- gc current grant congested
- gc cr grant congested

We'll describe the important Oracle RAC wait events and the possible action plan when a wait event appears in the "Top 5 Timed Wait Events" section of the AWR report. We'll also look at messages and data flow during some of the important global cache wait events.

gc current block 2-way As indicated in Figure 10-2, an instance requests authorization for a block to be accessed in current mode to modify the block. The instance mastering the corresponding resource receives the request. The master has the current version of the block and sends the current copy of the block to the requestor via the Cache Fusion mechanism. This event indicates write/write contention.

https://t.me/learningnets

Technet24

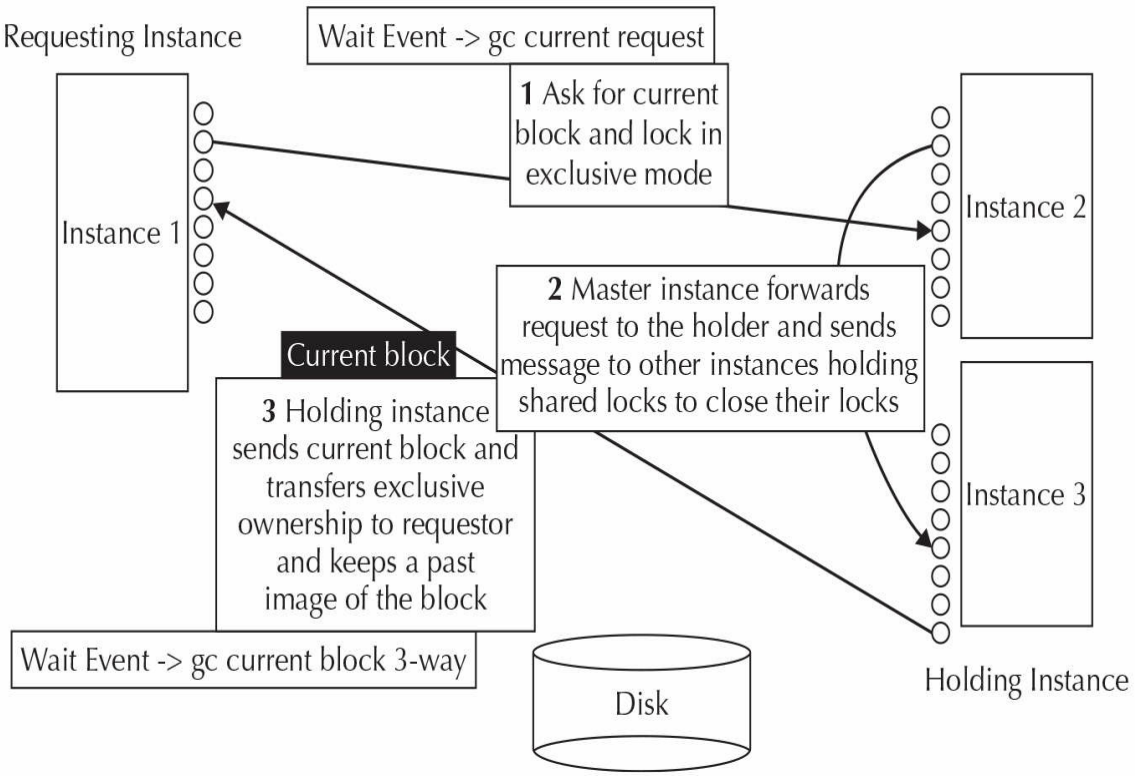


FIGURE 10-2. *Wait event: gc current block 2-way*

NOTE

The appearance of "gc current block 2-way" and the following three events in the "Top 5" event section doesn't necessarily indicate performance issues. It merely indicates that the instances are accessing copies of data present in each other's cache and that the Cache Fusion mechanism is being used to transfer copies of the data among the instances. However, if the average wait time for each event is very high, it might be impacting performance and need further analysis.

If this wait event appears in the "Top 5" timed events, do the following:

- Analyze the contention. Segments in the "Current Blocks Received" section of the AWR report should help you identify the top contentious objects.
- Ensure that good database object design practices, database object layout, and space management practices are being followed for the top contentious objects.
- Optimize the application contention by using an appropriate application-partitioning scheme.
- Look at "Tips for Avoiding Oracle RAC Wait Events," later in this chapter.

gc current block 3-way Figure 10-3 shows this wait event. An instance requests

https://t.me/learningnets

authorization for a block to be accessed in current mode. The instance mastering the corresponding resource receives the request and forwards the message to the current holder, telling it to relinquish ownership. The holding instance sends a copy of the current version of the block to the requestor via the Cache Fusion mechanism and transfers the exclusive lock to the requesting instance. This event indicates write/write contention.

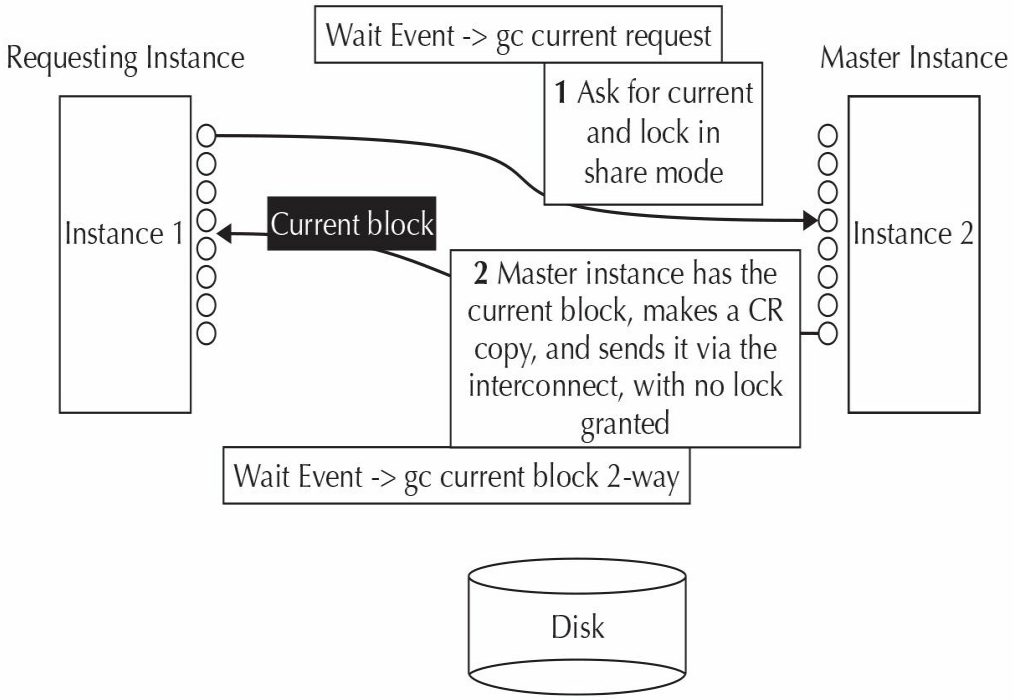


FIGURE 10-3. *Wait event: gc current block 3-way*

If this wait event appears in the "Top 5" timed events list, your plan of action should be similar to that for a "gc current block 2-way" event.

gc cr block 2-way Figure 10-4 shows a pictorial representation of the event. An instance requests authorization for a block to be accessed in CR mode. The instance mastering the corresponding resource receives the request. The master has the current version of the block. It makes a CR copy using the current block and undo data it has and sends the CR copy of the block to the requestor via the interconnect. This event indicates write/read contention.

https://t.me/learningnets
Technet24

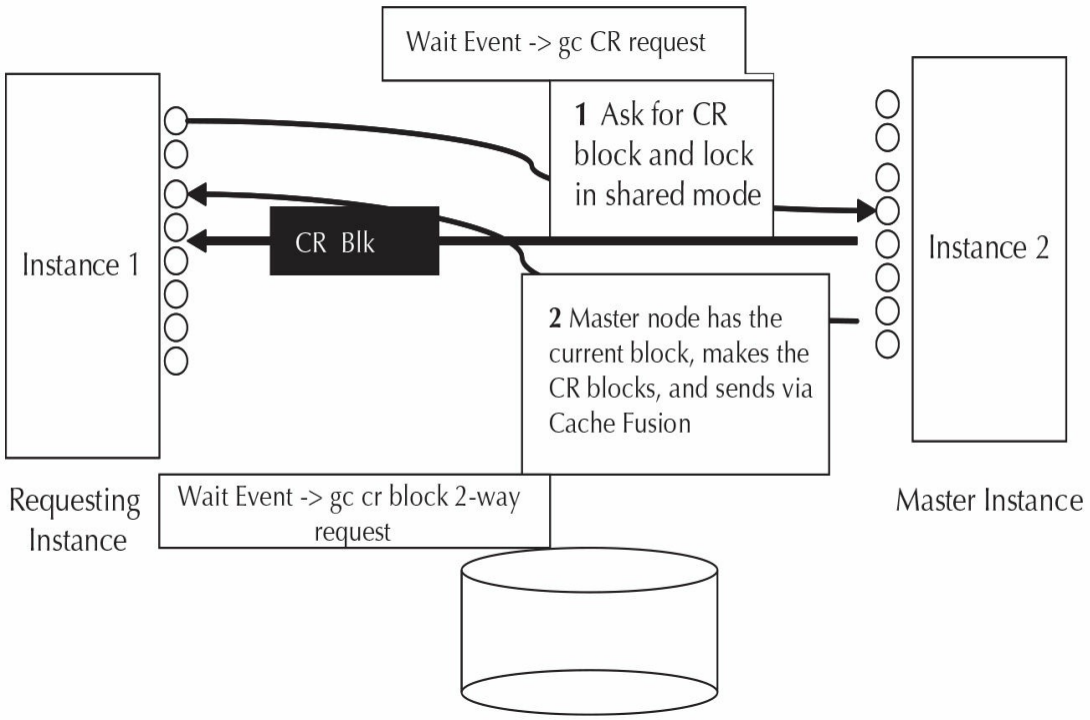


FIGURE 10-4. *Wait event: gc cr block 2-way*

If this wait event appears in the "Top 5" timed events, then do the following:

- Analyze the contention. The "Segments by CR Blocks Received" section of the AWR report should help you to identify the top contentious objects.
- Optimize the application contention by using an appropriate application-partitioning scheme.
- Look at the "Tips for Avoiding Oracle RAC Wait Events" section, later in this chapter.

gc cr block 3-way Figure 10-5 shows a pictorial representation of this event. An instance requests authorization for a block to be accessed in CR mode. The instance mastering the corresponding resource receives the request. The master forwards the request to the current holder of the block. The holding instance sends a CR copy of the block to the requestor via the Cache Fusion mechanism. This event indicates write/read contention.

https://t.me/learningnets

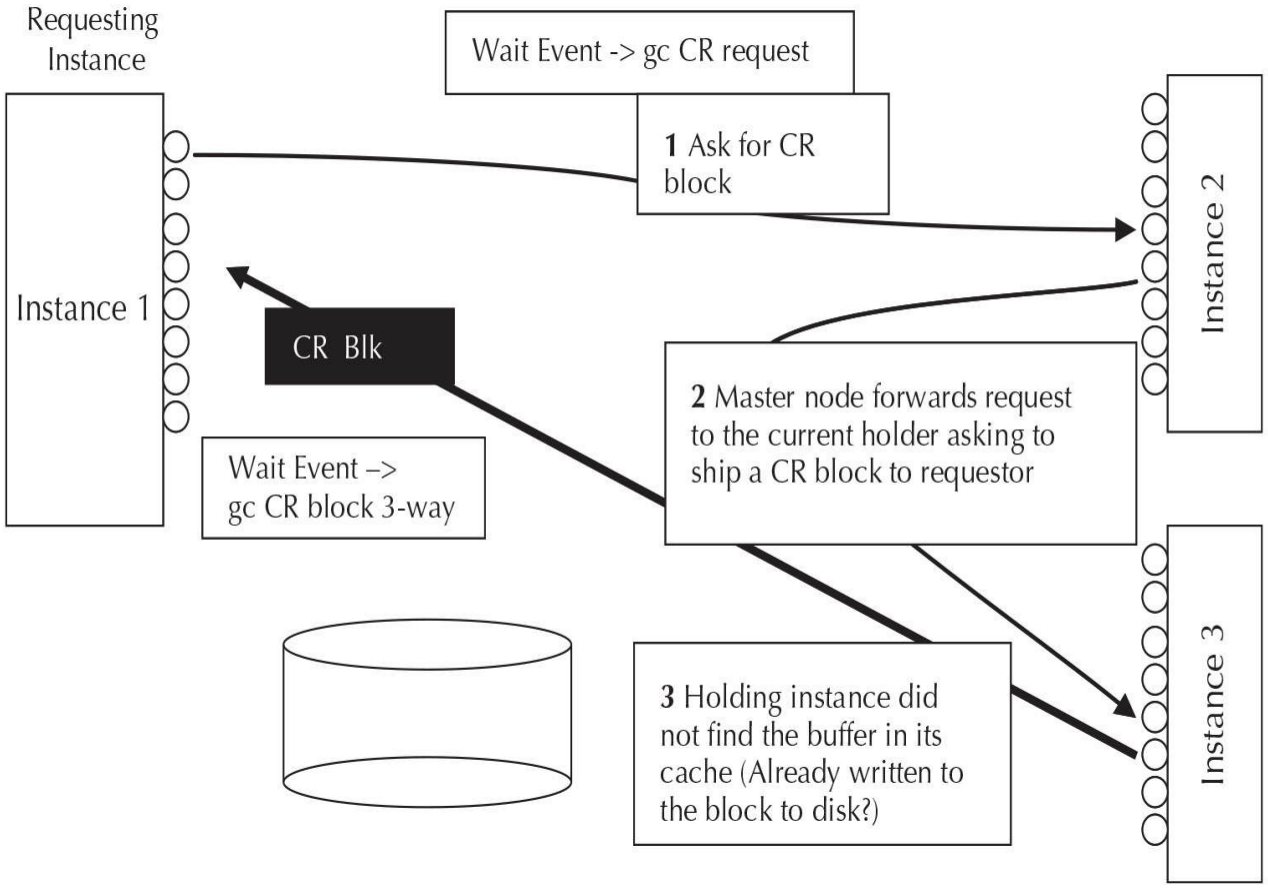


FIGURE 10-5. *Wait event: gc cr block 3-way*

If this wait event appears in the "Top 5" timed events list, the plan of action should be similar to that for the "gc current block 2-way" event.

gc current grant 2-way Figure 10-6 shows a pictorial representation of this event. When an instance needs a block in current mode, it sends the request to the master instance. The master instance finds that currently no instances, including self, have any lock on the requested block. It sends a message back to the requesting instance, granting it the shared lock on the block. The requesting instance then reads the block from the disk. This event doesn't indicate any contention.

https://t.me/learningnets

Technet24

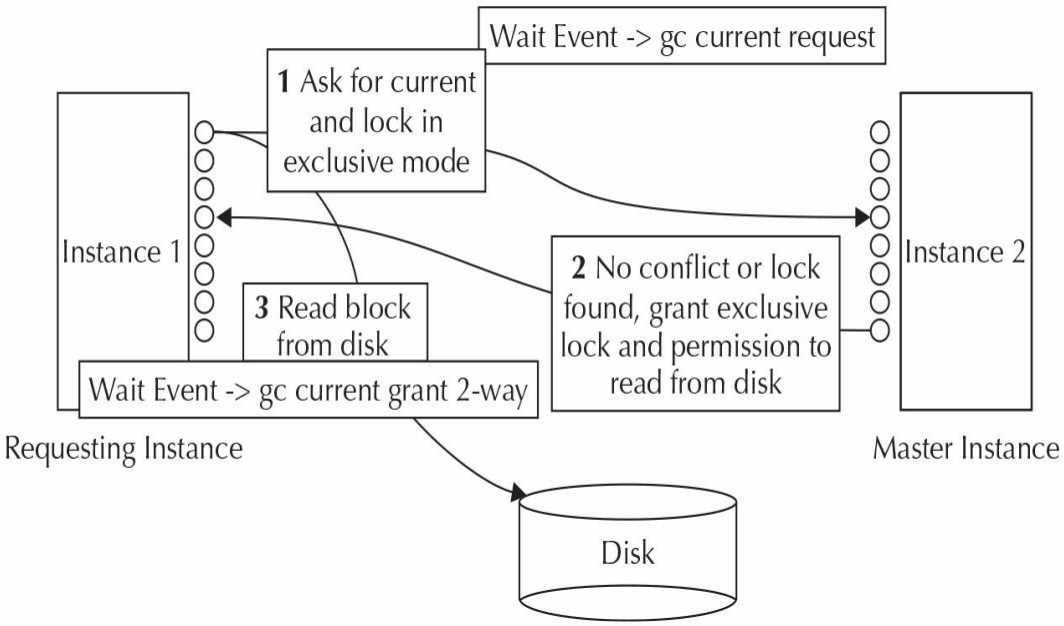


FIGURE 10-6. *Wait event: gc current grant 2-way*

The presence of this event in the "Top 5" list indicates that the instance is spending a significant amount of time in obtaining the locks. Here's a list of what you should do:

- Tune the SQL to optimize the number of blocks accessed by the application, thereby reducing the number of blocks it is requesting.
- Look at the "Tips for Avoiding Oracle RAC Wait Events" section, later in this chapter.

gc current block busy When a request needs a block in current mode, it sends a request to the master instance. The requestor eventually gets the block via Cache Fusion transfer; however, the block transfer is delayed due to either of the following reasons:

- The block was being used by a session on another instance.
- The block transfer was delayed because the holding instance could not write the corresponding redo record to the online redo log file immediately.

You can use the session-level dynamic performance views V$SESSION and V$SESSION_EVENT to find the programs or sessions causing the most waits on this event:

https://t.me/learningnets

```
select e.sid, e.time_waited, s.program, s.module
from v$session_event e, v$session s
where s.sid=e.sid
and e.event='gc current block busy'
order by e.time_waited;
```

To drill down further, you can use the following query to find out the module- or program-level details:

```
select st.sid, st.value, s.program, s.module
from v$sesstat st, v$session s, v$statname n
where s.sid=st.sid
and st.statistic#=n.statistic#
and n.name = 'gc CPU used by this session'
order by st.value;
```

You should take care while interpreting the results of this query because the longer a session has been running, the larger the values are for total wait times and statistics. To eliminate this potential fallacy, you should run the query a couple of times, spaced by a few seconds, and compare the difference (that is, not the absolute values). After identifying some candidate sessions for causing performance problems, your analysis should focus on all wait events and statistics for those sessions.

This event indicates significant write/write contention. If the event appears in the "Top 5" list of the AWR report, do the following:

- Ensure that the Log Writer (LGWR) is tuned. Refer to the "Oracle Database Performance Tuning Guide" for information about tuning the LGWR performance.
- Use appropriate application partitioning to avoid contention.
- Look at the "Tips for Avoiding Oracle RAC Wait Events" section, later in this chapter.

gc cr disk read Figure 10-7 shows a pictorial representation of the "gc cr disk read" event. This wait event is returned when an instance receives a request to send a CR buffer but the buffer is not cached locally. The requester waits on the gc cr request, and when the read disk status is returned, the requester reads the block requested from disk. This typically happens during long-running queries that do not find undo blocks in the local cache of the instance that owns the undo segment, or in any other cache in the global cache.

https://t.me/learningnets

Technet24

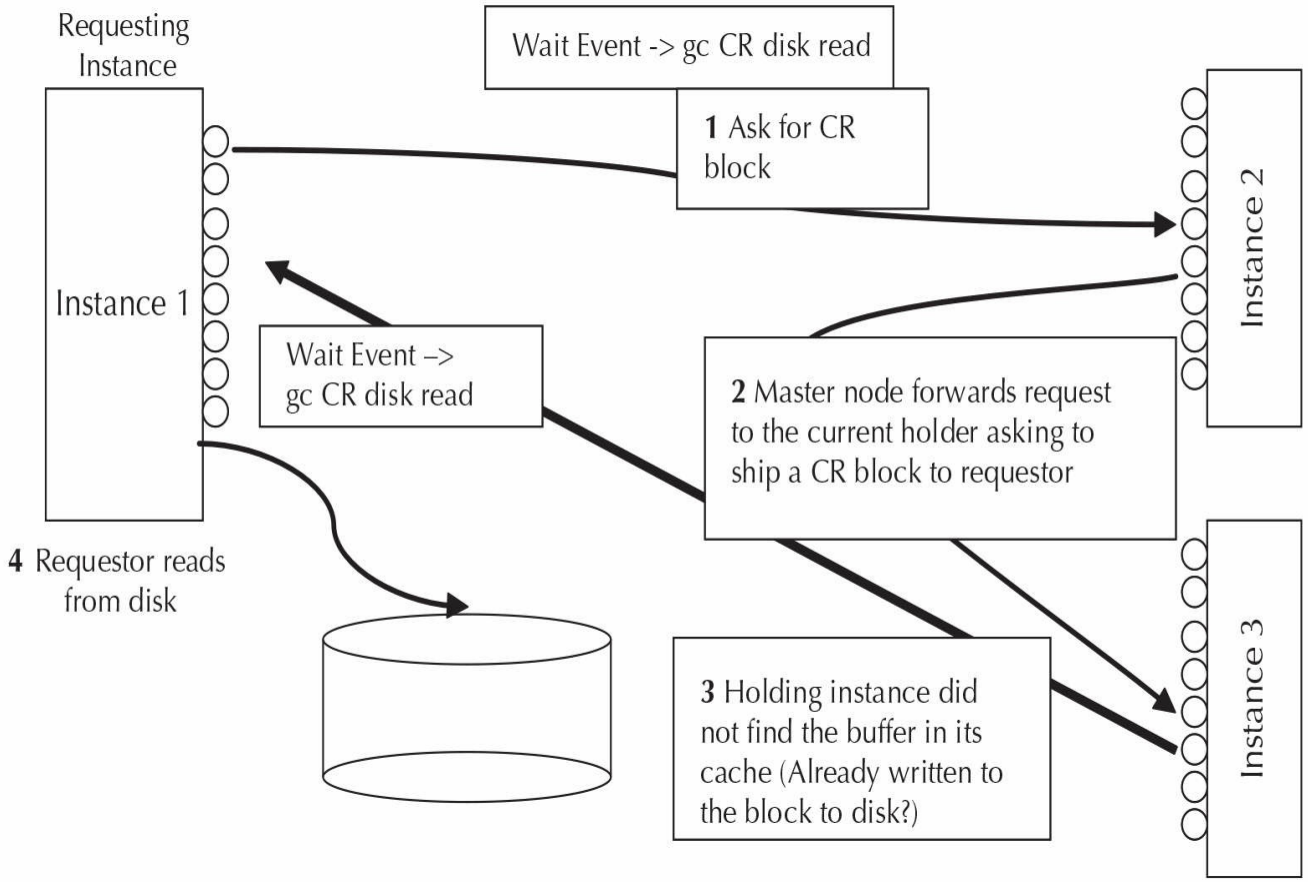


FIGURE 10-7. *Wait event: gc cr disk read*

We get this event when the node we hoped would be able to serve the CR block did not have the block in the cache—perhaps because it was still reading the block from disk or because the block was just pushed out of cache. However, this is a very rare occurrence and not a common performance issue.

To get further information about this event, you can set the 10708 event to level 5. It should dump the lock element covering the resource and other information, so you may be able to find which objects are causing this event as well as how you ended up hitting this event. It is usually recommended that you set the event at the process level or session level rather than the instance level because this tracing would produce a huge amount of trace files.

buffer busy global cr When an instance needs to generate a CR version of the current block, the block can be in either the local cache or a remote cache. If the latter, then LMS on the other instance will try to create the CR block; when the former, the foreground process executing the query will perform the CR block generation. When a CR version is created, the instance (or instances) needs to read the transaction table and undo blocks from the rollback/undo segment that are referenced in the active transaction table of the block. The details of the CR block fabrication and the role of transaction tables are discussed in the next chapter.

Sometimes this cleanout/rollback process may cause several lookups of local and

https://t.me/learningnets

remote undo headers and undo blocks. The remote undo header and undo block lookups will result in a “global cache cr request.” Because undo headers are frequently accessed, a buffer wait may also occur. Until the CR image can be generated, the process will wait for the event “buffer busy global cr.”

gc cr failure Failure is not an option in cluster communications because lost messages or blocks may potentially trigger node evictions. This wait event is triggered when a CR block is requested from the holder of the block and a failure status message is received from the requestor.

This happens only when there’s an unforeseen event, such as a lost block or checksum or an invalid block request, or when the holder cannot process the request. You’ll see multiple timeouts for the placeholder wait (such as gc cr request) before receiving the “gc cr failure” event.

This is also one of the events you should very rarely notice in the production environment. And if you do see them, they warrant immediate attention on your part. You can query the system statistics view V$SYSSTAT for “gc blocks lost” or “gc claim blocks lost.”

The most common reasons for the block failures at the global cache layer can be any of the following; however, the causes for the blocks lost are not known to Oracle. You would need to investigate the hardware and network layers (using **netstat** or OS logs, for example) to find clues.

- Lost blocks during transfer
- Checksum errors
- Invalid format or SCN in the block

If you frequently notice the “gc blocks lost” wait event, you should engage the system administrator and network administrator to investigate the hardware and network layers. The following discussion lists some potential problem areas to consider while investigating this issue.

Interconnect-Related Issues

Oracle Global Cache and Global Enqueue performance and stability are greatly dependent on the interconnect for synchronization. Any potential delay in communication or disruption will affect the stability of the cluster.

Large User Datagram Protocol (UDP) datagrams may be fragmented and sent in multiple frames based on maximum transmission unit (MTU) size. These fragmented packets need to be reassembled on the receiving node. High CPU utilization (sustained or frequent spikes) as well as inadequate reassembly of buffers and UDP buffer space can cause packet reassembly failures. Fragmented packets have a time-to-live value for reassembly. Packets that are not reassembled are dropped and requested again. If there is no buffer space for arriving fragments, they are silently dropped.

Also look for misconfiguration in the bonding driver or jumbo frames. While using

https://t.me/learningnets

Technet24

jumbo frames, you should configure the entire I/O path (NIC cards, switches, and routers) for an MTU of 9000. Also, if there are mismatches in the interconnect communication path or a duplex mode mismatch in the NIC cards, you will find packets lost.

If the CPU is saturated at the host level, you might see the delays in network transfer latencies. Similarly, if the interconnect is saturated, you will notice dropped packets. If the interconnect is running at full capacity, consider upgrading the interconnect to a higher bandwidth (for example, from GigE to 10GigE or InfiniBand). If the hardware does not support higher capacity interconnect, consider bonding the NIC cards for additional capacity.

The **netstat -s** output for the IP layer will look like the following, and you must pay close attention to the "packets dropped" or "packet assembly failed" statistic. Also, check the firmware updates for network interface cards (NICs), although diagnosis of this layer requires OS/hardware vendor intervention.

```
Ip:
    26290871 total packets received
    12 with invalid addresses
    0 forwarded
    0 incoming packets discarded
    19983565 incoming packets delivered
    18952689 requests sent out
    8 fragments dropped after timeout
    6924738 reassemblies required
    766457 packets reassembled ok
    8 packet reassembles failed
    522626 fragments received ok
    3965698 fragments created
```

NOTE

On some platforms, the extended statistics for the interconnect are available from the internal view X$KSXPIF. This view contains the interface statistics, such as send_errors, receive_errors, packets_dropped, and frame_errors at the interface level. You can use this information in analyzing network errors.

gc cr block busy This event is the same as a "gc current block busy" event, except in this case the requesting instance has requested the block in CR mode.

gc current buffer busy This event is also similar to the "gc current block busy" event. In this case, the reason the session does not wait isn't because the transfer from another instance was delayed but rather because another session on the same instance has already initiated the lock request and is waiting for the response from the master

https://t.me/learningnets

instance. Thus, multiple sessions on the local instance are accessing the same block simultaneously. This event indicates local contention for the block.

If this wait event appears in the “Top 5” list in the AWR report, tune the SQL statements to optimize the number of blocks touched by the application, thereby reducing the number of blocks being requested.

gc current block congested When an instance needs a block in current mode, it sends the request to the master instance. The requestor gets the block via Cache Fusion; however, the block-transfer process is delayed due to heavy workload on Cluster Group Services (GCS). As mentioned earlier, if any of the message or block transfers wait for more than 1 ms in the IPC queue before being picked up by the LMS process, you will see a congested wait.

This event doesn’t indicate any concurrency or contention at the block level. However, it does indicate that GCS is heavily loaded and the background processes involved with its work need more CPU time. Lack of CPU resources on the holding instance might cause this to happen. Adding more LMS processes also sometimes helps to overcome this situation.

If this wait event appears in the “Top 5” list of the AWR report, consult the upcoming “Tips for Avoiding Oracle RAC Wait Events” section.

gc cr block congested This event is the same as the “gc current block congested” event, except that in this case the requesting instance has requested the block in CR mode.

Tips for Avoiding Oracle RAC Wait Events

Following are some generic quick checks for avoiding excessive waits on the global cache:

- Ensure that the system has enough CPU resources. Monitor the average run queue length to make sure it stays below 1 in order to avoid CPU starvation for critical Oracle RAC–related processes such as LMS.
- Interconnect network transfer speed and bandwidth should not impact the Cache Fusion data transfer. Have a look at the AWR report, including how much network traffic is generated by each instance. Ensure that the overall network traffic is well below the bandwidth of the private interconnect.
- If multiple Oracle RAC database instances are running on each node, evaluate the network bandwidth against the sum total of traffic generated by all instances. Refer to the “Tuning the Cluster Interconnect” section, near the end of this chapter.
- Ensure that the socket send and receive buffers are configured appropriately. Refer to the platform/protocol information available from Oracle.

https://t.me/learningnets

Technet24

Global Cache Statistics

Global cache statistics are accounted for across the cluster level. Like a well-audited financial statement, all the numbers for blocks served from the source instance should be equal to the blocks received at the target instance(s). The statistics also include blocks lost and blocks retransmitted during failures.

In this section, we take a close look at the most common Oracle RAC–related system statistics and their significance in analyzing performance issues.

gc current blocks received This statistic represents the number of current blocks received through the interconnect from other instances. This counter is incremented by a foreground or background process when a request for a current block or a "global cache cr request" results in the shipping of a current block at the other instance. On the other side, the sender increments the "gc current blocks served" statistic.

gc current blocks served This counter is incremented when the LMS processes are instructed by the GCS to send the current block to the requesting instance. When there is a request for the current block, the holding instance receives a BAST (blocking asynchronous system trap) to release the ownership on the block and sends a data block as a result. However, sometimes the holder can ignore the request and continue working on the block up to _gc_defer_time, when some critical operation such as block cleanout is pending.

This gives additional buffer time to respond to the request while allowing local processes to finish the job. However, the holder must flush the pending redo on the block to disk and write the block-written record (BWR) to disk. Block-written records are discussed in Chapter 9.

gc cr blocks served This statistic represents the number of the CR block requests served by the instance. When a CR block is requested, the holder clones the buffer into its local buffer cache, applies the undo information to make the buffer a point-in-time copy, and then ships the block to the requesting instance.

Because the undo for that block is cached locally, it is normally efficient in building the CR version at the holding instance. The serving of a CR block is usually a very fast operation and is not blocked. However, sometimes it may include the flushing of a pending redo to disk, if a current block is shipped for CR.

In some cases, a complete block CR version may involve reading data or an undo block from the disk or from the other instance's cache. In these cases, the incomplete copy is sent to the requestor. The requestor will be given the block version with all the changes, which could be rolled out on the holder side, and can then resume the CR block creation itself.

We discuss the CR block fabrication process in Chapter 11.

gc cr blocks received This counter is incremented in a foreground or background process when a search for a consistent read version of a block fails to find a local

https://t.me/learningnets

buffer with the required snapshot of the data. It also means that the current version for that particular block is not cached locally. Therefore, an attempt is made to get the block from a remote instance. The holder may ship either a CR or a current version of the block; the receiver copies it as a CR buffer. The block may be cached with exclusive access rights in another instance, if it was recently modified there, or with shared access rights.

gc prepare failures When a remote cache request is initiated, a request handle and a buffer ID (BID) are allocated in the Program Global Area (PGA) of the requestor. Moreover, a buffer is allocated in the System Global Area (SGA) for remote direct memory access (DMA). The number of buffers that can be allocated is port specific, depending on the characteristics of the underlying IPC protocol. If a buffer cannot be allocated because too many buffers are allocated already, the allocation is tried again immediately. Under normal circumstances, buffers are prepared and unprepared all the time so that the probability of failure is very low.

Background processes such as SMON and PMON may prepare a lot of buffers for transaction or instance recovery, respectively. The occurrence of failures is normally very low, but is sometimes observed during recovery of large recovery sets. Usually, no intervention is required.

gc blocks lost This is a very important statistic and should be 0 or close to 0. This counter is incremented when the message for the block has arrived and the actual block has not arrived. This could happen when there is contention in the network or hardware errors in the network layer. You will notice this error sometimes when the public network is used for cluster communication.

gc blocks corrupt Before the database ships a block, it computes a checksum and stores it in the buffer header. When the block is received, the checksum will be recomputed and compared to the stored value. If the checksum does not match, the block will be ignored, the request cancelled, the receive buffer freed (“unprepared”), and the statistic incremented. The request will then be retried.

The occurrence of corrupt blocks indicates that the block was damaged during the transfer, usually owing to network transmission or adapter problems. A retry is attempted. By default, the checksum is computed. In order to save CPU cycles, sometimes the following setting is used to turn off the checksum function:

```
_interconnect_checksum = FALSE
```

Turning off the checksum function is only advisable for use in benchmarks and when reliable IPC protocols such as RDG (Reliable Data Gram) are used. Do *not* disable interconnect checksum in production environments.

Global Cache Statistics Summary

Table 10-2 summarizes the global cache statistics discussed so far, their related

https://t.me/learningnets

Technet24

statistics and wait events, and the V$ views containing additional details and initialization parameters influencing the behavior of the statistics.

Global Cache Statistic	Related Statistics and Wait Events	Related Views and Initialization Parameters
gc current blocks received	gc current blocks served gc current block 2-way gc current block 3-way gc current block busy gc current block congested gc current block receive time	V$INSTANCE_CACHE_TRANSFER V$SEGMENT_STATISTICS V$SQLAREA _gc_global_lru
gc current blocks served	gc current blocks received gc current block busy	V$CURRENT_BLOCK_SERVER _gc_defer_time
gc cr blocks served	gc cr block flush time gc cr block build time gc cr block send time	V$CR_BLOCK_SERVER _fairness_threshold _cr_grant_local_role
gc cr blocks received	gc cr blocks received gc cr block 2-way gc cr block 3-way gc cr block busy gc cr block congested	V$CR_BLOCK_SERVER V$SEGMENT_STATISTICS V$SQLAREA V$INSTANCE_CACHE_TRANSFER
gc blocks lost	cr request retry	gcs_server_processes _lm_lms OS utilities such as netstat
gc blocks corrupt		_interconnect_checksum _reliable_block_sends

TABLE 10-2. *Global Cache Statistics with Their Associated Wait Events and Parameters*

Global Cache Service Times

The Global Cache Services layer also tracks the various latencies for most of the service times associated with the block transfers. The timing mechanism is well instrumented and provides a wealth of information about the performance of Cache Fusion at the global level. In this section, we discuss the most important service times in the global cache layer and their significance in managing the performance of Oracle RAC.

gc current block receive time Like any other Oracle kernel calls, when one

https://t.me/learningnets

instance makes a request for a resource, the kernel starts a timer to measure the service times. Each individual request is timed from the point when the request is made until it completes. This statistic represents the cumulated end-to-end elapsed time or latency for a current block request.

The start time of the request is stored in a global cache structure representing the state of a cached data block. The average request time (end-to-end round trip time for an average current block receipt) can be obtained by using the following formula:

```
Average Current Block Receive Time  in ms =  (gc current block receive time /
gc current blocks received)
```

NOTE

The Oracle kernel instruments the timing in centiseconds (1/100th of a second), and we multiply the number by 10 to convert it to milliseconds (1/1000th of a second).

The very same calculation that is used in the "Cache Fusion Statistics" section of the AWR report is displayed as "Avg global cache current block receive time (ms)."

gc cr block receive time This statistic represents the accumulated end-to-end elapsed time or latency for a request to receive a CR block from other instances' cache. Because CR requests are never blocked, the latency component is just processing and messaging times.

The average request time (end-to-end round trip for an average cr block receive time) can be obtained by using the following formula:

```
Average time per CR block in ms =  (gc cr block receive time /
gc cr blocks received) * 10
```

gc current block flush time The current block flush time is part of the processing time for a current block. When there is a request (usually BAST) for a current block from the master, the pending redo *must* be flushed to the log file by LGWR before LMS sends the block to the requestor. However, the operation is asynchronous in that LMS queues the request, posts LGWR, and continues processing. LMS would check its log flush queue for completions and then send the block, or go to sleep and be posted by LGWR. The performance of the redo log writes and redo log sync time can influence the overall service time significantly.

You can obtain the average time for gc current block flush time through the following formula:

```
Average time for gc current block flush time = (gc current block flush time /
V$CURRENT_BLOCK_SERVER.(FLUSH1+FLUSH10+FLUSH100+FLUSH1000+FLUSH1000) ) *10
```

https://t.me/learningnets

Technet24

NOTE

Not all current block transfers require a log flush. If the block is not modified or the redo is already written to disk, the block can be shipped directly to the other instance.

gc current block pin time The pin time indicates how much elapsed time is required to process a BAST. It does not include the flush time and the send time, but it does include the time for which a release is deferred. Normally a very high average pin time indicates contention on current blocks and requires tuning to reduce the block-level concurrency. In some cases this is associated with a high incidence of deferring the block shipping (sometimes until _gc_defer_time) of the hot block and is commonly observed on index blocks.

The average global cache current block pin time (ms) can be obtained by using the following formula:

```
Avg global cache current block pin time (ms) =  (gc current block pin time /
gc current blocks served) *10
```

gc current block send time The send time is measured when a block is handed to the IPC layer to be sent. The requests are queued in a completion queue, and the send time is measured from when a block is handed off to the IPC layer until the system call returns a completion. The average latency is usually small and expected to be in the microsecond range.

gc cr block flush time The CR block flush time is part of the service time for a CR buffer and is similar to processing for a current buffer. A normal CR buffer processing does not include the “gc cr block flush time” because CR processing is designed for highest efficiency. However, in some cases a log flush for a CR block transfer is required when a CR buffer is cloned from a current buffer that has redo pending. A high percentage is indicative of hot blocks with frequent read-after-write access.

The average global cache cr block flush time (ms) can be obtained by using the following formula:

```
Avg global cache cr block flush time (ms) =  ( gc cr block flush time /
V$CR_BLOCK_SERVER.FLUSHES) *10
```

gc cr block build time The cr block build time shows the accumulated time from the point when the database cache layer is notified of a CR request until it is either sending it or needs to wait for a log flush for that block. This is the time it takes to find or construct the read-consistent version of a block. The build time is mostly CPU time incurred scanning the cache for CR versions of blocks and/or creating CR copies.

The process of building or finding the block should never be blocked and never require non-local memory or disk accesses. This is known as the “light-work rule.”

https://t.me/learningnets

Also, an instance can refuse to ship CR blocks if it was asked for too many copies of the same CR block. In this case, the instance will disown the ownership of that block and the requestor can go and read the block from disk after appropriate grants from the master. Additional details about CR block fabrication and the light-work rule are provided in Chapter 11.

Global Cache Service Times Summary

Table 10-3 summarizes the Global Cache Service time statistics discussed so far, their related statistics and wait events, and the V$ views containing additional details and initialization parameters influencing the behavior of those statistics.

Global Cache Service Time Statistic	Related Statistics and Wait Events	Related Views and Initialization Parameters
gc current block receive time	gc current block pin time gc current block flush time gc current block send time	V$CURRENT_BLOCK_SERVER _gc_defer_time
gc cr block receive time	gc cr block flush time gc cr block build time gc cr block send time	V$CR_BLOCK_SERVER
gc current block flush time	Redo write time Log file sync	V$CURRENT_BLOCK_SERVER
gc current block pin time		V$CURRENT_BLOCK_SERVER
gc current block send time	msgs sent queue time on ksxp (ms) msgs sent queued on ksxp	
gc cr block flush time	Redo write time msgs sent queue time on ksxp (ms) msgs sent queued on ksxp gc cr block busy log file sync	
gc cr block build time	gc cr block ship time log file sync	V$CR_BLOCK_SERVER _fairness_threshold

TABLE 10-3. *Global Cache Service Statistics with Associated Wait Events and Parameters*

Enqueue Tuning in Oracle RAC

Oracle RDBMS preserves the integrity of user data during simultaneous use of the same data block by using GCS. In addition to data blocks, end users can concurrently

https://t.me/learningnets

Technet24

access many other shared resources. Oracle uses a queuing mechanism to ensure proper use of these shared resources. In an Oracle RAC environment, Global Enqueue Services (GES) protects and regulates access to these shared resources.

Enqueue wait is the time spent by a session waiting for shared resources. If a user session spends a long time waiting for a shared resource (or an enqueue), it might adversely affect the user's response time, and hence the database performance.

Waiting for the updating of the control file (CF enqueue), of an individual row (TX enqueue), and of an exclusive lock on a table (TM enqueue) are all examples of enqueue waits.

In an Oracle RAC environment, some enqueues are similar to their single-instance counterparts and need to be coordinated only at the instance level. However, many of the enqueues need to be coordinated globally. GES is responsible for coordinating the global enqueues. Due to global coordination, some enqueues might have higher performance impacts in the Oracle RAC environment.

The number of enqueue resources allocated during the instance startup is calculated as

```
GES Resources = DB_FILES + DML_LOCKS + ENQUEUE_RESOURCES
                + PROCESSES + TRANSACTIONS + 200) × (1 + (N - 1) / N)
```

where *N* is the number of Oracle RAC instances.

The dynamic performance view V$RESOURCE_LIMIT contains the initial_allocation, current_utilization, max_utilization, and limit_value statistics for enqueues. The following SQL session output shows enqueue-related statistics from V$RESOURCE_LIMIT:

```
SQL> column current_utilization heading CURRENT
SQL> column MAX_UTILIZATION heading MAX_USAGE
SQL> column  INITIAL_ALLOCATION heading INITIAL
SQL> column resource_limit format a23
SQL>select * from v$resource_limit;
RESOURCE_NAME               CURRENT  MAX_USAGE INITIAL    LIMIT_VALU
----------------------- ---------- ---------- ---------- ----------
processes                        35         44        150        150
sessions                         40         49        170        170
enqueue_locks                    16         35       2261       2261
enqueue_resources                16         52        968  UNLIMITED
ges_procs                        33         41        320        320
ges_ress                          0          0       4161  UNLIMITED
ges_locks                         0          0       6044  UNLIMITED
ges_cache_ress                  346       1326          0  UNLIMITED
ges_reg_msgs                     46        225       1050  UNLIMITED
ges_big_msgs                     22        162        964  UNLIMITED
ges_rsv_msgs                      0          0        301        301
gcs_resources                  7941      10703      13822      13822
 <<<output lines not relevant are deleted >>>>
```

https://t.me/learningnets

Oracle AWR Report

The Automatic Workload Repository (AWR) is Oracle's mechanism for gathering and preserving statistics useful for performance analysis. The background process MMON (Manageability Monitor) performs the work necessary to generate the AWR report.

Every 15 minutes, the MMON background process takes a snapshot of statistics needed for performance tuning and diagnostics. The database automatically purges snapshots older than one week. Snapshot data is stored in a set of tables in the SYSAUX tablespace; these tables are owned by SYS.

AWR is fully Oracle RAC aware and is active by default on all the instances. One of the MMON processes acts as the master and coordinates snapshots on all the active instances. AWR takes concurrent snapshots on all the instances and statistics from all the instances stored in the AWR repository with the same snap_id. The database uses the column inst_id to differentiate statistics for different instances from the same snapshot.

The package DBMS_WORKLOAD_REPOSITORY is available for the manual management of snapshots. You can use this package to create, drop, and modify snapshots. You can also use this package to establish baseline snapshots. We'll now look at some sample commands using this package.

The following command takes an immediate snapshot and archives the data in the repository:

```
sql> execute DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT ();
```

To modify the snapshot retention interval to 30 days (43,200 minutes) from its default setting of 7 days, you can use the following command:

```
sql> execute DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS
( retention => 43200);
```

You can also change the default snapshot interval of 60 minutes to any other desired value. The following command changes the snapshot interval to 4 hours (240 minutes):

```
sql> execute DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS
( interval => 240);
```

A baseline snapshot is a snapshot that is taken when the database instances and the application are running at the optimal performance level. Baseline snapshots are used for comparative analysis. To mark a range of snapshots as baseline snapshots, use the following command:

```
sql> execute DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE
(start_snap_id => 20, end_snap_id => 25, baseline_name => 'normal  baseline');
```

Baseline snapshots are not removed during normal purge operations and need to be

https://t.me/learningnets

Technet24

removed manually using the DROP_BASELINE procedure, as follows:

```
sql> execute DBMS_WORKLOAD_REPOSITORY.DROP_BASELINE
(baseline_name => 'optimal baseline', cascade => FALSE);
```

Even though AWR is Oracle RAC aware and snapshots are taken simultaneously on all the active instances, reporting and analysis need to be done at the individual instance level. The script $ORACLE_HOME/rdbms/admin/awrrpt.sql is used to create an AWR report. Here is a sample session to generate an AWR report:

```
SQL> @awrrpt.sql
Current Instance
~~~~~~~~~~~~~~~~
   DB Id          DB Name      Inst Num Instance
-----------       ------------ -------- ------------
 3553717806       PROD                2  PROD2
Specify the Report Type
~~~~~~~~~~~~~~~~~~~~~~~
Would you like an HTML report, or a plain text report?
Enter 'html' for an HTML report, or 'text' for plain text
Defaults to 'html'
Enter value for report_type:text

<<<lines for other snapshots deleted from here>>>
                              1384 30 Jul 2005 10:00      1
                              1385 30 Jul 2005 11:00      1
Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap: 1384
Begin Snapshot Id specified: 1385
Specify the Report Name
~~~~~~~~~~~~~~~~~~~~~~
The default report file name is awrrpt_2_1384_1385.txt. To use this
name, press <return> to continue, otherwise enter an alternative.

Enter value for report_name:PROD2_SAT_JULY3005_10to11.txt
```

TIP

Instead of accepting the default name for the report file, you can give it a name that includes the instance name, the day and date, and the interval for which the report is generated. This will make the report comparison and tracking easy.

https://t.me/learningnets

Interpreting the AWR Report

Oracle performance tuning is a complex subject—several entire books have been written about it. It's not possible for us to explain the contents of an AWR report in a few paragraphs here. However, we'll attempt to familiarize you with the structure of the report as well as point out some important Oracle RAC–specific statistics to which you should pay attention.

An AWR report that you generate in an Oracle RAC environment contains the following Oracle RAC–specific sections, which aren't present in an AWR report generated in a single-instance database environment:

- Number of instances
- Instance global cache load profile
- Global cache efficiency percentage
- GCS and GES—workload characteristics
- Messaging statistics
- Service statistics
- Service wait class statistics
- Top 5 CR and current blocks segments

Number of Instances Section

This section lists the number of instances at the beginning and end of the AWR report interval:

```
RAC Statistics  DB/Inst: PROD/PROD1  Snaps: 2239-2240
                                  Begin   End
                                  ----- -----
                Number of Instances:     3     3
```

Global Cache Load Profile Section

This section contains information about the inter-instance Cache Fusion data block and messaging traffic:

```
Global Cache Load Profile
~~~~~~~~~~~~~~~~~~~~~~~~~          Per Second      Per Transaction
                                   ----------      ---------------
  Global Cache blocks received:    312.73                12.61
  Global Cache blocks served:      230.60                 9.30
   GCS/GES messages received:      514.48                20.74
  GCS/GES messages sent:           763.46                30.78
  DBWR Fusion writes:               22.67                 0.91
```

The first two statistics indicate the number of blocks transferred to or from this

https://t.me/learningnets

Technet24

instance. If the database does not contain tablespaces with multiple block sizes, you can use these statistics to calculate the amount of network traffic generated by the instance. Assuming the database block size is 8K, you can calculate the amount of data sent by this instance like so:

230 × 8192 = 1,884,160 bytes/sec = 1.9 MB/sec

You can also calculate the amount of data received by this instance:

313 × 8192 = 2,564,096 bytes/sec = 2.5 MB/sec

To determine the amount of network traffic generated due to messaging, you first need to find the average message size. Use the following message SQL query to find the average message size:

```
select sum(kjxmsize*
(kjxmrcv+kjxmsnt+kjxmqsnt))/sum((kjxmrcv+kjxmsnt+kjxmqsnt))
from x$kjxm
where kjxmrcv > 0 or kjxmsnt > 0 or kjxmqsnt >0 ;
```

For the system from which the sample report was taken, the average message size was about 300 bytes.

Calculate the amount of messaging traffic on the network like this:

300 (763 + 514) = 383,100 = 0.4 MB

In case the system for which the report is being analyzed is not available to determine the average message size, you can add 10 to 12 percent of the data traffic to estimate the messaging traffic. Assuming the report is from a two-node Oracle RAC environment, you can calculate the total network traffic generated by Cache Fusion activity like this:

= 1.9 + 2.5 + 0.4 = 4.8 MBytes/sec

= 4.8 × 8 = 38.4 Mbits/sec

To estimate the network traffic generated in an Oracle RAC environment consisting of two or more nodes, first generate an AWR report from all the instances for the same interval. Then calculate the total network traffic generated due to Cache Fusion activity:

= ∑ block received + ∑ Msg Recd × Avg Msg size

Note that ∑ block received = ∑ block served, and ∑ Msg Sent = ∑ Msg Recd. These calculations would require an AWR report from all the instances for the same interval.

https://t.me/learningnets

NOTE

The AWR report contains a line that indicates the interconnect traffic generated by the instance "Estd Interconnect traffic (KB)." A quick sum of this statistic from all the instances will provide the overall interconnect traffic, thus avoiding the trouble of the calculations.

The "DBWR Fusion writes" statistic in this section indicates the number of times the local DBWR was forced to write a block to disk due to remote instance(s). This number should be low; it is better to analyze this as a fraction of overall DBWR writes, which is available as the "Physical writes" statistics in the "Load Profile" section of the report:

```
Load Profile
~~~~~~~~~~~~                            Per Second       Per Transaction
                                        ----------       ---------------
                  Redo size:            700,266.45        28,230.88
              Logical reads:             17,171.74           692.27
              Block changes:              2,394.61            96.54
             Physical reads:                208.42             8.40
            Physical writes:                215.54             8.69
                 User calls:                275.03            11.09
                     Parses:                 22.06             0.89
```

In this case, "DBWR Fusion writes" is approximately 10.5 percent of the overall Physical writes. Establish a baseline percentage when the performance is good. You can fine-tune this number by ensuring that instances do not step on each other's data or partitioning of the application.

Global Cache Efficiency Percentages Section

This section of the report shows how the instance is getting all the data blocks it needs:

```
Global Cache Efficiency Percentages (Target local+remote 100%)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Buffer access -  local cache %:   97.19
Buffer access - remote cache %:    1.82
Buffer access -         disk %:    0.99
```

The most preferred method is to get data in the local buffer cache, followed by the remote instances' cache, and lastly from the disk. The sum of the first two rows gives the "cache hit ratio" for the instance. The value for the remote cache hit should typically be less than 10 percent. Consider implementing an application-partitioning scheme if this value is higher than 10 percent.

https://t.me/learningnets

Technet24

GCS and GES Workload Characteristics Section

This section contains timing statistics for global enqueue and global cache. The statistics are further subdivided into the following four subsections:

- Average time (in milliseconds) to obtain an enqueue
- Time the instance has to wait before receiving a block in consistent read (CR) mode or in current mode
- Amount of time/delay for an instance to process a CR request
- Amount of time/delay for an instance to process a current block request

```
Global Cache and Enqueue Services - Workload Characteristics
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

                    Avg global enqueue get time (ms):    951.5

         Avg global cache cr block receive time (ms):      3.9
    Avg global cache current block receive time (ms):      3.0

           Avg global cache cr block build time (ms):      0.7
            Avg global cache cr block send time (ms):      0.3
     Global cache log flushes for cr blocks served %:     50.5
           Avg global cache cr block flush time (ms):     10.1

       Avg global cache current block pin time (ms):      1.4
      Avg global cache current block send time (ms):      0.3
Global cache log flushes for current blocks served %:     1.4
     Avg global cache current block flush time (ms):      4.4
```

As a rule of thumb, all timings related to a CR block should be less than 5 ms, and all timings related to current block processing should be less than 10 ms. However, nowadays close to 1–5 ms is considered acceptable due to faster networks, but these numbers are good and conservative. We have seen these numbers at less than a millisecond in high-performance systems such as Oracle Exadata Database Machine.

GCS and GES Messaging Statistics Sections

The first section contains statistics related to sending a message, and generally all these statistics should be less than 1 millisecond. The second section details the breakup of direct and indirect messages.

https://t.me/learningnets

```
Global Cache and Enqueue Services - Messaging Statistics
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                    Avg message sent queue time (ms):          0.5
             Avg message sent queue time on ksxp (ms):          1.7
                Avg message received queue time (ms):          0.2
                   Avg GCS message process time (ms):          0.5
                   Avg GES message process time (ms):          0.2

                       % of direct sent messages:    52.22
                     % of indirect sent messages:    46.95
                   % of flow controlled messages:     0.83
```

Direct messages are the messages sent by an instance foreground process or the user process to remote instances, whereas indirect messages are the messages that are not urgent and are pooled and then sent. Indirect messages are low-priority messages. These statistics generally depend on the nature of the workload among the instances, and not much can be done to fine-tune them. Establish a baseline for these during normal user workloads. Also, observe these statistics after any significant change in the user workload and establish a new baseline after any changes.

Service Statistics Section

Statistics in this section show the resources used by all the service instance supports:

```
Service Statistics  DB/Inst: PROD/PROD1  Snaps: 2239-2240
-> ordered by DB Time
-> us - microsecond - 1000000th of a second
                                                        Physical     Logical
Service Name               DB Time (s)   DB CPU (s)      Reads        Reads
------------------------  ------------  ------------  ----------  --------------
PROD                      1,198,708.4   17,960.0        491,498    9,998,798,201
SYS$USERS                     3,903.3      539.7        245,724        2,931,729
SYS$BACKGROUND                   29.3        4.8          7,625        4,801,655
```

The instance PROD1 is serving the workload connected using the service PROD. If multiple services are configured, the breakup will appear here. There are two internal services in addition to the application services defined by the DBA. SYS$BACKGROUND is used by all background processes. SYS$USERS is the default service for user sessions that are not associated with applications services—for example, connecting as “sqlplus / as sysdba.”

Service Wait Class Statistics Section

This section summarizes waits in different categories for each service. If a service response is not acceptable, these statistics can show where the service is waiting.

```
Service Wait Class Stats  DB/Inst: PROD/PROD1  Snaps: 2239-2240
-> Wait Class info for services in the Service Statistics section.
-> Total Waits and Time Waited displayed for the following wait
```

https://t.me/learningnets

Technet24

```
   classes:  User I/O, Concurrency, Administrative, Network
-> Time Waited (Wt Time) in centisecond (100th of a second)

Service Name
-------------------------------------------------------------------------------
 User I/O  User I/O  Concurcy  Concurcy     Admin     Admin   Network   Network
Total Wts   Wt Time Total Wts   Wt Time Total Wts   Wt Time Total Wts   Wt Time
--------- --------- --------- --------- --------- --------- --------- ---------
PROD
  2227431   4136718   3338963  95200428         0         0   1660799     15403
SYS$USERS
  2 59502    188515       274       486         0         0      1676         3
SYS$BACKGROUND
    10412      1404      4135     12508         0         0         0         0
          ---------------------------------------------------------------------
```

Top 5 CR and Current Blocks Segments Sections

These sections contain the names of the “Top 5” contentious segments (index or tables). If a table or an index is being subject to a very high percentage of CR and current block transfers, you need to analyze its usage pattern, database layout characteristics, and other parameters that might cause contention.

```
Segments by CR Blocks Received  DB/Inst: PROD/PROD1  Snaps: 2239-2240
                                                                    CR
            Tablespace                          Subobject  Obj.     Blocks
Owner          Name     Object Name              Name     Type    Received  %Total
---------- ---------- -------------------- ---------- ----- ------------ -------
ES_MAIL    ESINFREQID ES_INSTANCE                       TABLE      136,997   58.65
ES_MAIL    ESINFREQID ES_INSTANCE_IX_TYPE               INDEX       21,037    9.01
ES_MAIL    ESFREQTBL  ES_FOLDER                         TABLE       14,616    6.26
ES_MAIL    ESFREQTBL  ES_USER                           TABLE        6,251    2.68
ES_MAIL    ESSMLTBL   ES_EXT_HEADER                     TABLE        2,467    1.06
          -------------------------------------------------------------

Segments by Current Blocks Received  DB/Inst: PROD/PROD1  Snaps: 2239-2240
                                                                    Current
            Tablespace                          Subobject  Obj.     Blocks
Owner          Name     Object Name              Name     Type    Received  %Total
---------- ---------- -------------------- ---------- ----- ------------ -------
ES_MAIL    ESINFREQID ES_INSTANCE                       TABLE      602,833   80.88
ES_MAIL    ESSMLTBL   ES_EXT_HEADER                     TABLE       18,527    2.49
ES_MAIL    ESINFREQID ES_INSTANCE_IX_TYPE               INDEX       13,640    1.83
ES_MAIL    ESFREQTBL  ES_FOLDER                         TABLE       11,242    1.51
ES_MAIL    ESINFREQID ES_INSTANCE_IX_FOLDE              INDEX        5,026     .67
          -------------------------------------------------------------
```

You’ll also find the following information in these sections:

```
-> Total Current Blocks Received:              2,328
-> Captured Segments account for               93.1% of Total
```

https://t.me/learningnets

The additional information allows you to compare the top segment's activity with respect to the overall system activity in that category.

ADDM

Automatic Database Diagnostic and Monitor (ADDM) is Oracle's powerful tool that contributes to making the database self-tuning. ADDM contains the following infrastructure components:

- **MMON** Manageability Monitor is a new background process that does all the work required for ADDM.
- **AWR** Automatic Workload Repository is a set of database objects used to gather and store database performance statistics.
- **Packages** The package DBMS_ADVISOR is available to manage ADDM.
- **Parameters** The STATISTICS_LEVEL initialization parameter should be set to either TYPICAL or ALL for ADDM to function. The DBIO_EXPECTED parameter needs to be set to represent the average read time for a database block. Its default value is 10000 (10 milliseconds). Use the following command to set this parameter:

```
Sql> execute DBMS_ADVISOR.SET_DEFAULT_TASK_PARAMETER
('ADDM','DBIO_EXPECTED'. 30000)
```

- **Auxiliary tablespace** Oracle uses an auxiliary tablespace (SYSAUX) as the storage space for the tools and other components (AWR, ADDM, and so on). This tablespace is created by default during database creation.

The goal of ADDM is to optimize the time spent by the database for servicing the user workload. Its sole objective is to reduce DB Time, which consists of the following two components:

- **Wait time** Time spent by user sessions while waiting for any resources
- **CPU time** Time spent by user sessions while processing user work

MMON analyzes the data stored in the AWR repository and uses predefined, built-in rules to suggest how DB Time can be reduced. It generates a report with recommendations but does not implement them automatically. Its findings include the following:

- **System capacity issues** CPU and I/O subsystem usage is analyzed.
- **Instance management** Instance memory management parameters such as SGA, redo log buffers, and buffer cache are analyzed. Also other initialization parameters are analyzed for optimal settings.
- **Java, SQL, and PL/SQL tuning** High resource-consuming Java, SQL, and PL/SQL statements are analyzed, as well as whether the optimal data-access path is being used.

https://t.me/learningnets

Technet24

- **Contention** In a single-instance environment, contention is analyzed in single instances as well as in the Oracle RAC environment. This takes into account buffer_busy_waits and all global cache–related wait events.
- **Database structure** The online redo log file size, among other things, is analyzed.
- **Miscellaneous** Application connection patterns are analyzed.

ASH Reports

You can view the Active Session History (ASH) reports for Oracle RAC to gain insight into your Oracle RAC cluster performance. ASH reports provide valuable statistics concerning session activity in an Oracle database. The database gathers all information about active sessions across all instances of an Oracle RAC database and stores the data in the System Global Area (SGA). Because ASH reports contain statistics just for active sessions, these reports help you understand how the database is handling its workload.

An ASH report has several sections, but the most important ones for reviewing the performance of an Oracle RAC database are the following:

- **Top Cluster Events** The Top Cluster Events section is folded into the Top Events report. This section shows the session activity in the cluster wait class event. You can learn which events and instances are responsible for most of the cluster wait events from this section.
- **Top Remote Instance** The Top Remote Instance section of an ASH report is contained in the Top Load Profile report specific to Oracle RAC. This section shows the cluster wait events and the instances that are responsible for most of the cluster waits.

Tuning the Cluster Interconnect

Oracle uses the cluster interconnect for sending messages to coordinate data block access and for sending copies of data blocks from one instance’s cache to another instance’s cache to optimize disk I/O. Therefore, cluster interconnect performance is crucial to Oracle RAC performance. Its configuration and proper functioning is one of the most important parts of Oracle RAC performance tuning.

Verifying That Private Interconnect Is Used

You can specify the private interconnect while configuring the CRS. Ensure that the private interconnect is used for Cache Fusion traffic. Also ensure that the private interconnect isn’t used by other network-intensive functions such as **ftp** and **rcp** commands between the cluster nodes because this might impact Oracle RAC performance.

https://t.me/learningnets

You can use the following commands to verify that the intended private interconnect is being used for the Cache Fusion traffic:

```
SQL> oradebug setmypid
Statement processed.
SQL> oradebug ipc
Information written to trace file.
SQL> oradebug tracefile_name
/u02/app/oracle/diag/rdbms/alpha/alpha1/trace/alpha1_ora_3766.trc
```

This will dump a trace file to the user_dump_dest. The output will look something like this:

```
SSKGXPT 0xb466a1e0 flags  sockno 13 IP 10.1.0.201 UDP 42773
SKGXPGPID Internet address 10.1.0.201 UDP port number 42773
```

This indicates that IP 10.1.0.201 with a UDP protocol is being used for Cache Fusion traffic. Also, cluster interconnect information is printed in the alert.log:

```
Interface type 1 eth1 10.1.0.0 configured from GPnP Profile for use
 as a cluster interconnect
Interface type 1 eth0 10.1.1.0 configured from GPnP Profile for use
 as a public interface
```

You can query the internal view X$KSXPIA to find out the network interface used for Cache Fusion traffic. Also, this view lists the source of the information (from where it picks the IP address, such as OCR or from the cluster_interconnects parameter). The column value for PICKED_KSXPIA is listed as OCR if it picks from the Oracle Cluster Registry and is listed as CI if it picks the address via the cluster_interconnects initialization parameter.

```
SELECT INST_ID,PUB_KSXPIA,PICKED_KSXPIA,NAME_KSXPIA,IP_KSXPIA
FROM X$KSXPIA;

INST_ID PUB_KS PICKED NAME_KSXPIA     IP_KSXPIA
------- ------ ------ --------------- ----------------
      1 N      OCR    eth1            10.1.0.201
      1 Y      OCR    eth0            10.1.1.201
```

Interconnect Latencies

The background process “ping” periodically measures the network statistics. It wakes up periodically (about every 5 seconds) and measures the latencies for message transfers and block transfers. On every wakeup, it sends two messages to all the cluster nodes (of 500 bytes and 8192 bytes, respectively), calculates the round-trip latencies, and maintains the information internally. AWR also uses this information to display the “Interconnect Ping Latency Stats” section of the report. The internal data is

https://t.me/learningnets

Technet24

exposed via X$KSXPPING.

Be forewarned that the observed latencies may include the scudding latency from the operating system running the **ping** command. Take this data with a grain of salt because long latency could be reported when there is a long scheduling latency or high IPC latency due to host load. Because ping is not a mandatory process for the functioning of the cluster but rather a mere diagnostic process, it is not scheduled to run in real time. Scheduling the ping process in real time would potentially overcome the aforementioned issues; however, this is not recommended due to the importance (or unimportance) of the process. It is advisable to run the ping process in default time-sharing schedule mode.

As always, it's a good idea to establish baseline statistics when the cluster is performing normally and compare the interconnect latencies from X$KSXPPING when you notice a performance issue.

NOTE

If you notice the ping process consuming a lot of the CPU processing, you can turn off the process by setting the parameter _ksxp_ping_enable to FALSE.

The internal view X$KSXPCLIENT provides the total overview network traffic generated by the other diagnostic processes in the Oracle 11*g* RAC environment. You can look up the network traffic generated by the ping process by using the following SQL query:

```
select * from X$KSXPCLIENT where name ='ping';

ADDR             INDX INST_ID NAME BYTES_SENT  BYTES_RCV
---------------- ------ ------- ---- ---------- ----------
00000000907CC0E8      9       1 ping     650904     650904
```

Verifying That Network Interconnect Is Not Saturated

As mentioned in the preceding sections, you can use the AWR report to analyze the following:

- Network traffic generated by Oracle instances. Ensure that at no point in time the traffic generated is saturating the private interconnect.
- Whether the network delays and latency, as indicated in the AWR report, are excessive. You can use the OEM to monitor the interconnect traffic.

Consider upgrading the network interconnect to a higher capacity when you see the

https://t.me/learningnets

interconnect traffic is reaching about 50 percent of its theoretical capacity.

Summary

Performance tuning in the Oracle RAC environment is similar to tuning in a single-instance environment, with a few additional considerations for Oracle RAC. Application partitioning and performance features such as ASM also help greatly in Oracle RAC. Caching frequently used sequences and choosing the right block size both have a big positive impact on performance.

Oracle RAC wait events and system statistics provide the fine-grained diagnostic information about the block transfers and inter-instance messaging. New optimization techniques and improved statistics-collection metrics at the global cache and enqueue services layers take performance analysis to the next level.

Automatic Workload Repository (AWR) is the performance warehouse for Oracle databases, and AWR reports provide tremendous information related to Oracle RAC performance. The AWR, ADDM, and ASH reports are an excellent place to start analyzing performance in an Oracle RAC environment. These tools simplify the analyzing of Oracle RAC performance.

In the next chapter, we look at resource management in an Oracle RAC environment.

https://t.me/learningnets

Technet24

PART IV

Advanced Oracle RAC Concepts

https://t.me/learningnets

CHAPTER 11

Global Resource Directory

Welcome to the world of resource management and administration in a parallel computing environment. This chapter provides a detailed discussion about resource management and locking issues as well as the inter-instance coordination activities that take place in the cluster. Understanding the internal workings of locking will help you in architecting Oracle RAC solutions and administering the Oracle RAC environment.

The Oracle RAC environment includes numerous resources, such as multiple versions of data block buffers in buffer caches in different modes. Oracle uses different types of locking and queuing mechanisms within a cluster to coordinate lock resources, data, and inter-instance data requests. We will examine how Oracle coordinates these resources to maintain the integrity of shared data and resources.

Coordination of concurrent tasks within a cluster is called *synchronization*. Resources such as data blocks and locks must be synchronized between the nodes because nodes within a cluster acquire and release ownership of them frequently. The synchronization provided by the Global Resource Directory (GRD) maintains cluster-wide concurrency of the resources and in turn ensures the integrity of shared data. The amount of synchronization required depends on the amount of resources and the number of users and tasks working on those resources. Little synchronization may be

https://t.me/learningnets

Technet24

needed to coordinate a small number of concurrent tasks, but with many concurrent tasks, significant synchronization is required.

Buffer cache management also requires synchronization because it's divided into multiple caches, and each instance is responsible for managing its own local version of the buffer cache. Copies of data blocks may be exchanged between the nodes in the cluster. This concept is also known as the *global cache*, although each node's buffer cache is separate and copies of blocks are exchanged through traditional distributed locking mechanisms.

Global Cache Services (GCS) maintains the cache coherency across the buffer cache resources. Global Enqueue Services (GES) controls the resource management across a cluster's non-buffer cache resources.

Resources and Enqueues

A *resource* is an identifiable entity—that is, it has a name or reference. The referenced entity is usually a memory region, a disk file, a data block, or an abstract entity; the name of the resource *is* the resource.

A resource can be owned or locked in various states, such as *exclusive* and *shared*. By definition, any shared resource is lockable. If it isn't shared, no access conflict can occur. If the resource is shared, the database must resolve access conflicts, typically with a lock. Although the terms *lock* and *resource* refer to separate objects, it's common to use the two terms interchangeably.

A global resource is visible and used throughout the cluster. A local resource is used by only one instance. It may still have locks to control access by the multiple processes of the instance, but no access to it is available from outside the instance.

Each resource can have a list of consumers for that resource; this is called the *grant queue*. Each of these clients might want the same resource in a different locking mode. A *convert queue* is a queue of client processes waiting to lock the resource in a particular lock mode.

Conversion is the process of changing a lock from the mode it currently holds to a different mode. Even if the mode is NULL, it's regarded as holding a lock without any conflicts of interest on that resource. Sometimes, when someone holds a NULL lock on something, it's also called an *anti-lock*.

Conversion is meant for optimizations such as the reader lock, which allows people to get read-only access to a resource, in which case the full global lock resource doesn't need to be enabled. This same concept is also available in enqueues like TX (transaction), where one can have a global TX or a local TX. The size of the enqueue resource by default is geared toward a local call, and when multiple users access that same resource, the TX lock is blown up to a full global resource.

Figure 11-1 shows the grant and convert queue representation for a block and its structure. The data block resource—also known as a *Parallel Cache Management (PCM) resource*—is identified using the following format:

https://t.me/learningnets

```
[0x10000c5][0x1],[BL]
Grant Queue
Convert Queue
Lock Value Block
```

FIGURE 11-1. *Grant queue and convert queue*

```
[LE] [Class] [BL]
[0x10000c5][0x1],[BL]
Grant Queue
Convert Queue
Lock Value Block
```

- **LE** Lock element that denotes the data block address (DBA—not to be confused with the other DBA, database administrator). It is visible in V$BH.LOCK_ELEMENT_ADDR.
- **Class** Class of the block. Visible in V$BH.CLASS# (data, undo, temp, and so on).
- **BL** Buffer cache management locks (or buffer locks).

Data buffer cache blocks are the most obvious and most commonly used global resource. Other data item resources are also global in the cluster, such as transaction enqueues and library cache and shared pool data structures.

GCS (Parallel Cache Management, or PCM) handles the data buffer cache blocks. GES (non–Parallel Cache Management, or non-PCM) handles the non–data block resources. The Global Resource Manager (GRM, the old DLM) keeps the lock information valid and correct across the cluster.

Grants and Conversions

The database places locks on a resource grant or convert queue. If the lock mode changes, the database moves the lock between the queues. If several locks exist on the grant queue, they must be compatible. Locks of the same mode aren’t necessarily compatible with each other.

NOTE

Chapter 2 discusses lock conversation and the compatibility matrix of DLM locks. We discuss the management of GES locks later in this chapter.

A lock leaves the convert queue under any of the following conditions:

https://t.me/learningnets

Technet24

- The process requests the lock termination (that is, removes the lock).
- The process cancels the conversion; the lock is moved back to the grant queue in the previous mode.
- The requested mode is compatible with the most restrictive lock in the grant queue and with all the previous modes of the convert queue, and the lock is at the head of the convert queue.

Figure 11-2 shows the lock conversion mechanisms. Convert requests are processed in a first in, first out (FIFO) fashion.

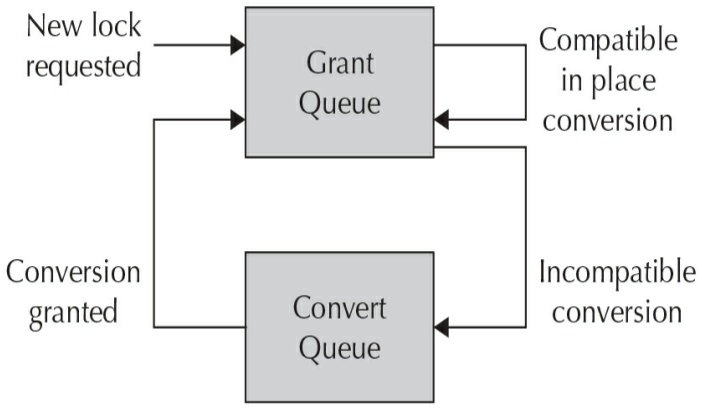


FIGURE 11-2. *Lock conversion*

The grant queue and convert queue are associated with every resource that GES manages. The following example discusses the sequence of operations in detail with respect to the compatibility of lock modes. This is an example of the database placing locks on a resource's grant or resource queues:

1. The database grants a shareable read lock.
2. The database grants another shareable read lock. The two read locks are compatible and can reside on the grant queue.
3. Another shareable read lock is placed on the grant queue.
4. One lock converts to shareable NULL. This conversion can be done in place because it is a simple downgrade.
6. Another lock attempts to convert to exclusive write. The lock must be placed on the convert queue.

Figure 11-3 shows the interested resource's grant and convert queue structure in all five stages.

https://t.me/learningnets

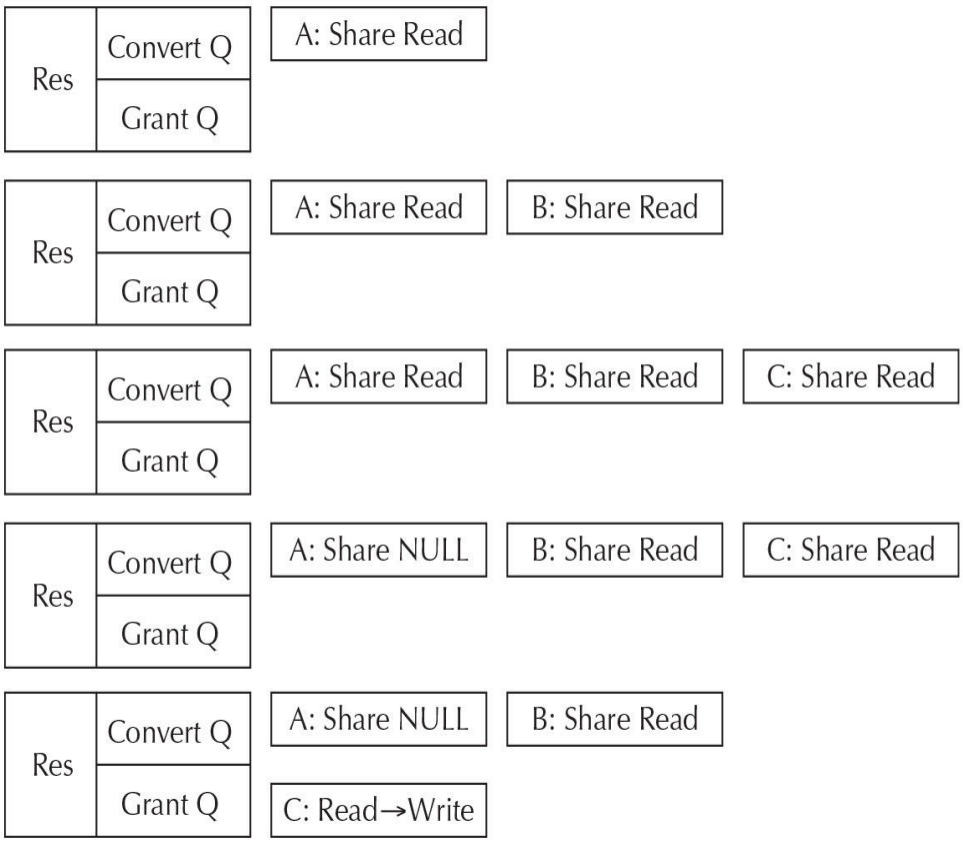


FIGURE 11-3. *Grant and convert queue operations*

Locks and Enqueues

Enqueues are locks that support queuing mechanisms and can be acquired in different modes. An enqueue can be held in exclusive mode by one process, and others can be held in non-exclusive mode depending on the type.

Enqueues in an Oracle RAC database are the same as the enqueues used in a single instance of the Oracle RDBMS, except for the scope. A few of the enqueues are local to the instance, and others are global. A TX enqueue protects a transaction, irrespective of whether it is for a single instance or a clustered environment. However, the temporary tablespace enqueue is local to the instance where it's held.

Cache Coherency

In *cache coherency*, the contents of the caches in different nodes are in a well-defined state with respect to each other. Cache coherency identifies the most up-to-date copy of a resource. In case of node failure, no vital information is lost (such as a committed transaction state) and atomicity is maintained. This requires additional logging or copying of data but is not part of the locking system.

Cache coherency is the mechanism by which multiple copies of an object are kept consistent between Oracle instances. The background process LCKx is responsible for

https://t.me/learningnets

Technet24

this important task. This process dialogues with LMD0 to synchronize access to resources.

The lock and resource structures for instance locks reside in the GRD, a dedicated area within the shared pool. The GRD maintains information about the shared resources such as data blocks.

The GCS maintains the details about the data block resources and cached versions. The GES maintains additional details—such as the location of the most current version, state of the buffer, role of the data blocks (local or global), and ownership details such as most current versions.

Each instance maintains a part of the GRD in its System Global Area (SGA). The GCS and GES nominate one instance, called the *resource master*, to manage all information about a particular resource. Each instance knows which instance masters which resource. This is done using a simple hash function on the data block address of the buffer to determine its master so that only a subset of resources need to be remastered when a node joins or leaves a cluster. GCS maintains cache coherency by using the Cache Fusion algorithm.

GES manages all non–Cache Fusion inter-instance resource operations and tracks the status of all Oracle enqueue mechanisms. The primary resources of the GES controls are dictionary cache locks, library cache locks, and standard enqueues. GES also performs deadlock detection of all deadlock-sensitive enqueues and resources.

Global Enqueue Services

GES coordinates the requests of all global enqueues (any non-buffer cache resources). This is the single point of contact for lock management in the Oracle RAC. It is also involved in deadlock detection and request timeouts. During normal operation, it manages caching and performs the cleanup operation during cluster reconfiguration.

Latches and Enqueues

The two types of local locks are latches and enqueues. *Latches* are instance specific and do not affect database-wide operations. Latches do not affect the global operations of clustered environments. Enqueues, however, can be both local to an instance and global to a cluster.

Latches are lightweight, low-level serialization mechanisms that protect in-memory data structures in the SGA. They do not support queuing and do not protect database objects such as tables or datafiles. Latches are usually held for short durations. Latches don’t support multiple levels and are always acquired in exclusive mode. Because latches are synchronized within a node, they do not facilitate internode synchronization.

Enqueues are shared structures that serialize access to database resources. Enqueues support multiple modes and are held longer than latches. They have a database-wide scope because they protect persistent objects such as tables or library cache objects. For example, if you update a row in a block, no one from any other instance should be

https://t.me/learningnets

able to update the same row. A transaction (TX) enqueue protects the update for your operation; this enqueue is a global lock in Oracle RAC.

Enqueues are associated with a session or transaction, and Oracle can use them in any of the following modes:

- Shared or protected read
- Exclusive
- Protected write
- Concurrent read
- Concurrent write
- NULL

Depending on the operation on the resource, the database obtains an enqueue lock on that resource. The following listing provides the most common locking modes for the associated operations:

```
Operation                 Lock Mode LMODE Lock Description
------------------------- --------- ----- ----------------
 Select                     NULL     1    null
 Lock For Update            SS       2    sub share
 Select For update          SS       2    sub share
 Insert/Delete/Update       SX       3    sub exclusive
 Lock Row Exclusive         SX       3    sub exclusive
 Create Index               S        4    share
Lock Share                  S         4    share
Lock Share Row Exclusive    SSX       5    share/sub exclusive
Alter/Drop/Truncate table   X         6    exclusive
Drop Index                  X         6    exclusive
Lock Exclusive              X         6    exclusive
Truncate table              X         6    exclusive
```

When a session is waiting on a specific enqueue wait event, it is waiting for an Oracle lock held by another user (or sessions) in a mode that's incompatible with the requested mode. The following is the compatibility matrix of the enqueue modes:

Compatible	NULL	SS	SX	S	SSX	X
NULL	Yes	Yes	Yes	Yes	Yes	Yes
SS	Yes	Yes	Yes	Yes	Yes	No
SX	Yes	Yes	Yes	No	No	No
S	Yes	Yes	No	Yes	No	No
SSX	Yes	Yes	No	No	No	No
X	Yes	No	No	No	No	No

https://t.me/learningnets

Technet24

Enqueues are acquired in various *lock modes*. Table 11-1 summarizes each lock mode and provides a detailed description; this information is common to both single-instance and multi-instance databases.

Mode	Summary	Description
NULL	Null mode. No lock is on the resource.	Conveys no access rights. Typically, a lock is held at this level to indicate that a process is interested in a resource, or it is used as a placeholder. Once created, NULL locks ensure the requestor always has a lock on the resource; there is no need for the DLM to create and destroy locks constantly when ongoing access is needed.
SS	Subshared mode (concurrent read). Read—there may be writers and other readers.	The associated resource can be read in an unprotected fashion; other processes can read from and write to the associated resource. This lock is also known as an RS (row share) table lock.
SX	Shared exclusive mode (concurrent write). Write—there may be other readers and writers.	The associated resource can be read from or written to in an unprotected fashion: other processes can both read from and write to the resource. This lock is also known as an RX (row exclusive) table lock.
S	Shared mode (protected read). Read—no writers are allowed.	A process cannot write to the associated resource, but multiple processes can read it. This is the traditional shared lock. Any number of users can have simultaneous read access to the resource. Shared access is appropriate for read operations.
SSX	Subshared exclusive mode (protected write). One writer only—there may be readers.	Only one process can hold a lock at this level. This allows a process to modify a resource without allowing other processes to modify the resource at the same time. Other processes can perform unprotected reads. This traditional update lock is also known as an SRX (shared row exclusive) table lock.
X	Exclusive mode. Write—no other access is allowed.	When a lock is held at this level, it grants the holding process exclusive access to the resource. Other processes cannot read from or write to the resource. This is the traditional exclusive lock.

TABLE 11-1. *Lock Mode Summary*

NOTE

For a complete and detailed discussion on the fundamentals of locking, refer to the Oracle document "Oracle Database Concepts (Oracle 12c Release 2)." Specifically,

https://t.me/learningnets

see the topic "Overview of the Oracle Database Locking Mechanisms" in Chapter 9, "Data Concurrency and Consistency."

Global Locks Database and Structure

Each node holds directory information for a set of resources. To locate a resource, the directory service uses a hashing function on the resource name to determine which nodes hold the directory tree information for the resource. The database sends the lock request directly to the holding node. The GRD also stores information on resource locks and the converter and granter queues. When a process requests a lock that is owned by the same node, the structure is created local to the node. In this case, the directory node is the same as the master node.

The GES layer in Oracle RAC synchronizes global locks among all active instances in a cluster. Global locks are mainly of two types:

- Locks used by the GCS for buffer cache management. Those locks are called *Parallel Cache Management (PCM) locks*.
- Global locks, such as global enqueues, that Oracle synchronizes within a cluster to coordinate non-PCM resources. The database uses these locks to protect the enqueue structures, and they are managed by GES. Global locks are called *non-PCM locks*. An example of a non-PCM lock is a controlfile enqueue that RMAN needs to obtain during one of its backups.

GES tracks the status of all Oracle locks and their corresponding resources. Global locks are allotted and created during instance startup, and each instance owns or masters some set of resources or locks.

Background processes within a cluster's instances, rather than transactions, hold the global locks. An instance *owns* a global lock that*protects* a resource, such as a data block or data dictionary entry, when the resource enters the instance's SGA. The GES manages locking only for resources accessed by more than one instance. In the following sections, we discuss non-PCM coordination (non-PCM locks) only.

GES Locks

Many non-PCM locks control access to datafiles and controlfiles and also serialize inter-instance communication. They also control library caches and dictionary caches. These locks protect datafiles, not the datafile blocks inside the file. Examples of these are Data Manipulation Language (DML) enqueues (table locks), transaction enqueues, and Data Definition Language (DDL) locks or dictionary locks. The System Change Number (SCN) and the mount lock are global locks, not enqueues.

Transaction Locks or Row-Level Locks Oracle's row-level locking is one of the most sophisticated features in the RDBMS; it is also one of the least understood topics. Row-level locks protect selected rows in a data block during a transaction. A transaction acquires a global enqueue and an exclusive lock for each individual row

https://t.me/learningnets

Technet24

modified by one of the following statements: INSERT, UPDATE, DELETE, or SELECT with the FOR UPDATE clause.

The database stores these locks in the block, and each lock refers to the global transaction enqueue. Because the database stores the locks at the block level, their scope is wider at the database level. A transaction lock is acquired in exclusive mode when a transaction initiates its first change. It is held until the transaction does a COMMIT or ROLLBACK. SMON also acquires it in exclusive mode when recovering (undoing) a transaction. The database uses transaction locks as a queuing mechanism for processes awaiting the release of an object locked by a transaction in progress.

Internal Implementation of Row-Level Locks An Oracle database creates every data block in the datafile, with the exception of the temp and rollback segments, with a predefined number of transaction slots. (Undo segments have different types of transaction slots, called *transaction tables*.) These transaction slots are called *interested transaction lists (ITLs)* and are controlled by the INITRANS parameter. The default value for INITRANS is 2 for tables and 3 for index segments.

Every transaction slot occupies 24 bytes of free space in the variable part of the data block header. The maximum number of transaction slots is controlled by the MAXTRANS parameter. However, the size of the variable part of the database block header cannot exceed 50 percent of the data block size. This limits the total number of transaction slots in the data block.

ITL slots are acquired for every DML lock that affects that data block. An ITL contains the transaction ID (XID), which is the pointer to an entry in the transaction table of a rollback segment. Another transaction can always read the data from the rollback segment. If new transactions want to update the data, they must to wait until the current transaction commits or rolls back.

Any transaction that is interested in getting a DML lock on the rows belonging to that block *must* get an ITL slot before proceeding. An ITL entry consists of the XID, the undo byte address (UBA), the flags indicating the transaction status (Flag), and the lock count (Lck) showing the number of rows locked by this transaction within the block and the SCN at which the transaction is updated. The XID uniquely identifies the transaction and provides the information about the undo for that transaction. Figure 11-4 shows the relational links between the block and ITL and the transaction tables.

https://t.me/learningnets

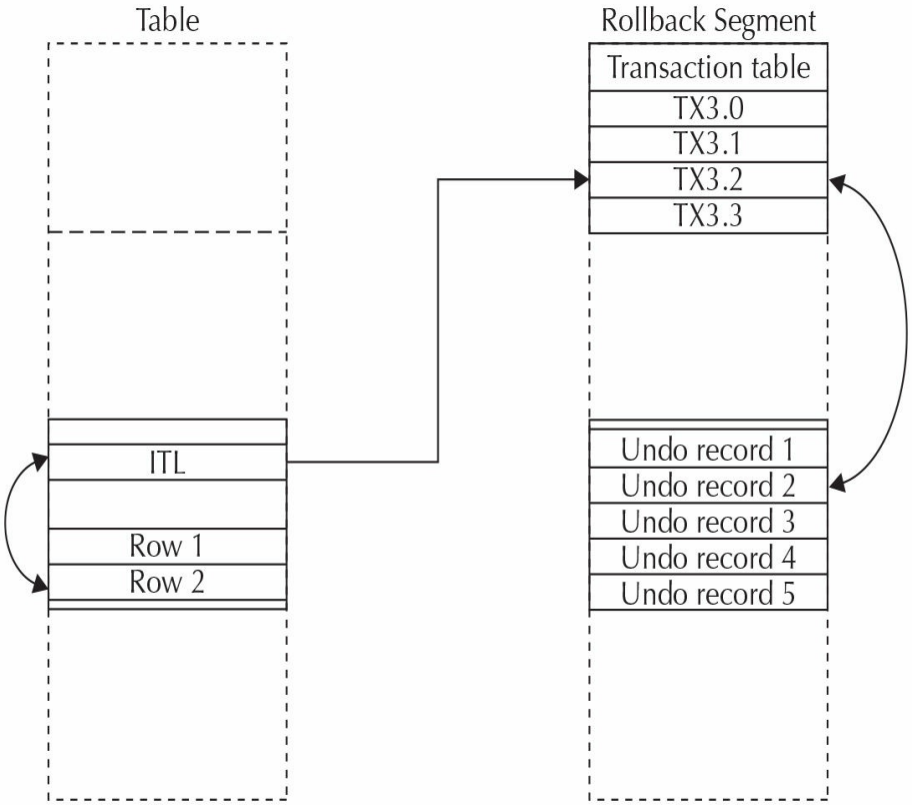


FIGURE 11-4. *Internal implementation of row-level locks*

Transaction Table

The transaction table is the data structure within the rollback segment that holds the transaction identifiers of the transactions using that rollback segment. The number of rows in the transaction table is equal to the number of transaction slots in that rollback segment and is visible via the internal view X$KTUXE (available only when logged in as SYS).

The transaction ID (XID) consists of three parts: the undo segment number, the undo segment slot number, and the wrap number in the undo segment (USN.SLOT#.WRAP#). The dynamic performance view V$TRANSACTION can be queried to get more details on that particular transaction.

While the transaction commits, Oracle completes the bare minimum requirements for the transaction commit operation as it optimizes the system for higher throughput. This involves updating the flag in the transaction table in the rollback segment, and the block isn't revisited. This process is known as*fast commit*. During this time, the ITL in the data block (called *open ITL)* is still pointing to the transaction table of the corresponding rollback segments. If the instance crashes before the transaction is committed (or rolled back), transaction recovery is performed while the database is opened the next time by the data from the rollback segments.

https://t.me/learningnets

Technet24

If at the same time or at a later time another transaction visits the data block (which has an open ITL) to get a consistent-read copy (CR), the transaction looks up the transaction table to find the status of the transaction. If the transaction is uncommitted, the second transaction creates a copy of the data or index block in memory, gets the UBA from the ITL, reads the data from the undo, and uses it to roll back the change defined by the undo. If the undo doesn't populate the ITL with an SCN, or if the SCN is too old, another copy of the data block is made and the undo is read from the undo block once again to undo the next change. The UBA in the transaction table is used to roll back the entire transaction as a result of a rollback command, process failure, or shutdown immediate command.

If the status of the transaction in the table is committed, the transaction is deemed committed. Now the rows are no longer locked by the transaction, but the lock byte in the row header isn't cleared until the next time the database performs a DML operation on the block. The lock byte cleanout is piggybacked with the DML operation.

The block cleanout is delayed by some discrete time interval because of the fast commit—this is called *delayed block cleanout*. This cleanout operation closes the open ITLs for the committed transactions and generates the redo information, because a block cleanout may involve updating the block with a new SCN. This is why you see the redo generation for some SELECT statements.

Table Locks

Table locks (TM) are DML locks that protect entire tables. A transaction acquires a table lock when a table is modified by one of the following statements: INSERT, UPDATE, DELETE, SELECT with the FOR UPDATE clause, or LOCK TABLE. A table lock can be held in any of several modes: null (N), row share (RS), row exclusive (RX), share lock (S), share row exclusive (SRX), or exclusive (X).

Messaging in Oracle RAC

The synchronization effort to achieve parallel processing among nodes ideally uses a high-speed interconnect linking the parallel processors. For parallel processing within a single node, messaging isn't necessary; the database uses the shared memory instead. The DLM handles messaging and locking between nodes. Interrupts are used when more than one process wants to use the processor in a uniprocessor architecture. Shared memory and semaphores are used when more than one process wants to communicate in a symmetric multiprocessing (SMP) system.

In Oracle RAC, GES uses messaging for inter-instance communication. The implementation of inter-process/inter-node communication is done by messages and asynchronous traps (ASTs). The database uses messaging for both intra-instance communication (between the processes of the same instance on the same node) and inter-instance communication (between processes on other nodes).

The LMON process of an instance communicates with the other LMON processes on other servers and uses the same messaging framework. Similarly, the LMD process from one instance communicates with other LMD processes using messages. Any

https://t.me/learningnets

process's lock client performs direct sends.

Although the GRD is a part of the RDBMS kernel, processes requiring the lock handle on the resource do not directly access the resource directory. This is done by a background message from the requesting instance's LMD to the master instance.

Once the message is received, the lock handle in the GRD is updated with the new information. The returning message confirms the acknowledgment of the grant for a set of operations on a resource. This helps in deadlock detection and avoidance. You can obtain the messaging traffic information from the fixed view V$GES_MISC.

Three-Way Lock Messages

GES uses messaging or inter-instance and inter-process communication. Inter-instance messaging is for passing messages between the LMON process from one instance and the LMON process of another instance. Intra-instance or inter-process communication is for when LMON wants to communicate with the LMD process.

In inter-instance messaging, a maximum of three parties are involved. Let's review a typical three-way communication. The three-way lock message involves up to a maximum of three nodes—namely, the master (M) instance, holding (H) instance, and requesting (R) instance. Figure 11-5 shows the sequence of messages, where the requesting instance R is interested in block B1 from the holding instance H. The resource is mastered in master instance M.

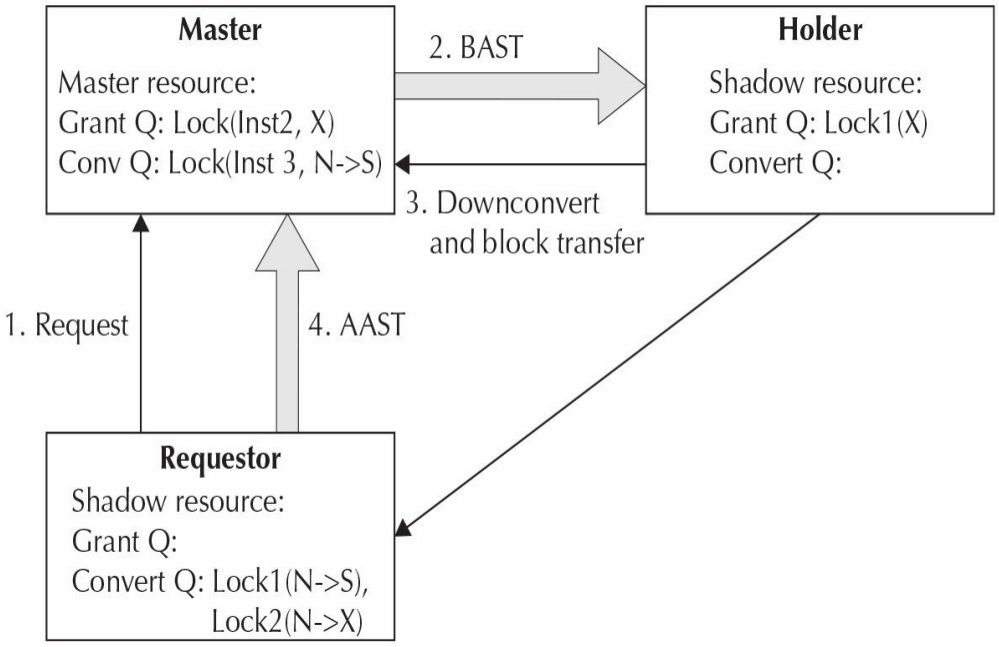


FIGURE 11-5. *Three-way lock messages*

1. Instance R gets the ownership information about a resource from the GRD, which maintains details such as lock ownership and current state. Instance R then sends the message to master instance M, requesting access to the resource.

https://t.me/learningnets
Technet24

The database sends this message via a direct send because it is critical.

2. Instance M receives the message and forwards that request to holding instance H. This message is also sent via a direct send. This message to the holder is known as a *blocking asynchronous trap (BAST)*, which we discuss in the next section.
3. Instance H sends the resource to instance R. It uses the high-speed interconnect to transfer the resource. The database copies the resource to the requesting instance's memory.
4. Upon receipt of the resource and lock handle on that resource, instance R sends an acknowledgment to instance M. The database queues the message for propagation; it isn't a critical message. This process is called *acquisition asynchronous trap (AAST)*, and we discuss it in the next section.

Asynchronous Traps

When a process requests a lock on a resource, the GES sends a blocking AST (BAST) to notify the holder, which is another process that currently owns locks on that resource in incompatible modes. When the holder receives the BAST from the requestor, owners of locks can relinquish them to permit access by the requester.

When the requesting process obtains a lock, an acquisition AST (AAST) is sent to tell the requester that it now owns the lock. LMD or LMS delivers ASTs to the process submitting the lock request. This is done by sending all messages to the LMD on the remote node, which then forwards the messages to the actual waiting process in earlier versions.

TX Enqueue Affinity Optimization

An instance that starts the transaction will become the master of the TX enqueue for it by mastering those resources. This optimization seeks to reduce the TX enqueue waits on the node running active transactions. This resource mastership optimization is an exception to the hashing-based resource mastership.

The optimization currently works with 16 nodes, and when there are more than 16 nodes, the optimization is turned off. However, you can enable this by increasing the value of the parameter **_lm_tx_delta** to a higher value to suit the total of instances. When the event "enq: TX – contention" is among the top wait events in an Oracle RAC environment, it may be because transactions are waiting for remotely mastered TX enqueues.

By default, TX enqueues should be locally allocated, but in some circumstances the database doesn't enforce that optimization. Currently, when the value of **cluster_database_instances > _lm_tx_delta**, the TX optimization is disabled. Because the value of the _lm_tx_delta (undocumented) initialization parameter defaults to 16, systems with more than 16 instances or with **cluster_database_instances** set too high may experience this kind of wait. However, this shouldn't be an issue in the Oracle RAC 11.2 release onward, and therefore you

https://t.me/learningnets

may not find a need to increase the value of this parameter for Oracle RAC 12*c*.

Messaging Deadlocks

Deadlocks can occur if one process (A) is waiting to send a BAST to another process to acquire a lock, and another process (B) is waiting on the lock that the waiting process holds. In this condition, the first process (A) will not be checking on BASTs, so it will not see that it is blocking another process. Because too many writers are trying to send messages (known as *BAST-only messages)* and no one is reading messages to free up buffer space, a deadlock occurs.

Message Traffic Controller (TRFC)

GES depends on messages for resource management. These messages are typically small (128 bytes), and the interconnect plays a great role if the number of messages is high, depending on the number of resources. Ideally, the interconnect should be lightweight or low latency so that it can reduce deadlocks. To streamline the messaging traffic and to avoid deadlocks, the message traffic controller introduces a ticketing mechanism for a smooth flow of messages.

Normally, messages are sent with the message sequence number along with tickets that control the message flow. This way, the receiver ensures that all the messages are received from the sender. The TRFC controls the DLM traffic between all the nodes in the cluster by buffering sender sends (in case of network congestion) and making the sender wait until the network window is large enough to accommodate the traffic.

The traffic controller will keep a predefined number of tickets in the reserve pool. The size of the ticket pool is dependent on the function of the network send buffer size.

Any process that sends messages should acquire a ticket before sending the messages, and then return the ticket to the pool once the message is sent. The database releases tickets used by the receivers (LMS or LMD) back into the pool, according to the remote receiver report of how many messages the remote receiver has received. This is similar to the transaction slot entry in the data blocks, where any DML lock should get the ITL slot before modifying the block.

If no tickets are available, the database buffers the messages at the sender side and sends them when a ticket is available. You can control the number of tickets with the hidden initialization parameter **_lm_tickets**. You normally don’t need to tweak this parameter, unless a big send queue is affecting system performance.

You can obtain the ticket availability and send queue details from the V$GES_TRAFFIC_CONTROLLER view. A node relies on messages to return from the remote node to release tickets for reuse. This ticketing mechanism allows for a smooth flow of messages.

The following query provides a snapshot of ticket usage details:

https://t.me/learningnets

Technet24

```
SQL> SELECT LOCAL_NID LOCAL ,REMOTE_NID REMOTE,TCKT_AVAIL AVAILABILITY,
  2  TCKT_LIMIT LIMIT,SND_Q_LEN SEND_QUEUE,TCKT_WAIT WAITING
  3  FROM V$GES_TRAFFIC_CONTROLLER;

     LOCAL     REMOTE AVAILABILITY      LIMIT SEND_QUEUE WAITING
---------- ---------- ------------ ---------- ---------- ----------
         0          1          750       1000          0 NO
         0          1          750       1000          0 NO
         0          1          750       1000          0 NO
```

Oracle RAC databases employ another mechanism to help avoid running out of tickets. If the number of available tickets goes below 50, the “active send back” feature is enabled for aggressive messaging. This helps the traffic controller maintain speed. The number of tickets that triggers active send back is configurable via the **_lm_ticket_active_sendback** parameter.

You can run the preceding query to monitor ticket usage and availability. Pay particular attention to the TCKT_WAIT column, which shows YES when messages are waiting for tickets. If it is waiting for tickets, you need to check the TCKT_LIMIT and TCKT_AVAIL columns, which show the ticket limit and ticket availability at that time, respectively.

In extreme situations, the preceding query may hang, because to get information about ticket availability, tickets are required. Under those conditions, the sessions will be waiting for the “KJCTS: client waiting for tickets” wait event. Alternatively, using the **lkdebug -t** option to **oradebug** can also dump the ticket information to the trace files:

```
oradebug setmypid
oradebug unlimit
oradebug lkdebug -t
```

Global Cache Services

GCS also uses locks to coordinate shared data access by multiple instances. These are different from the enqueue locks used by GES. These GCS locks protect only data blocks in the global cache.

A GCS lock can be acquired in shared mode or exclusive mode. Each lock element can have the lock role set to either local or global. If the lock role is local, the block can be handled as it is usually handled in a single instance. Local mode reads the block from the disk if it is not seen in the buffer cache and can write the block to disk when it holds the lock mode X.

In a global role, three lock modes are possible—shared, exclusive, and null. The instance can modify the block only with the exclusive (X) mode. When the lock role is global, the instance cannot read the block from the disk as it does in single-instance mode. It has to go through the master and can write to or read from the disk only when directed by the master. Otherwise, it has to wait for the buffer from the other instance.

https://t.me/learningnets

GCS maintains the lock and state information in the SGA. It is stored in an internal data structure called *lock elements*. It also holds a chain of cache buffers that are covered by the corresponding lock element. The lock elements are exposed as fixed table view V$LOCK_ELEMENT, which shows the lock element, the buffer class, and the state of the buffer.

Lock Modes and Lock Roles

A lock mode describes the access rights to a resource. The compatibility matrix is cluster-wide. For example, if a resource has an S lock on one instance, an X lock for that resource cannot exist anywhere else in the cluster. A lock role describes how the resource is to be handled. The treatment differs if the block resides in only one cache.

Cache Fusion changes the use of PCM locks in the Oracle server and relates the locks to the shipping of blocks through the system via IPC. The objectives are to separate the modes of locks from the roles assigned to the lock holders, and to maintain knowledge about the versions of past images of blocks throughout the system.

Global Lock Modes

GCS locks use the following modes: exclusive (X), shared (S), and null (N). An instance can acquire a global lock that covers a set of data blocks in either shared or exclusive mode, depending on the access type required.

Exclusive (X) lock mode is used during UPDATE or any of the DML operations. DML operations require that the blocks be in exclusive mode. If one instance needs to update a data block that is owned by another instance in an incompatible mode, the first instance asks GES to request that the second instance disown the global lock.

Shared (S) lock mode allows the instance to read (SELECT) blocks. Multiple instances can own a global lock in shared mode as long as they are reading the data. All instances can read the block without any change in the lock state of the other instance. This means instances do not have to disown global locks to allow another instance to read the data block. In other words, reading does not require any explicit locking or lock conversion, whether it is a single instance or a cluster implementation of Oracle.

Null (N) lock mode allows instances to keep a lock without any permission on the block(s). This mode is used so that locks need not be continually created and destroyed. Locks are simply converted from one mode to another.

Lock Roles

Lock roles handle the Cache Fusion functionality. A lock role can be either local or global. The lock role for a resource is local if the block is dirty only in the local cache. The role becomes global if the block is dirty in a remote cache or in several caches.

Initially, a block is acquired in a local lock role with no past images. If the block is modified locally and other instances express interest in the block, the instance holding

https://t.me/learningnets

Technet24

the block keeps a past image (PI) and ships a copy of the block, and then the role becomes global.

A PI represents the state of a dirty buffer. Initially, a block is acquired in a local role, with no PIs present. The node that modifies the block keeps PIs, as the lock role becomes global after another instance expresses interest in this block. A PI block is used for efficient recovery across the cluster and can be used to satisfy a CR request, remote or local.

The node must keep a PI until it receives notification from the master that a write to disk has completed covering that version. The node then logs a block-written record (BWR). The BWR is not necessary for the correctness of recovery, so it need not be flushed.

When a new current block arrives on a node, a previous PI is kept untouched because some other node might need it. When a block is pinged out of a node carrying a PI and the current version, they might or might not be combined into a single PI. At the time of the ping, the master tells the holder whether a write is in progress that will cover the older PI. If a write is not in progress, the existing current block replaces the older PI. If a write is in progress, this merge is not completed and the existing current block becomes another PI. An indeterminate number of PIs can exist.

Local and Global Roles In the local role, only S and X modes are permitted. All changes are on the disk version, except for any local changes (mode X). When requested by the master instance, the holding instance serves a copy of the block to others. If the block is globally clean, this instance's lock role remains local. If the block is modified by this instance and passed on dirty, a PI is retained and the lock role becomes global. The lock mode reads from disk if the block is not in the cache and may write to the block if the lock mode is X.

The local role states that the block can be handled similarly to the way it is handled in single-instance mode. In the local role, the lock mode reads from disks and writes the dirty block back to disk when it ages out without any further DLM activity.

In the global lock role, the possible lock modes are N, S, and X. When the lock role for a block is global, that block may be dirty in any of the instances, and the on-disk version may be obsolete. Therefore, interested processes can modify the block only with mode X. The instance cannot read from disk because it is not known whether the disk copy is current. The holding instance may send a copy to others when instructed by the master. The instance may write a block in X mode or the PI, and the write requests must be sent to the master.

Lock Elements A lock element (LE) holds lock state information (converting, granted, and so on). LEs are managed by the lock processes to determine the mode of the locks. LEs also hold a chain of cache buffers that are covered by the LE and allow the Oracle database to keep track of cache buffers that must be written to disk in case an LE (mode) needs to be downgraded (X → N).

Figure 11-6 shows the LEs and hash chains.

https://t.me/learningnets

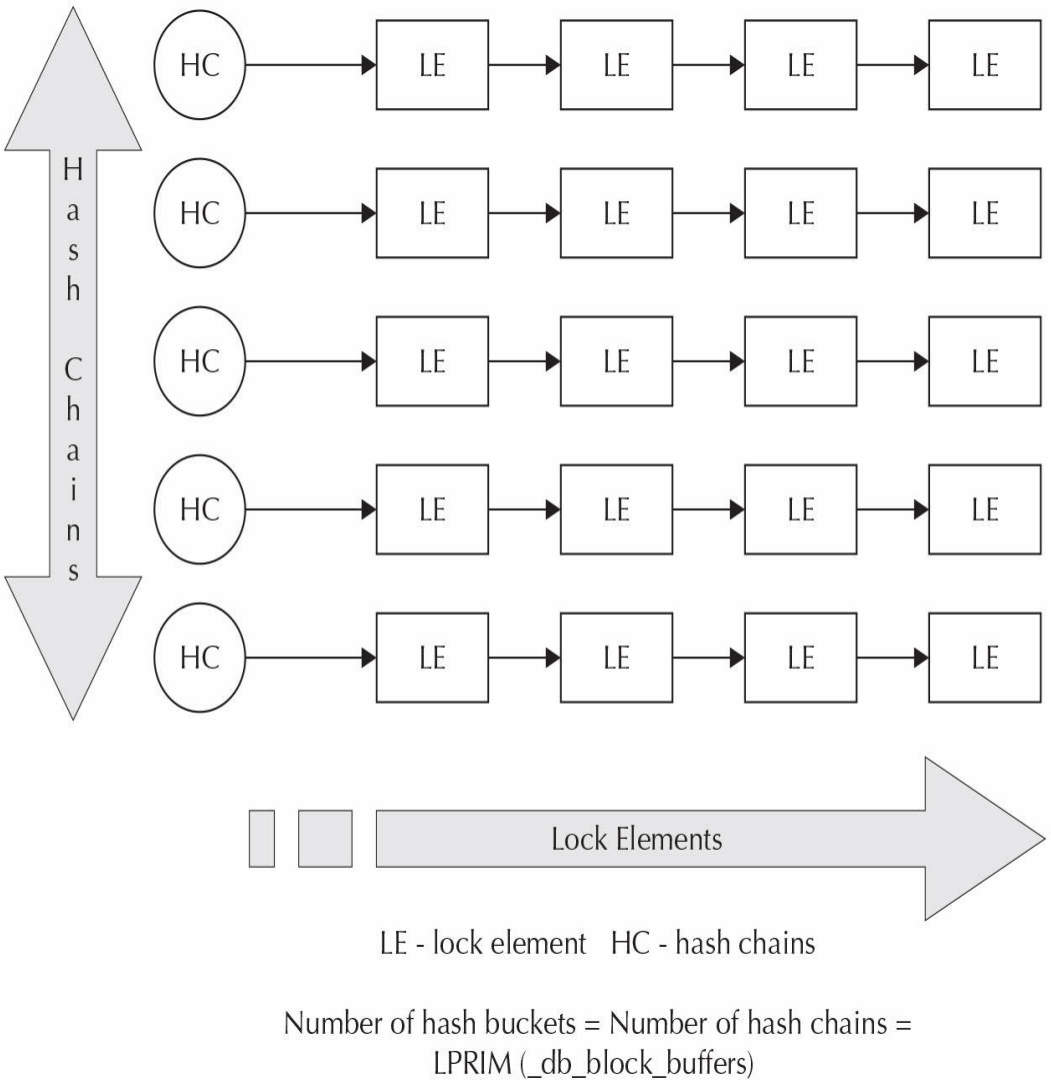


FIGURE 11-6. *Hash chains and lock elements*

LEs protect all the data blocks in the buffer cache. The following describes the classes of the Oracle data blocks, which are managed by the LEs using GCS locks. The class of the lock elements directly maps to X$BH.CLASS. The column state of X$BH can contain one of the following values:

- 0 or FREE
- 1 or EXLCUR
- 2 or SHRCUR
- 3 or CR
- 4 or READING
- 5 or MRECOVERY
- 6 or IRECOVERY
- 7 or WRITING

https://t.me/learningnets

Technet24

- 8 or PI

GCS manages PCM locks in the GRD. PCM locks manage the data blocks in the global cache. Because the buffer cache is global, if a buffer is not found in the local cache, a block may be cached in the other instance's buffer cache. If GCS doesn't find the block in any of the instances, it instructs the requesting instance for a disk read.

The GCS monitors and maintains the list and mode of the blocks in all the instances. Based on the request and availability, the GCS asks the holding instance to ship the block to the requesting instance, or instructs the requesting instance to read the block from the disk. All types of blocks are copied between instances in a coherent manner, and requests can be for write or read.

The database blocks from the shared database can be cached in the buffer cache of any instance that is used by the application. A local cache miss will consult the GCS to determine whether a requested block is cached by another instance. If the block is in cache, the GCS will inform the instance which node holds a lock on the block (the potential holder) of the request. If that node still has the block in its cache, it will send a copy of the block to the requestor. If the block is not in the node's cache, the GCS will grant an access right to the requestor and the block will be read from disk into the requestor's cache. The frequency and distribution of these events depends on the application.

Each database block has a master instance, where the complete lock state of the block is known. Each instance will be a master for some data blocks. Resource instances and master instances have a one-to-one relation. One resource cannot be mastered in more than one instance, but one instance can master more than one resource.

Although more than one instance may use the same set of resources in the Oracle RAC, only one instance is allowed to modify a block. GCS ensures cache coherency by requiring that instances acquire a lock before modifying or reading a database block. GCS locks are *not* to be confused with the row-level locks in single-instance environments. Row-level locks are still used in conjunction with PCM locks.

Row-level locks are independent of GCS locks. GCS locks ensure the block is used by only one instance, and row-level locks manage blocks at the row level. GCS ensures that the current mode buffer is exclusive for one instance and pre-images are allowed for other instances in shared mode.

If another instance needs to read a block, the current version of the block may reside in many buffer caches under shared locks. Thus, the most recent copy of the block in all SGAs contains all changes made to that block by all instances, regardless of whether any transactions on those instances have been committed.

At the same time, if a data block is updated in one buffer cache, the cached copies in other buffer caches are no longer current. New copies can be obtained after the modification operation completes. This is called *CR block fabrication*. Normally CR building occurs when a request is made for a block in that mode for a query. In simple terms, if a query wants a copy of a modified block, a CR copy is built. The process is quite simple and straightforward in a single-instance environment because the process

https://t.me/learningnets

can find all the required information in the local cache. In Oracle RAC, additional processing and messaging take place to have the buffer cache synchronize across all the nodes.

In CR building, the master of a block may be the local instance where the request is made, the instance where the block is cached, or any other instance. In clusters with only two nodes, blocks cached in another instance can be accessed after two hops, because the master is on the requesting mode or the caching node. In clusters with more than two nodes, a block access may require at most three hops but may complete in two hops if the master is the local node or the caching node. If the block is not cached anywhere, the request will complete after two hops.

NOTE

Block access between instances is done on a per-block basis. When an instance locks a block in exclusive mode, other instances cannot access the block in the same mode. Every time Oracle tries to read a block from the database, it must obtain a global lock. Ownership of the lock is thus assigned to the instance. Block locks are also known as BL type enqueues, which look like enqueues but aren't.

Consistent Read Processing

A SELECT doesn't require any lock in the Oracle RDBMS, which means that readers never block writers. The consistent read (CR) mechanism in the RDBMS kernel enables read consistency across the instances.

Read consistency ensures that the behavior of data returned by a query or transaction is consistent from the start of the query to the end of the query or transaction. At the same time, the transaction sees its own committed and uncommitted changes. Read consistency is guaranteed with the help of undo segments. The consistent read version of blocks is called *CR blocks*, and its processing is called *CR processing*.

During the start of the query or transaction, the database obtains the Snap SCN (the SCN of the block during that particular point in time) along with the UBA. During the transaction, any time the other interested session is looking for the CR buffer, the database obtains the block version at a particular point in time from the undo segments.

The interested process clones the buffer and scans the undo chain to construct the CR copy of the buffer at a point in time. Sometimes the process has to loop to take the buffer back to the required point in time. The information to go back to a previous point in time is available from the undo segments. The CR building process is the same for a single instance and Oracle RAC.

During CR processing, the database scans one ITL during every pass and applies the undo records to the cloned buffer. Applying undo information sometimes involves

https://t.me/learningnets

Technet24

block cleanout. Once the block is cleaned or no open ITLs are listed, or when the Snap SCN is less than or equal to the query SCN, the block is deemed valid and that buffer is returned as a CR buffer.

The database implements a few rules to make the CR building efficient in both single-instance databases and Oracle RAC. For example, a buffer can't be cloned an infinite number of times because a built-in mechanism limits the number of CR copies per DBA. This controls the CR versions of a hot object filling the buffer cache. The hidden parameter **_db_block_max_cr_dba** limits the number of CR copies per DBA on the buffer cache. However, you don't need this parameter for Oracle Database versions 11.2 onward.

In a multi-instance environment, when a buffer is transferred to another instance during Cache Fusion transfer, the past image buffer is kept in the home instance. Undo records can also be applied on top of this PI buffer during CR processing. CR processing does not generate any redo information.

This process becomes a little complex when the block is held in the other instance's cache. When the requestor wants a CR copy, it sends a message to the holder of the block. On receipt of the CR request, the holder walks through the undo block(s), constructs the CR copy, and ships the block to the requesting instance. Figure 11-7 shows the steps in the CR fabrication process.

https://t.me/learningnets

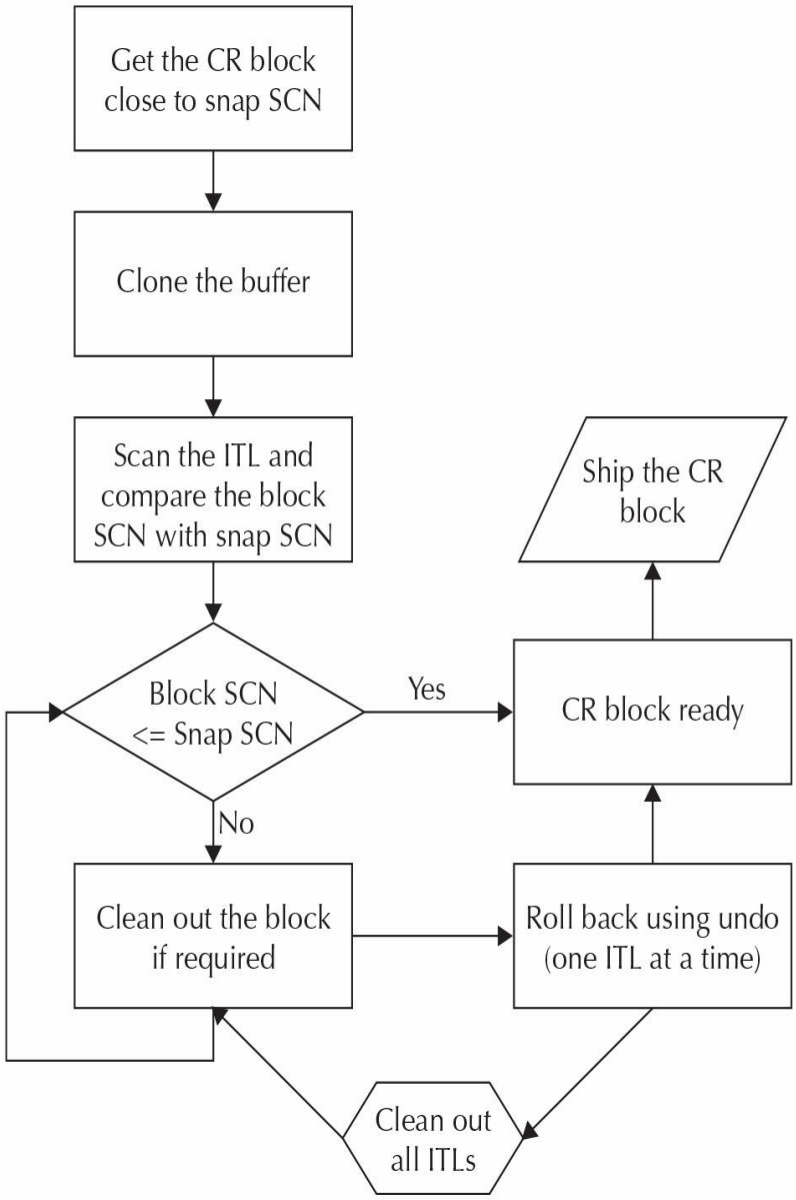


FIGURE 11-7. *CR block fabrication process*

Sometimes creating a CR block to the Snap SCN may involve too much work for the holder (for example, going back too far in time or reading the blocks from disk); in that case, the holder may not construct the CR block because it will be too expensive. The holder does a "favor" for the requestor by constructing the CR block and sending it. In this case, the holder just ships the block to the requesting instance and then CR fabrication is done at the requestor's instance.

Holder construction of CR copies to the requesting instances sometimes impacts the equilibrium of the instances, if one instance is always used for queries. In this case, the reading instance always has the privilege of getting all the required blocks to its cache, because CR fabrication is done by another instance. Because CR fabrication can be expensive when the fabricator has to scan through the chain of undo blocks and read the blocks from disks, the database uses a couple of optimizations to avoid CR request thrashing. One such technique is the *light-work rule*, which prevents the LMS

https://t.me/learningnets

Technet24

processes from going to disk while responding to CR requests for data, undo, or undo segment header blocks. This rule can prevent the LMS process from completing its response to the CR request.

Light-work Rule and Fairness Threshold

When too many CR requests arrive for a particular buffer, the holder can disown the lock on the buffer and write the block to the disk. Then the requestor can read the block from the disk after acquiring the required lock on the object. The process of disowning the block is technically known as *fairness down-convert*.

The number of consecutive requests after the holder converts the lock elements is configurable via the **_fairness_threshold** parameter. This parameter defaults to 2, which is usually enough for most instances. However, it requires a special setting when one instance in the cluster is always used for queries. This parameter has been used as a workaround for some bugs, but lately (from Oracle Database 1.2.0.4 onward), there has been no real need to set or change this parameter.

The light-work rule is invoked when CR construction involves too much work and no current block or PI block is available in the cache for block cleanouts. Any additional disk I/O to serve the CR block will kick the light-work rule. The light-work statistics are recorded in X$KCLCRST.LIGHT1, whereas X$KCLCRST.LIGHT2 is incremented when the buffer is in instance recovery mode.

The number of times the instance performs a fairness down-conversion and the light-work rule is invoked is shown in the V$CR_BLOCK_SERVER view. The V$CR_BLOCK_SERVER view also lists details about the CR request processing and the distribution of block requests:

```
V$CR_BLOCK_SERVER
Name                      Type        Notes
------------------------  ---------   ------------------------
CR_REQUESTS               NUMBER      CR+CUR =Total Requests
CURRENT_REQUESTS          NUMBER
DATA_REQUESTS             NUMBER
UNDO_REQUESTS             NUMBER
TX_REQUESTS               NUMBER      DATA+UNDO+TX= CR+CUR
CURRENT_RESULTS           NUMBER
PRIVATE_RESULTS           NUMBER
ZERO_RESULTS              NUMBER
DISK_READ_RESULTS         NUMBER
FAIL_RESULTS              NUMBER
```

https://t.me/learningnets

```
FAIRNESS_DOWN_CONVERTS    NUMBER     # of downconverts from X
FAIRNESS_CLEARS           NUMBER     # of time Fairness counter cleared
FREE_GC_ELEMENTS          NUMBER
FLUSHES                   NUMBER     Log Flushes
FLUSHES_QUEUED            NUMBER
FLUSH_QUEUE_FULL          NUMBER
FLUSH_MAX_TIME            NUMBER
LIGHT_WORKS               NUMBER     # of times light work rule evoked
ERRORS                    NUMBER
```

You can obtain the number of times a down-convert occurs and the light-work rule is invoked from the instance startup by executing the following query:

```
SQL> SELECT CR_REQUESTS, LIGHT_WORKS ,DATA_REQUESTS, FAIRNESS_DOWN_CONVERTS
  2 FROM
  3 V$CR_BLOCK_SERVER;

CR_REQUESTS LIGHT_WORKS DATA_REQUESTS FAIRNESS_DOWN_CONVERTS
----------- ----------- ------------- --------------------
      80919        5978         80918                17029
```

When the data-request-to-down-convert ratio is more than 40 percent, lowering the value of the _fairness_threshold parameter value may improve the performance and significantly reduce the interconnect traffic for the CR messages. We're presenting this information mainly for historical purposes, as you don't need to do this for newer releases (especially since Oracle Database 11.2.0.4). This parameter can be set to 0 when the systems are used for query purposes only. Setting the _fairness_threshold parameter to 0 disables the fairness down-converts, but this isn't a recommended practice.

Lost Block Optimization

In Oracle 11*g*, lost blocks handling was improved to eliminate false lost blocks resulting from scheduling delays. Also, side-channel batching was improved to accommodate the increase in logical side-channel messages. A new algorithm was added to favor disk reads over CR serving once a block has been lost. This quickly reduces the network congestion from lost block retries. The scheme is gradually switched back to CR serving based on load, I/O rate, and CR receive time.

Global Cache Services detects lost blocks using side-channel messages. Lost blocks and retransmits for the lost blocks may increase the load significantly, especially when the system is loaded fully and the lost blocks are part of a multiblock request. The side channel batch timeout is handled by the initialization parameter **_side_channel_batch_timeout_ms**, which defaults to 500 ms. This parameter specifies the timeout (in milliseconds) before shipping out the batched side-channel messages.

https://t.me/learningnets

Technet24

GCS Resource Mastering

GRD is like a central repository for locks and resources. It is distributed in all the nodes, and no single node has information about all the resources. Only one node maintains complete information about one resource, and that node is called the *master node* for that resource. Other nodes need to maintain information only about locally held locks on that resource. The process of maintaining information about the resources is called *lock mastering* or *resource mastering*.

GCS resources and locks share a one-to-many relationship in the GRD. There's one "resource" initiated for every data block address (DBA) present in the global cache, and more than one lock can be allocated for that resource. Each instance that has a copy of the block in its cache may hold a lock on that resource.

In other words, the GCS lock structure is physically located on each node of the cluster interested in that data block. The resource lock is allocated only in the master node. The master node is responsible for synchronizing access on that resource to the other nodes, and the master node for the resource is known as the *resource master*.

Resource Weight

You can define the "weight" of an instance as the number of resources it is "willing to master." By default, this weight is the number of resources an instance can accommodate in its SGA with respect to the other instances. The database computes the weights during startup and then broadcasts them to the member nodes as well as publishes them in the LMON trace files. This discussion is purely for academic purposes because Oracle RAC handles the resource allocations automatically and seamlessly. The following is an extract from the LMON trace showing the publication of the resource weight.

```
Name Service frozen
kjxgmcs: Setting state to 0 1.
kjxgrdecidever: No old version members in the cluster
kjxgrssvote: reconfig bitmap chksum 0x6694d07e cnt 1 master 1 ret 0
kjfcpiora: published my fusion master weight 39280
kjfcpiora: published my enqueue weight 64
kjfcpiora: publish my flogb 3
kjfcpiora: published my cluster_database parameter=0
kjfcpiora: publish my icp 1
```

NOTE

All of the calculations shown in this section are for understanding the behind-the-scenes work the database performs. It's not recommended that you set the initialization parameters and lock settings and values shown here any longer because

https://t.me/learningnets

the database automatically does that for you.

The following formula helps calculate the GCS resources. A minimum of 2500 resources are allocated in the shared pool if the calculated resources are less than 2500:

```
nres = (pcmlocks + pcmlocks/10);
                if (nres <= 2500)
                  nres = 2500;
```

You can query the GV$RESOURCE_LIMIT view to find out the details about the GCS resources:

```
SQL> SELECT RESOURCE_NAME,LIMIT_VALUE
     FROM GV$RESOURCE_LIMIT
     WHERE RESOURCE_NAME='gcs_resources';
```

You can describe the default relative weight with the following function; the value of the relative weight is between 0 and 1:

$$\text{Relative Weight (i)} = \frac{\text{SizeInBlocksOfCache(Local)}}{\sum \text{SizeInBlocksOfCache(Global)}}$$

If the initialization parameters for SGA are the same across all the instances, all the instances will have the same relative weight. When the relative weight is the same, the resources in the GCS layer are evenly mastered. If the buffer cache and shared pool settings are different from one another, the relative weight will be different and the number of resources they master will also be different.

Resource Master

The database uses a composite HASH*MAP integer function as the rule that associates the data block address (DBA) with instance numbers. Under normal conditions, each active instance of the cluster will receive the mastership of a number of resources proportional to the weight it has, as shown here:

```
Instance = MAP(HASH(M,RESOURCE)) = n
```

In this case, **M** is a number that is a multiple of **n**, and **n** is the maximum number of instances, as specified by the initialization parameter **cluster_database_instances. MAP** is a discrete function specified by an array.

The construction of **MAP(.)** is done by taking into account the relative weight of each instance.

Modifying the Mastership Distribution

Normally, the number of resource “slots” preallocated is proportional to the number of

https://t.me/learningnets

Technet24

cache buffers defined for the local cache. This simple rule guarantees that enough resource slots will be preallocated globally to accommodate all the resources that'll protect the DBAs (data block addresses) present in the global cache during the activity. Actually, the total number of resources is 10 percent greater than the total number of buffers defined in the global cache. When modifying the default distribution, you will need to take into account this aspect to avoid excessive recourse to freelists in the shared pool.

It's possible to force the number of resource slots a node publishes and hence remodulate the WEIGHT function using the parameter **_gcs_resources**, which otherwise defaults to the number of local buffers in the cache. Setting **_gcs_resources** to 0 totally disables resource mastering for that particular node.

Read-Mostly Locking

Starting with Oracle 11*g*, a new and improved locking mechanism helps optimize the performance of applications with very little read-write contention. The database can handle most of the lock requests on an object (or partition) known for reads better at the GCS layer by pre-granting read access to all the blocks. This will reduce the locking overhead associated with shared read access. This also reduces the messaging and CPU costs associated with block grants.

However, current block (write) access on these objects mastered by this protocol would come up with a slightly greater cost than the standard Cache Fusion transfers. This optimization is called *read-mostly locking*.

Internals of Read-Mostly Locking

The Global Cache Layer maintains the object-level statistics for Shared and Exclusive lock requests and invokes the read-mostly policy for the objects for the standard Dynamic Resource Mastering protocol.

Theoretically, an object can transform into a read-mostly object in as short a period as 10 minutes based on the access, and it can be taken out of read-mostly status within the next 10 minutes if the access pattern changes. If the object is a standard read-only object, it's always better to place it into the read-only tablespace. Placing the object in the regular read-only tablespace eliminates the GCS locking overhead and yields better performance than read-mostly locking.

When an object becomes read-mostly, any instance can just go ahead and read the blocks belonging to that object without waiting for grants from the master, because the shared (S-affinity) access on that object is pre-granted to all the nodes. This reduces the lock requests, grant messages, and block transfers on that object. The read-mostly status of the object is dissolved when there is a write request on the object, and this is done through the lock element called *anti-lock*.

When a write request is made on the read-mostly object, GCS broadcasts a message to all instances asking them to open an anti-lock on the object. When an anti-lock is present on an object, all access to that block will be controlled through the standard

https://t.me/learningnets

Cache Fusion protocol, even if the object is read-mostly. Anti-locks are dissolved when the object is taken out of read-mostly status.

Keep in mind that when the object is read-mostly, you will see an increase in disk I/O because the blocks on that object are *always* read from disk, although they are cached in another instance's buffer cache. Therefore, if your system is I/O bound, read-mostly locking will complicate the situation, so we recommend that you turn off the read-mostly feature. You can disable the read-mostly feature by setting the parameter **_gc_read_mostly_locking** to FALSE. This is a workaround for several known issues where you might notice a database hang.

Resource Affinity and Dynamic Resource Remastering

The Global Resource Directory includes a feature called *resource affinity* that allows for the mastering of the frequently used resources on its local node. It uses dynamic resource remastering to move the location of the resource masters. This technique optimizes the system for situations where the database is always executing certain transactions on one instance. This happens in an application-partitioning environment or an activity-based, load-balancing environment.

When activity shifts to another instance, the resource affinity will correspondingly move to the new instance. If activity isn't localized, the resource ownership is distributed to the instances equitably.

Dynamic resource remastering is the ability to move the ownership of a resource between instances of Oracle RAC during runtime without affecting availability.

When one of the nodes in an Oracle RAC cluster frequently accesses a resource or a set of resources compared to the other nodes, it would be better to have that resource mastered by the frequently requesting instance.

In normal conditions, a resource can be mastered only during instance reconfiguration, which happens when an instance joins or leaves the cluster. Other than a node joining or leaving the cluster, the ownership is remastered from one node to another node (outside the reconfiguration jurisdiction) based on the frequency of the usage. This online resource mastership change is called *dynamic remastering*.

Some applications perform an application partitioning, and this means that the database will always direct a set of requests to one particular node. This happens when application partitioning is implemented, and also where some third-party load balancers are employed to route the connections based on the user or application type.

In both cases, one node may always be requesting a particular subset of resources. During these situations, it'll be beneficial to have those resources mastered at the same instance. This reduces interconnect messaging traffic as the resources are mastered by the requesting instance. The resource can be of an object or file, based on the version of Oracle.

Currently remastering occurs at the object level, which helps fine-grained object remastering. For example, table partitions will have a different object ID and can be remastered to different instances. Figure 11-8 shows dynamic remastering.

https://t.me/learningnets

Technet24

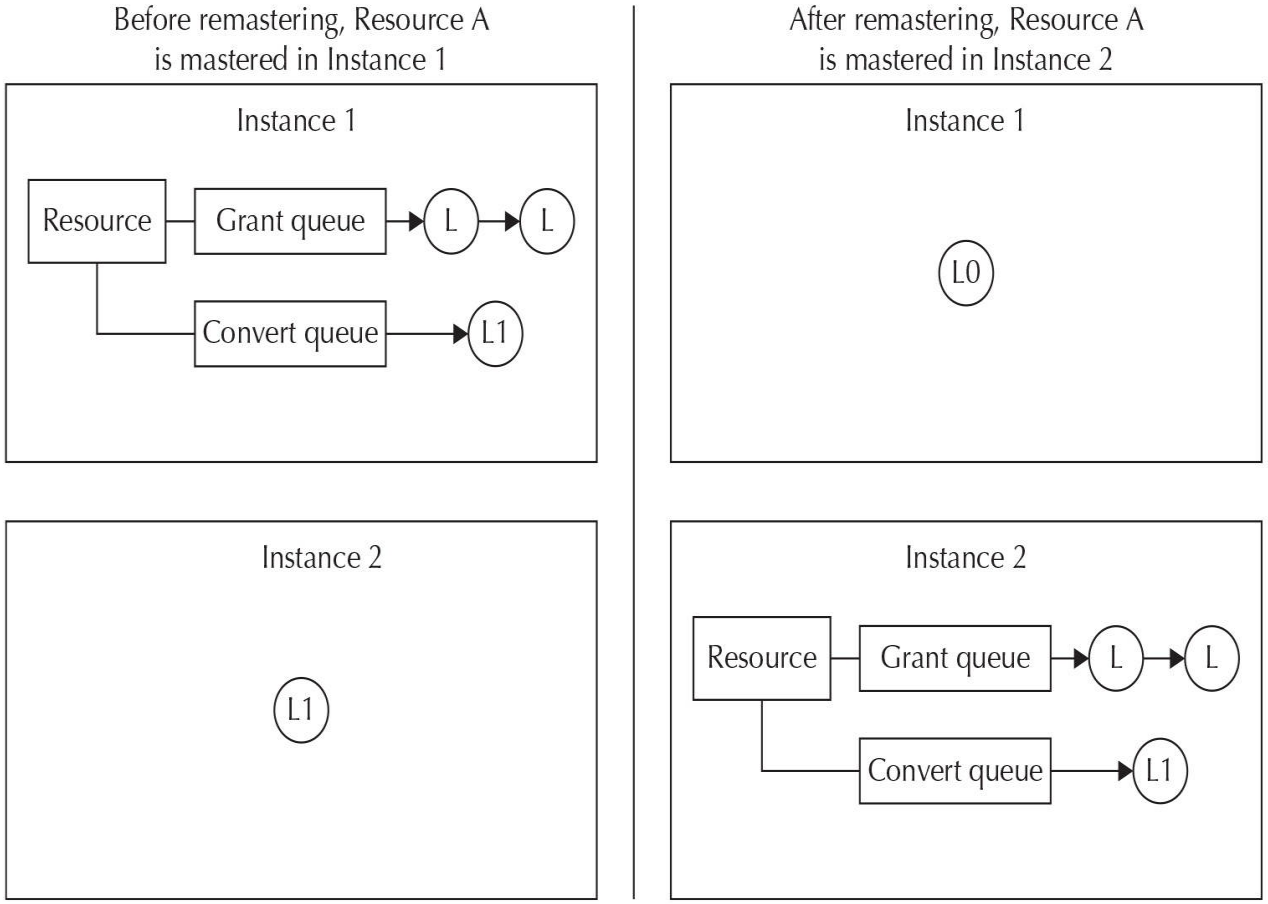


FIGURE 11-8. *Dynamic remastering*

Dynamic Resource Mastering Policy

Oracle RAC performs remastering only under very stringent conditions. The database evaluates the resources and their access patterns at predefined intervals.

The database only considers those resources that are heavily accessed as candidates for resource mastering. Once it identifies the resources as hot or frequently used resources, those resources become eligible for a remastering evaluation. The following parameters influence the dynamic resource mastering protocol.

- **_gc_policy_time** This parameter controls the dynamic mastering window for candidate selections. On every window of **_gc_policy_time**, the object-level statistics are sampled to select the candidates for remastering. The database analyzes the access patterns at this interval and moves the objects to the DRM (Dynamic Resource Manager) queue. The database also evaluates the decisions to establish or dissolve read-mostly locking at this interval.

  This parameter defaults to 10 minutes, which is usually good for most of the systems. However, if you notice the dynamic resource mastering happening too frequently, consider increasing this to a higher value. Setting this parameter to 0 disables the object affinity and read-mostly policy enablement, which

https://t.me/learningnets

disables the remastering in practical terms.

- **_gc_affinity_ratio** This parameter works as a filter condition during object selection. To get into the selection queue, a node must access the object more than the number of times specified by this parameter. This parameter defaults to 50, which means that a node should access an object 50 times more than the other node to master the interested resource. This simulates the application partitioning at the resource directory.
- **_gc_policy_minimum** This parameter works in conjunction with **_gc_affinity_ratio** during object selection. It defines the minimum number of global cache operations per minute to qualify for affinity or read-mostly selection. The parameter's value defaults to 2400 global cache operations (open, convert, and close) per minute, helping to pick the hot objects in the GRD.

  Depending on the access distribution of objects in the GRD, you may want to increase or decrease the default value of this parameter. Raising the parameter's value reduces the number of objects evaluated for dynamic remastering.
- **_gc_read_mostly_locking** This parameter defaults to TRUE, which means that the database will consider the read-mostly policy evaluation during the DRM window. You should carefully benchmark any changes to this parameter.
- **_gc_transfer_ratio** This parameter works in conjunction with **_gc_read_mostly_locking**. An object is read-mostly when the number of block transfer is less than 50 percent of the number of disk reads. This parameter defaults to 2 (which is 50 percent), and changing this to a lower value usually reduces the likelihood of an object becoming read-mostly.
- **_lm_drm_max_requests** At the dynamic remastering policy evaluation interval, if the objects selected for affinity or read-mostly are less than 100, the database processes them in a single batch. If more than 100 objects are queued for evaluation, the database handles them in batches, and this initialization parameter limits the number of objects per batch. This parameter defaults to 100, and there is no need to change it unless Oracle Support recommends doing so.

Object Affinity Statistics

GRD maintains the object-level statistics in the SGA, and you can query the X$OBJECT_POLICY_STATISTICS view to check out the statistics. You can query them at any time to find out the global cache operations on a particular object. LCK0 maintains these statistics, and the database uses them for read-mostly and object affinity evaluation.

The database refreshes this information every policy interval, which is 10 minutes. So, when you query the X$OBJECT_POLICY_STATISTICS view, the value you see is the object affinity statistics from the last policy evaluation, which is less than or

https://t.me/learningnets

Technet24

equal to 10 minutes. You can query the view V$GCSPFMASTER_INFO to list the objects under the DRM protocol (affinity and read-mostly). The following shows the interesting columns for DRM analysis:

```
Name                                      Null?    Type
----------------------------------------- -------  -----------
INST_ID                                            NUMBER
FILE_ID                                            NUMBER
DATA_OBJECT_ID                                     NUMBER
GC_MASTERING_POLICY                                VARCHAR2(11)
CURRENT_MASTER                                     NUMBER
PREVIOUS_MASTER                                    NUMBER
REMASTER_CNT                                       NUMBER
```

You can join DATA_OBJECT_ID with DBA_OBJECTS.DATA_OBJECT_ID to get the object_name. When you set the initialization parameters **_gc_undo_affinity** and **_gc_undo_affinity_locks** to TRUE (the default), the database also lists undo segments in this view. Undo segments will have an object ID value of 4294950912+USN. For example, undo segment 8 will have the object ID 4294950920 in the DATA_OBJECT_ID column.

The GC_MASTERING_POLICY column will contain the type of mastering policy (affinity or read-mostly). The columns CURRENT_MASTER and PREVIOUS_MASTER list the current and previous masters of that object, respectively. When the value of the CURRENT_MASTER column is 32767, the object has dissolved its affinity or read-mostly during the latest policy evaluation. A value of 32767 for the PREVIOUS_MASTER column means the object has never been in the object affinity table in the evaluation period.

The REMASTER_CNT column lists the number of times the object was remastered. This view does not keep the record of dissolved objects, and the statistics are reset when they are removed from the affinity table.

Dynamic resource remastering requires the acquisition of the RM (remastering) enqueue at the exclusive level. The RM enqueue is a single-threaded enqueue, and only one enqueue exists per database. So, at any point in time, only one node can be performing the remastering operation, and the other node must wait for the current node to complete the operation.

Once the node acquires the RM enqueue, it broadcasts the message to all other nodes and starts the remastering of resources selected based on the candidate selection algorithm. If too many resources are queued for remastering, the remastering occurs in batches. Up to 100 resources are selected for remastering after acquisition of the RM enqueue. The LMD processes from all the instances request the corresponding LMON process on the respective instances to remaster the resource, and the database updates the GRD to reflect the current resource ownership.

Figure 11-9 explains the remastering process.

https://t.me/learningnets

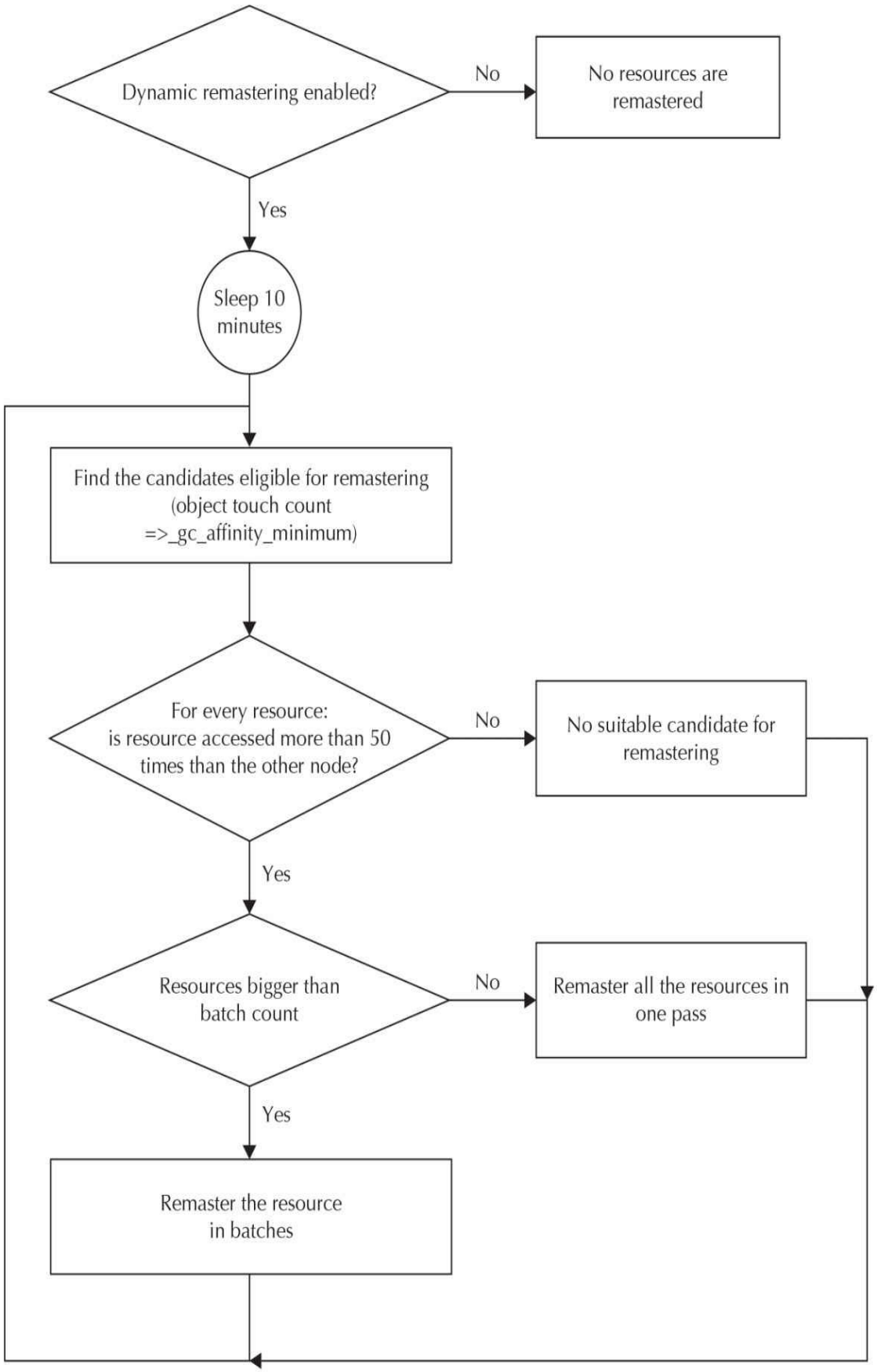


FIGURE 11-9. *Flowchart of dynamic remastering*

The Oracle RAC database initially remasters resources to instances based on a

https://t.me/learningnets

Technet24

simple hashing algorithm. In a busy cluster, resource remastering is expensive in terms of the messages, and resource ownership transfers to the other instances.

For benchmarking, or in an application-partitioned environment that doesn't require dynamic remastering, you can disable dynamic remastering by setting the **_gc_policy_time** hidden initialization parameter to 0.

Summary

Congratulations! You have just completed learning about the foundations of resource management in Oracle Real Application Clusters. Global Resource Directory consists of Global Enqueue Services and Global Cache Services, and it is the central reference in resource management in Oracle RAC.

Messaging plays a key role in the lock management and conversion operations. Resources are placed in grant and convert queues, and the traffic controllers avoid message traffic deadlocks with the ticketing mechanism. This chapter also looked at Oracle's implementation of row-level locks and its CR block fabrication process.

Affinity and read-mostly mechanisms in dynamic resource mastering help GCS to master the resources close to the operation. We have explored the algorithms as well as the inner workings and references of dynamic resource mastering. Next up is a closer look at Cache Fusion, which is the foundation for high performance in Oracle RAC systems.

https://t.me/learningnets

CHAPTER 12

A Closer Look at Cache Fusion

A scalable, high-performance system should spend as little time as possible in transmitting messages, and use the most efficient communication methods for I/O processing. Cache Fusion makes this possible. In this chapter, we look under the hood of Cache Fusion to understand the internal operations of various kinds of block transfers that occur between instances. Understanding the internal operations of the block transfers will help you appreciate the intelligence built into this component of Real Application Clusters as well as aid you in designing scalable database applications.

We also take a look at how and why Oracle moves data blocks from one instance to another instance in an Oracle RAC environment. We also explain concepts closely associated with the Cache Fusion technology and wherever possible provide relevant database instance statistics.

Oracle RAC appears to merge the database buffer cache of multiple instances so that the end user sees only a single large buffer cache. In reality, the buffer caches of each node remain separate; copies of data blocks are shared through distributed locking and messaging operations. A data block present in an instance's buffer cache can be copied to another instance's buffer cache over the high-speed interconnect, thereby eliminating a reread from disk by other instance(s). Figure 12-1 depicts the

https://t.me/learningnets

Technet24

notion of a single large buffer cache spread across all the database instances among the nodes of a cluster.

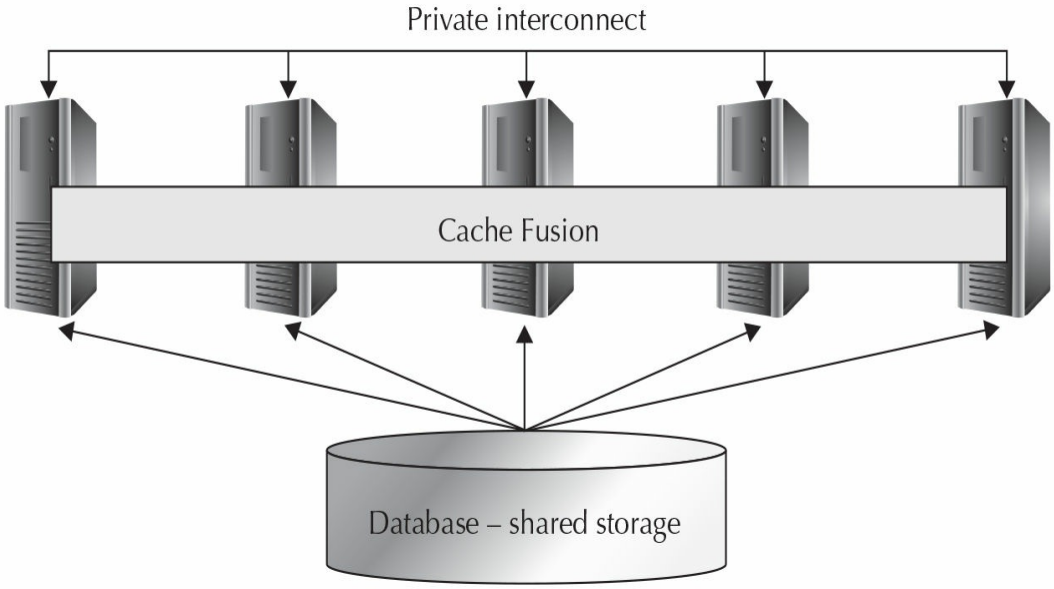


FIGURE 12-1. *Cache Fusion creates merged buffer caches of participating instances*.

In Oracle Database version 8.1.5, Oracle introduced the idea of sharing data using private interconnects between the nodes, an entity that was meant only for messaging purposes in previous versions. This protocol is known as *Cache Fusion Protocol*. Data blocks are shipped throughout the network, similar to messages, thus reducing the most expensive component of data transfer (disk I/O) to data sharing.

Using the private interconnect greatly increases performance and scalability while preserving availability benefits of the classic implementation of a shared disk architecture in which the disk is the medium of data sharing. Simply put, while the basic data sharing and coherency mechanisms remain largely the same (distributed locks and messaging), copies of data blocks can now be passed from one node to another with one high-speed network hop instead of the two relatively slow hops required for a disk ping.

The Cache Fusion technology has evolved and has been refined to run applications efficiently without any specific changes to those applications. As one would expect, with every release of the Oracle RDBMS, Oracle has introduced newer concepts and improved efficiency.

Key Components in Cache Fusion

The following sections cover a few terms and concepts used in the discussion of Cache Fusion. In simple terms, every resource in the global cache (such as data blocks, index blocks, and undo blocks) potentially handles up to three parties. At startup, a simple hashing algorithm divides the management role of the resources into available

https://t.me/learningnets

instances. Each instance controls the access to the resources, and its role is to act as the resource "master" for the resources. Any instance that has acquired the permission (to read or modify) is the current "owner" of the resource. Someone else requesting access on that resource can be a "requestor" of that resource. Although this explanation is oversimplified, it is relevant to our discussion.

Ping

The transfer of a data block from one instance's buffer cache to another instance's buffer cache is known as a *ping*. Whenever an instance needs a block, it sends a request to the master of the resource to obtain a lock in the desired mode. If another lock resides on the same block, the master will ask the current holder to downgrade/release the current lock. As you learned in an earlier chapter, this process is known as a *blocking asynchronous trap (BAST)*. When an instance receives a BAST, it downgrades the lock as soon as possible. However, before downgrading the lock, it might have to write the corresponding block to disk. This operation sequence is known as a *disk ping* or *hard ping*.

NOTE

The definitions of AST and BAST used in this book are based on an Oracle context. Various implementations (such as VMS) use different definitions for AST and BAST.

Starting with Oracle 9*i*, disk pings have been reduced because blocks are transferred from one instance's buffer cache to another instance's buffer cache via the private interconnect in many cases. This type of transfer is known as a *Cache Fusion block transfer*. Some amount of block transfers in the Oracle RAC environment is healthy because it indicates sharing of data among the instances and minimizes disk I/O. However, excessive block transfers can impact performance, consuming CPU and network resources.

As we explained in Chapter 10, you can view the amount of data sent and received by an instance via Cache Fusion in the Automatic Workload Repository (AWR) report—specifically, in the "Global Cache Load Profile" section.

Deferred Ping

Oracle uses various mechanisms to mitigate pinging. When an instance receives a BAST, it might defer sending the block or downgrading the lock by tens of milliseconds. This extra time might allow the holding instance to complete an active transaction and mark the block header appropriately. This will eliminate any need for the receiving instance to check the status of the transaction immediately after receiving/reading a block.

https://t.me/learningnets

Technet24

Checking the status of an active transaction is an expensive operation that might require access (and pinging) to the related undo segment header and undo data blocks as well. You can use the hidden Oracle initialization parameter **_gc_defer_time** to define the duration for which an instance deferred downgrading a lock.

Past Image (PI) Blocks

PI blocks are copies of blocks in the local buffer cache. Whenever an instance needs to send a block it has recently modified to another instance, it preserves a copy of that block, marking it as PI.

An instance is obliged to keep PIs until the current owner of the block writes that block to the disk. The database discards the PIs after it writes the latest version of the block to disk. When the database writes a block to disk and is known to have a global role, indicating the presence of PIs in other instances' buffer caches, Global Cache Services (GCS) informs the instance holding the PIs to discard them.

With Cache Fusion, the database writes a block to disk to satisfy checkpoint requests and so on, not to transfer the block from one instance to another via disk.

When an instance needs to write to a block—to satisfy a checkpoint request, for example—the instance checks the role of the resource covering the block. If the role is global, the instance must inform the GCS of the write requirement. The GCS is responsible for finding the most current block image and informing the instance holding that image to perform the block write. The GCS then informs all holders of the global resource that they can release the buffers holding the PI copies of the block, allowing the global resources to be released.

The database places a block-written record (BWR) in its redo log buffer when an instance writes a block covered by a global resource, or when it's told it can free up a PI buffer. Recovery processes to indicate that redo information for the block is not needed for recovery prior to this point use this record. Although the BWR makes recovery more efficient, the instance doesn't force a flush of the log buffer after creating it because it is not essential for accurate recovery.

You can query the fixed table X$BH to find the number of past image blocks present in the buffer cache:

```
SQL> select state, count(state) from X$BH group by state;
     STATE  COUNT(STATE)
----------  -----------
         1  403
         2  7043
         8  15
         3  659
```

There are 15 past image blocks indicated in the STATE column (with a value of 8) of the X$BH table.

https://t.me/learningnets

Lock Mastering

The memory structure where GCS keeps information about a data block usage (and other sharable resources) is known as the *lock resource*. The database distributes the responsibility for tracking locks among all the instances, and the required memory also comes from the System Global Area (SGA) of the participating instances.

Due to the distributed nature of the resource ownership, a master node exists for each lock resource. The master node maintains complete information about current users and requestors for the lock resource. The master node also contains information about the PIs of the block.

Whenever an instance needs a data block, it must contact the master of lock resources corresponding to that data block. The master instance for the required lock resource could be local or remote. If local, the database needs to send fewer or no messages over the interconnect to obtain the required lock for that block. (For more information, see Chapter 11.)

Types of Contention

Resource contention occurs when two or more instances want the same resource. If an instance is using a resource such as a data block and that resource is needed by another instance at the same time, the result is contention for that resource.

There are three types of contention for data blocks:

- **Read/read contention** Read/read contention is never a problem because of the shared disk system. Other instances can read a block read by another instance without the intervention of GCS. Also, when the object is read-mostly, any instance can read the block(s) without additional messages from/to GCS.
- **Write/read contention** Write/read contention is addressed by a consistent read. The holding instance (most of the time) constructs the CR block and ships the requesting instance using interconnects.
- **Write/write contention** Cache Fusion technology addresses write/write contention. The database uses the cluster interconnect in some cases to ship data blocks among the instances that need to modify the same data block simultaneously.

It’s important to note that although the interconnect is the route for the current block to be shipped, the database stores locking information in the GRD to avoid corruptions in the blocks in memory due to conflicting access. There is no user- or application-level tuning or modification for this.

In the following sections, we look at how these components work seamlessly in a real-world situation. We explore how these contentions were addressed before Cache Fusion was introduced, and how they work with Cache Fusion.

https://t.me/learningnets

Technet24

Cache Fusion I, or Consistent Read Server

Oracle introduced Cache Fusion I (also known as *consistent read server)* in version 8.1.5. Oracle keeps a list of recent transactions that have changed a block. The database preserves the original data contained in the block in the undo segment, which the database utilizes to provide consistent read versions of the block to other readers or to recover an uncommitted transaction after an instance or process crash.

When the database commits a transaction, the server process places a flag in the undo segment header, indicating the transaction status as "committed." To save time, it doesn't stamp each and every block that it modifies with a commit flag. Thus, in the future, whenever another process reads that block and finds an active transaction in it, the reader refers to the corresponding undo segment header to determine whether the transaction has been committed.

If the undo segment header indicates that the transaction has been committed, the reader updates the block with the system commit number (SCN) in the transaction layer of the block to signify the commit and then moves on to read the block. However, if the transaction has not yet been committed, the reader locates the undo block where the original data is stored and makes a consistent read image of the block for its use.

This discussion leads us to infer the following:

- When a reader reads a recently modified block, it might find an active transaction in the block.
- The reader will need to read the undo segment header to decide whether or not the transaction has been committed.
- If the transaction isn't committed, the process creates a consistent read (CR) version of the block in the buffer cache using the data in the block and the data stored in the undo segment.
- If the undo segment shows the transaction is committed, the process must revisit the block, clean out the block, and generate the redo for the changes.

In an Oracle RAC environment, if the process of reading the block is on an instance other than the one that has modified the block, the reader will have to read the following blocks from the disk:

- **Data block** To get the data and/or transaction ID and the undo byte address (UBA)
- **Undo segment header block** To find the last undo block used for the entire transaction
- **Undo data block** To get the actual undo record to construct a CR image

Before these blocks can be read, the instance modifying those blocks will have to write them to disk. This will result in at least six disk I/O operations. The number of

https://t.me/learningnets

I/O operations can be optimized if the modifying instance, knowing that an instance needs a CR copy of the block it recently modified, can construct a CR copy of the block and send it to the requesting instance. It is highly probable that the data block, undo segment header block, and undo block will still be in this instance's buffer cache. Thus, by making a CR copy and sending it over the private interconnect, the database potentially replaces six disk I/Os with a single data block transfer over the interconnect.

Prior to Cache Fusion, whenever an instance needed to modify a data block that had been recently modified by another instance, the sequence of operations shown next and in Figure 12-2 occurred:

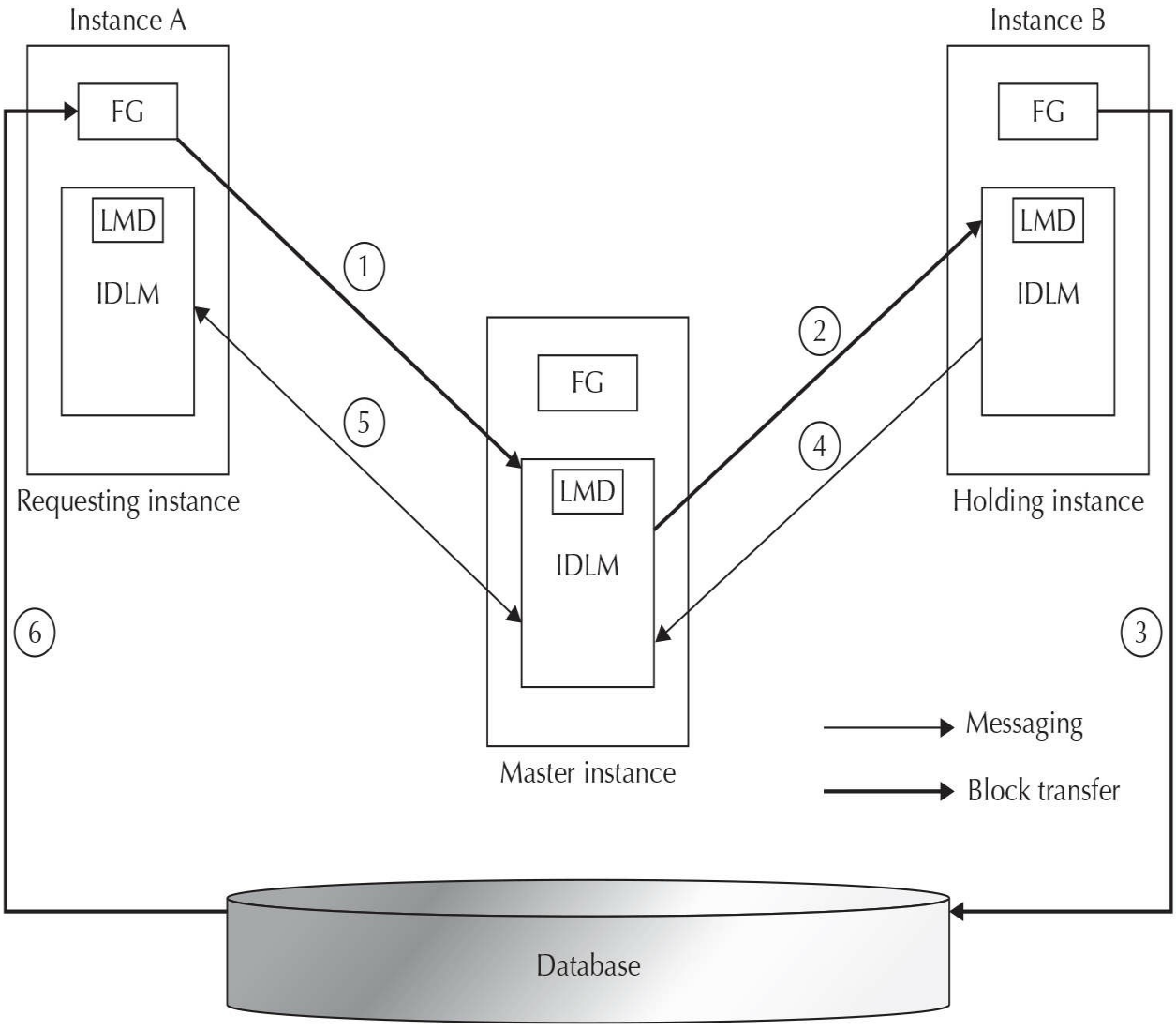


FIGURE 12-2. *Write/read contention before Cache Fusion*

1. An instance sends a message to the lock manager requesting a shared lock on the block.
2. Following are the possibilities in the global cache:

https://t.me/learningnets
Technet24

- If there is no current user for the block, the lock manager grants the shared lock to the requesting instance.
- If another instance has an exclusive lock on that block, the lock manager asks the owning instance to downgrade the lock.

3. Based on the result, either of the following can happen:
   - If the lock is granted, the requesting instance reads the block from the disk.
   - The owning instance writes the block to disk and disowns the lock on that resource.
4. The owning instance also informs the lock manager and the requesting instance that it has released the lock.
5. The requesting instance now has the lock granted. The lock manager updates the Integrated Distributed Lock Manager (IDLM) with the new holders of that resource.
6. The requesting instance reads the block from disk. If it is looking for the previous version of the block, it can proceed with the regular CR block fabrication in its own buffer cache.

CR fabrication can be an expensive operation, especially when the undo information is cached in the remote node. The requesting instance must make a few additional disk I/Os to get the blocks to its own cache. Because the disk is used as a data transfer medium, latency issues will crop up and application scalability and performance will degrade when the data isn't partitioned across the instances.

If the CR block fabrication occurs at the holding instance, part of the disk I/O can be avoided because the data blocks and undo blocks are local to that instance. Upon undo block fabrication, the holder can ship the processed block to the requestor. Read/write contention is avoided by building CR in the holding instance, and this is the first phase in Cache Fusion.

Following is the sequence of operations with the implementation of the CR server engine in Oracle 8*i*. The newly introduced background process—the block server process (BSP)—makes the CR fabrication at the holder's cache and ships the CR version of the block across the interconnect. The sequence of operations is shown in Figure 12-3.

https://t.me/learningnets

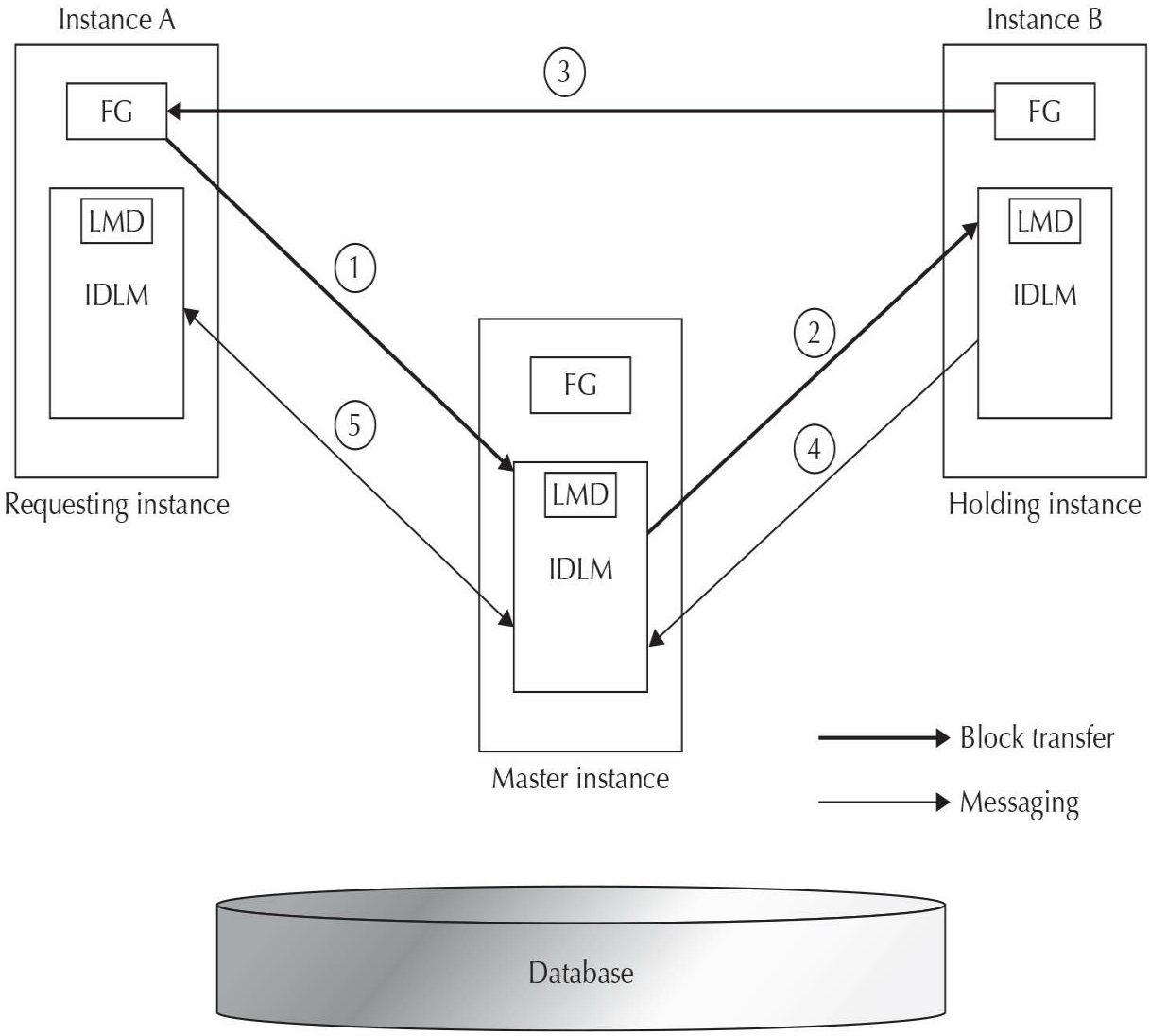


FIGURE 12-3. *CR block transfer in Cache Fusion*

1. An instance sends a message to the lock manager requesting a shared lock on the block.
2. Following are the possibilities in the global cache:
   - If there is no current user for the block, the lock manager grants the shared lock to the requesting instance.
   - If another instance has an exclusive lock on that block, the lock manager asks the owning instance to build a CR copy and ship it to the requesting instance.
3. Based on the result, either of the following can happen:
   - If the lock is granted, the requesting instance reads the block from disk.
   - The owning instance creates a CR version of the buffer in its own buffer cache and ships it to the requesting instance over the interconnect.
4. The owning instance also informs the lock manager and the requesting instance

https://t.me/learningnets

Technet24

that it has shipped the block.

5. The requesting instance now has the lock granted. The lock manager updates the IDLM with the new holders of that resource.

While making a CR copy needed by another instance, the holding instance may refuse to do so if either of the following occurs:

- It doesn't find any of the needed blocks in its buffer cache. It will not perform a disk read to make a CR copy for another instance.
- It is repeatedly asked to send a CR copy of the same block. After sending CR copies of the same block four times, it will voluntarily relinquish the lock, write the block to disk, and let other instances get the block from disk. The number of copies it will serve before it does so is governed by the hidden parameter **_fairness_threshold**.

While downgrading the lock and writing the block to disk, the holding instance will send a copy of the block it has constructed so far, and the requesting instance will continue the construction of the CR copy needed by it. This is known as the *light-work rule*. Refer to Chapter 11 for a detailed discussion on the light-work rule and fairness threshold.

Cache Fusion II, or Write/Write Cache Fusion

The CR server addressed read/write contention in the first phase of Cache Fusion. The second phase of Cache Fusion, introduced in Oracle 9*i*, implements the framework for current block transfers for write/write contention. Prior to Oracle 9*i*, whenever an instance needed to modify a data block that had been recently modified by another instance, the following sequence of operations would take place, as illustrated in Figure 12-4:

https://t.me/learningnets

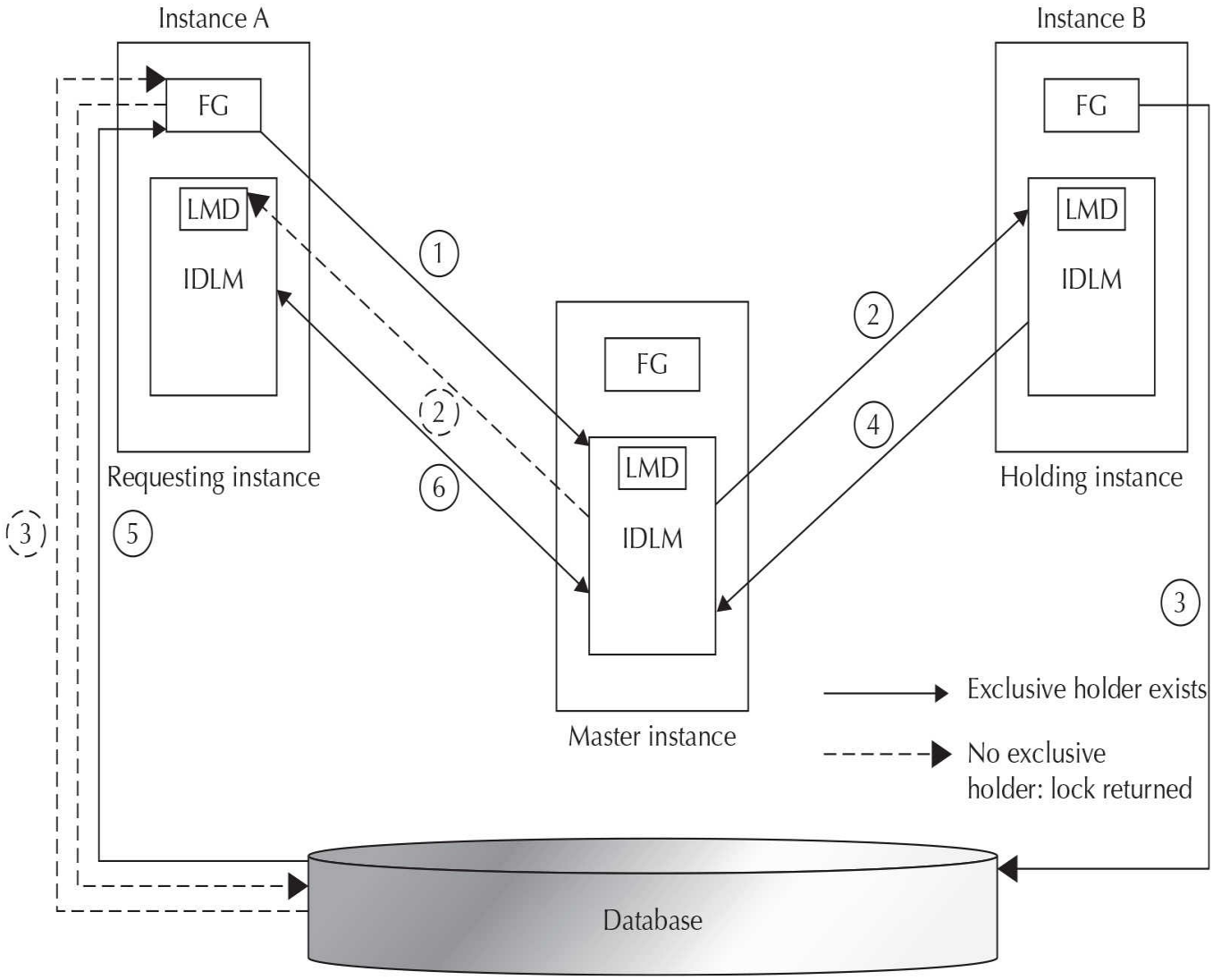


FIGURE 12-4. *Write/write contention in pre–Cache Fusion*

1. An instance sends a message to the lock manager requesting an exclusive lock on the block.
2. Following are the possibilities in the global cache:
   - If there is no current user for the block, the lock manager grants the exclusive lock to the requesting instance.
   - If another instance has an exclusive lock on that block, the lock manager asks the owning instance to release the lock.
3. Based on the result, either of the following can happen:
   - If the lock is granted, the requesting instance reads the block from the disk.
   - The owning instance writes the block to disk and disowns the lock on that resource.
4. The owning instance also informs the lock manager and the requesting instance that it has released the lock.

https://t.me/learningnets

Technet24

5. The requesting instance now has the exclusive lock. It can read the block from disk and proceed with the intended changes to the block.
6. The lock manager updates the resource directory with the current information about the holder.

Often, the requesting instance will find an active transaction in the active transaction list of the block, and it might need the corresponding undo block header and undo data block to clean out the block. Thus, the requesting instance might need multiple disk I/Os before the instance can modify the block. Cache Fusion makes this operation efficient and fast, as shown in Figure 12-5:

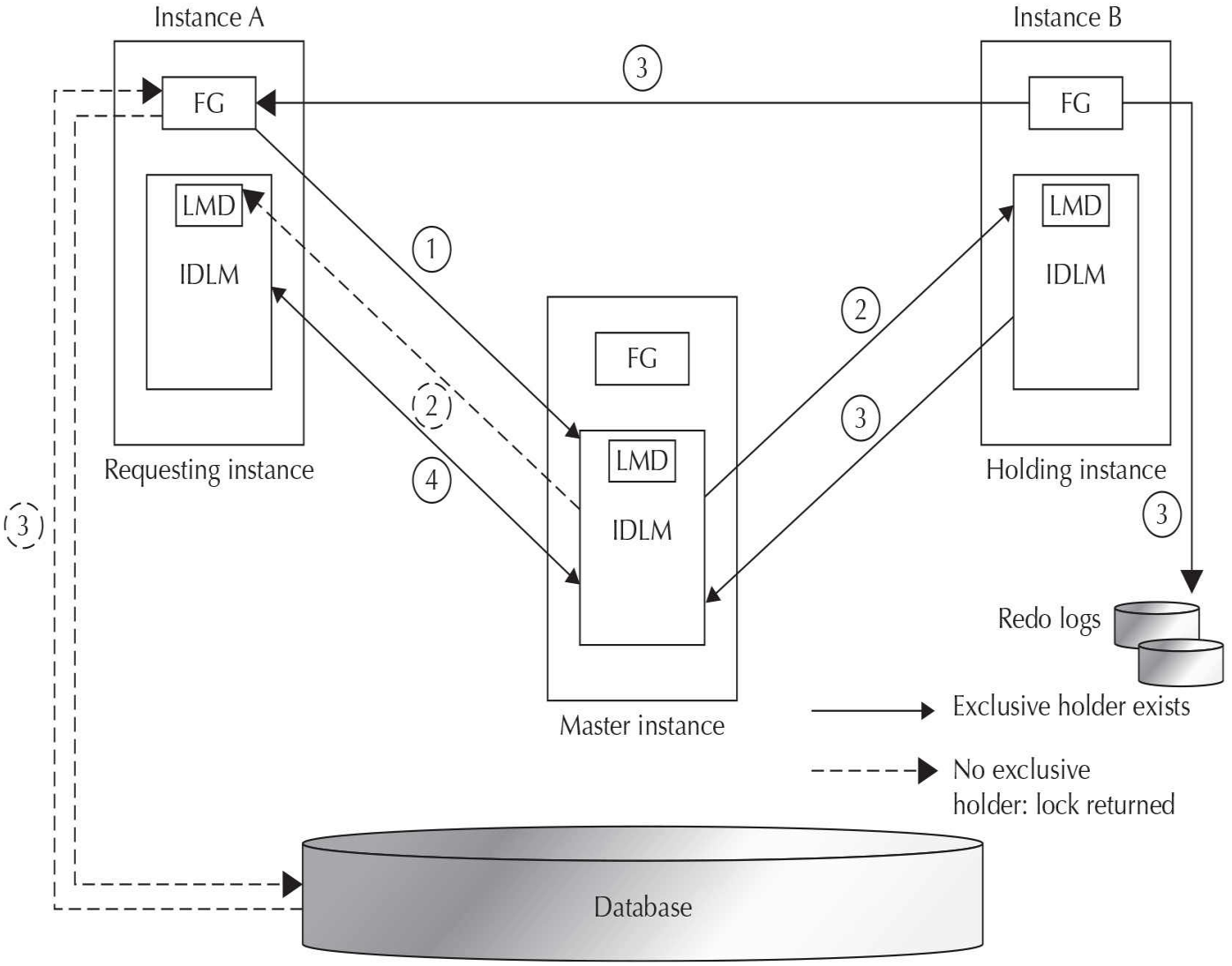


FIGURE 12-5. *Cache Fusion current block transfer*

1. The instance sends a message to the lock manager to request an exclusive lock on the block.
2. Following are the possibilities in the global cache based on whether any other user is holding the lock:

https://t.me/learningnets

- If there is no current user for the block, the lock manager grants the exclusive lock to the requesting instance.
- If another instance has an exclusive lock on that block, the lock manager asks the owning instance to release the lock.

3. Based on the result, either of the following can happen:
   - If the lock is granted, the requesting instance reads the block from the disk.
   - The owning instance sends the current block to the requesting instance via the high-speed interconnects. To guarantee recovery in the event of instance death, the owning instance writes all the redo records generated for the block to the online redo log file. It will keep a past image of the block and inform the master instance that it has sent the current block to the requesting instance.
4. The lock manager updates the resource directory with the current holder of the block.

Cache Fusion in Operation

Before we get into Cache Fusion operation, here's a quick recap of GCS resource roles and modes. A GCS resource can be local or global. It's local if it can be acted upon without consulting other instances, and it's global if it cannot be acted upon without consulting or informing remote instances. The database uses GCS as a messaging agent to coordinate manipulation of a global resource.

Depending on the type of operation an instance intends to perform on a block (resource), an instance will acquire the block in shared or exclusive mode. When an instance needs a block to satisfy a query request, it requires the block to be in consistent read mode or in shared mode. When an instance needs a block to satisfy a DML (update, insert, delete) request, it needs the block in exclusive mode. The Null (N) mode indicates that the instance does not currently hold the resource in any mode. This is the default status for each instance.

An instance having a resource with a local role can have the resources in X, S, or N mode. The following table denotes the different states of a resource:

Mode/Role	Local	Global
Null: N	NL	NG
Shared: S	SL	SG
Exclusive: X	XL	XG

- **SL** When an instance has a resource in SL form, it can serve a copy of the block to other instances and it can read the block from disk. Because the block is not modified, there is no need to write to disk.
- **XL** When an instance has a resource in XL form, it has sole ownership and interest in that resource. It also has the exclusive right to modify the block. All

https://t.me/learningnets

Technet24

changes to the block are in its local buffer cache, and it can write the block to disk. If another instance wants the block, it'll contact the instance via GCS.

- **NL** An NL form protects consistent read blocks. If a block is held in SL mode and another instance wants it in X mode, the current instance will send the block to the requesting instance and downgrade its role to NL.
- **SG** In SG form, a block is present in one or more instances. An instance can read the block from disk and serve it to other instances.
- **XG** In XG form, a block can have one or more PIs, indicating multiple copies of the block in several instances' buffer caches. The instance with the XG role has the latest copy of the block and is the most likely candidate to write the block to disk. GCS can ask the instance with the XG role to write the block to disk or to serve it to another instance.
- **NG** After discarding PIs when instructed by GCS, the block is kept in the buffer cache with the NG role. This serves only as the CR copy of the block.

Consider the following: a four-instance Oracle RAC environment, with instances A, B, C, and D. Instance D is the master of the lock resources for the data block BL. In this example, we'll work with only one block (BL) to begin with, and it'll reside on disk at SCN 987654.

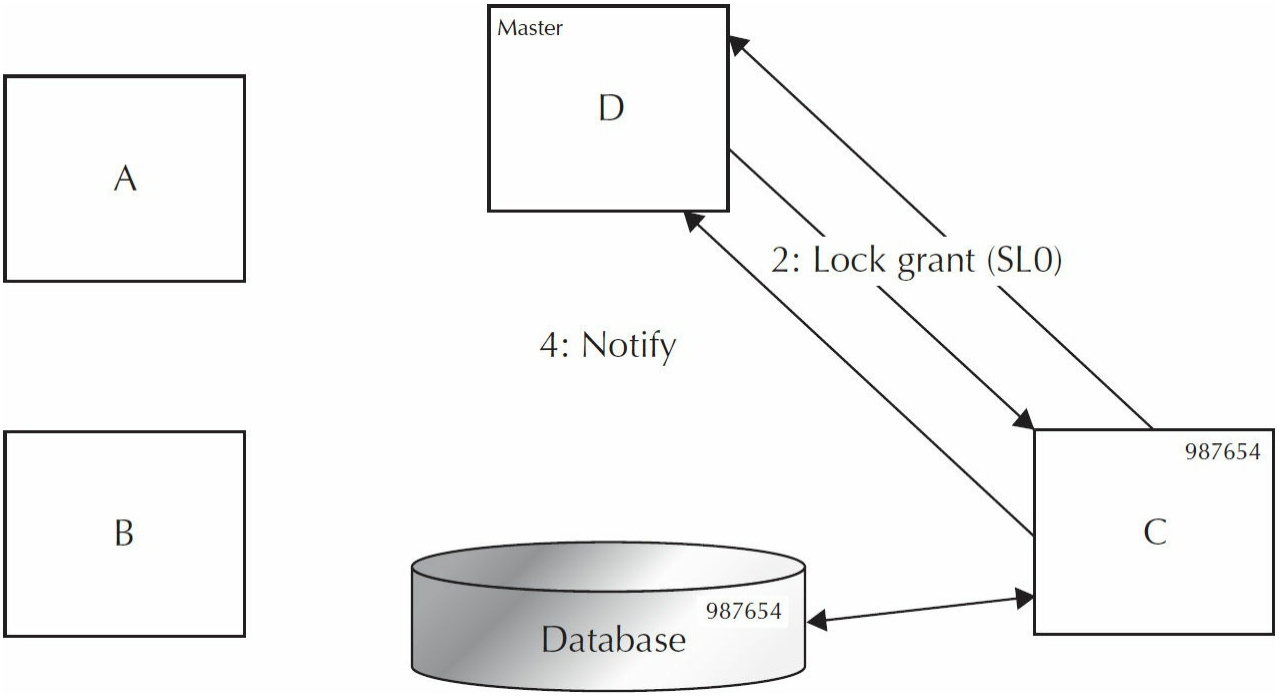
We'll choose an alphanumeric code for the lock state: The first letter will indicate the lock mode: N = Null, S = Shared, and X = Exclusive. The second letter will indicate the lock role: L = Local and G = Global. The third letter will be a digit indicating the past image: 0 = block does not have a past image, and 1 = a PI of the block.

The following examples will undergo a sequence of operations to help you understand the current block and CR block transfers from the instances. These scenarios include reading from disk, reading from cache, getting the block from cache for update, performing an update on a block, performing an update on the same block, reading a block that was globally dirty, performing a rollback on a previously updated block, and reading the block after commit. Although these examples don't cover all the functionalities of Cache Fusion, they're the most common operations in a real-world environment.

Example 1: Reading a Block from Disk

In this example, instance C wants to read the block. It will request a lock in shared mode from the master instance.

https://t.me/learningnets



1. Instance C requests the block by sending a shared lock request (S, C) to master instance D. (S represents the type of lock requested, and C represents the requesting instance.)

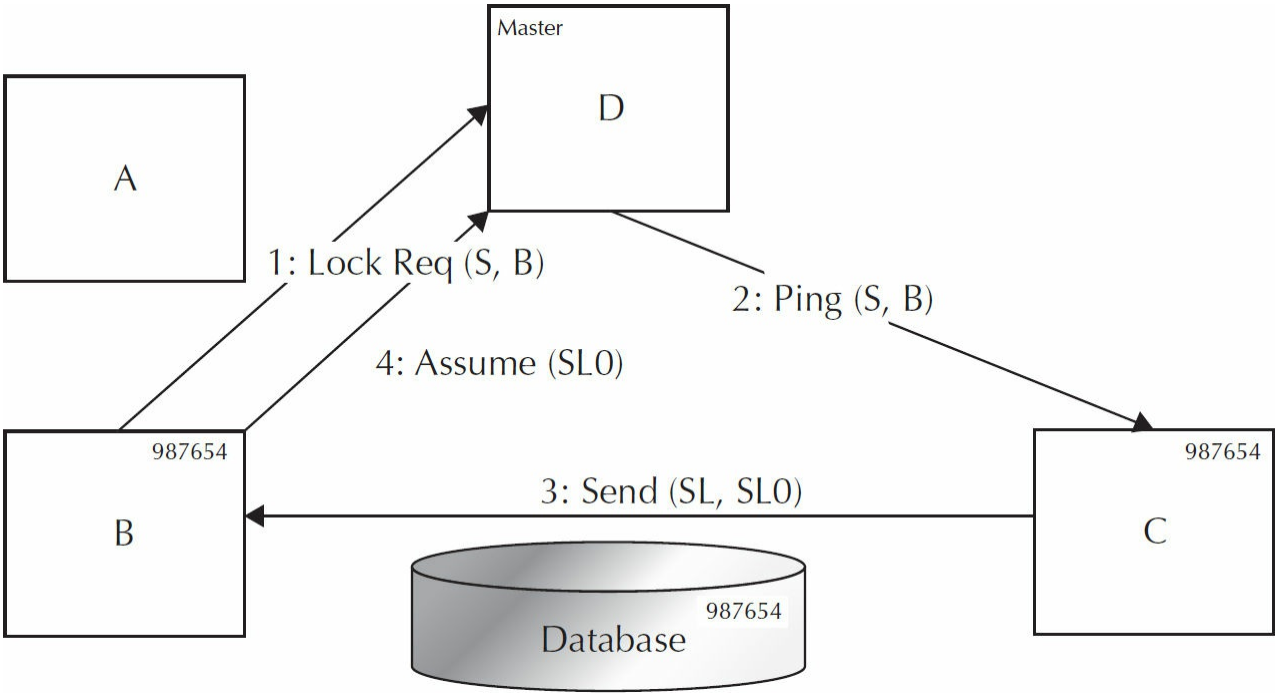
2. The block has never been read into the buffer cache of any instance, and it is not locked. Master instance D grants the lock to instance C. The lock granted is SL0, indicating that it is a shared lock with local interests, and no past images of this block exist.
3. Instance C reads the block from the shared disk into its buffer cache.
4. Instance C has the block in shared mode. The lock manager updates the resource directory.

Example 2: Reading a Block from the Cache

We begin at the end of example 1. Instance B wants to read the same block that is cached in instance C.

https://t.me/learningnets

Technet24

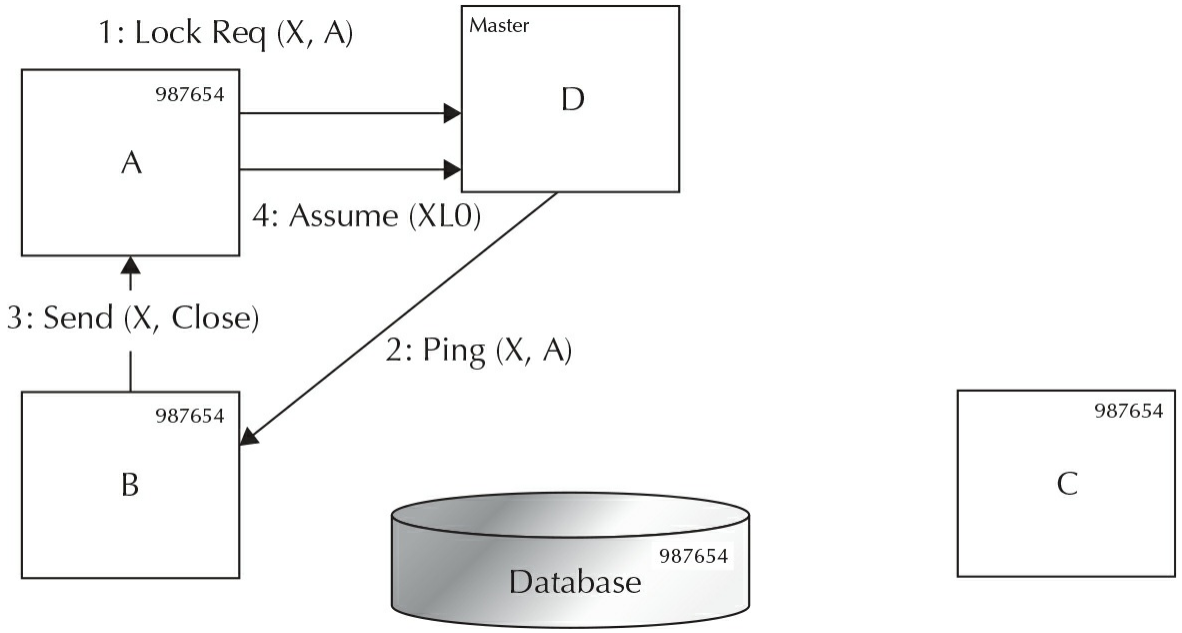


1. Instance B sends a shared lock request, Lock Req (S, B), to master instance D.
2. The lockmaster at instance D knows that the block may be available at instance C and hence sends a ping message to instance C.
3. Instance C sends the block to instance B over the interconnect. Along with the block, instance C also indicates that instance B should take the current lock mode and role from instance C.
4. Instance B sends a message to instance D that it has assumed the SL lock for the block. This message is not critical (yet it is important) for the lock manager; hence, this message is sent asynchronously.

Example 3: Getting a (Cached) Clean Block for Update

We begin at the end of example 2. Instance A wants to modify the same block that's already cached in instances B and C.

https://t.me/learningnets



1. Instance A sends an exclusive lock request, Lock Req (X, A), to master instance D.

2. The lockmaster at D knows that the block may be available at instance B in SCUR mode and at instance C in CR mode. It also sends a ping message to the shared lock holders. The most recent access was at instance B, and instance D sends a BAST message to instance B.

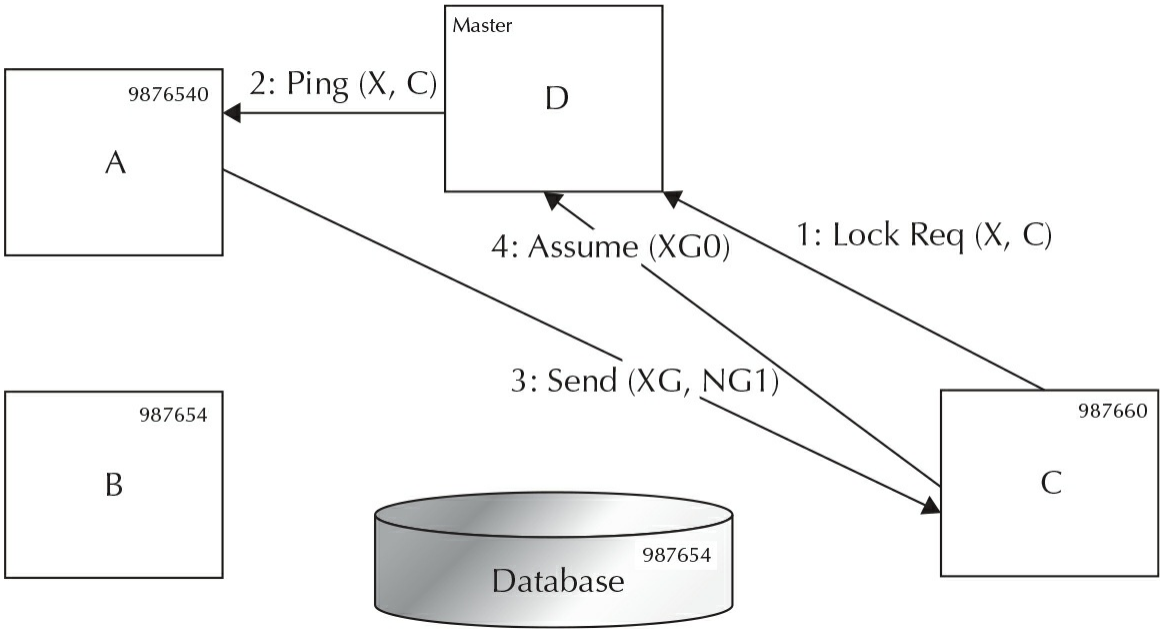
3. Instance B sends the block to instance A over the interconnect and closes its shared (SCUR) lock. The block may still be in its buffer to be used as CR; however, it releases any lock on it.
4. Instance A now has the block in exclusive mode and sends an assume message to instance D. Instance A holds the lock in XL0 mode.
5. Instance A modifies the block in its buffer cache. The changes are not committed, and the block has not yet been written to disk; therefore, the SCN on disk remains at 987654.

Example 4: Getting a (Cached) Modified Block for Update and Commit

This example begins at the end of example 3. Instance C now wants to update the block BL. Note that instance A has already updated the block, and the block has an open transaction on it. If instance C tries to update the same row, it will wait for the previous update to commit or roll back. However, in this case, instance C updates a different row, and after the update it issues a commit.

https://t.me/learningnets

Technet24



1. Instance C sends an exclusive lock request, Lock Req (X, C), to master instance D.
2. The lockmaster at D knows that instance A holds an exclusive lock on the block and hence sends a ping message to instance A.
3. Instance A sends the dirty block to instance C over the interconnect. It downgrades the lock on the buffer from XCUR to NULL. It keeps a PI version of the block and disowns any lock on that buffer. Before shipping the block, instance A has to create a PI and flush any pending redo for the block changed. The block mode on instance A is now NG1.
4. Instance C sends a message to instance D indicating it now has the block in exclusive mode. The block role G indicates that the block is in global mode, and if it needs to write the block to disk, it must coordinate this with the other instances that have past images of that block. Instance C modifies the block and issues a commit. This takes the SCN to 987660.

NOTE

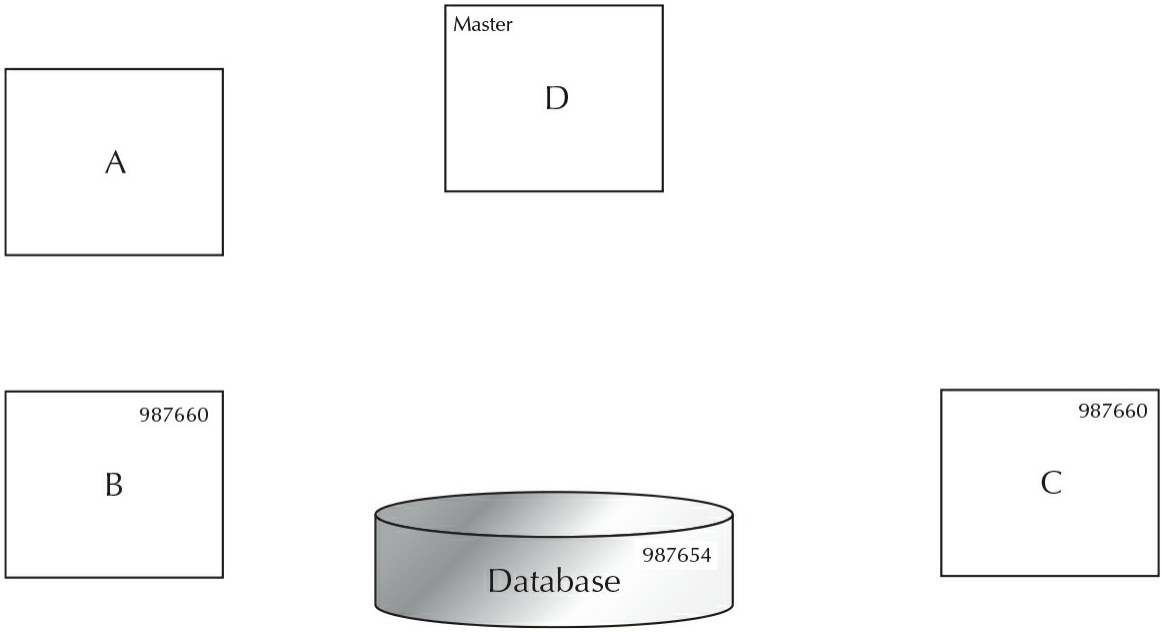
Row-level locks and block locks operate independently in the Oracle RAC environment. In this example, we update the same block in two different instances, but not the same row. Refer to Chapter 11 for a detailed discussion of Oracle's implementation of row-level locking. Row locks (known as enqueue type TX) and GCS buffer locks (known as enqueue type BL) can run in parallel for a transaction.

Example 5: Commit the Previously Modified Block and Select the Data

Now instance A issues a commit to release the row-level locks held by the transaction

https://t.me/learningnets

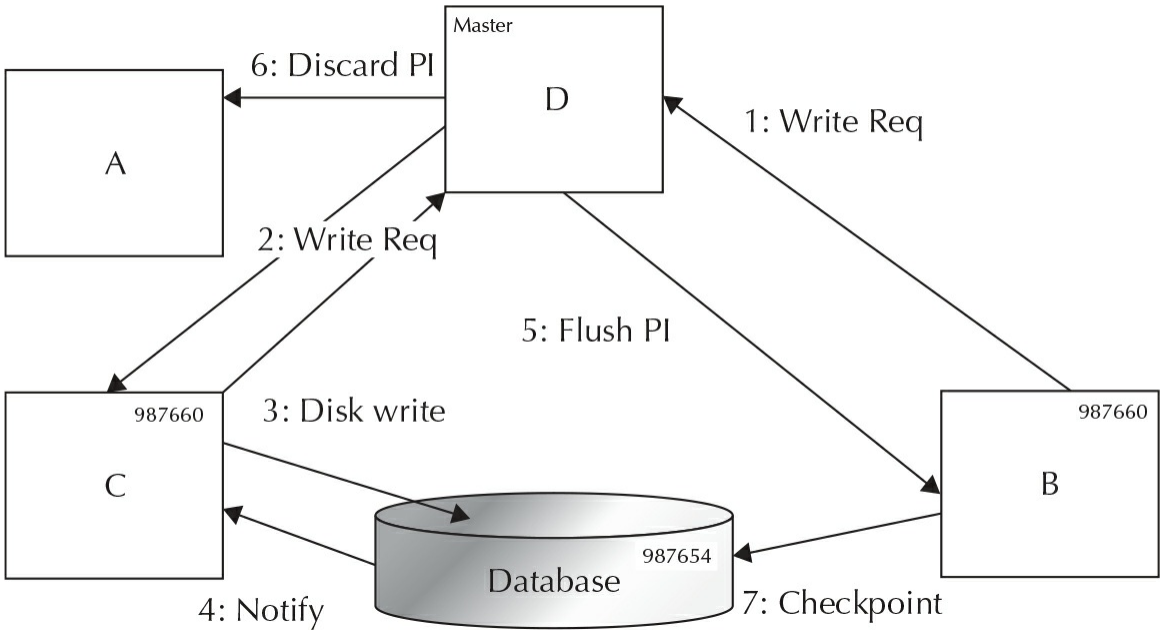
and to flush the redo information to the redo log files. Global locks on the resources remain the same.



1. Instance A wants to commit the changes. Commit operations do not require any synchronous modifications to the data block.
2. The lock status remains the same as the previous state, and change vectors for the commits are written to the redo logs.

Example 6: Write the Dirty Buffers to Disk Due to Checkpoint

Continuing at the end of example 5, instance B writes the dirty blocks from the buffer cache due to a checkpoint.



1. Instance B sends a write request to master D with the necessary SCN.
2. The lockmaster at D knows that the most recent copy of the block may be available at instance C and hence sends a message to instance C asking to write.

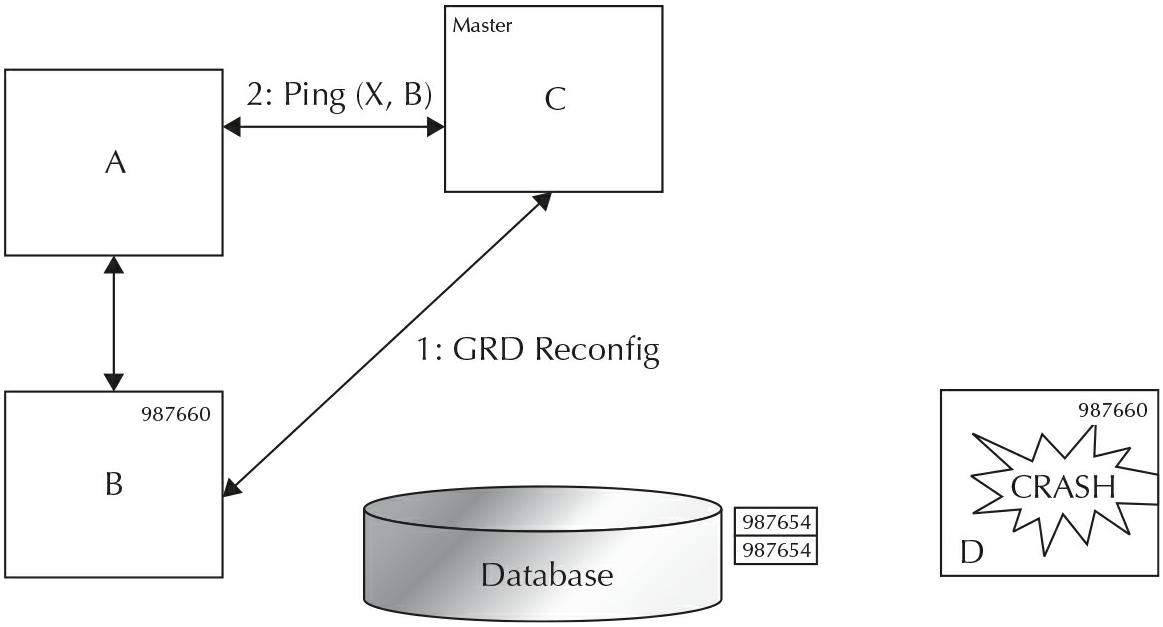
https://t.me/learningnets

Technet24

3. Instance C initiates a disk write and writes a BWR into the redo log file.
4. Instance C gets the write notification that the write is completed.
5. Instance C notifies the master that the write is completed.
6. On receipt of the notification from C, master instance D tells all PI holders to discard their PIs, and the lock at instance C writes the modified block to the disk.
7. All instances that have previously modified this block will also have to write a BWR. The write request by instance C has now been satisfied, and instance C can now proceed with its checkpoint as usual.

Example 7: Master Instance Crash

This example continues from example 6.



1. The master instance D crashes.
2. The Global Resource Directory (GRD) is frozen momentarily, and the resources held by master instance D will be equally distributed in the surviving nodes.

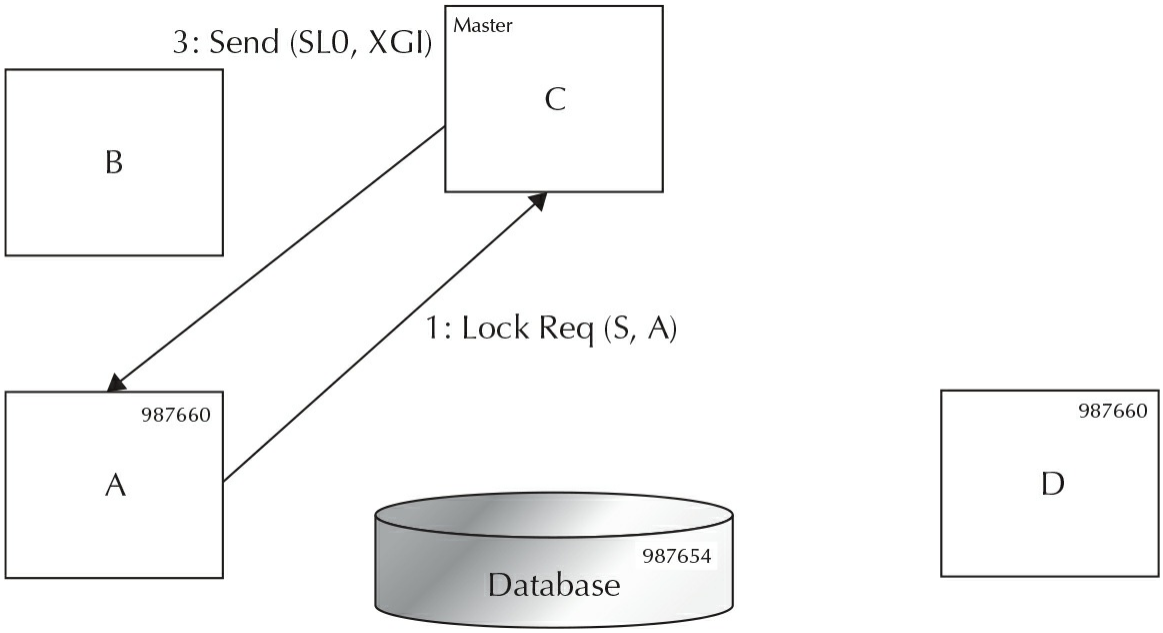
NOTE

Mode details on the GRD are provided in Chapter 11, and GRD and fast reconfiguration after node failure are discussed in Chapter 9.

Example 8: Select the Rows from Instance A

Now instance A queries the rows from that table to get the most recent data.

https://t.me/learningnets



1. Instance A sends a shared lock request, Lock Req (S, A), to master instance C.
2. Master instance C knows that the most recent copy of that block may be in instance C (which is the current master) itself, and asks the holder to ship the CR block to instance A.
3. Instance C ships the CR block to instance A over the interconnect.

Sequence of Operations

Table 12-1 lists the sequence of operations on the nodes and the corresponding modes of the buffers in various caches.

https://t.me/learningnets

Technet24

Example	Operation on Node A	Operation on Node B	Operation on Node C	Operation on Node D	Buffer Status			
					A	B	C	D
1			Read the block from disk.				SCUR	
2		Read the block from cache.				CR	SCUR	
3	Update the block.				XCUR	CR	CR	
4			Update the same block.		PI	CR	XCUR	
5	Commit the changes.				PI	CR	XCUR	
6		Trigger checkpoint.				CR	XCUR	
7				Instance crash.				
8	Select the rows.				CR		XCUR	

TABLE 12-1. *Sequence of Operations on Nodes and Buffer Status in Examples 1 through 8*

Cache Fusion Walkthrough

In the following Cache Fusion example, we'll use the V$SEGMENT_STATISTICS, X$BH, XLE, XKJBL, V$DLM_RESS, and V$GES_ENQUEUE views to trace blocks and the lock status in a four-node environment.

The table we use in this example is named EMP. This table has a primary key on column EMPNO to make it more selective for the retrieval. We also manually master the EMPLOYEES table to instance D for demonstration purposes. You can execute the following command to manually remaster an object to one instance:

```
oradebug lkdebug -m pkey <objected>
```

Run the following SQL script as any DBA user to get the data_object_id for the EMPLOYEES table owned by the user SMITH. The data dictionary view dba_objects contains details about all the objects in the database. We query this view to get details about the data_object_id for the EMPLOYEES table:

https://t.me/learningnets

```
REM We get the object id of the EMPLOYEES table
column owner format a10
column data_object_id format 999999
column object_name format a15
select owner, data_object_id, object_name
from dba_objects
where owner='SMITH'
and object_name='EMPLOYEES';

SQL> @GET_OBJECT_ID

OWNER      DATA_OBJECT_ID OBJECT_NAME
---------- -------------- ---------------
SMITH               51151 EMPLOYEES
```

The data dictionary view dba_extents keeps detailed storage information for all segments in the database. We can get the details about the file_id and block_id of the EMPLOYEES table by querying this view. We can query the dba_extents view or use the dbms_rowid package to get the details:

```
REM We get the File ID, Starting Block ID for EMPLOYEES table
col owner format a8
col segment_name format a12
select owner,segment_name,file_id,block_id,blocks
from dba_extents
where owner='SMITH'
and segment_name='EMPLOYEES'

OWNER    SEGMENT_NAME    FILE_ID   BLOCK_ID     BLOCKS
-------- ------------ ---------- ---------- ----------
SMITH    EMPLOYEES      4         25          8
```

Alternatively, we can get the details about the file number and block number of the rows from the EMPLOYEES table using the following method. The rowid is the physical address of the row inside the database and uniquely identifies a row in the database.

https://t.me/learningnets

Technet24

```
SQL> select rowid,empno,ename from SMITH.employees where empno in(7788,7876);
ROWID                   EMPNO ENAME
------------------ ---------- ----------
AAAMfPAAEAAAAAgAAH       7788 SMITH

AAAMfPAAEAAAAAgAAK       7876 ADAMS

SQL> select
  2  dbms_rowid.rowid_block_number('AAAMfPAAEAAAAAgAAH') Block_no
  3  from dual;
  BLOCK_NO

----------
        32

 SQL> select
   2  dbms_rowid. rowid_relative_fno('AAAMfPAAEAAAAAgAAH') File_no
   3  from dual;
  FILE_NO
----------
         4
```

Queries to Get the Lock and Buffer Status

Following is a quick explanation of the queries you can use to get the lock and buffer status information. We ran the following four queries, and you must run them in a session logged in with SYS as sysdba.

GET_SEGMENT_STATS.SQL The dynamic performance view v$segment_statistics contains statistics for each segment, such as number of reads, writes, row lock waits, and ITL waits. You can use this view to get information about how data blocks belonging to the EMPLOYEES table are received by the instances.

```
column stat format a35
column value format 99999
select STATISTIC_NAME stat, value val
from v$segment_statistics
where value > 0
and OWNER='SMITH' and OBJECT_NAME='EMPLOYEES';
```

GET_BUFFER_STAT.SQL The internal view X$BH contains the status of the block in the buffer cache. We can get the status of any cached buffer in the buffer cache from this view. The predicate obj=51151 is used to restrict our interest on the EMPLOYEES table. To get the status of the block that contains the rows of EMPNO 7788 and 7369, dbablk=32 is used. The last condition in the predicate, class=1, is used to get the details about the data blocks of the EMPLOYEES table.

https://t.me/learningnets

```
SELECT
state, mode_held, le_addr, dbarfil, dbablk, cr_scn_bas, cr_scn_wrp
FROM x$bh
WHERE obj=51151
AND dbablk=32
AND class=1;
```

Column STATE of X$BH contains information about the mode of the block in the buffer cache. You can also query V$BH to get the same information. However, querying V$BH may be an expensive operation in production because it accesses all buffers (joining with X$LE) and may consume a large amount of CPU cycles. Hence, it's better to query the X$BH view whenever you access the buffer cache information.

The following table contains information about the various block statuses in the buffer cache:

State	Explanation
0	Buffer is free, unused.
1	Buffer current, locked in X mode.
2	Buffer current, locked in S mode.
3	Consistent read buffer.
4	Buffer being read.
5	Buffer under media recovery.
6	Buffer under instance recovery.
7	Write clone buffer.
8	Past image (PI) buffer.

State values 1, 2, 3, and 8 are the most commonly seen buffers modes in the buffer cache; we will use these values in our demo example.

Getting the GCS Resource Name You can get the GCS resource name for use in the following section by first running the following query:

https://t.me/learningnets

Technet24

```
column hexname format a25
column resource_name format a15
select
b.kjblname hexname,b.kjblname2 resource_name,
b.kjblgrant,b.kjblrole,b.kjblrequest
from
X$LE a, X$KJBL b
where a.le_kjbl=b.kjbllockp
and a.le_addr=(select le_addr from
x$bh where dbablk=32 and obj=51151
and class=1 and state! =3);

HEXNAME                   RESOURCE_NAME   KJBLGRANT   KJBLROLE KJBLREQUE
------------------------- --------------- --------- ---------- ---------
[0x20][0x40000],[BL]      32,262144,BL     KJUSEREX           0 KJUSERNL
```

The kjblname column in the internal view X$KJBL provides the resource name in hexadecimal format, and kjblname2 provides the resource name in decimal format. We will be using the hex format ([id1],[id2],[type]) to query the resource in the V$GC_ELEMENT and V$DLM_RESS views.

Monitoring the Resource Allocation We ran the following query to monitor the resource allocated to the data block from the EMPLOYEES table:

```
column resource_name format a22
column state format a8
column mast format 9999
column grnt format 9999
column cnvt format 9999
select
a.resource_name,b.state,a.master_node mast,a.on_convert_q cnvt ,
a.on_grant_q grnt,b.request_level,b.grant_level
from v$dlm_ress a,
V$ges_enqueue b
where upper(a.resource_name)=upper(b.resource_name1)
and a.resource_name like '%[0x20][0x40000],[BL]%';

RESOURCE_NAME          STATE     MAST  CNVT  GRNT REQUEST_L GRANT_LEV
---------------------- -------- ----- ----- ----- --------- ---------
[0x20][0x40000],[BL]   GRANTED      3     0     1 KJUSERNL  KJUSERPR
[0x20][0x40000],[BL]   GRANTED      3     0     1 KJUSERNL  KJUSERPR
```

Sample Operation Details

We'll perform a sample operation. At the end of each SQL operation, we'll take a snapshot using the aforementioned four queries to monitor the status change in blocks and locks in the participating instances. User statements should be run as SMITH, and

https://t.me/learningnets

monitoring queries should be run as user SYS because they refer to internal views.

The sequence of operations is as follows:

1. Select instance C to get the row with EMPNO=7788 from the EMPLOYEES table:
   ```
   SELECT EMPNO, ENAME, SAL from EMP where EMPNO=7788;
   ```
2. Select the same row in instance B:
   ```
   SELECT EMPNO, ENAME, SAL from EMP where EMPNO=7788;
   ```
3. Update the row in instance A:
   ```
   UPDATE EMP SET SAL=SAL+1000 where EMPNO=7788;
   ```
4. Update a new row (EMPNO=7369) in instance C:
   ```
   UPDATE EMP SET SAL=SAL*2 where EMPNO=7369;
   ```
5. Commit the changes in instance A:
   ```
   COMMIT;
   ```
6. Write the changes to the disk because of a checkpoint in instance B:
   ```
   ALTER SYSTEM CHECKPOINT; ALTER SYSTEM SWITCH LOGFILE;
   ```
7. A master node crash occurs.
8. Select the data from instance A and verify the status of the current master:
   ```
   SELECT EMPNO,ENAME,SAL FROM EMP WHERE EMPNO IN (7788,7369)
   ```

At the beginning of the example, the buffer cache does not contain any block belonging to the EMPLOYEES table:

https://t.me/learningnets

Technet24

```
Lock status queries run on instance A at the beginning
$sqlplus '/as sysdba'

SQL> @GET_SEGMENT_STATS
no rows selected
SQL>@GET_BUFFER_STAT
no rows selected
SQL> @GET_RESOURCE_NAME
no rows selected
SQL>@GET_RESOURCE_STAT
no rows selected

Lock status queries run on instance B at the beginning
$sqlplus '/as sysdba'
SQL> @GET_SEGMENT_STATS
no rows selected
SQL>@GET_BUFFER_STAT
no rows selected
SQL> @GET_RESOURCE_NAME
no rows selected
SQL>@GET_RESOURCE_STAT
no rows selected

Lock status queries run on instance C at the beginning
$sqlplus '/as sysdba'
SQL> @GET_SEGMENT_STATS
no rows selected
SQL>@GET_BUFFER_STAT
no rows selected
SQL> @GET_RESOURCE_NAME
no rows selected
SQL>@GET_RESOURCE_STAT
no rows selected

Lock status queries run on instance D at the beginning
$sqlplus '/as sysdba'

SQL> @GET_SEGMENT_STATS
no rows selected
SQL>@GET_BUFFER_STAT
no rows selected
SQL> @GET_RESOURCE_NAME
no rows selected
SQL>@GET_RESOURCE_STAT
no rows selected
```

https://t.me/learningnets

As you can see, the buffer cache doesn't contain any blocks belonging to the table EMP. Because no blocks are in the buffer cache, no resources are needed to keep track of them. Therefore, no rows are returned by the queries. The table EMP is mastered by instance D, and every request on the table will be routed through instance D.

Example 1: Retrieve the Data from Disk

The first example retrieves the data from disk and monitors the resource movements between master and holder instances. To start, we'll query the data from the EMPLOYEES table on instance C. Connect as SMITH and run the following query to load a buffer from disk to the buffer cache of instance C. Instance D is the master of the resource. You can query the X$KJBR.KJBRMASTER view to find the master node for a particular resource.

```
REM Let us query the EMP data from instance C
REM This query is run as user SMITH

SQL> select empno,ename,sal from employees
  2  where empno=7788;
     EMPNO ENAME              SAL
---------- ---------- ----------
      7788 SMITH             3000
```

Now we run the lock status queries on instance C. We run the monitoring queries as user SYS, because we query the internal tables. These queries yield the following results:

```
REM #### Result of lock queries on instance C,
REM after a select from EMPLOYEES table on instance C
REM This query is run as user SYS
SQL> @get_segment_stats
OBJECT_NAM STAT                                VALUE
---------- ------------------------------------ -------
EMP        physical reads                            1

SQL> @get_resource_name
HEXNAME                  RESOURCE_NAME   KJBLGRANT   KJBLROLE KJBLREQUE
------------------------ --------------- --------- ---------- ---------
[0x20][0x40000],[BL]     32,262144,BL    KJUSERPR           0 KJUSERNL
SQL> @get_resource_stat
no rows selected
SQL> @get_buffer_stat
STATE  MODE_HELD      FILE#     BLOCK#   SCN_BASE   SCN_WRAP
----- ---------- ---------- ---------- ---------- ----------
    2          0          4         32          0          0
```

Observe the following:

https://t.me/learningnets

Technet24

- The block was retrieved from disk, indicated by “physical reads” for the segment in the v$segment_statisics view.
- The block is held in the current shared mode, as indicated by 2 in the STATE column of the x$bh table. X$BH STATE is 2 (SCUR).
- The master node is node 3 (where instance D is running; the node numbering starts with 0).
- A protected read lock (shared read), as indicated by KJUSERPR, has been granted on this resource.
- The row data resides in File# 4 and Block# 32 in the database. The following excerpt from the block dump shows that all 14 rows of the EMPLOYEES table reside in the same block. We’ve edited the block dump output for brevity:

```
Start dump data blocks tsn: 4 file#: 4 minblk 32 maxblk 32
buffer tsn: 4 rdba: 0x01000020 (4/32)
scn: 0x0000.000adca3 seq: 0x01 flg: 0x04 tail: 0xdca30601

Block header dump:  0x01000020
 Object id on Block? Y
 seg/obj: 0xc7cf  csc: 0x00.adca2  itc: 2  flg: E  typ: 1 - DATA
     brn: 0  bdba: 0x1000019 ver: 0x01 opc: 0
     inc: 0  exflg: 0

Itl           Xid                  Uba         Flag  Lck        Scn/Fsc
0x01   0x0014.017.00000102   0x01400239.0023.06  C---    0  scn 0x0000.000a9cb9
0x02   0x000a.012.000001ea   0x0080118f.00d1.11  C-U-    0  scn 0x0000.000ad663
data_block_dump,data header at 0x64c6664
===============
tsiz: 0x1f98
hsiz: 0x2e
pbl: 0x064c6664
bdba: 0x01000020
     76543210
flag=--------
ntab=1
nrow=14
```

In this block dump, nrow=14 shows that all 14 rows are stored in the same block. The data block address (bdba) is 0x01000020; this will be used later to verify the block-written records in the redo logs.

The block dump belongs to the EMPLOYEES table, and the object ID of the block is shown as seg/obj:0xc7cf, where the object ID 51151 is represented as 0xc7cf.

Here’s the result of lock queries on master instance D after a select from EMPLOYEES table on instance A:

https://t.me/learningnets

```
REM Lock status on the Master Node D
SQL> @get_segment_stats
no rows selected
SQL> @get_buffer_stat
no rows selected
SQL> @get_resource_name
no rows selected
SQL> @get_resource_stat
RESOURCE_NAME          STATE     MAST  CNVT  GRNT REQUEST_L GRANT_LEV
---------------------- -------- ----- ----- ----- --------- ---------
[0x20][0x40000],[BL]   GRANTED      3     0     1 KJUSERNL  KJUSERPR
```

Observe the following:

- There is no physical read or logical read on the EMPLOYEES table in the master node. The query GET_SEGMENT_STATS on V$segment_statistics confirms this.
- Because there is no read on the EMPLOYEES table, no blocks from the EMPLOYEES table are cached in the master node.
- Because no blocks are cached, no resource is allocated to protect that block in instance D.
- The query GET_RESOURCE_STAT tells us that the resource is granted on a protected read level.

Example 2: Read the Block from Cache

Continuing from example 1, we run the same query from instance B. The data is already cached in instance C, and we should get the block from instance C. The block was read to instance C from disk by the disk read, and the block is not changed yet. The on-disk version and the cache version are the same because no modifications were made from instance C.

https://t.me/learningnets

Technet24

```
REM Execute the following query on instance B: (User SMITH's session)
SQL> select empno,ename,sal from employees
  2  where empno=7788;
     EMPNO ENAME             SAL
---------- ---------- ----------
      7788 SMITH            3000
#### Result of lock queries on instance B,
after a select from EMPLOYEES table on instance B
SQL> @get_segment_stats
OBJECT_NAM STAT                                VALUE
---------- ------------------------------------ -------
EMP        gc cr blocks received                     1

SQL> @get_buffer_stat
     STATE  MODE_HELD      FILE#     BLOCK#   SCN_BASE   SCN_WRAP
---------- ---------- ---------- ---------- ---------- ----------
         3          0          4         32     665905          0
SQL> @get_resource_name
no rows selected
SQL> @get_resource_stat
no rows selected
```

Observe the following:

- The block was retrieved from the cache and not from the disk, as indicated by “gc cr blocks received” for the segment in the v$segment_statisics view.
- The block is held in the current shared mode as indicated by 3 in the STATE column of the x$bh table. X$BH.STATE is 3 (CR).
- Because the buffer is in CR state, it does not require any locks to protect it, and no resources are allocated for that buffer in the master node.
- Master node is node 3 (where instance D is running; the node numbering starts with 0).

Now we’ll look at the change in lock status in the master instance D and the instance that was served the block, which is instance C.

Here are the lock statistics in the master instance:

https://t.me/learningnets

```
SQL> @get_resource_stat
RESOURCE_NAME          STATE     MAST  CNVT  GRNT REQUEST_L GRANT_LEV
---------------------- -------- ----- ----- ----- --------- ---------
[0x20][0x40000],[BL]   GRANTED      3     0     1 KJUSERNL  KJUSERPR

#### Result of lock queries on instance C
after a select from EMPLOYEES table on instance B

SQL> @get_buffer_stat
STATE  MODE_HELD      FILE#     BLOCK#   SCN_BASE   SCN_WRAP
----- ---------- ---------- ---------- ---------- ----------
    2          0          4         32          0          0
SQL> @get_resource_name
HEXNAME                   RESOURCE_NAME   KJBLGRANT   KJBLROLE KJBLREQUE
------------------------- --------------- --------- ---------- ---------
[0x20][0x40000],[BL]      32,262144,BL    KJUSERPR           0 KJUSERNL
```

Observe the following:

- The master instance granted the shared access to instance C.
- The SCUR lock mode is compatible with another CR mode without any conflicts.
- Instance C shipped the block to instance B over the interconnect as instructed by master instance D.
- The column KJBLGRANT shows KJUSERPR for the lock in SCUR mode in instance C.
- No additional locks are required to protect the CR buffer in instance A, so no lock status changes are in the master instance.
- A block present in the buffer cache in CR mode can be used locally by the instance for the query. However, it cannot be used for update and cannot be served to other instances; hence, the block does not require a lock.

At the end of example 2, the global cache has two copies of the buffer in two of its caches. Instance C has the buffer in SCUR mode, and instance B has the buffer in CR mode. The buffer is still in the local role, no one has yet updated the block, and there is no past image in any of the caches.

Example 3: Update the Row in Instance A

The block that holds the data about user SMITH (EMPNO=7788) is now cached in instances B and C. Instance C reads the block from disk and serves it to instance B via the interconnect. Then instance A updates the data and makes the buffer incompatible. Instance A does not commit the changes.

Now let’s update the block on instance A:

https://t.me/learningnets

Technet24

```
REM Let us update DML on instance A:

      UPDATE EMP SET SAL=SAL+1000 WHERE EMP=7788;

#### Result of lock queries on instance A

SQL> @get_segment_stats
OBJECT_NAM STAT                                    VALUE
---------- ---------------------------------- -------
EMP  gc current blocks received                        1

SQL> @get_buffer_stat
STATE  MODE_HELD      FILE#     BLOCK#   SCN_BASE   SCN_WRAP
----- ---------- ---------- ---------- ---------- ----------
    1          0          4         32          0          0

SQL> @get_resource_name
HEXNAME                   RESOURCE_NAME   KJBLGRANT   KJBLROLE KJBLREQUE
------------------------- --------------- --------- ---------- ---------
[0x20][0x40000],[BL]      32,262144,BL    KJUSEREX           0 KJUSERNL
SQL> @get_resource_stat
no rows selected
```

Observe the following:

- State 1 in x$bh shows that the block is held by instance A in exclusive mode in the current state.
- Statistic “gc current blocks received” indicates that the block is obtained from the global cache through Cache Fusion and not from the disk.
- Column KJBLGRANT has a value of KJUSEREX, indicating that an instance has an exclusive lock on this resource.
- Instance A has upgraded its lock from Null to X mode.

https://t.me/learningnets

```
REM Lock and Block statistics from node B and C

REM Buffer Status in Node B (After Instance A updated the Block)
SQL> @get_buffer_stat
STATE  MODE_HELD      FILE#     BLOCK#   SCN_BASE   SCN_WRAP
----- ---------- ---------- ---------- ---------- ----------
    3          0          4         32     669341          0
SQL> @get_resource_name
no rows selected

SQL> @get_resource_stat
no rows selected

REM  Buffer Status from Node C
SQL> @get_buffer_stat
STATE  MODE_HELD      FILE#     BLOCK#   SCN_BASE   SCN_WRAP
----- ---------- ---------- ---------- ---------- ----------
    3          0          4         32     669340          0
SQL> @get_resource_name
no rows selected

SQL> @get_resource_stat
no rows selected

REM Resource Statistics from Master Node
SQL> @get_resource_stat
RESOURCE_NAME          STATE     MAST  CNVT  GRNT REQUEST_L GRANT_LEV
---------------------- -------- ----- ----- ----- --------- ---------
[0x20][0x40000],[BL]   GRANTED      3     0     1 KJUSERNL  KJUSEREX
```

Note the following:

- Instance C downgraded its lock from SCUR to Null mode and has the buffer in CR mode only.
- The buffer status in instance C is changed from 2 (SCUR) to 3 (CR).
- Instances B and C no longer have any lock on the resource. A block present in the buffer cache in CR mode can be used locally by the instance for query. However, the block cannot be used for update and cannot be served to other instances, so it does not require any lock.
- The master instance shows the lock covering the resource is granted in exclusive mode.
- Instance A has the most recent block in the buffer cache. Now any instance requesting the block will be served by instance A.

Example 4: Update a New Row (in the Same Block) in Instance C

https://t.me/learningnets

Technet24

Continuing from example 3, we update the same block from another instance, but here we update a different row. Updating the same row will cause waits for the TX enqueue because we are working with a single instance. The buffer holding block 32 is held exclusively by instance A and protected by a BL enqueue. Row-level locks in an Oracle RAC instance operate similarly to those in a single instance. In this example, we update the same block but a different row.

Let's update the same block on instance C. The block already has an open transaction operating on empno 7788.

```
REM On instance C's SMITH session:
UPDATE EMP SET SAL = SAL*2 WHERE EMPNO=7369;
COMMIT;

Result of lock queries on instance C, after an update to EMPLOYEES
table on instance C

SQL> @get_segment_stats
OBJECT_NAM STAT                                    VALUE
---------- ---------------------------------- -------
EMP        physical reads                            1
EMP        gc current blocks received                1

SQL> @get_buffer_stat
STATE  MODE_HELD      FILE#     BLOCK#   SCN_BASE   SCN_WRAP
----- ---------- ---------- ---------- ---------- ----------
    1          0          4         32          0          0
SQL> @get_resource_name
HEXNAME                   RESOURCE_NAME   KJBLGRANT   KJBLROLE KJBLREQUE
------------------------- --------------- --------- ---------- ---------
[0x20][0x40000],[BL]      32,262144,BL    KJUSEREX           0 KJUSERNL

SQL> @get_resource_stat
no rows selected
```

Observe the following:

- The statistic "gc current blocks received …1" indicates that the instance received one block belonging to this segment (table) from a remote instance via GCS.
- A value of 1 in the STATE column of the X$BH view on instance C indicates that the block is current and is locked exclusively by this instance.
- The resource is held in exclusive mode as shown as KJUSEREX in the third query.

Now let's query the master node to see the lock statistics:

https://t.me/learningnets

```
REM Statistics from Master Node (Instance D)

SQL> @get_resource_stat
RESOURCE_NAME          STATE     MAST  CNVT  GRNT REQUEST_L GRANT_LEV
---------------------- -------- ----- ----- ----- --------- ---------
[0x20][0x40000],[BL]   GRANTED      3     0     1 KJUSERNL  KJUSEREX
[0x20][0x40000],[BL]   GRANTED      3     0     1 KJUSERNL  KJUSERNL
```

Now we'll go back to the instance that served the block to this instance; instance A served the current block to instance C:

```
REM Resource statistics in node A
SQL> @get_buffer_stat
     STATE  MODE_HELD      FILE#     BLOCK#   SCN_BASE   SCN_WRAP
---------- ---------- ---------- ---------- ---------- ----------
         8          0          4         32          0          0

SQL> @get_resource_name
HEXNAME                   RESOURCE_NAME   KJBLGRANT   KJBLROLE KJBLREQUE
------------------------- --------------- --------- ---------- ---------
[0x20][0x40000],[BL]      32,262144,BL    KJUSERNL           0 KJUSERNL
```

Note the following:

- Instance A down-converts the block from XCUR to PI (STATE equals 8).
- A Null mode lock, as indicated by KJUSERNL, protects the PI.
- Master node D shows the locks covering instance C as exclusive (KJUSEREX) and the lock covering instance A (KJUSERNL) for the PI image.
- The master keeps track of the PI images across the buffer cache. All these instances will be asked to discard the PI copies once the current block is written to disk.

If you look at examples 3 and 4 closely, you can understand that the buffer locks are independent of row-level locks. When the block is updated in instance C, a row-level lock is placed in the block. When the block is updated again, another row-level lock is placed in the block.

The block-level storage parameter INITRANS controls the initial number of row-level locks that can be taken on that block. As demand grows, the number of row-level locks in the block grows as long as the block has enough space to record the transaction information. Refer to Chapter 11 for a better understanding of Oracle's internal implementation of row-level locks.

Example 5: Commit the Block in Instance A and Select the Rows

Now the process operating on instance A decides to commit the changes. Commit operations in the Oracle RAC environment work the same as operations in a single-

https://t.me/learningnets

Technet24

instance environment as far as transactional locks are concerned. In the global cache, level commits do not make any lock status changes in the lock management layer. Now we will explore the query results after issuing a commit in instance A:

```
REM Let us commit our update done in example 3
REM In SMITH's session, we will issue a commit.
COMMIT;
```

The following are the results in instance A after issuing a commit:

```
Resource statistics in node A
SQL> @get_buffer_stat
     STATE  MODE_HELD      FILE#     BLOCK#   SCN_BASE   SCN_WRAP
---------- ---------- ---------- ---------- ---------- ----------
         8          0          4         32          0          0
SQL> @get_resource_name
HEXNAME                   RESOURCE_NAME   KJBLGRANT   KJBLROLE KJBLREQUE
------------------------- --------------- --------- ---------- ---------
[0x20][0x40000],[BL]      32,262144,BL    KJUSERNL           0 KJUSERNL

SQL> @get_segment_stats
OBJECT_NAM STAT                                 VALUE
---------- ---------------------------------- -------

EMP        gc current blocks received               1

REM Resource Statistics in  Instance C
SQL> @get_segment_stats

OBJECT_NAM STAT                                 VALUE
---------- ---------------------------------- -------

EMP        physical reads                           1
EMP        gc current blocks received               1
```

https://t.me/learningnets

```
SQL> @get_buffer_stat
STATE  MODE_HELD      FILE#     BLOCK#   SCN_BASE   SCN_WRAP
----- ---------- ---------- ---------- ---------- ----------
    1          0          4         32          0          0
SQL> @get_resource_name
HEXNAME                  RESOURCE_NAME   KJBLGRANT   KJBLROLE KJBLREQUE
------------------------ --------------- --------- ---------- ---------
[0x20][0x40000],[BL]     32,262144,BL    KJUSEREX           0 KJUSERNL

REM Statistics from Master Node (Instance D)
SQL> @get_resource_stat
RESOURCE_NAME          STATE     MAST  CNVT  GRNT REQUEST_L GRANT_LEV
---------------------- -------- ----- ----- ----- --------- ---------
[0x20][0x40000],[BL]   GRANTED      3     0     1 KJUSERNL  KJUSEREX
[0x20][0x40000],[BL]   GRANTED      3     0     1 KJUSERNL  KJUSERNL
```

Observe the following:

- Commits do not make any major changes in the global cache locks.

Example 6: Disk Writes Due to a Checkpoint

Continuing from example 5, the modified buffer from instance C is written to disk due to a checkpoint. We will explore the resource statistics after the checkpoint.

```
REM Let us trigger a checkpoint in Instance C
SQL> alter system checkpoint global;
System altered.

REM Resource Statistics in Instance A
SQL> @get_buffer_stat
STATE  MODE_HELD      FILE#     BLOCK#   SCN_BASE   SCN_WRAP
----- ---------- ---------- ---------- ---------- ----------
    3          0          4         32     669387          0
SQL> @get_resource_name
no rows selected
SQL> @get_resource_stat
no rows selected

REM Resource Statistics in  Instance C
SQL> @get_segment_stats
OBJECT_NAM STAT                                    VALUE
---------- ---------------------------------- -------

EMP        physical reads                            1
EMP        gc current blocks received                1
```

https://t.me/learningnets

Technet24

```
SQL> @get_buffer_stat
STATE  MODE_HELD      FILE#     BLOCK#    SCN_BASE   SCN_WRAP
----- ---------- ---------- ---------- ---------- ----------
    1          0          4         32          0          0
SQL> @get_resource_name
HEXNAME                  RESOURCE_NAME   KJBLGRANT   KJBLROLE KJBLREQUE
------------------------ --------------- --------- ---------- ---------
[0x20][0x40000],[BL]     32,262144,BL    KJUSEREX           0 KJUSERNL

Statistics from Master Node (Instance D)

SQL> @get_resource_stat
RESOURCE_NAME          STATE     MAST  CNVT  GRNT REQUEST_L GRANT_LEV
---------------------- -------- ----- ----- ----- --------- ---------
[0x20][0x40000],[BL]   GRANTED      3     0     1 KJUSERNL  KJUSEREX
```

Here are some observations:

- The checkpoint request from instance C to the master does not change any lock status in the current node.
- The PI from node A is discarded, and the buffer is changed to CR mode. Hence, the status of the buffer is changed from 8 to 3.
- Because the PI buffer is changed to CR, there is no need for the master to protect that resource. Therefore, the KJUSERNL lock is destroyed.
- Instance C also generates a BWR and writes it to redo log files. The BWR dump is shown here:

```
Block Written Record Dump in Redo Log File of Instance C

CHANGE #1 MEDIA RECOVERY MARKER SCN:0x0000.00000000 SEQ:  0 OP:23.1
 Block Written - afn: 4 rdba: 0x01000020 BFT:(1024,16777248) non-BFT:(4,32)
                   scn: 0x0000.001778e1 seq: 0x01 flg:0x00

REDO RECORD - Thread:1 RBA: 0x000006.00004d4a.0010 LEN: 0x0070 VLD: 0x06
SCN: 0x0000.00178185 SUBSCN:  1 08/19/2010 22:36:35
CHANGE #1 MEDIA RECOVERY MARKER SCN:0x0000.00000000 SEQ:  0 OP:23.1
```

Example 7: Instance Crash

At this time, instance D (the master instance for our interested resource EMP) crashes. Here's the reconfiguration information from the alert log of instance A:

https://t.me/learningnets

```
Reconfiguration started (old inc 8, new inc 10)
List of instances:
 1 2 3 (myinst: 1)
 Global Resource Directory frozen
 * dead instance detected - domain 0 invalid = TRUE
 Communication channels reestablished
* domain 0 not valid according to instance 3
 * domain 0 not valid according to instance 2
 Master broadcasted resource hash value bitmaps
 Non-local Process blocks cleaned out
LMS 0: 1 GCS shadows cancelled, 0 closed, 0 Xw survived
 Set master node info
 Submitted all remote-enqueue requests
 Dwn-cvts replayed, VALBLKs dubious
 All grantable enqueues granted
 Post SMON to start 1st pass IR

Instance recovery: looking for dead threads

Setting Resource Manager plan DEFAULT_MAINTENANCE_PLAN via parameter
Beginning instance recovery of 1 threads
 Submitted all GCS remote-cache requests
 Post SMON to start 1st pass IR
 Fix write in gcs resources
Reconfiguration complete
```

Example 8: Select the Data from Instance A

After instance reconfiguration, we select the data from instance A. Here are the resource statistics on the master node:

```
SQL> @get_segment_stats
OBJECT_NAM STAT                                    VALUE
---------- ---------------------------------- -------
EMP        gc cr blocks received                       1
EMP        gc current blocks received                  1

SQL> @get_buffer_stat

     STATE  MODE_HELD      FILE#     BLOCK#   SCN_BASE   SCN_WRAP
---------- ---------- ---------- ---------- ---------- ----------
         3          0          4         32     670125          0
SQL> @get_resource_name
no rows selected
SQL> @get_resource_stat
no rows selected
```

Upon instance recovery, the EMPLOYEES table is remastered in instance C. We

https://t.me/learningnets

Technet24

query X$KJBR with the resource name and confirm the resource is remastered in instance C. Alternatively, we can use **oradebug lkdebug -a <pkey>** to determine the details about the master of that resource.

```
REM Resource Statistics from the (new) master node
SQL> @get_master_info
   INST_ID KJBRNAME                  KJBRROLE KJBRMASTER
---------- ---------------------- ---------- ----------

         3 [0x20][0x40000],[BL]              0          2


SQL> @get_resource_stat
RESOURCE_NAME          STATE     MAST  CNVT  GRNT REQUEST_L GRANT_LEV
---------------------- -------- ----- ----- ----- --------- ---------
[0x20][0x40000],[BL]   GRANTED       3     0     1  KJUSERNL  KJUSEREX
```

Observe the following:

- Instance ID 3 is instance C, because instance numbers for the INST_ID start from 1. KJBRMASTER shows 2 because for KJBR the numbering starts from 0.
- Upon the master instance crash, the resources are equally distributed to the surviving nodes and the sample object (obj=51151, EMP) is remastered to instance C.

Resource Mastering and Remastering

If you have carefully read the concepts of Cache Fusion so far, the following should be clear:

- Resource usage (including locks on the data blocks) is tracked by GCS. GCS resources and responsibilities are equally distributed among all the instances.
- Every resource has a master, and the master instance knows the complete lock state of the resource at any point in time.
- When an instance needs a resource, it requests the desired lock from the master instance by sending a message.
- Based on the current status of the resource, the master grants access to the resource or asks the holder to disown the lock on that resource.
- If one instance accesses a particular resource more often than the others, it would be beneficial to master the resource to that instance to reduce the number of messages on ownership transfers.
- At any point in time a resource can be accessed at a maximum of three hops. This is true irrespective of the number of instances because only three parties are involved in any resource management operation.

https://t.me/learningnets

- If the application's data-access patterns are such that a set of data blocks is generally accessed by only one instance, then making the local instance the master of those resources can optimize the amount of messages sent to remote instances. Oracle keeps track of data block access patterns by instances and remasters the resources to optimize the messaging. By default, the Oracle kernel invokes the remastering algorithm every 10 minutes.
- Objects with a high number of read lock requests are treated specially with the "read-mostly" locking protocol. Object affinity for read-mostly and dynamic remastering can be controlled by the **_gc_affinity_locking** parameter. This parameter defaults to TRUE, which enables read-mostly and object remastering validation periodically.
- Undo affinity and object affinity are enabled by default. However, you can disable undo affinity by setting _gc_undo_affinity=false, and you can disable object affinity by setting _gc_affinity_locking=false. However, you can disable the affinity locks for undo and object affinity by setting _gc_affinity_locks=false.
- The undocumented initialization parameter **_gc_read_mostly_locking** controls the read-mostly policy evaluation periodically. Setting _gc_read_mostly_locking=false disables the read-mostly policy evaluation. You can manually force an object to be read-mostly using the following command:

  ```
  oradebug lkdebug –m pkey_readmostly <obj#>
  ```

  And you can dissolve the read-mostly status for an object using this command:

  ```
  oradebug lkdebug –m dpkey <obj#>
  ```

Background Processes and Cache Fusion

The following sections describe the functions performed by each background process as it relates to GCS.

LMON: Lock Monitor Process

The LMON background process maintains GCS memory structures. It handles the abnormal termination of processes and instances. The LMON process handles the reconfiguration of locks and resources when an instance joins or leaves the cluster. Activities that occur during instance reconfigurations are tracked by LMON in its trace file.

LMON is responsible for executing dynamic lock remastering every 10 minutes. It is known as the *Global Enqueue Services monitor*.

LMS: Lock Manager Server

LMS is the most active Oracle RAC background process; it can become very active,

https://t.me/learningnets

Technet24

consuming significant amounts of CPU time. Oracle recommends that you allocate this process the necessary CPU time by increasing its priority.

Oracle implements a feature to ensure that the LMS process doesn't encounter CPU starvation. LMS is also known as the *Global Cache Service (GCS) process*. This process is responsible for receiving remote messages and executing the desired GCS requests, which include the following operations:

- Retrieving requests from the server queue that are queued by LMD (discussed in the next section) to perform requested lock operations
- Rolling back any uncommitted transactions for any blocks that are being requested for consistent read by the remote instance
- Copying blocks from the holding instance's buffer cache and sending a read-consistent copy of the block to the requesting foreground process on the requesting instance to be placed into the buffer cache

LMS also sends messages to remote instances that aren't sent by the user processes. Each Oracle RAC instance can have two or more LMS processes. The internal view X$KJMSDP contains statistics for the work performed by each LMS process for the instance.

You can also set the number of LMS processes with the initialization parameter **GCS_SERVER_PROCESSES.** Oracle has been optimizing the default number of LMS processes, which varies from version to version. In most of the versions, the default value is as follows:

```
MIN(CPU_COUNT/2, 2))
```

In a single CPU machine, however, Oracle starts just one LMS process. You can consider increasing the value of this parameter if the global cache activity is very high.

LMD: Lock Manager Daemon Process (LMDn)

The LMD process performs global lock deadlock detection. It also monitors for lock conversion timeouts. LMD manages lock manager service requests for GCS resources and sends them to a service queue to be handled by the LMSn process. This process primarily handles lock requests for Global Enqueue resources. The internal view X$KJMDDP contains statistics for the work done by each LMD process for the instance.

LCKn: Lock Process (LCK0)

This process manages instance resource requests and cross-instance calls for shared resources. During instance recovery, it builds a list of invalid lock elements and validates lock elements.

https://t.me/learningnets

DIAG: Diagnostic Daemon (DIAG)

The diagnostic daemon process is a part of the diagnosability framework. It regularly monitors the health of the instance. It also checks for instance hangs and deadlocks. Most importantly, it captures the vital diagnostics data for instance and process failures.

Summary

Cache Fusion contributes to the extreme performance of Oracle Database RAC. In this chapter, you saw examples of how it works and what happens “under the hood” to enable the transparency of the sessions and connected applications. Understanding the workings of Cache Fusion will help you to appreciate its functionalities as well as enable you to design highly scalable system architectures.

Past image (PI) blocks and block-written records (BWRs) optimize the block transfer between instances and guarantee faster block recovery during instance failures in high-concurrency environments. The dynamic remastering and read-mostly affinity algorithms reduce the messages between master and requestor based on specific rules.

The background processes Lock Manager Daemon (LMD) and Global Cache Services (GCS) handle the block and lock transfers between the instances and play a key role in the implementation of Cache Fusion between the instances. The DIAG process enhances diagnosability, and the LCK processes play a significant role in Cache Fusion recovery.

https://t.me/learningnets

Technet24

CHAPTER 13

Workload Management, Connection Management, and Application Continuity

Managing various types of workloads is a challenging task in the Oracle RAC environment. Oracle RAC makes this process simpler by enabling you define services, which allow you to split the workload and distribute it among the available instances. You can create services and assign them to applications in order to manage workloads or groups of applications. Thus, if you have both online users and batch jobs accessing a RAC database, you can assign dedicated services specific characteristics and then assign them to the online users and the batch jobs.

Services are tightly integrated with the Oracle Resource Manager, allowing distributed transactions in the Oracle RAC database using services. Also, Oracle RAC can relocate services among instances in response to planned and unplanned outages, which greatly extends the availability and scalability of Oracle RAC environments.

Using services, although not mandatory, is highly recommended by Oracle to optimize the performance of applications to automate workload management. Ideally, you should create services so that all users who share them have similar service-level

https://t.me/learningnets

requirements.

Although services, also called *dynamic database services*, are the foundation of workload management in an Oracle RAC database, Oracle provides several important features to enable you to maintain a robust RAC database. Some of these features are brand new in Oracle Database 12*c*.

This chapter introduces you to the following key topics pertaining to workload management in Oracle RAC; they provide both high availability and scalability for all your applications:

- Oracle services (dynamic database services)
- Connection load management
- Fast application notification
- Transaction Guard
- Distributed transaction processing

Understanding Dynamic Database Services

Database services (or simply, *services)* are logical abstractions that help you manage workloads in an Oracle database by representing each workload with common attributes, service-level thresholds, and priorities.

Services help you manage the workload in an Oracle RAC environment. They provide a logical approach to group a workload so that users employing the same set of data, functionalities, and service-level requirements can be grouped together to use the same services. For example, an online transaction processing (OLTP) service can be defined for the users who execute small, short-lived transactions; a BATCH service can be defined for users executing long-running batch transactions; a data warehousing (DW) service can be defined for users employing data warehousing features; and so on. Another useful way to group users could be according to the type of functionalities used, such as sales, human resources, order entry, and so on.

In an Oracle database, a database service can span multiple instances, help balance the workload across the cluster, and provide high availability, with full location transparency. Following are some of the important features of services:

- You use services to efficiently distribute the workload. The Oracle RAC Load Balancing Advisory provides workload execution efficiency and system resource availability information to listeners and mid-tier servers to route the incoming workload in an optimal manner.
- You can configure services to provide high availability. You can configure multiple instances to provide the same services. If an instance or a node failure occurs, the workload can be relocated to another existing instance.
- Services provide a transparent way to direct workload. In many cases, users might not even notice the instance, node, or the machine from where the

https://t.me/learningnets

Technet24

services are being provided.

- You can define services for both policy-based and administrator-based databases.
- Services are tightly integrated with Oracle Resource Manager, and consumer groups are automatically assigned to sessions based on a session's service.
- Services are tightly integrated with the scheduler, which allows better workload management of the work done by the scheduler.
- Services allow capturing various metrics about the workload, thus making it possible to measure the amount of resources used for that workload or application. These are exposed in the dynamic views V$SERVICEMETRIC and V$SERVICE_STATS.

To manage workloads, you can define and assign services to a particular application or to a subset of an application's operations. You can also group other types of work under services. For example, online users can be a service, batch processing can be another, and reporting can be yet another service type. It's recommended that all users who share a service have the same service-level requirements. You can define specific characteristics for services, and each service is a separate unit of work. A service is the building block that can be used to divide work into logical workloads.

Services allow applications to benefit from the reliability of the redundant parts of the cluster. The services hide the complexity of the cluster from the client by providing a single system image for managing work.

Each service represents a workload with the following:

- Globally unique name
- Common service-level policies—performance, HA, and so on
- Common priority
- Common function and footprint

Services supply many benefits. They provide a single system image to manage competing applications, and they allow each workload to be managed in isolation and as a unit. Using standard user interfaces in DBCA, NetCA, SRVCTL, and Oracle Enterprise Manager (OEM) with services, you can configure, administer, enable, and disable a workload, as well as measure it as a single entity. OEM supports viewing and operating services as a whole, with the ability to drill down to the instance level when needed. A service can span one or more instances of an Oracle database or multiple databases in a global cluster, and a single instance can support multiple services. The number of instances offering the service is transparent to the application.

Services enable the automatic recovery of work. This is achieved according to business rules. Following outages, the service is recovered quickly and automatically at the surviving instances. When instances are later repaired, any services that aren't running are restored quickly and automatically by the Cluster Ready Services (CRS)

https://t.me/learningnets

daemon. The service immediately changes state from down to up status, and a notification is available for applications using the service, to trigger immediate recovery and load-balancing actions.

Services are dynamic and tightly integrated with Oracle Resource Manager. The resources assigned to a service can be augmented when the load increases, and they can be reduced when the load declines. This dynamic resource allocation provides a cost-effective solution for meeting demands as they occur. For example, services are measured automatically, and the performance is compared to service-level thresholds. Performance violations are reported to OEM, allowing the execution of automatic or scheduled solutions.

A number of RDBMS features act in concert to support services. The Automatic Workload Repository (AWR) collects a service's performance metrics, recording the service performance, including SQL execution times, wait classes, and resources consumed by the service. AWR alerts you when workloads exceed the service response time thresholds. The following service-related dynamic views help monitor various aspects of running services:

- **V$SERVICES** Displays information about the services in the database.
- **V$SERVICEMETRIC** Displays service metric values measured on the most recent time interval period for services currently running.
- **V$SERVICE_STATS** Shows a minimal set of performance statistics, such as how well a node is processing the SQL calls issued as part of a service.
- **V$SERVICE_EVENT** Shows the aggregated wait counts and wait times for each wait statistic.

Global Data Services (GDS) is a new Oracle Database service, introduced in Oracle Database 12*c*. GDS extends the traditional concept of services to a globally replicated environment that can include any combination of an Oracle RAC database, a single-instance database, Oracle Active Data Guard, and Oracle GoldenGate.

When you implement GDS, applications will see the set of replicated databases as a single data source. The goal is to allow you to deploy services anywhere within the global configurations in order to support load balancing of your workload, provide high availability, and offer regional affinity. GDS is based in dynamic workload management with the help of services, in addition to replication through Oracle Active Data Guard and Oracle GoldenGate.

Global services extend traditional database services and support all attributes of traditional services. In addition, you can configure attributes such as global service placement, replication lag, and region affinity. GDS offers load balancing and service failover across multiple Oracle databases that are replicated through Active Data Guard or GoldenGate.

Service Characteristics

https://t.me/learningnets

Technet24

Services are highly configurable, allowing you to set up complex and efficient workload management in an Oracle RAC database. Following are the characteristics you use when configuring a service:

- Service name
- Service edition
- Database role for a service
- Instance preference of a service
- Service management policy
- Server pool assignment
- Load-balancing of services
- Connection load-balancing goal

In the following sections, we discuss the service attributes you can define when creating a service.

Service Name

You must assign a unique name when creating a service. Clients accessing the RAC database will use this service name to connect to an instance.

Service Edition

Oracle's edition-based redefinition feature lets you change the structure of an object while the object is in use. You can also set the edition attribute of a service when creating or modifying a service. Services use the service edition you set as their initial session edition. By default, the initial session edition is the database edition.

Database Role for a Service

You can specify a database role for a service to indicate that the service should be automatically started when the Oracle Data Guard database role matches one of the service roles you specify when creating a service. You can use the **role** parameter when creating a service to specify the database role of a service. Here's how you can specify the **role** parameter within a **srvctl add service** command:

```
-role "[PRIMARY] [, PHYSICAL-STANDBY]
       [, LOGICAL_STANDBY] [, SNAPSHOT_STANDBY]
```

Instance Preference of a Service

When using an administrator-managed database, you can define which specific instances will support a service by designating the instances as either preferred or available. *Preferred* instances for a service are instances in which the service will be started, and these instances will serve the users. *Available* instances are the backup instances; the service will be started on these instances in the event of failure of

https://t.me/learningnets

preferred instances.

To view a service's high availability (HA) characteristics, you can use the **srvctl** command:

```
[oracle@alpha3 ~]$ srvctl config service -database PROD -service PROD_OLTP
Service name: PROD_OLTP
Service is enabled
Server pool: PROD_PROD_OLTP
Cardinality: 1
Disconnect: false
Service role: PRIMARY
Management policy: AUTOMATIC
DTP transaction: false
AQ HA notifications: false
Failover type: NONE
Failover method: NONE
TAF failover retries: 0
TAF failover delay: 0
Connection Load Balancing Goal: LONG
Runtime Load Balancing Goal: NONE
TAF policy specification: NONE
Preferred instances: PROD1
Available instances: PROD2, PROD3
```

NOTE

Users can specify the available and preferred database instances for administrator-managed databases only.

There is no need to define the available and preferred instances for policy-managed Oracle RAC databases because Oracle will manage the placement of services automatically.

You can define the available and preferred instances for an administrator-managed Oracle RAC database using DBCA, SRVCTL, or Oracle Enterprise Manager. This definition process creates a number of high availability (HA) resources that are managed by the clusterware to keep the services available. The DBCA and SRVCTL interfaces for a service ask the administrator to enter the list of instances that are the preferred locations for the service, plus any additional instances that are available to support the service in the event of an outage or planned operation. Once the service is created, Database Resource Manager can be used to create consumer groups that control the priority of the service.

When you define a service for an administrator-managed Oracle RAC database, you define which instances will normally support that service. These are known as the

https://t.me/learningnets

Technet24

preferred instances. You can also define other instances to support a service if the service's preferred instance fails. These are known as *available* instances.

When you specify the preferred instances, you are specifying the number of instances on which a service will initially run. For example, on a three-node cluster, the preferred configuration may offer the Payroll service from instance 1 and the ERP service from instances 2 and 3.

The number of preferred instances also sets the initial *cardinality* for that service, which means that the service will start on at least a number of database instances equal to the number of preferred instances. For example, if a service is configured with two preferred instances, Oracle Clusterware will set the cardinality for this service to 2, which means Oracle will always try to keep this service running on two database instances when the preferred instance fails.

Afterward, due either to instance availability or planned service relocations, a service may be running on an available instance. When a service moves to an available instance, Oracle does not move the service back to the preferred instance when the preferred instance comes back online. This is because the service is already running on the desired number of instances. All this provides a higher level of continual service availability and avoids a second service outage due to failback processing.

Service Management Policy

Oracle Database 12*c* allows you to specify the management policy for each server, which means you can configure the startup options for a database service. You can do this by specifying the **-policy** parameter when adding a service with the SRVCTL utility. The management policy specifies whether Oracle Clusterware should start an Oracle Net Service automatically when a database starts.

You can set the management policy to either manual or automatic. When you create a service, specify the **-policy** parameter with either AUTOMATIC or MANUAL as the value. The default value is AUTOMATIC. If you set the management policy to automatic (the default), the service starts automatically when you start the database. If you set the management policy to manual, you must manually start the service after the database starts. In some special occasions, such as when troubleshooting, the DBA may want to start the database services manually.

Oracle Net Services provides connection load balancing that lets you to spread user connections across all the instances supporting a service. Oracle RAC uses Fast Application Notification (FAN) to notify applications about configuration changes and the current service level provided by each instance where the service is enabled.

The easiest way to use FAN is to use the Oracle clients that have Fast Connection Failover (FCF), which means the clients have been integrated with the HA FAN events. These include Java Database Connectivity (JDBC), Oracle Call Interface (OCI), and ODP.NET. OCI clients can include clients with Transparent Application Failover (TAF) enabled. For OCI and ODP.NET, you must enable the service to send FAN high availability events—in other words, set AQ_HA_NOTIFICATIONS to TRUE.

https://t.me/learningnets

By default, when you create a database, Oracle defines one special database service for your Oracle RAC database. The default database service is always available on all instances in an Oracle RAC environment, unless the instance is in restricted mode. You cannot alter this service or its properties.

Server Pool Assignment

When configuring a policy-managed database, you can assign services to a server pool with the **-serverpool** parameter. You can specify the service as UNIFORM if it runs on all instances in the server pool, or as SINGLETON if it runs on only a single instance.

Oracle selects the instance where a service runs, if you configure singleton services. If that instance fails, Oracle automatically fails over the service to a different instance in the server pool. Note that Oracle Database Quality of Service Management (QoS Management) manages all singleton services if the maximum size of the **-serverpool** parameter is 1.

Load Balancing of Services

An Oracle service in Oracle RAC databases allows load balancing at two different levels (namely, client-side load balancing and server-side load balancing):

- *Client-side load balancing* implements the basic load-balancing methods by distributing the workload using a simple round-robin approach within the available servers, whereas the *server-side load balancing* method uses the Load Balancing Advisory to redirect the database connection to the least-loaded database instance or the best database instance meeting the service-level requirements.
- Under server-side load balancing, the SCAN listener uses the service-level "goal" settings (that is, **-clbgoal** and **-rlbgoal**) to channel connection requests to the best instance. We discuss goals in the following section.

Upon creation of an Oracle database with DBCA, server-side load balancing is automatically configured and a sample client-side load-balancing connection is defined in the tnsnames.ora file.

Connection Load-Balancing Goal

You must configure the service-level goal to implement the server-side load-balancing feature so that connection requests to the database services are redirected to the best database instance. The type of a service goal suggests the type of workload an Oracle Net Service is required to serve—or in simple terms, a service goal suggests the nature of a database connection.

Oracle Net Services can have one service goal: either LONG or SHORT. A LONG service goal is for long-lived connections, where a database connection is assumed to be active for a time and actively processing in the Oracle database instance, such as

https://t.me/learningnets

Technet24

Oracle Form sessions or sessions in the database connection pool.

A SHORT service goal, as the name suggests, is for short-duration database connections such as an application accessing the metadata from the master tables. Oracle balances the workload based on the goal of a service. Oracle uses the Load Balancing Advisory for services defined with a SHORT goal, whereas for a LONG goal service, Oracle balances the connection per instance using a session count per service.

You can use the SRVCTL utility to implement server-side load balancing. For example, the following SRVCTL command implements a SHORT goal for the PROD_OLTP service on the PROD database:

```
$srvctl modify service -database prod -service prod_oltp -clbgoal SHORT
```

And the following example shows how to modify a service named PROD_OLTP to define a connection load-balancing goal for long-lived sessions:

```
$ srvctl modify service -database prod -service prod_oltp -clbgoal LONG
```

Connection load balancing helps to efficiently distribute the database workload among the instances, and we discuss it in detail later in this chapter. You can select between two types of load-balancing goals: one based on the best service quality and the other based on throughput. You can also use the Load Balancing Advisory to let the client know the current service levels being provided by instances and get recommendations about redistributing workload among the instances. The Load Balancing Advisory is an attribute you can enable when creating a database service.

Services and Policy-Managed Databases

You can define services for both admin- and policy-managed databases, but there are some restrictions while creating services for a policy-managed database. Services in a policy-managed database are assigned to a server pool and can be defined as a SINGLETON or UNIFORM service.

You assign services to the server pool in which the database is hosted by specifying the parameter **-serverpool** with the SRVCTL utility. The **-cardinality** parameter lets you define a service as either UNIFORM or SINGLETON:

- A SINGLETON service runs on only one database instance in its server pool, and the user does not have control over which instance will serve the service. In case of failover, Oracle will automatically relocate the SINGLETON service to another available database instance within the same server pool. One of the main reasons for using a SINGLETON service is to host applications that don’t behave well in an Oracle RAC environment, or for applications that need to be isolated from other workloads in the same database.
- A UNIFORM service runs on all database instances in its server pool. The administrator-managed database uses the conventional method of assigning

https://t.me/learningnets

services to preferred and available database instances. Most of the discussion in this chapter pertains to service management in an administrator-managed database.

Resource Management and Services

Oracle Database Resource Manager is tightly integrated with Oracle Net Services. Oracle Database Resource Manager binds the services to the respective consumer groups and accordingly prioritizes the services within the database instance. When users connect to the database using a service, Oracle transparently assigns a database session to a consumer group mapped with the connected database service at runtime.

For example, suppose you have two services named HIGHPRI and LOWPRI that are assigned to high-priority and low-priority resource consumer groups. Oracle will prioritize the sessions that connect using the HIGHPRI service over those that connect using the LOWPRI service.

Use the DBMS_RESOURCE_MANAGER package to assign a resource group to a service.

Using Services with Oracle Scheduler

You can assign services to jobs created in the scheduler so that Oracle can efficiently manage the workload and so that the same jobs can utilize the full benefits of Oracle RAC. When a service is assigned to a job, Oracle identifies the work of the scheduler for workload management.

Administering Services

The following tools are available for administering services:

- **Database Configuration Assistant (DBCA)** An easy and intuitive GUI tool for creating and modifying services.
- **Oracle Enterprise Manager/Grid Control (OEM/GC)** An easy and intuitive GUI tool for managing services.
- **Server Control (SRVCTL)** A comprehensive command-line Oracle RAC administration tool. DBCA and OEM, in turn, call this tool.

Oracle highly recommends the use of SRVCTL and OEM for managing services.

NOTE

You shouldn’t use the DBMS_SERVICE package to modify a service that requires persistence across restarts. It is not recommended that you use the DBMS_SERVICE

https://t.me/learningnets

Technet24

package to modify a service that requires persistence across restarts. Service modifications using DBMS_SERVICE do not update the attributes of the cluster resources in the Oracle Cluster Registry (OCR). Oracle Clusterware updates the resource attributes stored in the OCR, and the changes made by the DBMS_SERVICE package will be lost. SRVCTL is the recommended tool for administering and managing services.

Default Database Services

During installation of the Oracle RAC database, Oracle automatically creates a database service that has the same name as the Oracle RAC database and is available to all database instances in the Oracle RAC database. In addition to the default database service, Oracle also employs the following two internal services:

- **SYS$BACKGROUND** This service is used by an instance's background processes only.
- **SYS$USERS** When users connect to the database without specifying a service, they use this service.

These two internal services are available on all the instances all the time, as long as the instance is up. You cannot relocate or disable these two services.

NOTE

It is recommended that you always create your own set of services rather than use the default SYS$USERS service because it does not give you the granular control you will need to manage the workload.

Creating a Service Using Oracle Enterprise Manager

OEM provides a GUI for managing cluster services. The Create Service page provides all the parameters used to create a service. This page can be located by navigating to the Cluster Database home page and then to the Cluster Managed Database Services page. Click the Create Service on the Cluster Managed Database Services page. This Create Service page allows you to configure various attributes of the Oracle service explained in the previous sections.

Creating a Service Using the Server Control Utility

Use the **srvctl add service** command to create a service for an administrator-managed database, as shown here:

```
$ srvctl add service -database prod -service my_service -preferred prod1,prod2
-available prod3,prod4
```

https://t.me/learningnets

For example, the following command creates a service named “test” that defines instances prod1 and prod2 as the preferred instances, instances prod3 and prod4 as available instances, and a basic TAF policy, as well as configures the service to start automatically using automatic management policy:

```
$ srvctl add service -database prod -service test - -preferred prod1, prod2
-available prod3, prod4 -tafpolicy  basic -policy AUTOMATIC
```

Note that the default value for the **-policy** attribute, which assigns the service management policy, is AUTOMATIC.

The following example creates a singleton service test for policy-managed database ORCL using server pool SRVPL1:

```
$ srvctl add service -db prod -service test -serverpool SRVPL1
-cardinality singleton -policy AUTOMATIC
```

Note that you use preferred and available instances with server-managed databases. In order to create a service for a policy-managed database, use server pools instead of preferred/available instances by specifying the **-serverpool** parameter, as shown here:

```
$ srvctl add service -database prod -service my_service -serverpool spool1
```

You can also use the new **eval** parameter with the previous command, to evaluate the impact of creating the new service, without actually creating one. Here’s an example.

```
$ srvctl add service -database prod -service my_service -serverpool spool1 -eval
```

If you’re using pluggable databases (PDBs), specify the pluggable database name with the **-pdb** parameter when creating the service. You can do this for both administrator-managed and policy-managed databases. The following example shows how to do this for a policy-managed database:

```
$ srvctl add service -database prod -service my_service -serverpool spool1
-pdb -mypdb1
```

By specifying the **-pdb** property here, you’ve associated the service my_service with the PDB named mypdb1.

You can add several parameters to the **srvctl add service** command, to specify features such as TAF, Application Continuity, and the Load Balancing Advisory—all features we discuss later in this chapter. Here’s an example that shows what a more complex service-creation statement would look like:

```
$ srvctl add service -database prod -service my_service -notification TRUE \
   -failovermethod BASIC -failovertype SELECT -clbgoal 5
```

We’ll explain a number of the attributes you can specify for services throughout this

https://t.me/learningnets

Technet24

chapter, when we introduce the relevant concepts.

The following is the syntax for creating a service using the server control utility:

```
$ srvctl add service -database database_name -service service_name
-preferred preferred_list -available available_list  -serverpool  server_pool
-policy  [AUTOMATIC | MANUAL] -TAFPOLICY [BASIC | NONE | PRECONNECT]
```

Modifying a Service

You can modify existing services with the **srvctl modify service** command. For example, you can use this command to move services to a different instance, change serviced attributes such as the failover delay, or add/remove a preferred or available instance. Here's an example that shows how to move a service from one instance to another:

```
$ srvctl modify service -database prod -service finance -oldinst prod1 -newinst node2
```

Starting and Stopping a Service

In order for a service to be available for client connections, you must start the service. You can start a service with the **srvctl start service** command, as shown here:

```
$ srvctl start service -db db_unique_name [-service service_name_list]
[-instance inst_name] [-startoption start_option]
```

Use the **srvctl stop service** command to stop a service:

```
$ srvctl stop service -db db_unique_name [-service service_name_list]
[-instance inst_name] [-stopoption stop_option]
```

Note that if you've configured a service for automatically starting up, by specifying the automatic management policy (**-policy AUTOMATIC**), you cause the services to start up automatically.

Enabling and Disabling Services

You enable a disabled service in the following manner:

```
$ srvctl enable service -db db_unique_name [-service service_name_list]
[-instance inst_name]
```

You can disable a service with the following command:

```
$ srvctl disable service -db db_unique_name [-service service_name_list]
[-instance inst_name]
```

Relocating a Failed Service

Once a set of services fails over to another surviving instance and eventually when the failed instance is restarted, all the migrated services do not automatically come back to

https://t.me/learningnets

the restarted instance. For example, once the failed node (say, node1) restarts, it reclaims its own virtual IP (VIP), which was temporarily taken over by another node (say, node2).

You must manually relocate services. An example of the ERP service is provided next, where the service is relocated from node2 back to node1:

```
[node1]/> srvctl relocate service -db PROD -service ERP -oldinst  RAC2 -newinst RAC1
```

NOTE

You may also move a service from one instance to another after adding or deleting nodes, in order to balance the workload across the cluster.

Checking the Status of a Service

You check the status of a service as shown here:

```
$ srvctl status service –db  prod
```

Checking the Configuration of a Service

You can check the high availability configuration of a service with the following command:

```
$ srvctl config service –db prod –service finance
```

Using Views to Get Service Information

The V$SERVICES view contains information about services that have been started on that instance—that is, services being currently served by that instance. Here’s some sample output from V$SERVICES:

```
SQL> SELECT NAME,NETWORK_NAME, CREATION_DATE, GOAL, DTP, AQ_HA_NOTIFICATION,
CLB_GOAL FROM V$SERVICES;

NAME             NETWORK_NAME    CREATION_ GOAL         D AQ_ CLB_G
---------------- --------------- --------- ------------ - --- -----
prodXDB           prodXDB        10-AUG-09 NONE         N NO  LONG
prod              prod           10-AUG-09 NONE         N NO  LONG
```

https://t.me/learningnets

Technet24

```
SYS$BACKGROUND                    30-JUN-09 NONE         N NO  SHORT
SYS$USERS                         30-JUN-09 NONE         N NO  SHORT

SQL> SELECT NAME,NETWORK_NAME, CREATION_DATE, GOAL, DTP, AQ_HA_
NOTIFICATION,CLB_GOAL FROM V$SERVICES;

NAME              NETWORK_NAME  CREATION_   GOAL          D AQ_ CLB_G
---------------   ------------  ---------   ------------ - --- -----
PROD_BATCH        PROD_BATCH    23/AUG/10   NONE          N NO  LONG
PROD_OTLP         PROD_OLTP     23/AUG/10   NONE          N NO  LONG
PRODXDB           PRODXDB       20/AUG/10   NONE          N NO  LONG
PROD              PROD          20/AUG/10   NONE          N NO  LONG
SYS$BACKGROUND                  15/AUG/09   NONE          N NO  SHORT
SYS$USERS                       15/AUG/09   NONE          N NO  SHORT
```

Note that services that have not been started on the instance are not shown in this view.

Distributed Transaction Processing

A distributed or global transaction (XA transaction) can span across the database instances in a cluster database. Distributed transactions are tightly integrated with the Oracle 12*c* RAC database; thus, they can take advantage of the numerous high availability and stability benefits offered by Oracle RAC. In order to use Oracle RAC with global transactions, set the distributed transaction parameter (**-dtp**) to TRUE using SRVCTL.

In order to support distribution transaction processing, Oracle uses a background process called the global transaction process (GTX*n)*, where *n* is the number of the Oracle Global Transaction process. The number of global transaction processes are controlled by the Oracle database initialization parameter GLOBAL_TXN_PROCESSES. Setting this parameter to 0 disables the XA support for the Oracle RAC database. Oracle sets this parameter to a value of 1 so that, by default, Oracle RAC databases span distributed transactions across the available database instances.

Users are still allowed to add a singleton Distributed Transaction Processing (DTP) service in case they observe some performance degradation with the distributed transaction. Users can direct all branches of the distributed transactions to one database instance, similar to Oracle singleton services in prior releases of Oracle RAC databases.

When configuring DTP services, you must create at least one DTP service for each instance that handles a distributed transaction. If you have multiple distributed transactions, choose a DTP service running on a separate instance for each such transaction. Because all branches of a distributed transaction will run on a single instance, you can balance the total workload of all DTP transactions through multiple singleton services.

https://t.me/learningnets

DTP services ensure that all globally distributed transactions performed have their tightly coupled branches running on a single Oracle RAC instance. By default DTP is disabled on a singleton service, but you can modify a singleton service by setting the **dtp** parameter of the service to TRUE. The following examples show how to configure DTP services in your cluster.

Create a singleton service as shown here. If you're using administrator-managed databases, you must define only one instance as the preferred instance.

```
$ srvctl add service -db crm -service xa_01.example.com -preferred    ORCL1
    -available ORCL2, ORCL3
```

If you're using a policy-managed database, you must specify a server pool instead of a preferred instance. You must also set the service cardinality to SINGLETON, as shown in the following example:

```
$ srvctl add service -db prod -service xa_01.example.com -serverpool dtp_pool
    -cardinality SINGLETON
```

Configure DTP by setting the **dtp** parameter to TRUE. By default, DTP is turned off.

```
$ srvctl modify service -db prod -service xa_01.example.com -dtp TRUE
```

Although we've discussed how you can set up DTP services in your cluster, they aren't the only means to handle distributed transactions in a cluster. Instead of using DTP services, you can also use XA affinity. Oracle recommends (starting with Oracle 11.1) that you use global transactions with XA affinity instead of configuring DTP-enabled services.

AQ_HA_Notifications

When AQ_HA_Notifications is set to TRUE, information about an instance being up or down (or about similar events) will be sent to the mid-tier servers via the advance queuing mechanism. Some applications such as .NET applications cannot use FAN events, and in such cases AQ notifications are useful. You may need to enable the AQ HA events for services while configuring client failover in an Oracle Data Guard environment for OCI applications using these services.

Using Application Contexts

Oracle collects important statistics for the work done by a service, and user applications can qualify a service by action or module name to easily identify transactions within a service. It is strongly recommend that you set module and action names because this allows you to identify the culprit transactions or application code while optimizing the database or application. The DBMS_APPLICATION_INFO package can be used to set module and action names for the application programs.

https://t.me/learningnets

Technet24

For example, suppose we are creating a table called TAB1 and have inserted a record into the table and then updated the same record from two different sessions at the same time. Therefore, we'll set the application context information on another session where the update statement is expected to wait, and we'll query the V$SESSION dynamic performance view for additional application information on the problematic session:

```
CREATE TABLE TAB1 (COL1 VARCHAR2 (10));
INSERT INTO TAB1 VALUES ('FIRST');
UPDATE TAB1 SET COL1 = 'SECOND';
```

From another database session, let's try to update the recently inserted record. However, before executing the DML, let's set the module name, action name, and client info using the DBMS_APPLICATION_INFO package:

```
exec DBMS_APPLICATION_INFO.SET_MODULE
( MODULE_NAME => 'UPDATE RECORD', ACTION_NAME => 'UPDATE TAB1');
exec DBMS_APPLICATION_INFO.SET_CLIENT_INFO ( CLIENT_INFO => 'SUNRAY');
UPDATE TAB1 SET COL1 = 'UPDATED';
```

This session will appear to hang (wait forever) because the same row is being updated by another session and not committed. Because we set the application context using the DBMS_APPLICATION_INFO package, we can easily identify the problematic code location via a simple query to the V$SESSION dynamic performance view:

```
SELECT SID,EVENT,MODULE,ACTION,CLIENT_INFO
FROM V$SESSION WHERE STATE='WAITING' AND
WAIT_CLASS='Application';
SID EVENT                          MODULE         ACTION        CLIENT_INFO
388 enq: TX - row lock contention  UPDATE RECORD  UPDATE TAB1   SUNRAY
```

As shown here, V$SESSION exposes the application context information set in the DBMS_APPLICATION_INFO package, which can be quite handy while troubleshooting application performance issues.

Refer to the "Oracle Database PL/SQL Packages and Types Reference" for details and more examples using this wonderful package that can make your life a lot easier, especially when you are troubleshooting custom code.

Workload Distribution and Load Balancing

Workload distribution can sometimes be called *load balancing*. It involves managing user connections in such a way that the work they perform is distributed uniformly across Oracle RAC nodes or cluster database instances. However, this does not necessarily mean that the *work itself* is evenly spread between nodes due to this mechanism. Instead, the distribution of work is an indirect result of the connection

https://t.me/learningnets

distribution.

The workload distribution strategy must be determined during the test implementation phase of application deployment (if not earlier) to make it clear how the application is actually going to connect to the various nodes and database instances. Designers and architects should work out the application behavior with adequate testing to understand how the system performs under a given workload; this information provides a benchmark for the live implementation.

Workload distribution can occur on a per-connection basis, based on server-CPU load or on application/module usage. Certain strategies may be more appropriate than others, depending on the implementation and type of usage. The implementation architecture also plays a crucial role in the performance of the system as well as the way the workload is distributed.

Consider, for example, a four-node Oracle RAC database running Oracle E-Business Suite R12*i*. Multiple applications and web servers on the system may or may not be clustered. A total of 8000 users (connections) access the database via a dedicated server mechanism, and no external, intelligent network devices route the connection request(s) to a particular Oracle RAC instance based on some predetermined criteria.

Using Oracle's workload distribution mechanism, you can ensure that about 2000 clients connect to each Oracle RAC node, thereby ensuring that the *connections* are balanced—but not necessarily the *work* these connections perform. In this system, one particular node is more heavily loaded than the others, because the users connected to that node are firing many batch jobs, while the other nodes are running user modules—Inventory, Payables, GL, and the like. Nevertheless, you would like these connections to perform equal units of work, thereby achieving the ever-elusive and mirage-like *load-balanced* system.

One of the most difficult tasks to achieve is a complete and perfectly balanced system—unless the application behavior is very well analyzed, the users are well educated, and the application access is restricted to one node, ensuring a uniform usage pattern. This is easier said than done, however, as many of us have discovered over the years. If only one database server responds to all the incoming requests, the capacity of the server will not be able to handle high volumes of incoming traffic once the workload increases. Needless to say, application and database response times take a beating—and so do the DBAs. An increase in work and connections to the database server can lead to a point at which upgrading the server hardware will no longer be cost effective.

To achieve server scalability, more servers need to be added to distribute the load among the cluster. Distributing the load among these clustered servers is known as *load balancing*. Load balancing and clustering go hand in hand and can be applied to various types of applications such as Internet web servers, telecom applications, and network information servers.

Generic load balancing can become a highly technical and complicated topic and is best treated by network and systems professionals who operate large networks. Load balancing is decided at the system architecture design level, although the

https://t.me/learningnets

Technet24

implementation mechanisms and methods could differ, depending on the application and type of load balancing being done. In this discussion, we look at load balancing particularly from an Oracle perspective.

In Oracle database applications, load balancing is the attempt to achieve a balanced number of database connections across all the database servers. Load balancing also distributes the services across all the nodes in a cluster implementation. Load balancing can be accomplished using a hardware device or by using Oracle's own software mechanism, as you will learn in the following few sections.

Oracle attempts load balancing by introducing the workload distribution principle. By ensuring a uniform set of connections across all the nodes, we can hope that all the servers are equally loaded to achieve maximum scalability—of course, depending on many other factors (too many to discuss here). We can hope that each user will perform equivalent amounts of work, irrespective of the node to which each user is logged in.

Application designers and implementers like to restrict usage of an application/module to a particular node only in hopes of achieving balance. For example, if we have deployed General Ledger, Payables, Accounts Receivable, and Inventory in our organization on a four-node cluster for 8000 users, we might restrict usage of General Ledger and Payables to nodes 1 and 2, respectively. Nodes 3 and 4 can run only Accounts Receivable and Inventory, respectively. This method—commonly known as *application partitioning*—can sometimes cause an imbalanced cluster. In such cases, each node accepts a maximum of 2000 connections, and a node runs only one application at a time.

Using this method, we could end up in a situation where nodes 1 and 2 are loaded evenly and the other two nodes are heavily loaded. It is also possible that between the last two nodes, node 4 could be running at near 100 percent CPU and memory utilization due to the intense workload generated by the Inventory users. Therefore, it is quite possible that an acceptable approach, such as application partitioning, can sometimes yield unexpected or undesirable results.

Problems occur because the work patterns and resource usage of each application can be different. The way they scale and their behavior with increased workload can be vastly different. Hence, it is extremely crucial to test and measure workload profile and performance statistics *before* implementing an application and/or before deploying soft load balancing (application partitioning).

NOTE

Designers and architects should also work out whether a policy-managed database or an administrator-managed database would suit the workload. A policy-managed database suits big installations where multiple cluster databases are running from the same Oracle Grid Infrastructure, whereas an administrator-managed database suits small installations where the DBA can decide the placement of the cluster database

https://t.me/learningnets

instance resources within the cluster.

Hardware and Software Load Balancing

The deciding factors for choosing one method of load balancing over another depends on the requirements, available features, complexity of implementation, and cost. We'll consider hardware- and software-based load balancing here, because they are more applicable to the Oracle database application.

Hardware load-balancing devices can route TCP/IP packets to various servers in a cluster. These types of load balancers provide a robust topology with high availability but come at a much higher cost. On one hand, this method uses a circuit-level network gateway to route traffic. However, its implementation is more expensive than software implementations.

The most commonly used load balancers are software based and are used mainly as additional/optional features of software applications such as web servers. Oracle provides some form of load balancing as an integrated component of its database software. Software-based load balancers are cheaper than their hardware counterparts. Software is more configurable based on requirements, and it can incorporate intelligent routing based on multiple input parameters. On the other hand, additional hardware might be needed to isolate the load-balancing application.

Many applications like to balance their workloads across all the instances of an Oracle RAC database. Usually, load balancing with Oracle (excluding the parallel feature) means connection load balancing—that is, the connections to the database are balanced on a "round-robin" basis, on CPU load, or on session count at connect time. You can implement two types of load balancing: one on the client side and the other on the server side. We look at these two types of load balancing in the following sections.

Client-Side Load Balancing

Oracle's client-side load-balancing feature enables clients to randomize connection requests among all the available listeners. Oracle Net Services progresses through the list of protocol addresses in a round-robin sequence, balancing the load on various listeners. This normally is referred to as *client-side connect-time load balancing*. Without client-side load balancing, Oracle Net Services progresses through the list of protocol addresses sequentially until one succeeds. Figure 13-1 illustrates client-side load balancing.

https://t.me/learningnets

Technet24

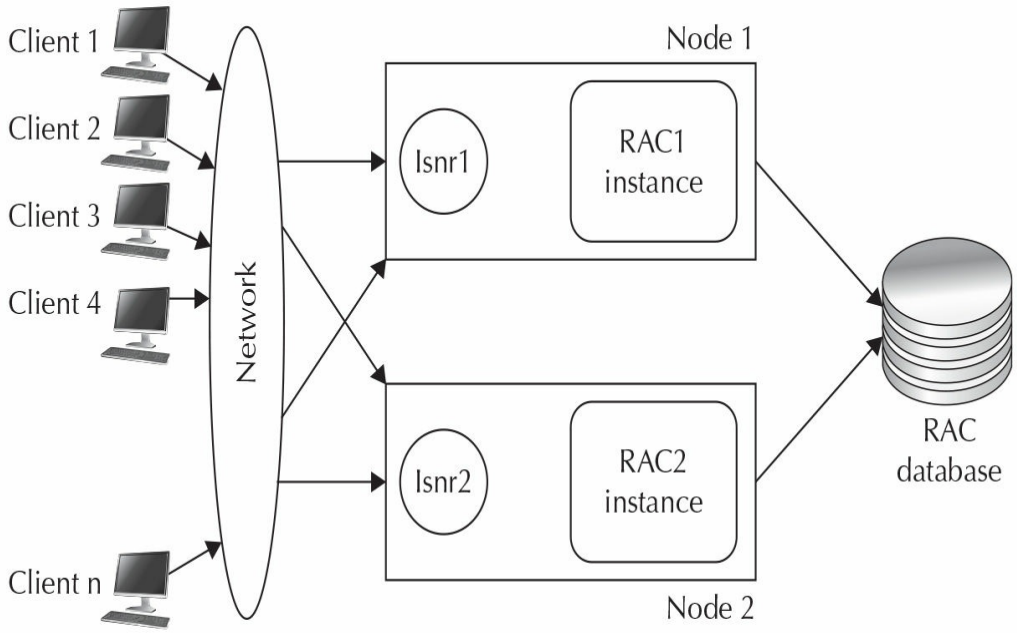


FIGURE 13-1. *Client-side load balancing*

You define client-side load balancing in the client connection definition file (tnsnames.ora) by setting the parameter LOAD_BALANCE = ON. When this parameter is set to ON, Oracle Net Services progresses through the list of listener addresses in a random sequence, balancing the load on the various listeners. When it's set to OFF, the addresses are tried sequentially until one succeeds. This parameter must be correctly specified in the Net Services name (connect descriptor). By default, this parameter is set to ON for the DESCRIPTION_LIST.

When SCAN (Single Client Access Name) is used in the connection definition, the Oracle database randomly connects to one of the available SCAN listeners in a round-robin fashion and balances the connections on the three SCAN listeners. Therefore, client-side load balancing is irrelevant for clients supporting SCAN access, because Oracle Net Services automatically load-balance the client connection requests across all IP addresses defined as part of the SCAN. Here is a sample tnsnames.ora (client and server) configuration for a four-node Oracle RAC database using SCAN:

```
RAC =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = SCAN-HOSTNAME)(PORT = 1701))
    (CONNECT_DATA =
      (SERVICE_NAME = RAC)
    )
  )
```

Upon receiving a connection request from the users using the preceding connection details, SCAN will connect to one of the SCAN listeners defined by SCAN in a round-robin fashion. Once SCAN is connected to a SCAN listener, the SCAN listener will

https://t.me/learningnets

identify the least-loaded cluster database instance providing the requested service. Once the least-loaded instance is identified, the SCAN listener will redirect the client to the local listener of the cluster node running the least-loaded cluster database instance. The local listener will connect the client to the database instance.

NOTE
Oracle does not load-balance when the client connects to the database using EZConnect.

Load balancing can also be specified for an ADDRESS_LIST or associated with a set of addresses or set descriptions. If you use ADDRESS_LIST, LOAD_BALANCE = ON should be within the (ADDRESS_LIST =) portion. If you do not use ADDRESS_LIST, LOAD_BALANCE = ON should be within the (DESCRIPTION =) portion. Here is a sample tnsnames.ora (client and server) configuration for a four-node Oracle RAC database:

```
RAC =
  (DESCRIPTION =
    (LOAD_BALANCE = ON)
    (ADDRESS = (PROTOCOL = TCP)(HOST = node1-vip)(PORT = 1541))
    (ADDRESS = (PROTOCOL = TCP)(HOST = node2-vip)(PORT = 1541))
    (ADDRESS = (PROTOCOL = TCP)(HOST = node3-vip)(PORT = 1541))
    (ADDRESS = (PROTOCOL = TCP)(HOST = node4-vip)(PORT = 1541))
    (CONNECT_DATA = (SERVICE_NAME = RAC)
  ))
```

Alternatively, you can configure it in another way. Because LOAD_BALANCE is ON by default, the reference is not needed unless you want to turn it off. Most examples show LOAD_BALANCE = ON when the Oracle Connection Manager is being used.

```
RAC =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = node1-vip)(PORT = 1541))
      (ADDRESS = (PROTOCOL = TCP)(HOST = node2-vip)(PORT = 1541))
      (ADDRESS = (PROTOCOL = TCP)(HOST = node3-vip)(PORT = 1541))
      (ADDRESS = (PROTOCOL = TCP)(HOST = node4-vip)(PORT = 1541))
    )
    (CONNECT_DATA = (SERVICE_NAME = RAC)
  ))
```

The listener.ora file on *each* node will look like this NODE1 file:

https://t.me/learningnets

Technet24

```
LISTENER=(DESCRIPTION=(ADDRESS_LIST=
(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER))))          # line added by Agent
LISTENER_SCAN3=(DESCRIPTION=(ADDRESS_LIST=
(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER_SCAN3))))    # line added by Agent
LISTENER_SCAN2=(DESCRIPTION=(ADDRESS_LIST=
(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER_SCAN2))))    # line added by Agent
LISTENER_SCAN1=(DESCRIPTION=(ADDRESS_LIST=
(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER_SCAN1))))    # line added by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN1=ON  # line added by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN2=ON  # line added by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN3=ON  # line added by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER=ON        # line added by Agent
```

The listener.ora file on each node will define three SCAN listeners and one local listener, as shown in the preceding listener.ora.

The tnsnames.ora file on each node (as well as your client PCs or other hosts) may also contain the individual entries for each of the Oracle RAC instances, as shown next. This is useful when a client program wants to connect to a specific node.

```
RAC1 =
  (DESCRIPTION =
    (ADDRESS_LIST =
```

https://t.me/learningnets

```
      (ADDRESS = (PROTOCOL = TCP)(HOST = NODE1-vip)(PORT = 1541))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = RAC)
      (INSTANCE_NAME = RAC1)
    ))

RAC2 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = NODE2-vip)(PORT = 1541))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = RAC)
      (INSTANCE_NAME = RAC2)
    ))

RAC3 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = NODE3-vip)(PORT = 1541))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = RAC)
      (INSTANCE_NAME = RAC3)
    ))

RAC4 =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = NODE4-vip)(PORT = 1541))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = RAC)
      (INSTANCE_NAME = RAC4)
    ))
```

Oracle automatically sets the value for the LOCAL_LISTENER database parameter, so there is no need to set this parameter manually. The LOCAL_LISTENER database parameter allows the database instance to know about the local listener.

Here’s a sample value for the LOCAL_LISTENER database parameter:

```
(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=NODE1-VIP)(PORT=1541))))
```

In this way, the tnsnames.ora file on each node should be able to resolve its local

https://t.me/learningnets

Technet24

listener, and listeners on each cluster node know about each other. With this, the configuration is complete; now when clients start to connect using the Oracle RAC service name (which matches the init.ora/SPFILE parameter SERVICE_NAME) from a client terminal using SQL*Plus or any other front-end application, the connections are randomly sent to each of the active nodes. This is the round-robin feature mentioned earlier. An **lsnrctl services** command can reveal the statistics for each node.

Note that provisioning a "random" method of a database connection mechanism can also be done using a hardware device that is configured appropriately. Cost is a major factor to be considered, and if the database instance is down for some reason, the hardware device has no way of knowing it, even though it has the intelligence to recognize that the node is up and running. This lack of database knowledge can cause delays or timeouts for connections. Oracle helps you overcome these limitations with its simple client-side load-balancing feature.

One of the disadvantages of the client-side method of load balancing is that client connections won't be able to gauge the load and other resource consumption factors on the instance/node to which they are trying to connect. Hence, timeouts can also occur if the node being attempted for connection is heavily loaded and unable to respond quickly.

In any load-balancing scheme, regardless of the connection method, once a connection is established, that connection continues to stay on that instance (until it terminates) and does not migrate to other less heavily loaded nodes in case the usage spikes up. This is an important misconception that even many DBAs and application technologists hold on to.

Server-Side Load Balancing

Server-side load balancing is also known as *listener-connection load balancing*. This feature improves connection performance by balancing the number of active connections among multiple dispatchers and instances.

In a single-instance environment (when shared servers are used), the listener selects the least-loaded dispatcher to handle the incoming client requests. When you create a database with DBCA, it automatically enables server-side load balancing. It also creates sample client-side load-balancing information in the tnsnames.ora file on the server.

The core of server-side load balancing is *dynamic service registration*, which is a feature where listeners on all the cluster nodes register themselves with other listeners in the cluster to know the database instances and services provided by the databases in the cluster. Dynamic service registration is implemented by PMON (Process Monitor), where a listener is always aware of all instances and dispatchers regardless of their location. Depending on the load information, a listener decides to which instance and, if a shared server is configured, to which dispatcher the incoming client request will be sent. Figure 13-2 illustrates typical server-side load balancing.

https://t.me/learningnets

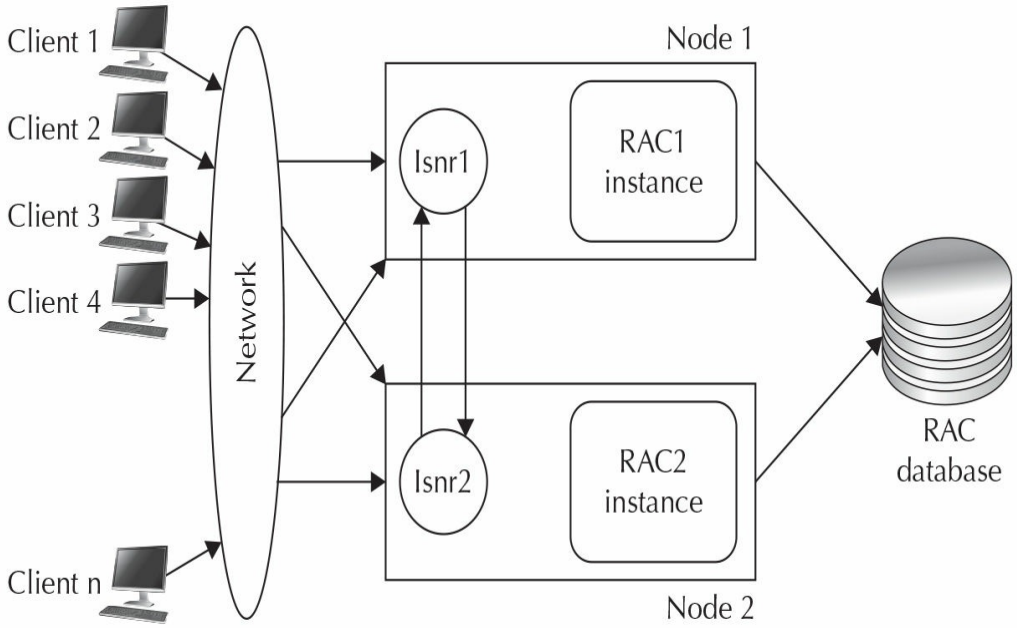


FIGURE 13-2. *Server-side load balancing*

In a shared server configuration, a listener selects a dispatcher in the following order:

1. Least-loaded node
2. Least-loaded instance
3. Least-loaded dispatcher for that instance

In a dedicated server configuration, a listener selects an instance in the following order:

1. Least-loaded node
2. Least-loaded instance

If a database service has multiple instances on multiple nodes, the listener chooses the least-loaded instance on the least-loaded node. If a shared server is configured, the least-loaded dispatcher of the selected instance is chosen.

You use the initialization parameter REMOTE_LISTENER to enable listener-connection load balancing. The entry on every node would look like this:

```
*.remote_listener=<SCAN_NAME>:<SCAN_PORT>
```

Oracle sets the REMOTE_LISTENER database parameter, and the Oracle Clusterware Database Agent manages the parameter.

The listener considers the current workload being executed on all the available instances for the requested database service. With server-side load balancing, the

https://t.me/learningnets

Technet24

listener routes incoming client connections to instances on the least-loaded nodes. The listener determines the connection distribution based on profile statistics that are dynamically updated by PMON. The higher the node load, the more frequently PMON updates the load profile. Updates occur in as little as 3 seconds on heavily loaded nodes and may take up to 10 minutes on lightly loaded nodes.

The service updates are printed to the listener log as follows:

```
(CONNECT_DATA=(SERVICE_NAME=PROD)(INSTANCE_NAME=PROD1)(CID=(PROGRAM=)(HOST=erpa
s2)(USER=applprod))) * (ADDRESS=(PROTOCOL=tcp)(HOST=130.188.1.214)(PORT=48604))
* establish * PROD * 0
19-NOV-2017 18:07:27 * service_update * PROD2 * 0
19-NOV-2017  18:07:29 * service_update * PROD1 * 0
19-NOV-2017  18:10:12 *
(CONNECT_DATA=(SERVICE_NAME=PROD)(INSTANCE_NAME=PROD1)(CID=(PROGRAM=)(HOST=erpd
b1)(USER=applprod))) * (ADDRESS=(PROTOCOL=tcp)(HOST=130.188.1.215)(PORT=47933))
* establish * PROD * 0
19-NOV-2017  18:10:12 *
(CONNECT_DATA=(SERVICE_NAME=PROD)(INSTANCE_NAME=PROD1)(CID=(PROGRAM=)(HOST=erpd
b1)(USER=applprod))) * (ADDRESS=(PROTOCOL=tcp)(HOST=130.188.1.215)(PORT=47934))
* establish * PROD * 0
19-NOV-2017 18:12:13 * service_update * PROD2 * 0
19-NOV-2017 18:12:15 * service_update * PROD1 * 0
19-NOV-2017 18:12:24 * service_update * PROD1 * 0
19-NOV-2017 18:13:42 * service_update * PROD1 * 0
```

Depending on the load information sent by each PMON process, the listener, by default, redirects the incoming connection request to the listener of the least-loaded instance on the least-loaded node. By default, Oracle uses the run queue length to distribute the load. To trace the load balancing, you can set the debug event 10237 in the init.ora file, and you can examine the PMON trace file in the database's diagnostic directory to confirm the load balancing:

```
event="10257 trace name context forever, level 16"
```

If you want to guarantee that the listener redirects the connection request based on session load, set PREFER_LEAST_LOADED_NODE_[listener name] to OFF in the listener.ora file. This is suited for applications that use connection pools.

When services are created using the DBMS_SERVICE.CREATE_SERVICE package, you also need to specify a connection load-balancing goal using the parameter CLB_GOAL. It accepts two values: CLB_GOAL_LONG and CLB_GOAL_SHORT.

LONG is the default connection load-balancing value, which means connections are held for longer periods, based on session count. You use the SHORT value to set up connection load balancing for those applications that use runtime load balancing.

Connection pooled applications for web-based applications connect at pool startup and keep the connection open to service short requests, hence needing the LONG setting. On the other hand, you need to specify the SHORT setting for one-to-one

https://t.me/learningnets

connections such as a SQL*Forms-based application where a connection is established for a session and is closed on completion. In reality, the SHORT setting is quite rare in modern applications, which are mostly connection pooled.

Here's an example that shows how to specify a method of load balancing when creating a database service:

```
EXECUTE DBMS_SERVICE.CREATE_SERVICE(service_name=>'rac.acme.com',-
network_name=>'rac.acme.com',-
goal=>dbms_service.goal_service_time,-
clb_goal=>dbms_service.clb_goal_short);
```

With CLB_GOAL_SHORT, connection load balancing uses the Load Balancing Advisory, when it is enabled (either GOAL_SERVICE_TIME or GOAL_THROUGHPUT). When CLB_GOAL_NONE (no Load Balancing Advisory) is used, connection load balancing uses an abridged advice based on CPU utilization.

With CLB_GOAL_LONG, connection load balancing balances the number of connections per instance using session count per service. This setting is recommended for applications with long connections. This setting can be used with Load Balancing Advisory when connection pools are used. As the names suggest, CLB_GOAL_LONG creates long-lived connections and CLB_GOAL_SHORT creates connections of short duration.

The parameter **goal=>dbms_service.goal_service_time** is concerned with connection pool load balancing or runtime load balancing, which also requires the configuration of database services and the configuration of Oracle Net Services connection load balancing. This feature is described in the section, where we discuss the Load Balancing Advisory.

When an application requests a connection, the connection pool is able to provide a connection to the instance that will best service the client request. This load-balancing algorithm is able to route work requests based on a defined policy. When a database service is created, a goal is specified along with it, which can be GOAL_THROUGHPUT, GOAL_SERVICE_TIME, or GOAL_NONE.

GOAL_THROUGHPUT means that the load balancing will be based on the rate that work is completed in the service plus the available bandwidth. If GOAL_SERVICE_TIME is specified, load balancing is based on the elapsed time for work done in the database service plus the available bandwidth to the service.

You can use the GOAL_THROUGHPUT setting for Concurrent Manager connections in Oracle EBS, and you can use the GOAL_SERVICE_TIME setting for interactive web-based apps connecting via connection pools. This option is better than using connection load-balancing (CLB).

If you use the connection pool load-balancing feature, the GOAL_* parameters are set using DBMS_SERVICE, and CLB_GOAL is ignored. Having PREFER_LEAST_LOADED_NODE_[listener_name] in listener.ora will disable/override new functionality.

In summary, without external hardware, Oracle provides two different types of load

https://t.me/learningnets

Technet24

balancing: namely, client-side and server-side load balancing. Here are some points to keep in mind:

- Client-side load balancing is affected by setting a parameter in tnsnames.ora. It simply distributes the sessions uniformly across all the available instances, regardless of whether a specific session carries too much of the load or a specific instance is already heavily loaded.
- Server-side load balancing is affected by the REMOTE_LISTENER parameter in init.ora. This takes into account the instance that is loaded, not just the count of the sessions in each.
- Both can be used concurrently. Therefore, it's possible that client-side load balancing directs a connection to a specific listener, but that listener may direct the connection to an instance on a different node.

Table 13-1 summarizes the overall details about the different types of load balancing, with their advantages and disadvantages.

	Hardware Load Balancing (Switch)	Software Load Balancing (Server)	Software Load Balancing (Client)
Implementation	Implemented at the network level. Depending on the address, the connection is routed to a specific node.	Implemented at server level. Based on the CPU usage of the server, the connection is accepted or forwarded to another node.	Implemented using client configuration files. Connections distributed to different nodes from the list of nodes.
Operation Parameters	At network level. Network address of the incoming connection.	Implemented at OS level; CPU usage statistics from OS.	Application level; from the application or module.
Cost	Additional cost involved, as it requires additional hardware.	No additional cost.	No additional cost.
Optimization for Logon Storms	Well optimized for logon storms (huge number of connection requests during very short time).	Not optimized for logon storms. All connections may land in one node.	Partial optimization for logon storms as requests are distributed to nodes on a round-robin basis.

TABLE 13-1. *Types of Load Balancing*

https://t.me/learningnets

Transparent Application Failover

The Transparent Application Failover (TAF) feature of the Oracle RAC database allows database connections to automatically relocate to another database instance in case the database instances serving the database connection fail or simply shut down. Active transactions roll back, but the new database connection, achieved using a different node, is identical to the original node. This is true regardless of how the connection fails. Oracle services implement this feature by specifying a TAF policy. Oracle TAF restarts the failed query on the new database instance, but the DML transactions (INSERT, UPDATE, and DELETE) aren't supported by Oracle TAF. Applications will have to roll back and reissue the transaction.

A TAF configuration specified on a service always overrides a TAF configuration on the client side. As you know, the Oracle RAC database in 12*c* can be either administrator or policy based. The Oracle TAF feature only applies to administrator-managed databases.

Users can choose from three TAF policies:

- **NONE** This policy is similar to not having any TAF policy.
- **BASIC** This policy restarts the failed query on the new database instance upon failover. The database reconnection is established at failover time.
- **PRECONNECT** This method is the same as the BASIC method, except that Oracle anticipates connection failure and creates a shadow connection to the other instance, which is always available to improve the failover time.

Here's how to change the TAF policy of the PROD_BATCH service of the PROD database to PRECONNECT:

```
$srvctl modify service -database PROD -service PROD_BATCH -tafpolicy PRECONNECT
```

TAF notifications are used by the server to trigger TAF callbacks on the client side. TAF is configured using either a client-side Transparent Network Substrate (TNS) connect string or server-side service attributes. However, if both methods are used to configure TAF, the server-side service attributes will supersede the client-side settings, because using server-side service attributes is the preferred way to set up TAF.

When a failure occurs, callback functions are initiated on the client side via Oracle Call Interface (OCI) callbacks. This will work with standard OCI connections as well as connection pool and session pool connections.

TAF Considerations

TAF works very well for the reporting and query-only applications, without requiring any additional code changes at the application layer. However, it does not fail over or protect the following elements:

https://t.me/learningnets

Technet24

- Active insert, update, and delete transactions
- PL/SQL server-side package variables
- Applications not using OCI 8

TAF supports only the failover of SELECT statements and client applications built using OCI. Custom application code may be required to trap errors or exceptions and perform alternative actions. Any application written using OCI 8 or later can take advantage of TAF. Many Oracle tools use OCI transparently; therefore, SQL*Plus, the Pro*C precompiler, the ODBC driver, and the thick Java Database Connectivity (JDBC) driver all work with TAF.

For an application to utilize the TAF feature, it must incorporate the appropriate OCI 8 (or later) usages and provide certain network configuration information. This requires the manual configuration of the tnsnames.ora and listener.ora files. It is possible to utilize TAF in a configuration where connected clients are performing update transactions; however, additional coding is required to achieve this.

Uncommitted changes at the time of failure are automatically rolled back by the database. To ensure that the application does not think that changes still exist, an error is returned to the application that can trigger the application to execute its own rollback statement. The application can then retry the transaction. In most cases, the application will receive another error during the rollback phase, indicating that the connection has been lost. This error must be trapped, but it can be ignored because the database will have already undone any transactional changes active during the failover.

Alternatively, Oracle Net Services can also preconnect users to a second node. If the primary node dies, the second connection will already be in place, bypassing network bottlenecks at reconnect time.

Configuring TAF

For a client to use TAF, it can either connect to the database using a connect descriptor that contains a FAILOVER_MODE portion or use server-side TAF defined for a service created using the SRVCTL utility. An example of a tnsnames.ora entry for a four-node Oracle RAC database follows:

```
RAC =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = NODE1-vip)(PORT = 1541))
    (ADDRESS = (PROTOCOL = TCP)(HOST = NODE2-vip)(PORT = 1541))
    (ADDRESS = (PROTOCOL = TCP)(HOST = NODE3-vip)(PORT = 1541))
    (ADDRESS = (PROTOCOL = TCP)(HOST = NODE4-vip)(PORT = 1541))
    (CONNECT_DATA =
      (SERVICE_NAME = RAC)
      (FAILOVER_MODE=(TYPE=SELECT)(METHOD=BASIC))
    ))
```

It is assumed that the SERVICE_NAME parameter in init.ora or SPFILE is

https://t.me/learningnets

accurately configured. Services can also be configured using the Server Control utility (SRVCTL), DBCA, or Enterprise Manager. Most importantly, the service entries have to be updated in the OCR repository. The tnsnames.ora files should be configured as shown earlier in the chapter. Initialization parameters such as LOCAL_LISTENER and REMOTE_LISTENER should also be accurately configured.

Note that we have not used FAILOVER=ON. This default parameter is set to ON if you use DESCRIPTION_LIST, DESCRIPTION, or ADDRESS_LIST in tnsnames.ora. FAILOVER=ON/TRUE/YES provides for connect-time failover, where Oracle Net Services tries the first address in the list and then fails over to the next address in the address list if the first one fails. Without this, Oracle would try only one address (if LOAD_BALANCE=ON, a random address is selected or else the first one in the list is used for a connection attempt) from the list and report an error on connection failure.

This type of service configuration for the service RAC tells Oracle Net Services to fail over to another instance providing the service RAC only when the instance to which we are currently connected fails.

TAF Configuration Options

The notable part of TAF configuration is shown here:

```
(FAILOVER_MODE=(TYPE=SELECT)(METHOD=BASIC))
```

The TYPE parameter can take any of the following three values:

- **SESSION** Set to fail over the session. If a user's connection is lost, a new session is automatically created for the user on the backup node using the service name specified in tnsnames.ora. This type of failover does not attempt to recover select operations.
- **SELECT** Set to enable users with open cursors to continue fetching on them after a failure. Using this mode can incur overhead on the client side in normal select operations.
- **NONE** This is the default. No failover functionality is used. This can also be explicitly specified to prevent failover from happening.

The METHOD parameter can accept either of the following two values and basically determines which method of failover takes place on instance failure from the primary node to the secondary:

- **BASIC** Set to establish connections at failover time. This option requires almost no work on the backup server until failover time.
- **PRECONNECT** Set to pre-established connections. This means that a separate database connection to the secondary (backup instance) is maintained all the time while there is a connection to the primary instance. This provides faster failover but requires that the backup instance be able to support all

https://t.me/learningnets

Technet24

connections from every supported instance. Note that the BACKUP= clause is mandatory when using the PRECONNECT method of failover.

Here is a sample configuration with connect-time failover, connection load balancing, and TAF for a four-node Oracle RAC database:

```
RAC =
(DESCRIPTION=
(LOAD_BALANCE=ON)
(FAILOVER=ON)
(ADDRESS=(PROTOCOL=tcp)(HOST=NODE1)(PORT=1541))
(ADDRESS=(PROTOCOL=tcp)(HOST=NODE2)(PORT=1541))
(ADDRESS=(PROTOCOL=tcp)(HOST=NODE3)(PORT=1541))
(ADDRESS=(PROTOCOL=tcp)(HOST=NODE4)(PORT=1541))
(CONNECT_DATA=
(SERVICE_NAME=RAC)
(FAILOVER_MODE=(TYPE=select)(METHOD=basic)(BACKUP=RAC2)))
```

The BACKUP=RAC2 clause is crucial here to avoid failover-related issues. Assume that we are connected to an instance and a failure occurs on that instance. When TAF is triggered, Oracle Net Services tries to fail over to another instance as per the TNS configuration. It will try connecting using the service name RAC and can attempt to reconnect to the failed node itself because Oracle RAC is a service being offered by *n* number of nodes. Although Oracle Net Services would fail over to another address in the list due to the FAILOVER=ON parameter (which provides connect-time failover), we can avoid unnecessary connect-time delays and errors. With the BACKUP=RAC2 clause, Oracle Net Services will ensure that once TAF is triggered, the connection request is only sent to the BACKUP=RAC2 node, and this is done immediately without any other address being tried.

It is recommended that TAF be configured using the BACKUP parameter in the CONNECT_DATA of the connect string; this makes it obvious which connect string will be used for failover. Without the BACKUP parameter, failover can still occur, but you may see some strange effects, such as disconnected sessions returning to the same node from which they were disconnected.

The TNS alias BACKUP is not actually verified to be a valid node of an Oracle RAC cluster that connects to the same database as the main connection. Therefore, it is possible to set BACKUP to a totally different machine and database with different schema objects. This could cause some confusion, such as a failed-over select returning “ORA-942: Table or view does not exist” if the table does not exist at the BACKUP connection. This is the most common failover mechanism used in Oracle Data Guard environments.

The TAF configuration mentioned in this section can also be configured on the server side on an Oracle Net Service using the SRVCTL utility.

Here is an example of creating a service called RACSRV with the TAF configuration explained in this section:

https://t.me/learningnets

```
SRVCTL ADD SERVICE -d RAC -s RACSRV -r RAC1,RAC2,RAC3,RAC4
SRVCTL START SERVICE -d RAC -s RACSRV
SRVCTL MODIFY SERVICE -d RAC -s RACSRV -q TRUE -P BASIC -e SELECT
```

Users can use other options of the SRVCTL utility to configure various other TAF configurations for a service, such as failover delays, failover retries, AQ HA notifications, and so on.

Validating the Failover

To check whether your TNS entries are working correctly and if TAF is indeed being triggered, use the following set of steps. (Ensure that this is a test system or that your database is running in ARCHIVELOG mode if data protection is important.)

1. As user *SYS*, log into the primary node where a user connection is made using the new TAF-enabled TNS service entry:

```
col sid format 999
col serial# format 9999999
col failover_type format a15
col failover_method format a15
col failed_over format a12

SQL> select instance_name from v$instance ;

INSTANCE_NAME
----------------
RAC1

SQL> select sid, serial#, failover_type, failover_method, failed_over
from V$SESSION where username = 'SCOTT';
       SID    SERIAL# FAILOVER_TYPE FAILOVER_METHOD FAILED_OVER
---------- ---------- ------------- --------------- -----------
        26          8 SELECT        BASIC           NO
```

This shows that the user SCOTT has logged into instance RAC1. You could also use the following method:

https://t.me/learningnets

```
SQL> select inst_id, instance_name from GV$SESSION;

INST_ID INSTANCE_NAME
------- -------------
      1 RAC1
      2 RAC2
      3 RAC3
      4 RAC4

SQL> select inst_id, sid, serial#, failover_type, failover_method,
failed_over from GV$SESSION where username = 'SCOTT';

INST_ID       SID    SERIAL# FAILOVER_TYPE FAILOVER_METHOD FAILED_OVER
------- --------- ---------- ------------- --------------- -----------
      1        26          8 SELECT        BASIC           NO
```

2. As user SCOTT, run a long query that would last for a period of 45 seconds or more. Here's a commonly used query:

```
SQL> select count(*) from
(select * from dba_source
union
select *  from dba_source
union
select *  from dba_source
union
select *   from dba_source
union
select *   from dba_source)
```

At this juncture, you can issue a SHUTDOWN ABORT on the instance where SCOTT is running the query—which is instance RAC1 in this case. SCOTT's user screen (SQL*Plus) should hang for a few seconds or more, depending on various factors such as cluster communication and node failure detection, network delays, and timeout parameters.

3. After the failover is completed internally by Oracle, the screen should return to normal and the query should continue to process and display results, as follows:

```
COUNT(*)
----------
128201
```

(Note that the resultant value 128201 is an example; the exact value you get depends on the number of objects in your system.)

4. As user SYS, run the query to find out if SCOTT's session was indeed failed

https://t.me/learningnets

over to the other node. The advantage of using GV$ views is that you don't need to log into each instance and query them individually to find out if SCOTT has been transferred there.

```
SQL> select inst_id, sid, serial#, failover_type, failover_method,
failed_over from GV$SESSION where username = 'SCOTT';

INST_ID       SID    SERIAL# FAILOVER_TYPE FAILOVER_METHOD FAILED_OVER
------- --------- ---------- ------------- --------------- -----------
      3        16         10 SELECT        BASIC           YES
```

The YES for FAILED_OVER specifies that SCOTT's session did fail over to instance RAC3 when RAC1 failed due to a shutdown abort.

Troubleshooting TAF Issues

Oracle uses a resource type of ora.service.type for each Oracle Net Service configured in the cluster database. Oracle Clusterware also manages the virtual IPs (VIPs) on which the listeners listen. If a node goes down, the service and VIPs that are managed by Cluster Ready Services (CRS) will fail over to a surviving node. By having the listener listen on the VIPs, we can avoid waiting for a TCP timeout if a node goes down. Connections will be routed to surviving nodes and will be able to resume their work when the failover is completed. When you're troubleshooting TAF issues, it is important to look at the CRS layer, the network layer, and the database layer to identify where a problem may reside.

The ocssd.log and crsd.log files located under $GRID_HOME/log/<hostname>/cssd and $GRID_HOME/log/<hostname>/crsd provide detailed output about the CRS reconfiguration and node membership details.

Workload Balancing

When it comes to workload balancing, connection placement uses connection load balancing, and work request placement uses runtime load balancing. The benefit of managing work across multiple servers as a single unit is that it increases the utilization of the available resources. Work requests can be distributed across instances offering a service according to the current service performance. Balancing work requests occurs at two different times—at *connect time* and at*runtime*.

Connection Pooling and the Load Balancing Advisory

Historically, load balancing within a pool of connections was based on the initial load information provided by Oracle to the middle tier. Therefore, the load information was outdated in no time because the workload on the system was bound to increase as more users connected. Hence, load balancing involved distributing an even number of connections across all the Oracle RAC instances. When an application asked for a connection, it was given a random connection (to an instance) from a set of inactive connections in the pool. There was no way of knowing how loaded the instance was

https://t.me/learningnets

Technet24

before connecting to it.

Oracle's Load Balancing Advisory provides advice regarding how to apportion new work to the best instances, so as to maintain an optimal quality of work. When you configure this feature, an Oracle RAC database will periodically send load information and the current service level to the connection pool. The information is collated from all active instances and sent as a FAN to the connection pool manager in the middle tier. Because there is a periodic update about the service information and the listener is able to route connections based on intelligence, this is also referred to as *runtime load balancing*.

Oracle RAC's Load Balancing Advisory monitors the work for each service in an instance and provides a percentage value of how much of the total workload should be sent to this instance as well as a service quality flag. The feedback is provided as an entry in the AWR, and a FAN event is published.

Well-designed and well-built applications generally tend to make good use of connection pools built into the middle tier of an application. For example, applications using Common Object Request Broker Architecture (CORBA), BEA WebLogic, and other Java-based applications can take advantage of the connection-pooling mechanism in the middle layer.

In simple terms, the connection pool is a named group of identical connections to the database. Applications borrow connections from the connection pool and use the connections to perform database calls. When that work is complete, the connection channel is released back to the connection pool to be reused.

Connection caching or pooling is used to preserve connections and minimize the overhead of creating new connections at the OS level. By maintaining the connection(s) for a long time, the OS does not need to spend time allocating process space, CPU time, and so on, every time a connection is made. Even disconnection from the database requires quite a bit of OS coordination, such as database lock cleanup and other housekeeping activities that can be avoided by using connection pooling.

Standard architectures that can use the Load Balancing Advisory include connection load balancing, TP monitors, application servers, connection concentrators, hardware and software load balancers, job schedulers, batch schedulers, and message queuing systems. All of these applications can allocate work.

Connection Load Balancing

Good applications connect once to the database server and stay connected. This is a best practice for all applications—connection pools, client/server, and server side. Because connections are relatively static, the method for balancing connections across a service should not depend on metrics that vary widely during the lifetime of the connection.

Three metrics are available for the listeners to use when selecting the best instance:

- **Session count by instance** For symmetric services and the same capacity

https://t.me/learningnets

nodes, the absolute session count evenly distributes the sessions.

- **Run queue length of the node** For asymmetric services or different capacity nodes, the run queue length places more sessions on the node with the least load at the time of connection.
- **Goodness of service** For all services and any capacity nodes, the goodness of the service is a ranking of the quality of service experienced at an instance. The ranking compares the service time to the service's threshold value. It also considers states such as whether access to an instance is restricted. Examples of goodness ratings are excellent, average, violating, and restricted. To avoid a listener routing all connections to the excellent instance between updates to the goodness values, each listener adjusts its local ratings by a delta as connections are distributed. The delta value used is the resource consumed by each connection using a service.

In the section dealing with connection load balancing, we explained how you can specify a connection load-balancing goal with the parameter CLB_GOAL when creating a database service. When an application requests a connection, the connection pool is able to provide a connection to the instance that will best service the client request. This load-balancing algorithm is able to route work requests based on a defined policy. When database services are created, a goal is specified along with them, which can be GOAL_THROUGHPUT, GOAL_SERVICE_TIME, or GOAL_NONE.

GOAL_THROUGHPUT means that the load balancing will be based on the rate at which work is completed in the service plus the available bandwidth. If GOAL_SERVICE_TIME is specified, load balancing is based on the elapsed time for work done in the database service plus the bandwidth available to the service.

If the connection pool load-balancing feature is used, the GOAL_* parameters are set using DBMS_SERVICE, and CLB_GOAL is ignored. Having PREFER_LEAST_LOADED_NODE_[listener_name] in listener.ora will disable/override new functionality.

The following two examples show how to configure the load balancing advisory by defining server-level goals for a service. The first example shows how to define SERVICE_TIME as the service goal. The second example shows how to specify the THROUGHPUT service goal.

```
$ srvctl modify service -database prod -service finance -rlbgoal SERVICE_TIME
-clbgoal SHORT
$ srvctl modify service -database prod -service finance -rlbgoal THROUGHPUT
-clbgoal LONG
```

You disable the Load Balancing Advisory for a service by setting the **-rlbgoal** parameter's value to NONE. You can check the current settings of service goals by querying the DBA_SERVICES, V$SERVICES, and V$ACTIVE_SERVICES views.

https://t.me/learningnets

Technet24

Runtime Load Balancing

Runtime load balancing (RLB) is used when selecting connections from connection pools. For connection pools that act on services at one instance only, the first available connection in the pool is adequate. When connection pools act on services that span multiple instances, for runtime load balancing, a metric is needed that is responsive to the current state of each service at each instance.

The AWR measures the response time, CPU consumption, and service adequacy for each service and is integrated with the Load Balancing Advisory. The views V$SERVICE_METRICS and V$SERVICE_METRICS_HISTORY contain the service time for every service and are updated every 60 seconds with one hour of history. These views are available to applications to use for their own runtime load balancing. For example, a middle-tier application using connection pools can use the service metrics when routing the runtime requests to instances offering a service.

The Database Resource Manager maps services (in place of users) to consumer groups, thereby automatically managing the priority of one service relative to others. The Oracle RAC high availability features keep the services available, even when some or all components are unavailable. The Oracle Data Guard Broker, in conjunction with Oracle RAC, is capable of migrating the primary service across various Oracle Data Guard sites for disaster tolerance.

Using Service Time for Load Balancing

Using the service time for load balancing is a major improvement over earlier approaches that use round-robin or run queue length. The run queue length metric does not consider sessions that are blocked in wait—for example, interconnect wait, I/O wait, and application wait—and unable to execute. The run queue length and session count metrics also do not consider session priority.

Using service time for load balancing recognizes machine power differences, sessions that are blocked in wait, failures that block processing, and competing services of different importance. Using service time prevents sending work to nodes that are overworked, hung, or failed.

The Load Balancing Advisory takes into consideration different-sized nodes, and in cases where a node is heavily loaded, the connection pool is capable of removing connections for that instance from the pool (as they become inactive or hung). The connection pool is then able to create new connections to another instance, providing better service levels.

The response times DB time and CPU time are available for each service at each instance in V$SERVICE_METRICS_HISTORY. Just as the listener uses this data to deal connections across the service, the connection pool algorithm can use the data when selecting connections from the connection pool. Using this approach distributes the work across instances that are serving the service well and avoids sending work to slow, hung, failed, and restricted instances.

Assuming that an ODP.NET-based application uses connection pooling with the grid runtime load-balancing attribute enabled, an ODP.NET connection pool registers

https://t.me/learningnets

for an RLB notification with a particular service in an Oracle RAC or grid environment. If the environment is properly configured, the service periodically sends notifications to ODP.NET with information on the responsiveness of each service member.

For optimal performance, ODP.NET dispenses connections to service members based on information received about their responsiveness. This attribute can be used only against Oracle instances within an Oracle RAC database. Its value is also ignored if pooling=False.

The grid runtime load-balancing attribute is enabled by setting the grid runtime load-balancing connection string attribute to True. The default is False. The following connection string enables the grid runtime load-balancing attribute:

```
"user id=apps;password=apps;data source=erp;grid runtime load" + "balancing=true;"
```

Measuring Workloads by Service

If configured, the Automatic Workload Repository (AWR) maintains performance statistics, including response times, resource consumption, and wait events, for all services and for all the work that is being done in the system. Selected metrics, statistics, wait events, and wait classes, plus SQL level traces, are maintained for the service, optionally augmented by module and action names. The statistics aggregation and tracing by service are global in scope and persist across instance restarts and service relocation for both Oracle RAC and single-instance Oracle.

Using services offers you a powerful way to monitor application performance, especially since most systems use anonymous sessions, making it impossible to track application performance with traditional session-based performance analysis. You should create modules and actions to really get to the bottom of most performance issues. You can associate actions in the modules you create with key business transactions, thus letting you examine the performance of those transactions in great detail.

Use the DBMS_MONITOR package to enable module and action monitoring. In the following example, we show how to create a module and actions to correlate a specific service named the ERP service. Once you do this, you can monitor the specific action we've created, named EXCEPTIONS PAY, in the PAYROLL module.

```
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE (SERVICE_NAME => 'ERP',
MODULE_NAME=> 'PAYROLL', ACTION_NAME => 'EXCEPTIONS PAY');
```

You can monitor all actions in the PAYROLL module with the following command:

```
EXECUTE DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE (SERVICE_NAME => 'ERP',
MODULE_NAME=> 'PAYROLL', ACTION_NAME => NULL);
```

Once you've created appropriate modules and actions, you may want to check that

https://t.me/learningnets

Technet24

monitoring is enabled for the service, module, and actions with the following command:

```
SQL> SELECT * FROM DBA_ENABLED_AGGREGATIONS;
AGGREGATION            SERVICE                MODULE    ACTION
------------           --------------------   --------  ---------
SERVICE_MODULE_ACTION  erp                    payroll   exceptions pay
SERVICE_MODULE         erp                    payroll
SERVICE_MODULE         hot_batch              posting
```

You can view service, module and action names in the V$SESSION, V$ACTIVE_SESSION_HISTORY, and V$SQL views. You can track the performance statistics using the V$SERVICE_STATS, V$SERVICE_EVENT, V$SERVICE_WAIT_CLASS, V$SERVICEMETRIC, and V$SERVICEMETRIC_HISTORY views. When you enable statistics collection for an important transaction, you can see the call speed for each service, module, and action name at each database instance using the V$SERV_MOD_ACT_STATS view. You can view the statistics for the performance data you're tracing for the modules and actions in each service in the V$SERV_MOD_ACT_STATS view, as shown here:

```
SELECT
    service_name
  , TO_CHAR(begin_time, 'HH:MI:SS') begin_time
  , TO_CHAR(end_time, 'HH:MI:SS') end_time
  , instance_name
  , elapsedpercall  service_time
  ,  callspersec  throughput
FROM
    gv$instance i
  , gv$active_services s
  , gv$servicemetric m
WHERE s.inst_id = m.inst_id
  AND s.name_hash = m.service_name_hash
  AND i.inst_id = m.inst_id
  AND m.group_id = 10
ORDER BY
   service_name
 , i.inst_id
 , begin_time ;
```

This query will fetch you the service equality statistics over a 30-second interval, letting you determine if you need to move work and balance services across your RAC cluster.

Using Service-Level Thresholds

https://t.me/learningnets

AWR lets you specify the response time for calls (ELAPSED_TIME_PER_CALL) and the CPU time for calls (CPU_TIME_PER_CALL) as performance thresholds for each service. You must set the two thresholds on all nodes in a RAC cluster. When you set the two performance thresholds described earlier, AWR will monitor the elapsed time and CPU time and will send alerts when performance metrics fall outside the thresholds. You can either schedule an Oracle Enterprise Manager (OEM) job to respond to the alerts or respond programmatically to the alerts. You can thus maintain your service availability in response to changes in workload demand, by changing a job's priority, stopping a process, or starting/stopping/relocating a service.

The following example shows how to set service thresholds using the DBMS_SERVER_ALERT package. The example sets a threshold based on ELAPSED_TIME_PER_CALL.

```
EXECUTE DBMS_SERVER_ALERT.SET_THRESHOLD (
METRICS_ID => DBMS_SERVER_ALERT.ELAPSED_TIME_PER_CALL
, warning_operator => DBMS_SERVER_ALERT.OPERATOR_GE
, warning_value => '500000'
, critical_operator => DBMS_SERVER_ALERT.OPERATOR_GE
, critical_value => '750000'
, observation_period => 30
, consecutive_occurrences => 5
, instance_name => NULL
, object_type => DBMS_SERVER_ALERT.OBJECT_TYPE_SERVICE
, object_name => 'payroll');
```

Service-level thresholds are useful for comparing the achieved service level with the minimum configured/accepted service levels. Setting appropriate service-level thresholds allows you to predict a system capable of achieving the required service levels. You can use the DBMS_SERVER_ALERT package to set service-level thresholds. By default, statistics are collected for the work attributed to every service. Each service can be further qualified by module name and action name to identify the important transactions within the service. This eliminates the hard work associated with measuring the time for all calls for the service. The service, module, and action name provide a user-explicable unit for measuring elapsed time per call and for setting and resource consumption thresholds.

The module and action are visible in V$SESSION. This naming format is a way of tying a portion of the application code to the database work done on its behalf. The module name is set to a user-recognizable name for the program that is currently executing (script or form). The action name is set to a specific action that a user is performing within a module (such as reading mail or entering new customer data). Setting these tags using OCI in 11*g* does not result in additional round trips to the database.

We can use the DBMS_MONITOR package to control the gathering of statistics that quantify the performance of services, modules, and actions. DBMS_MONITOR is also used for tracing services, modules, and actions.

https://t.me/learningnets

Technet24

NOTE

We can't overemphasize the need to use DBMS_MONITOR and DBMS_APPLICATION_INFO in all your applications.

Oracle RAC High Availability Features

For HA clients connected to an Oracle RAC database, HA event notification can provide a best-effort programmatic signal to the client in the event of a failure. Client applications can register a callback on the environment handle to signal interest in this information. When a significant failure event occurs (which applies to a connection made by this client), the callback is invoked, with information concerning the event (the event payload) and a list of connections (server handles) that were disconnected as a result of the failure.

The HA event notification mechanism improves the response time of the application in the presence of failure. With HA event notification, standalone, connection pool, and session pool connections are automatically broken and cleaned up by OCI, and the application callback is invoked within seconds of the failure event. If any of these server handles are TAF enabled, failover will also automatically be engaged by OCI.

Applications must connect to an Oracle RAC instance to enable HA event notification. Furthermore, these applications must initialize the OCI environment in OCI_EVENTS mode. Then these applications can register a callback that is invoked whenever an HA event occurs.

High Availability, Notifications, and FAN

One of the main principles of a highly available application is that it be able to receive fast notification when the status of a critical resource or component changes. This resource or component could be within a cluster or outside, but it is critical to the functioning of the application. The Fast Application Notification (FAN) feature provides a framework to achieve higher levels of availability and reliability for a cluster database. FAN is always aware of the current configuration of the server pools in the cluster, so Oracle always tries to make an application connection to the database instances that are able to respond. Oracle's high availability framework continually monitors the database and services and sends out appropriate event notifications using FAN. Oracle posts a FAN event on any state change in the cluster.

Using notifications, applications are able to perform some form of compensatory action or handle this event (by executing a program or script) so that application availability isn't compromised or a certain set of predetermined actions is taken to minimize the impact of this status change known as an event.

The timely execution of these handler programs minimizes the impact of cluster

https://t.me/learningnets

component failures by avoiding connection timeouts, application timeouts, and reacting to cluster resource reconfiguration in both planned and unplanned scenarios.

Event-Based Notification

The concept of event-based notification is not new. Customers who have deployed Oracle Enterprise Manager or monitored enterprise IT resources using third-party management console products that rely on low-overhead protocols such as SNMP are already familiar with the usage and benefits of receiving component up, down, or threshold violation alerts from the database layer. Production sites may also implement their own polling mechanisms by capturing events and delivering proper notifications to higher application layers.

Traditionally, client or mid-tier applications connected to the database have relied on connection timeouts, out-of-band polling mechanisms, or other custom solutions to realize that a system component has failed. This approach has huge implications in application availability, because downtimes are extended and more noticeable.

The other area where the traditional approach falls short is in the management of complex database cluster environments, because of the exponential growth in the number of server-side components, both horizontally (across all cluster nodes) and vertically (across cluster node components such as listeners, instances, and application services). Furthermore, with the introduction of grid computing to the enterprise, such levels of component complexity must be interconnected, monitored, and managed in a proactive and automated way.

When the state of an Oracle database service changes (up, down, restart failure), the new status is posted or relayed to interested subscribers through FAN events. Applications can use these events to achieve very fast failure detection and re-create, reestablish, and balance a connection pool following a failure, as well as to use failback detection to restore normal services.

A FAN event is issued when a service becomes available, when it becomes unavailable, or when it’s no longer being managed by CRS. The notification events occur for status changes in resources under CRS for services, service members, databases, and instances. Databases and instances are included to accommodate applications that depend on these resources directly. The notification event is used to eliminate the HA problems of sessions waiting to time out when blocked, and/or are wasting time hanging around after their service has terminated. This is done by providing a notification in the form of an out-of-band event to the client side. In this way, sessions are notified quickly when the application changes state.

Because important HA events are pushed as soon as they are detected, this method results in a more efficient use of existing computing resources and in better integration with enterprise applications, including mid-tier connection managers or IT management consoles such as trouble ticket loggers and e-mail/paging servers.

FAN is a distributed system that is enabled on each participating node. This makes it reliable and fault tolerant because the failure of one component is detected by another one. Therefore, event notification can be detected and pushed by any of the

https://t.me/learningnets

Technet24

participating nodes.

FAN events are tightly integrated with Oracle Data Guard Broker, Oracle JDBC implicit connection cache, and Oracle Enterprise Manager. JDBC applications managing connection pools do not need custom code development. They are automatically integrated with the Oracle Notification Service (ONS) if implicit connection cache and fast connection failover are enabled. Figure 13-3 shows the FAN architecture.

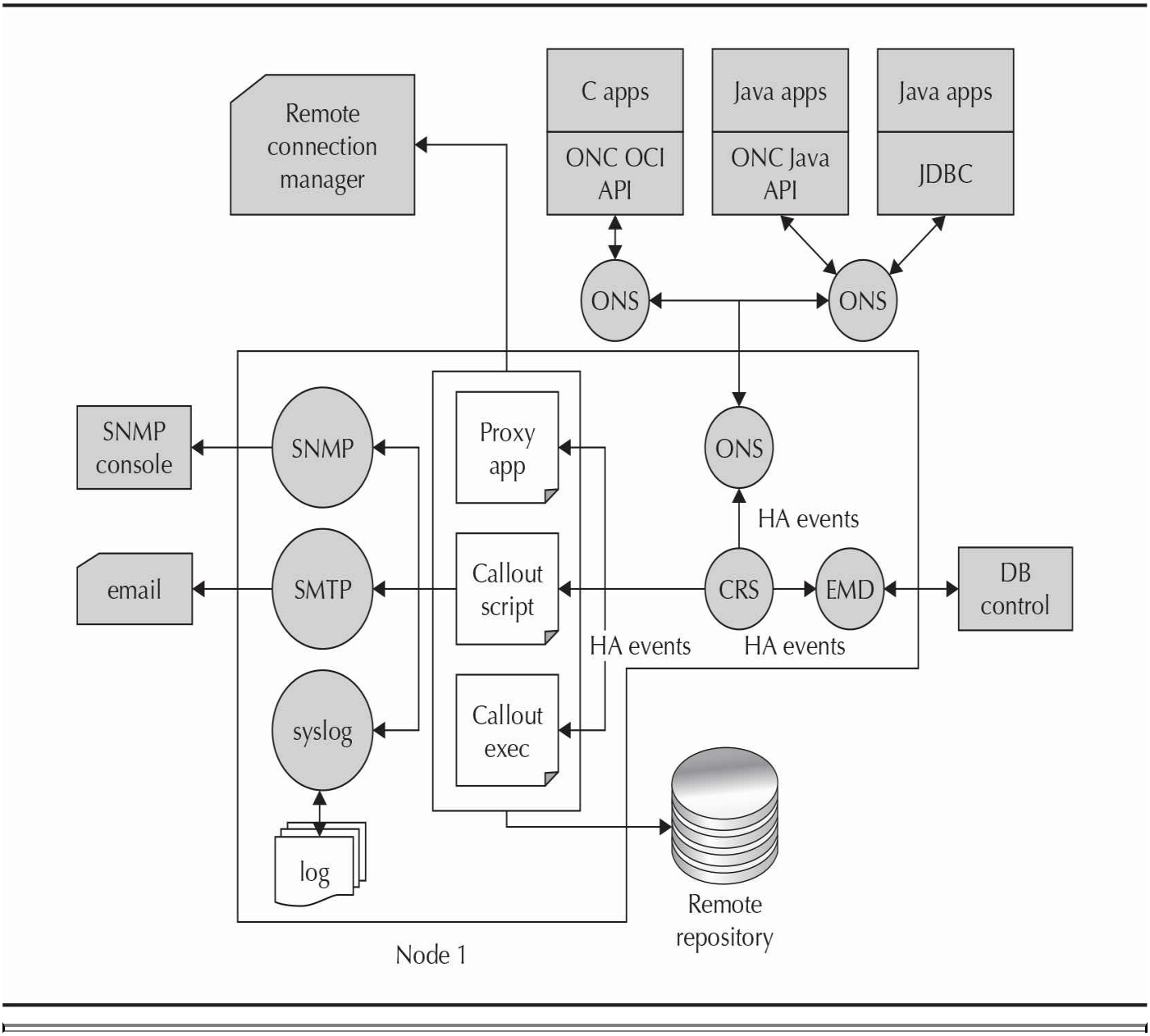


FIGURE 13-3. *FAN architecture ONS*

Application Failure Issues

When a service goes down in a cluster, applications end up wasting time trying one or some of these activities:

- Waiting for a TCP/IP timeout when a node fails without closing sockets and

https://t.me/learningnets

opening additional sockets for every subsequent connection while that particular IP is down.

- Attempting to connect when services are down and not connecting when services resume.
- Attempting to execute work on slow, hung, or dead nodes.

The existing mechanisms in Oracle do not allow for these types of problems to be reported, so accounting for downtime or performance issues caused by them is not possible—or at best is an estimation.

When a node fails without closing the network sockets, all sessions that are blocked in an I/O use the transport layer configuration (tcp_keepalive) for a predefined period before timing out. The problem could be worse for sessions that are processing something at the client end and are not aware that the server is down. The status is known only after the last result set is to be fetched or processed. This causes severe delays or timeouts.

For cluster configuration changes, the RAC HA framework publishes a FAN event immediately when a state change occurs in the cluster. Instead of waiting for the application to poll the database and detect a problem, applications receive FAN events and can react immediately to them.

On receipt of a FAN event, applications can abort sessions in communication with the failed instance or node, relocate them to another node, notify sessions waiting to resume operations, and reorganize incoming work on the fly as additional resources become available or lost. For example, if a service is down, a DOWN event is published. For DOWN events, the disruption to the application can be minimized because sessions to the failed instance or node can be terminated. Incomplete transactions are terminated and the application user is immediately notified. Application users who request connections are directed to available instances only.

You can also use server-side callouts to do the following:

- Page DBAs or open support tickets when resources fail to start.
- Automatically start dependent external applications that need to be co-located with a service.
- Change resource plans or shut down services when the number of available instances decreases—for example, if nodes fail.
- Automate the failback of a service to PREFERRED instances if needed.

For UP events, when services and instances are started, new connections can be created so that the application can immediately take advantage of the extra resources.

FAN also publishes Load Balancing Advisory events. Applications can take advantage of the Load Balancing Advisory FAN events to direct work requests to the instance in the cluster that is currently providing the best service quality.

Using FAN

https://t.me/learningnets

Technet24

FAN can be used in three ways. First, applications can take advantage of FAN without programmatic changes by utilizing an integrated Oracle client. The integrated clients for FAN events include Oracle Database JDBC, Oracle WebLogic Server Active GridLink for Oracle RAC, Oracle Database Oracle Data Provider for .NET (ODP.NET), OCI Connection Pooling, and Connection Manager.

Second, applications can use the ONS API to subscribe to FAN events and execute event-handling actions upon receipt. Third, applications can implement FAN server-side callouts on the database tier.

Applications can also enable the Oracle JDBC implicit connection cache and let FAN events be handled by the Oracle JDBC libraries directly. Oracle implicit connection cache is deprecated and has been replaced with Universal Connection Pool. Using this method, you no longer need to write any custom Java or C code to handle FAN events, nor do you need to invoke the ONS API directly. They are automatically integrated with ONS if implicit connection cache and fast connection failover are enabled. Consider this as an out-of-the-box integration with the ONS.

A typical Java application using FAN would require the application developer to download the current FAN .jar (known as simplefan.jar) from Oracle. The developer would then add this .jar file to the CLASSPATH of the application. There is no need to start any ONS daemon to use Oracle Universal Connection Pool on the client side because having this .jar file in the CLASSPATH is sufficient. The application developer would need to instantiate the FanManager class and use methods provided by this class to configure and use FAN.

Refer to the “Oracle JDBC Developer Guide” for detailed steps for implementing FAN.

FAN Events and Notification

FAN events consist of a header and payload information delivered as a set of name/value pairs accurately describing the name, type, and nature of the cluster event. Based on this payload, the event recipient can take concrete management, notification, or synchronization steps. These could include actions such as shutting down the application connection manager, rerouting existing database connection requests, refreshing stale connection references, logging trouble tickets, and sending a page to the database administrator. The objective is to deliver FAN events so that they precede any other polling intervals or connection timeouts.

Notification is the first step in the processes for application recovery, service recovery, offline diagnosis, and fault repair. Notification occurs in several forms:

- **In-band notification** Using strong and weak dependencies in CRS and special CRS events for check and fail actions, dependent resources receive an in-band notification to start and stop. These events occur as a result of starting or stopping interdependent resources, and as a result of dependent resources failing and restarting. These notifications are considered “in-band” because they are posted and processed synchronously as part of CRS managing the system.

https://t.me/learningnets

- **Out-of-band notification** FAN provides callouts, events, and paging; e-mail from the enterprise console, status changes, fault notifications, and fault escalation are forwarded to invoke repair and to interrupt applications to respond to the service change. These FAN events are considered "out-of-band" because they are issued asynchronously through gateway processes out of RAC/HA to listeners, enterprise console, and callouts.
- **Error and event logs** When a fault occurs in any layer, details of the error are reported to persistent event logs. This is the case for all error conditions, including those that are automatically recovered by the CRS system. For expediency, all event logs should have a consistent format and should be logged to a consistent location. Data collection is essential to ensure that a root cause for the condition can be found early. Once the problem is identified, a resolution can be produced.

Three categories of events are currently supported in FAN:

- **Service events** These events include both application services and database services.
- **Node events** These events include cluster membership states and native join/leave operations.
- **Load-balancing events** These events are sent from the Oracle RAC Load Balancing Advisory.

In addition to application services, Oracle RAC standardizes the generation, presentation, and delivery of events pertaining to managed cluster resources, which include these event types:

Event Type	Description
SERVICE	Primary application service event
SRV_PRECONNECT	Shadow application service event (mid-tiers and TAF using primary and secondary instances)
SERVICEMEMBER	Application service on a specific instance event
DATABASE	Oracle database event
INSTANCE	Oracle instance event
SERVICEMETRICS	Database service metrics
NODE	Oracle cluster node event

This table describes the event status for each of these managed cluster resources:

https://t.me/learningnets

Technet24

Event Status	Description
status=up	Managed resource comes up.
status=down	Managed resource goes down.
status=preconn_up	Shadow application service comes up.
status=preconn_down	Shadow application service goes down.
status=nodedown	Managed node goes down.
status=not_restarting	Managed resource cannot fail over to a remote node.
status=unknown	Unrecognized status.

The event status for each managed resource is associated with an event reason. The reason further describes what triggered the event, as shown in the following table:

Event Reason	Activity Type	Event Trigger
reason=user	Planned	User-initiated commands such as **srvctl** and **sqlplus**.
reason=failure	Unplanned	Managed resource polling checks detected a failure.
reason=dependency	Unplanned	Dependency of another managed resource that triggered a failure condition.
Event Reason	**Activity Type**	**Event Trigger**
reason=autostart	CRS boot	Initial cluster boot (managed resource has profile attribute AUTO_START=1 and was offline before the last CRS shutdown).
reason=boot	CRS boot	Initial cluster boot (managed resource was running before the last CRS shutdown).

Other event payload fields further describe the unique cluster resource whose status is being monitored and published. They include the following:

https://t.me/learningnets

Event Resource Identifier	Description
VERSION=<n.n>	Event payload version (currently VERSION=1.0)
timestamp=<eventDate> <eventTime>	Server-side date and time when the event was detected
service=<serviceName.dbDomainName>	Name of the (primary or shadow) application service (excluded from NODE events)
database=<dbName>	Name of the Oracle RAC database (excluded from NODE events)
instance=<sid>	Name of the Oracle RAC instance (excluded from SERVICE, DATABASE, and NODE events)
host=<hostName>	Name of the cluster node as returned by the clusterware (excluded from SERVICE and DATABASE events)
card=<n>	Service membership cardinality (excluded from all events except SERVICE status=up)

The combination of all these attributes and types results in a FAN event with the following payload structure:

```
<Event_Type> VERSION=<n.n>
service=<serviceName.dbDomainName>
[database=<dbName> [instance=<sid>]] [host=<hostname>]
status=<Event_Status> reason=<Event_Reason> [card=<n>]
timestamp=<eventDate> <eventTime>
```

Here's an example:

```
Service events:
SERVICE VERSION=1.0 service=homer.simpsons.com database=BART status=up
reason=user card=4

SERVICEMEMBER VERSION=1.0 service=barney.simpsons.com database=BEER
instance=DRAUGHT host=moesplace status=down reason=user

DATABASE VERSION=1.0 service=skywalker.starwars.com database=STARS
host=galaxy status=up reason=boot timestamp=08-Mar-2017 15:31:00

INSTANCE VERSION=1.0 service=robinhood.thieves.com database=RANSOM
instance=SCOTLAND host=locksley status=down reason=failure
ASM VERSION=1.0 instance=ASM1 host=apps status=down reason=failure
Node events:

NODE VERSION=1.0 host=sales status=nodedown
```

https://t.me/learningnets

Technet24

Configuring Server-Side Callouts

Server-side callouts are server-side scripts or executables that Oracle executes upon the posting of a FAN event. These executables are stored in the $GRID_HOME/racg/usrco directory on each cluster node. Users can store any number of executables in this directory, but Oracle does not guarantee that it will execute them in order. However, Oracle does guarantee that all the executables in the $GRID_HOME/racg/usrco directory will be executed. It is not a good practice to use multiple callouts; instead, use a single callout script that parses the FAN events and calls the respective executable store at another location.

Here's a sample parsing callout:

```
#!/bin/bash
FAN_LOGFILE=/tmp/FAN.log
EVENT_TYPE=$1
shift
for ARGS in $@ ; do
        PAYLOAD_PROPERTY='echo $ARGS | awk -F"=" '{print $1}'`
        VALUE='echo $ARGS | awk -F"=" '{print $2}'`
        case $PAYLOAD_PROPERTY in
                VERSION|version)        EVENT_VERSION=$VALUE ;;
                SERVICE|service)        EVENT_SERVICE=$VALUE ;;
                DATABASE|database)      EVENT_DATABASE=$VALUE ;;
                INSTANCE|instance)      EVENT_INSTANCE=$VALUE ;;
                HOST|host)              EVENT_HOST=$VALUE ;;
                STATUS|status)          EVENT_STATUS=$VALUE ;;
                REASON|reason)          EVENT_REASON=$VALUE ;;
                CARD|card)              EVENT_CARD=$VALUE ;;
                TIMESTAMP|timestamp)    EVENT_TIMESTAMP=$VALUE ;;
                ??:??:??)               EVENT_LOGTIME=$PAYLOAD_PROPERTY ;;
        esac
done

# Stat Event Handler Section
# Write your event specific actions here.
# End Event Handler Section
```

You have to create an executable script with the preceding contents and place this script under the $GRID_HOME/racg/userco directory on each cluster node. Whenever Oracle posts a FAN event, this executable script will be called. This script will parse the event payload properties and call event-specific tasks from the event handler section.

Using Transaction Guard for Efficient Client

https://t.me/learningnets

Failover

When a database outage occurs, communications between the clients and server are disrupted and clients receive error messages regarding this breakdown in communications. However, these messages don't tell the clients whether the operations in progress were committed, failed part way through, or are still running on the server, despite the disruption in the client connections to the server. Users might, as a result, submit duplicate modifications to the database or do other things that could cause some type of logical corruption of the data.

Oracle Database 12*c*'s new Transaction Guard feature protects against this type of communication breakdown and its attendant problems caused by both planned and unplanned outages as well as the possibility of duplicate committed transactions. Transaction Guard is a reliable protocol and API that returns the outcome of the last transaction.

When using Transaction Guard, applications make use of a logical transaction ID (LTXID) to confirm the status of the last open transaction in progress during an outage. You must configure database services in order to take advantage of the Transaction Guard feature. Here are the steps you must follow to do this:

1. Grant appropriate permissions to the users so that they can call the GET_LTXID_OUTCOME procedure in the DBMS_APP_CONT package:
   ```
   SQL> grant execute on DBMS_APP_COUNT to testuser;
   ```

2. Configure the services to use Transaction Guard with the following SRVCTL command, which applies to the traditional administrator-managed service:
   ```
   $ srvctl add service -db prod -service finance -preferred prod1, prod2
     -available prod3,prod4 -commit_outcome TRUE  -retention  86400 -
   notification TRUE
   ```

If you're configuring a policy-managed service, use the following command, specifying the **-serverpool** parameter instead of the **-preferred** and **-available** parameters:

```
$ srvctl add service -db prod -service finance -serverpool pool_1
  -commit_outcome TRUE   -retention   86400 -notification TRUE
```

The **commit_outcome** parameter enables the Transaction Guard feature. The **retention** parameter specifies how long (in seconds) the database should maintain the transaction history and the commit outcome. Setting the optional **notification** parameter to TRUE will enable FAN events. The examples here show how to configure the use of the Transaction Guard when creating a service. You can also configure the Transaction Guard for existing services in a similar fashion by using the **srvctl modify service** command. The following example shows how to do this:

```
$ srvctl modify service -db prod -service finance -serverpool pool_1
  -commit_outcome TRUE -retention 86400  -notification TRUE
```

https://t.me/learningnets

Technet24

By setting the **commit_outcome** parameter to TRUE, you ensure that the outcome of the COMMIT is durable, meaning that you can access the outcome status following a COMMIT operation when an outage occurs. Applications will then use the status of the COMMIT outcome to enforce the status of the very last transaction that occurred prior to the outage.

Application Continuity for Masking Database Outages

Oracle Database 12*c*'s new Application Continuity feature helps mask the impact of a database outage by rebuilding affected database sessions and automatically resubmitting the pending work. Database outages often cause the loss of in-flight work, leaving users and applications in doubt about the correct state of transactions. This potentially could lead to duplicate submissions of work, the rebooting of middleware, and other related issues.

Note that you can't use the Application Continuity feature to recover from the effects of nonrecoverable errors such as submitting invalid data values in a transaction. Only failures caused by recoverable errors are eligible to use the Application Continuity feature. These recoverable errors may be caused by a temporary outage such as a session, node, or network failure.

The goal of the Application Continuity feature is to safely attempt to replay the in-flight work following an outage. Planned outages due to configuration changes and patch application are also eligible for recovery. Application Continuity uses another new Oracle Database feature, the Transaction Guard, to perform its job. When it can successfully replay a "lost" in-flight transaction, Application Continuity can handle hardware, software, network, and storage-related errors and outages.

You can use Universal Connection Pool 12*c*, JDBC Type 4 Driver 12*c*, or Oracle WebLogic Server 12.1.2 to take advantage of the Application Continuity feature. Third-party Java applications must use the Oracle Database 12*c* JDBC Replay Driver.

Before configuring Application Continuity, make sure you have enough processing power and memory available in the both the middle tier and the database tier.

When Application Continuity is configured, if a DBA kills or disconnects a session, the feature will make the database automatically try to recover the terminated session. Use the NOREPLAY keyword with the **kill session** and **disconnect session** commands if you don't want the database to replay the session, as shown here:

```
$ alter system kill session 'sid, serial#, @inst' noreplay;
$ alter system disconnect session 'sid, serial#, @inst' noreplay;
```

In both commands, "inst" refers to the instance number.

How Application Continuity Works

Following is the logical sequence of steps involved in enforcing the Application

https://t.me/learningnets

Continuity feature:

1. Configure your services for application continuity by setting the **failovertype** parameter to TRANSACTION and the **commit_outcome** parameter to TRUE. You can do this when creating a service or by modifying an existing service.
2. When an application connects to the database using the service you configured in step 1, its session handle will include a logical transaction ID (LTXID) for every transaction.
3. When a recoverable error occurs, an error message in the form of a FAN message is sent back to the replay driver, which works closely with the server.
4. For JDBC (Type 4) applications, the Fast Connection Failover (FCF) feature will abort the dead session when the replay driver receives the error (or down) message. The replay driver will do the following at this stage:
   - It'll open a new session to replace the dead session.
   - It finds out if there were any open transactions, using the Transaction Guard for this purpose.
   - It prepares for a replay if there were indeed one or more in-flight transactions when the session died.
   - If the application developers registered a labeling callback or connecting callback, it uses them to call back for the initial session state.
   - It rebuilds the dead session and recovers the transactional and non-transactional states after validating the states.
5. Once the replay is imitated and the transaction replay begins, the database continues to validate the replay. If the database fails to correctly rebuild the session state to its original state, the replay driver will communicate the original error back to the application.

Let's say a user selects an item for purchase from a web checkout. The user transaction will wend its way through the application infrastructure before ending up as successful database transaction. Let's assume that there is an error somewhere in the infrastructure supporting the database right before the transaction is committed. Under Application Continuity, the JDBC driver will catch the failure and, using the Transaction Guard feature, will check with a different RAC node to confirm whether the transaction was committed, or if it should replay the transaction.

In case the transaction wasn't committed, Application Continuity will replay the transaction by submitting all the in-flight work to an alternative RAC node and ensure that the transaction is committed. The application is oblivious to the whole replay operation because it's silently managed behind the scenes by Application Continuity.

Configuring a Service for Application Continuity

In order to use the Application Continuity feature, you must configure a service. The same goes for the Transaction Guard feature as well, thus providing you more

https://t.me/learningnets

Technet24

incentive to configure services to manage and track workloads efficiently.

The two required parameters you need to set for a service in order to use Application Continuity are **failovertype** and **commit_outcome**. Set the **failover** parameter to the value TRANSACTION and the **commit_outcome** parameter to the value TRUE.

Here's an example showing how to configure a service for Application Continuity in your environment (we'll also explain the optional parameters):

```
$ srvctl add service -db prod -service myservice -serverpool srvpool1
    -failovertype TRANSACTION -commit_outcome TRUE -replay_init_time 1800
    -retention 86400 -notification TRUE -rlbgoal SERVICE_TIME -clbgoal SHORT
    -failoverretry 30 -failoverdelay 10
```

In this example, we specified several optional parameters beyond the two required parameters. Let's briefly review those optional parameters:

- **replay_init_time** Length of time in which the replay will start. The default is 300 seconds.
- **Retention** Length of time (in seconds) the database will store the commit outcome data. The default is one day.
- **Notification** If you set this to TRUE, FAN notifications will be sent for OCI and ODP
  .NET clients.
- **rlbgoal** Use this parameter for connection load balancing. Specify SHORT for runtime load balancing.
- **clbgoal** Use this parameter for runtime load balancing and set the value to SERVICE_TIME.
- **failoverretry** Sets the number of connection retries for each connection attempt. The recommended value is 10.
- **failoverdelay** The time in between each connection attempt. The recommended value is 10.

You can also modify an existing service by using the **srvctl modify service** command in a similar manner.

Summary

Oracle was one of the first commercial databases to foresee the need for an efficient mechanism to deal with a large user population and workload distribution in a single-instance database as well as multi-instance environments. Workload distribution is very critical for an Oracle RAC environment to have balanced resource utilization among the cluster nodes and to experience optimal performance.

The Transparent Application Failover (TAF) feature helps the connected sessions

https://t.me/learningnets

seamlessly migrate to another instance during an unexpected instance failure. Complex applications and applications not supported by TAF configurations can utilize the Fast Application Notification (FAN) feature discussed in this chapter. End-user applications can subscribe to FAN notifications and take appropriate actions based on the FAN events.

The new Oracle Database 12*c* feature Transaction Guard improves client failover by preventing transactions being replayed by the Application Continuity feature from being applied multiple times. You need to configure services when using the new Application Continuity and Transaction Guard features. The chapter explained how to set values for various service parameters for Application Continuity.

With so many rich features and flexible options in load balancing and connection management, users have plenty of choices for managing the connections.

https://t.me/learningnets

Technet24

CHAPTER 14

Oracle RAC Troubleshooting

Previous chapters in this book provided comprehensive information for understanding the architecture, functionalities, and processes of Oracle Real Application Clusters 12*c*. How efficiently and effectively you can troubleshoot this behemoth depends on your level of understanding of Oracle Real Application Clusters and the associated Oracle Grid Infrastructure components. In this chapter, we look into the details of debugging Oracle Real Application Clusters, from simple startup problems to complex system hang or crash problems.

As a single piece of software, the Oracle RDBMS is one of the most complex commercial products in the world, with lots of changes incorporated into Oracle RAC 12*c*. Troubleshooting the Oracle RAC database involves analyzing the debug information of Oracle Grid Infrastructure and database instances. But with the help of a solid and extensive diagnostics framework, you can usually diagnose even complex problems simply by viewing and interpreting Oracle's detailed debug trace files.

The initialization parameter DIAGNOSTIC_DEST specifies the location of the directory for storing diagnostic data. This location is also called the Automatic Diagnostic Repository (ADR) home. Each database instance in an Oracle RAC system stores its own alert log inside the $DIAGNOSTIC_DEST/diag/rdbms/<dbname>/SID/alert directory. This is the first

https://t.me/learningnets

place to look whenever the database encounters problems. Alert logs show detailed information about the basic settings of the database, including the nondefault parameters used.

Alert logs also contain information about startup and shutdown as well as details of nodes joining and leaving, with timestamps.

Just like a database instance has an alert log, each cluster node has a clusterware alert log (alert<hostname>.log) stored inside the $ORACLE_HOME/log/<hostname> directory. This will be the first log file to analyze if the database alert log file has reported symptoms/incidents related to Oracle Clusterware or ASM. Before getting into details on how to troubleshoot, it is very important to know the location of all debug files required to diagnose different processes and components of the Oracle RAC database.

Installation Log Files

Installing Oracle RAC involves the installation of software and the execution of post-installation configuration scripts such as root.sh. Therefore, you will be analyzing different types of debug log files, depending on at which stage of the installation the failure has occurred. Oracle stores software-installation log files inside the logs directory of Oracle central inventory (the location of central inventory is specified by the oraInst.loc file). This file is stored at /etc/oracle on the Linux operating system and at /var/opt/oracle on other UNIX operating systems. Oracle records all actions performed by Oracle Universal Installer in the log file named Install Actions<*timestamp*>.log.

Oracle stores the debug log file for the root.sh configuration script under the $GRID_HOME/cfgtools/crsconfig directory. Oracle records actions performed by the root.sh configuration script in the rootcrs_<hostname>.log debug log file.

NOTE

My Oracle Support Bulletin ID 811306.1, "RAC and Oracle Clusterware Starter Kit and Best Practices," from the Oracle RAC Assurance Support Team, provides the latest information on generic and platform-specific best practices for implementing an Oracle RAC cluster. This document is specifically for Linux, but there are separate documents for all operating systems.

Log Directory Structure in the Oracle RDBMS

Oracle stores all diagnostic data, such as trace files, core files, and the alert logs, inside a common directory structure called Automatic Diagnostic Repository (also known as

https://t.me/learningnets

Technet24

the ADR). The ADR has a unified directory structure across multiple database instances, and all major components involved in the Oracle RAC database, such as ASM, CRS, and database instances, store their diagnostic files inside their own dedicated directory within the ADR. Using a unified directory structure to store all diagnostic data in a common place allows Oracle support tools such as the ADR Command Interpreter (ADRCI) to collect and analyze diagnostic data across multiple databases and ASM instances.

The location of the ADR is specified by the database initialization parameter DIAGNOSTIC_DEST, which by default points to a location specified by the operation system environment variable $ORACLE_BASE. In case both the environment variable ORACLE_BASE and the database initialization parameter DIAGNOSTIC_DEST aren't defined, Oracle by default uses $ORACLE_HOME/log as the Automatic Diagnostic Repository. ADR contains one top directory named diag as well as subdirectories for technology components of the Oracle RAC database to store their diagnostic files. In a typical Oracle Database 11*g* installation, you will see the following subdirectories inside the $ADR_BASE/diag directory:

- asm
- clients
- crs
- diagtool
- lsnrctl
- netcman
- ofm
- rdbms
- tnslsnr

In an Oracle RAC 12*c* installation, where a separate user owns Oracle Grid Infrastructure and the Oracle RAC database, you may notice that the directories (except for rdbms, clients, and diagtool) are empty, because the software owner of Oracle Grid Infrastructure has its own dedicated ORACLE_BASE and the DIAGNOSTIC_DEST parameter isn't configured. You can query the database view V$DIAG_INFO to display the ADR locations. This view provides important information about ADR locations as well as the number of active incidents and problems reported inside the ADR.

NOTE

Users can use the ADR command-line interface tool ADRCI to view and package diagnostic data stored inside the ADR.

https://t.me/learningnets

On a database instance, the rdbms/<dbname>/<SID> subdirectory has the following important subdirectories that store the diagnostic files for the respective database instances:

- **cdump** This directory contains core dumps.
- **trace** This directory stores the database alert log and trace files of the foreground and background database processes.
- **alert** This directory stores the XML alert log.
- **hm** This directory stores the health check output files.
- **incident** This directory stores the incident directory created by critical errors, and each incident directory is named as the ID of the incident.
- **metadata** This directory contains the diagnostic metadata.

Log Directory Structure in Oracle Grid Infrastructure

As mentioned earlier, in an Oracle 12*c* RAC installation, a cluster node consists of a Grid home and an Oracle RDBMS home. The Grid home is where Oracle Grid Infrastructure is installed and is commonly known as GRID_HOME. Diagnostic log files for Oracle Clusterware and ASM are stored under the $GRID_HOME/log/<hostname> directory. Each component in the Oracle Grid Infrastructure stack has its respective directories created under the $GRID_HOME/log/<hostname> directory. Oracle Grid Infrastructure stores all Oracle Clusterware–related debug log files under the $GRID_HOME/log directory. In a typical Oracle 12*c* Grid Infrastructure home, you will find the following directories containing debug log files for different components of Oracle Grid Infrastructure:

- **$GRID_HOME/log** This is the main directory for storing trace and diagnostic log files of Oracle Clusterware and ASM on each cluster node.
- **$GRID_HOME/log/<hostname>** Contains trace and diagnostic log files of Oracle Clusterware and ASM for the local node only. Each cluster node has its own dedicated log directory inside its own GRID_HOME. This directory contains individual subdirectories for each component of Oracle Grid Infrastructure.
- **$GRID_HOME/log/<hostname>/agent** Contains trace and diagnostic log files for oraagent, orarootagent, oracssdagent, and oracssdmonitor for the CRSD and OHASD daemons.
- **$GRID_HOME/log/<hostname>/client** Contains trace and diagnostic log files for various Oracle Grid Infrastructure clients, such as CLSCFG, GPNP, OCRCONFIG, OLSNODES, and OIFCFG.
- **$GRID_HOME/log/<hostname>/crfmond** Contains trace and diagnostic log

https://t.me/learningnets

Technet24

files recorded by the System Monitor Service provided by Oracle Cluster Health Monitor (Oracle CHM).

- **$GRID_HOME/log/<hostname>/cssd** Contains Cluster Synchronization Services (CSS) logs, containing actions such as reconfigurations, missed check-ins, connects, and disconnects from the client CSS listener. In some cases, the logger logs messages with the category of auth.crit for the reboots done by Oracle. This could be used for checking the exact time when the reboot occurred.
- **$GRID_HOME/log/<hostname>/cvu** Contains the trace and debug log files generated by the Oracle Cluster Verification utility.
- **$GRID_HOME/log/<hostname>/evmd** Contains the trace and diagnostic files for the Event Volume Manager (EVM) and evmlogger daemons. This is not used as often for debugging as the CRSD and CSSD directories.
- **$GRID_HOME/log/<hostname>/gnsd** Contains the trace and debug log files to troubleshoot problems related to the Oracle Grid Naming Service, which is introduced in Oracle Grid Infrastructure.
- **$GRID_HOME/log/<hostname>/mdnsd** Contains the trace and diagnostic log files to troubleshoot the Multicast Domain Name Service. The Oracle Grid Naming Service uses this service to manage name resolution and service discovery.
- **$GRID_HOME/log/<hostname>/racg** Contains the trace and debug log files for each Oracle RACG executable.
- **$GRID_HOME/log/<hostname>/crflogd** Contains the trace and diagnostic log files recorded by Cluster Logger Services provided by the Oracle Cluster Health Monitor (Oracle CHM).
- **$GRID_HOME/log/<hostname>/crsd** Contains the trace and diagnostic log files of the Oracle CRSD daemon and is a good point to start with any Oracle Clusterware issues.
- **$GRID_HOME/log/<hostname>/ctssd** Contains the debug log files to troubleshoot the Oracle Cluster Time Synchronization Service introduced in Oracle Grid Infrastructure to synchronize the clocks on the cluster nodes.
- **$GRID_HOME/log/<hostname>/diskmon** Contains the debug log files to troubleshoot the Oracle Disk Monitor daemon.
- **$GRID_HOME/log/<hostname>/giplcd** Contains the debug and trace files to troubleshoot the Oracle Grid Interprocess Communication daemon.
- **$GRID_HOME/log/<hostname>/gpnpd** Contains the log and output files of the Oracle Grid Plug and Play daemon.
- **$GRID_HOME/log/<hostname>/ohasd** Contains the log and output files of the Oracle High Availability Services daemon. The OHASD log files are really important to diagnose cluster startup issues in Oracle 11*g* onward.
- **$GRID_HOME/log/<hostname>/srvm** Contains log files for the Oracle

https://t.me/learningnets

Server Manager service.

Troubleshooting a Failed Oracle Grid Infrastructure Installation

Troubleshooting a failed Oracle RAC install starts with analyzing different sets of log files, depending on the stage of the install where the failure has occurred. For example, to troubleshoot an installation failure that has occurred prior to executing the root.sh configuration script, completing most of the prerequisites in the installation documentation and fixing the errors reported by OUI in the Perform Prerequisites Checks screen ensures that the installation will succeed. However, there may be scenarios where the installation still fails even after you've ensured the prerequisites. In such cases, you review the installation log files stored inside the log subdirectory of the Oracle Central Inventory. Oracle Universal Installer records each install action in the installation log file, and reviewing this log file shows you the actual action that Oracle Universal Installer was performing when the installation failed.

Oracle Universal Installer is a Java program, and advanced Java trace can be enabled to trace detail debug information about actions being performed by Oracle Universal Installer. Chapter 4 explains the procedure to enable advanced Java tracing for Oracle Universal Installer. You have to cancel the existing installation session and restart the Oracle Universal Installer with advanced Java trace options if you wish to debug. Once enabled, the trace file provides the exact commands being executed by Oracle Universal Installer, along with the command's output. After reviewing this trace file, you can narrow down the command or action that causes the installation to fail and take the appropriate measures to resolve the underlying issue. Once the underlying issue is resolved, you can either proceed with the current installation or restart the Oracle Universal Installer, depending on at which stage the error occurred—the OUI offers "retry" options where it can.

Sometimes an Oracle RAC installation fails while executing a post-installation configuration script, such as root.sh, especially when Oracle Grid Infrastructure is being installed. The root.sh configuration script configures and starts the Oracle Clusterware and ASM if the OCR and voting files are stored inside the ASM. The root.sh script validates permissions on the file system and configures network resources, such as creating virtual IPs for configuring storage, creating the ASM disk group to store OCR and voting files, and establishing a link between third-party clusterware and Oracle Clusterware, if third-party clusterware is used. In most cases, misconfiguration of the network and shared storage are the main reasons behind failure of the root.sh configuration script—unless you are encountering an unwelcome Oracle bug.

Inside the root.sh configuration script, you can see that it executes rootmacro.sh, rootinstall.sh, setowner.sh, rootadd_rdbms.sh, rootadd_filemap.sh, and rootcrs.pl. In this case, rootcrs.pl is the main configuration script, and it performs most of the configuration done by root.sh. Other configuration scripts only validate permissions

https://t.me/learningnets

Technet24

and create the required directories on the file system. It is very easy to fix a failure of the root.sh script if it is happening in a script other than rootcrs.pl.

Oracle records all actions performed by the rootcrs.pl configuration script in the rootcrs_<hostname>.log file, stored in the $GRID_HOME/cfgtools/crsconfig directory, and this is the first file to be analyzed to identify the action/command that's causing root.sh failure. Once the failed configuration item is known, further diagnostic log files must be reviewed to identify the root cause of the failure. For example, if the rootcrs_<hostname>.log file is reporting a failure upon starting the Cluster Synchronization Services daemon, then you review the diagnostic log file for the OCSSD daemon stored at $GRID_HOME/log/<hostname>/cssd/ocssd.log to identify the root cause of the failure. Once the root cause is identified, the root.sh configuration script can be rerun, as the configuration script root.sh is restartable on the failed node.

To rerun root.sh, first deconfigure the cluster by executing the following two scripts as the root user. First, run the following script on all cluster nodes except the last node:

```
$GRID_HOME/crs/install/rootcrs.pl -verbose -deconfig -force
```

Then run the following script on the last cluster node. This script will erase data from the OCR and voting disk:

```
$GRID_HOME/crs/install/rootcrs.pl -verbose -deconfig -force -lastnode
```

It's a good idea to be patient while troubleshooting a root.sh failure because sometimes simply overlooking the available information can delay the overall troubleshooting response time. For example, root.sh fails with the following error on the terminal:

```
PROT-1: Failed to initialize ocrconfig
Command return code of 255 (65280) from command:
/u01/12.1.0/app/grid/bin/ocrconfig -upgrade oragrid oinstall
Failed to create Oracle Cluster Registry configuration, rc 255
```

As shown next, after reviewing the rootcrs_<hostname>.log file, you know that the root.sh configuration script is using software user "oragrid" to create and configure OCR inside the ASM, but the ASMLib drivers were configured using the software owner "oracle":

```
Executing as oragrid: /u02/12.1.0/app/grid/bin/asmca -silent
-diskGroupName DATA -diskList ORCL:ASMD1,ORCL:ASMD2
-redundancy EXTERNAL -configureLocalASM
Creating or upgrading OCR keys
Command return code of 255 (65280) from command:
/u01/app/12.1.0/grid/bin/ocrconfig -upgrade oragrid oinstall
```

In this example, the install issue is resolved after you reconfigure ASMLib with the oragrid user and rerun root.sh successfully.

https://t.me/learningnets

Oracle 12*c* RAC requires several prerequisites to be met prior to installing Oracle Grid Infrastructure. For example, multicasting must be enabled on the private network because Oracle uses multicasting to provide highly available IP (HAIP) to implement redundancy of the private interconnect using the Oracle stack; if multicasting is not enabled, the root.sh configuration script succeeds on the first cluster node but fails on subsequent cluster nodes to start Cluster Synchronization Services in clustered mode.

Oracle uses multicasting to establish the initial connection/link with other cluster nodes, and if multicasting is not enabled, other cluster nodes cannot join an existing cluster, which root.sh starts when executed on the first node. Oracle provides a program to validate whether or not multicasting on the private network is enabled. Refer to My Oracle Support Note 1212703.1, which details this requirement of enabling multicast for the private network. Sometimes the root.sh script fails to start Cluster Synchronization Services in clustered mode even when multicasting is enabled on the private network. This may happen if IGMP (Internet Group Management Protocol) snooping is enabled and is rejecting the multicast network packets on the private network, so ensure that IGMP snooping is configured appropriately to allow multicast networking.

NOTE

Many organizations do not allow multicasting within production networks. When such policies exist, make sure you let the Network Admins/Network Security teams know you'll need to use multicasting. This is a common reason for failures, and unless you understand this restriction, and remove it with help from the network teams, the installation can't proceed! The same goes for IGMP snooping.

Apart from the obvious reasons, certain bugs may cause root.sh to fail. Therefore, it's always a good practice to search the knowledge base and bug database to see if you are encountering a known bug to speed up the resolution process.

Inside the Database Alert Log

The directory where the install software for Oracle RDBMS is stored is known as ORACLE_HOME. The log files stored are related to each of the database instances running out of this Oracle home. Suppose, for example, that you have a database called TEST with a database instance called TEST1. All its background process trace files, along with alert.log, would be available in the $ORACLE_BASE/diag/rdbms/test/TEST1/trace directory, which is the default location.

Let's take a closer look at starting up an instance in a two-node Oracle 12*c* RAC cluster. We use SQL*Plus to issue the startup command on node1 and then on node2. The sequence of events follows, with additional explanations added as required.

https://t.me/learningnets

Technet24

NOTE

The IP addresses are masked in the following examples because they are taken from live databases.

```
SQL> startup nomount;
<from the alert log>
Cluster communication is configured to use the following interface(s) for this instance
  169.254.222.55
cluster interconnect IPC version:Oracle UDP/IP (generic)
IPC Vendor 1 proto 2
Wed Aug 23 16:37:47 2017
PMON started with pid=2, OS id=10254
```

As shown in this extract of an actual alert log for an Oracle RAC 12*c* environment, Oracle uses highly available IP as the interconnect from the reserved IP range 169.254.X.X. The correct interconnect must be used for the Cache Fusion traffic. Some may choose the public network for the interconnect traffic, but doing so will bring the database to its knees. To identify the network used for Cache Fusion or a private traffic, you can do any of the following:

```
 # /oifcfg getif
bond0  173.21.17.127  global  public
bond1  173.21.17.0  global  cluster_interconnect
#
```

As you can see, bond1 is used as the cluster interconnect, and the private virtual IP 169.254.222.55 is configured and running on this network interface, which is evident from the following command:

```
 # /sbin/ifconfig bond1:1
# /sbin/ifconfig bond1:1
bond1:1   Link encap:Ethernet  HWaddr 00:17:A4:77:0C:8E
          inet addr:169.254.222.55  Bcast:169.254.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
```

Depending on the source of the information, PICKED_KSXPIA is populated with the values GPnP, OSD, OCR, and CI. If the CLUSTER_INTERCONNECTS parameter is set in the SPFILE, the query will return the following output:

```
SQL> select INST_ID,PUB_KSXPIA,PICKED_KSXPIA,NAME_KSXPIA,IP_KSXPIA
from x$ksxpia where PUB_KSXPIA = 'N';;
  INST_ID P PICK NAME_KSXPIA  IP_KSXPIA
--------- - ---- ------------ ----------------
        1 N GPnP bond1:1      169.254.222.55
```

Another option is the conventional method of finding the interconnect information

https://t.me/learningnets

from an Interprocess Communications (IPC) dump. X$KSXPIA is exposed as GV$CONFIGURED_INTERCONNECTS.

Log in to the database as user SYS:

```
SQL> oradebug setmypid
SQL> oradebug tracefile_name
SQL> oradebug ipc
```

This will dump a trace file to DIAGNOSTIC_DIR/diag/rdbms/<dbname>/SID/trace. The output will look something like this:

```
SKGXP:[2b1b8d37ba58.44]{ctx}:   SSKGXPT 0x2b1b8d37d180 flags 0x5 { READPENDING }
sockno 5 IP 169.254.222.55 UDP 50842 lerr 0
SKGXP:[47397333351000.44]{ctx}:
```

You can see that we are using IP 169.254.222.55 with a User Datagram Protocol (UDP). To change the network that Oracle RAC uses, you can change the order of the networks in the operating system–dependent network configurations using the CLUSTER_INTERCONNECTS parameter.

The moment the first instance is started, its alert log gets populated with important information. The first thing to notice in the preceding example after the list of nondefault parameters is the IP address used for the interconnect. Ensure that you see the correct protocol you have configured for interconnect usage. The supported interconnect protocols with respect to various operating systems are listed in Chapter 3.

https://t.me/learningnets

Technet24

```
lmon registered with NM - instance number 1 (internal mem no 0)
Reconfiguration started (old inc 0, new inc 6)
List of instances:
 1 2 (myinst: 1)
 Global Resource Directory frozen
* allocate domain 0, invalid = TRUE
 Communication channels reestablished
 * domain 0 not valid according to instance 2
 * domain 0 valid = 0 according to instance 2
 Master broadcasted resource hash value bitmaps
 Non-local Process blocks cleaned out
 LMS 0: 0 GCS shadows cancelled, 0 closed, 0 Xw survived
 LMS 1: 0 GCS shadows cancelled, 0 closed, 0 Xw survived
 Set master node info
 Submitted all remote-enqueue requests
 Dwn-cvts replayed, VALBLKs dubious
 All grantable enqueues granted
 Post SMON to start 1st pass IR
 Submitted all GCS remote-cache requests
 Post SMON to start 1st pass IR
 Fix write in gcs resources
Reconfiguration complete
```

When an instance starts up, it's the job of Lock Monitor (LMON) to register with the Node Monitor (NM). That's what we see in the alert.log with the instance ID that is getting registered. When any node joins or leaves a cluster, the global resource directory undergoes a reconfiguration event. We see the start of the reconfiguration event along with the old and new incarnation. Next, we see the number of nodes that have joined the cluster. Because this was the first node to be started up, in the list of nodes we see only one node listed, and the number starts with 0. A reconfiguration event is a seven-step procedure, and upon completion the "reconfiguration complete" message is logged to alert.log.

The messages logged in alert.log are summaries of the reconfiguration event. The LMON trace file would have more information about the reconfiguration. Following are the contents of the LMON trace file:

```
kjxgmrcfg: Reconfiguration started, reason 1
kjxgmcs: Setting state to 0 0.
```

Here, you can see the reason for the reconfiguration event. The most common reasons would be 1, 2, and 3. Reason 1 means that the NM initiated the reconfiguration event, as typically seen when a node joins or leaves a cluster. A reconfiguration event is initiated with reason 2 when an instance death is detected. How is an instance death detected? Every instance updates the controlfile with a heartbeat through its Checkpoint (CKPT) process. If heartbeat information is not present for x amount of time (normally 30 seconds in UNIX variants), the instance is considered to be dead

https://t.me/learningnets

and the Instance Membership Recovery (IMR) process initiates reconfiguration. This type of reconfiguration is commonly seen when significant time changes occur across nodes, the node is starved for CPU or I/O time, or some problems occur with the shared storage.

A reason 3 reconfiguration event is due to a communication failure. Communication channels are established between the Oracle processes across the nodes. This communication occurs over the interconnect. Every message sender expects an acknowledgment message from the receiver. If a message is not received for a timeout period, then a “communication failure” is assumed.

This is more relevant for protocols such as UDP because Reliable Shared Memory (RSM), Reliable DataGram protocol (RDG), and Hyper Messaging Protocol (HMP) don’t need it since the acknowledgment mechanisms are built into the cluster communication and protocol itself. When the block is sent from one instance to another using wire, especially when unreliable protocols such as UDP are used, it is best to get an acknowledgment message from the receiver.

The acknowledgment is a simple side-channel message that is normally required for most of the UNIX systems, where UDP is used as the default IPC protocol. When user-mode IPC protocols such as RDG (on HP Tru64 UNIX TruCluster) or HP HMP are used, the additional messaging can be disabled by setting the value of the _RELIABLE_BLOCK_SENDS parameter to TRUE.

NOTE

The hidden database parameter _RELIABLE_BLOCK_SENDS is by default set to TRUE.

For Windows-based systems, it is always recommended to leave the default value as is:

```
Database mounted in Shared Mode (CLUSTER_DATABASE=TRUE).
Completed: alter database mount
```

Because this is an Oracle RAC database, every instance mounts the database in shared mode. Sometimes you want to mount the database in exclusive mode, as in completing the actions of a patch set application. Checking the alert log is one way to confirm that:

```
Wed Aug 23 16:38:04 2017
alter database open
```

This instance was first to open. The wait that occurs between when the time the Commit SCN is broadcasted to all nodes and the log writer writes the pending redo is minimal because the broadcast and commit are asynchronous. This means the system

https://t.me/learningnets

Technet24

waits until it is sure that all nodes have seen the Commit SCN. Any message with an SCN greater than Commit SCN is deemed sufficient.

When the instance is mounted, the database is opened. But before the opening, the cluster must make sure every instance has applied the changes from all instances. Ordinarily you would think that this process takes time. Fortunately, it doesn't. Why? It's because all the instance does is to wait for the broadcast SCN. Once it gets that, it opens the database. The other changes are processed asynchronously. Therefore, the database opening doesn't have to wait.

You can measure the progress of the database instance recovery by querying the V$INSTANCE_RECOVERY view. Before broadcasting the Commit SCN, the process checks whether it has already received a higher SCN from that instance.

RAC ON and OFF

In some cases, you may want to disable the Oracle RAC options for testing purposes—perhaps to run a benchmark or to convert the Oracle RAC binaries to single-instance binaries.

In such a case, you can use the following procedure to convert the Oracle RAC installation to non–Oracle RAC. Disabling and enabling Oracle RAC options are available only for UNIX platforms. Windows installations do not support relinking binaries with Oracle RAC ON and OFF.

Use the following steps to disable Oracle RAC (known as RAC OFF):

1. Log in as the Oracle software owner (which is typically the UNIX account "oracle") in all nodes.
2. Shut down all the instances from all the nodes using the NORMAL or IMMEDIATE option.
3. Change the working directory to $ORACLE_HOME/lib:
   ```
   cd $ORACLE_HOME/lib
   ```
4. Run the following **make** command to relink the Oracle binaries without the Oracle RAC option:
   ```
   make -f ins_rdbms.mk rac_off
   ```
   This normally runs for a few minutes and should not pose any errors.
5. Now relink the Oracle binaries:
   ```
   make -f ins_rdbms.mk ioracle
   ```

Now the Oracle binaries are relinked with the RAC OFF option. You may have to edit the init.ora or SPFILE parameters accordingly. If errors occur in step 4, you may need to contact Oracle Support and log a service request with the trace and log files.

Use the following steps to enable Oracle RAC (known as RAC ON):

https://t.me/learningnets

1. Log in as the Oracle software owner (typically the UNIX account “oracle”) in all nodes.

2. Shut down all the instances from all the nodes using the NORMAL or IMMEDIATE option.

3. Change the working directory to $ORACLE_HOME/lib:
   ```
   cd $ORACLE_HOME/lib
   ```

4. Run the following **make** command to relink the Oracle binaries with the RAC option:
   ```
   make -f ins_rdbms.mk rac_on
   ```
   This normally runs for a few minutes and should not pose any errors.

5. Now relink the Oracle binaries:
   ```
   make -f ins_rdbms.mk ioracle
   ```

Now the Oracle binaries are relinked with the RAC ON option. You may need to edit the init.ora or SPFILE parameters accordingly. If any errors occur in step 4, you may need to contact Oracle Support and log a service request with the trace and log files.

Database Performance Issues

Oracle RAC databases have more than one instance using the same set of resources, and a resource may be requested by more than one instance. Resource sharing is well managed by Global Cache Services (GCS) and Global Enqueue Services (GES). However, in some cases, the resource management operations could run into a deadlock situation and the entire database may hang because of serialization issues. Sometimes software bugs can cause database-hang issues, and these situations almost always require the intervention of Oracle Support in the form of a service request.

Database-hang issues can be placed in the following categories:

- Hung database
- Hung session(s)
- Overall instance/database performance
- Query performance

We will examine only the hung database because that is more critical and complex than the other categories and also related to our point of interest. What’s more, detailed texts are available for analyzing the database and query performance.

Hung Database

https://t.me/learningnets

Technet24

Oracle Support defines a “true” database hang as “an internal deadlock or a cyclical dependency between two or more processes.” When dealing with DML locks (that is, enqueue type TM), Oracle is able to detect this dependency and roll back one of the processes to break the cyclical condition. On the other hand, when this situation occurs with internal kernel-level resources (such as latches or pins), Oracle is usually unable to automatically detect and resolve the deadlock.

If you encounter a database-hang situation, you need to generate system-state dumps so that Oracle Support can begin to diagnose the root cause of the problem. Whenever you generate such dumps for a hang, it is important to make at least three of them a few minutes apart, on all instances of your database. That way, evidence shows whether a resource is still being held from one time to the next.

System-state dumps can be quite large and require sufficient storage. In addition, they require a lot of I/O and CPU resources. Unfortunately, you need to generate the dumps during a database hang, which sometimes could be a problem.

The maximum dump file size should be set to unlimited, as this will generate bigger and larger trace files, depending on the size of the System Global Area (SGA), the number of sessions logged in, and the workload on the system. The SYSTEMSTATE dump contains a separate section with information for each process. Normally, you need to generate two or three dumps in regular intervals. Whenever you generate SYSTEMSTATE dumps repetitively, make sure you reconnect every time so that you get a new process ID and also the new trace files. Expect HUGE trace files! If you’re generating multiple system state dumps, ensure that you have sufficient free space in the dump directory.

A SYSTEMSTATE dump includes session wait history information, and you don’t need to generate multiple system state dumps. The SYSTEMSTATE dump can be taken by any of the following methods.

From SQL*Plus:

```
alter session set max_dump_file_size = unlimited;
alter session set events 'immediate trace name systemstate level 10';
```

Using **oradebug**:

```
oradebug setmypid
oradebug unlimit
oradebug dump systemstate 10
```

When the entire database is hung and you cannot connect to SQL*Plus, you can try invoking SQL*Plus with the **prelim** option if you’re using Oracle 10*g* or later. This attaches the process to the Oracle instance and no SQL commands are run. No login triggers or pre-processing is done, and no SQL queries are allowed to run.

Alternatively, **oradebug** allows you to dump the global system state by connecting to one node. The following shows the global SYSTEMSTATE dump from **oradebug**:

```
oradebug -g all dump systemstate 10
```

https://t.me/learningnets

The **–g** option is used for Oracle RAC databases only. This will dump system states for all the instances. The SYSTEMSTATE dump/trace file can be found in the trace directory under ADR on the instance where the dump was generated.

Hanganalyze Utility

A severe performance problem can be mistaken for a hang. This usually happens when contention is so bad that it *seems like* the database is completely hung. Usually, a SYSTEMSTATE dump is used to analyze these situations. However, if the instance is large with more than a few gigabytes of SGA and with a heavy workload, a SYSTEMSTATE dump may take an hour or more and often fails to capture all the SGA and lock structures. Moreover, a SYSTEMSTATE dump has the following limitations when dealing with hang issues:

- It reads the SGA in a "dirty" manner, so it may be inconsistent when the time to dump all of the process is long.
- It usually provides a lot of information (most of which is not needed to determine the source of the hang), which makes it difficult to determine quickly the dependencies between processes.
- It does not identify "interesting" processes on which to perform additional dumps (ERRORSTACK or PROCESS STATE).
- It's often a very expensive operation for databases that have large SGAs. A SYSTEMSTATE dump can take hours, and generating a few continuous dumps within an interval of a few minutes is nearly impossible.

The **hanganalyze** command overcomes the limitations of the SYSTEMSTATE dump utility and provides cluster-wide information in Oracle RAC environments with a single shot. This command uses the DIAG daemon process in the Oracle RAC process to communicate between the instances. Cluster-wide, **hanganalyze** will generate information for all the sessions in the cluster regardless of the instance that issued the command.

You can invoke **hanganalyze** from SQL*Plus or through **oradebug** (which is available when you connect as SYS in the SQL*Plus utility). The following syntax can be used to get a **hanganalyze** trace when connected to SQL*Plus:

```
alter session set events 'immediate trace name hanganalyze level <level>';
```

Or when logged in as SYS:

```
oradebug hanganalyze <level>
```

Cluster-wide, **hanganalyze** can be obtained like so:

https://t.me/learningnets

```
oradebug setmypid
oradebug setinst all
oradebug -g def hanganalyze <level>
```

Here, <level> sets the amount of additional information that will be extracted from the processes found by **hanganalyze** (ERROSTACK dump) based on the STATE of the node. The following table describes the various levels and the trace information produced when they are set:

Level	Trace Information
1–2	Only **hanganalyze** output. No process dump at all.
3	Level 2 + dump only processes thought to be in a hang (IN_HANG state).
4	Level 3 + dump leaf nodes (blockers) in wait chains (LEAF, LEAF_NW, IGN_DMP state).
5	Level 4 + dump all processes involved in wait chains (NLEAF state).
10	Dump all processes (IGN state).

The **hanganalyze** utility uses internal kernel calls to determine whether a session is waiting for a resource, and it reports the relationships between blockers and waiters. In addition, it determines which processes are "interesting" to be dumped, and it may perform automatic PROCESS STATE dumps and ERRORSTACKs on those processes, based on the level used while the command is being executed.

NOTE

*The **hanganalyze** utility is not intended to replace a SYSTEMSTATE dump, but it may serve as a roadmap to interpret a system state while you're diagnosing complex issues. Performance issues related to row cache objects, enqueues, and latches can be analyzed only with **hanganalyze** and/or a SYSTEMSTATE dump. The process state information in these dumps provides an accurate snapshot of the locks/latches held by each process or session. It also tells us which event the process was waiting for and if any TM locks were held for a transaction. Problems associated with Data Definition Language (DDL) statement locks and row cache lock issues can be debugged using only the SYSTEMSTATE dumps.*

Debugging Node Eviction Issues

One of the most common and complex issues in Oracle RAC is performing the root cause analysis (RCA) of the node eviction issues. Oracle evicts a node from the cluster mainly due to one of the following three reasons:

- The node is not able to complete the network heartbeat (NHB).

https://t.me/learningnets

- The node is not able to complete the disk heartbeat (DHB).
- The node does not have enough CPU to perform either of the heartbeat operations.

The whole troubleshooting theory to debug node evictions in Oracle Clusterware revolves around the aforementioned reasons. You must read the previous chapters in this book to understand how Oracle performs network and disk heartbeats—it is very important to understand this process before you attempt to troubleshoot node evictions. The node eviction process is reported as Oracle error ORA-29740 in the alert log and LMON trace files. To determine the root cause, first look at the Oracle Clusterware alert_hostname.log.

The first step is to establish the reason why the failed node is evicted from the cluster. Was it not able to perform the network heartbeat (NHB) or disk heartbeat (DHB)? Which process was responsible for evicting the failed node? Was there an infrastructure problem or was the failed node just busy? To know the exact reason of the node eviction, you will review the clusterware alert log file, which has details of why the failed node was evicted and which process caused the failed node to be evicted from the cluster. Here is an example from the clusterware alert log of a failed cluster node:

```
reboot advisory message from host: racnode01, component: cssagent,
with timestamp: L-2017-11-05-10:03:25.340
reboot advisory message text: Rebooting after limit 28500 exceeded;
disk timeout 27630, network timeout 28500,
last heartbeat from CSSD at epoch seconds 1241543005.340,
4294967295  milliseconds ago based on invariant clock value of 93235653
```

As you can see in this example, the cssdagent caused the failed node to reboot, so now you can look into the diagnostic log file of the cssdagent process to get more details on why cssdagent failed and rebooted the node. The cssdagent process is responsible for spawning and monitoring the OCSSD process, and cssdagent can cause a cluster node to be evicted if the cluster node is very busy and the OCSSD process cannot be scheduled on the failed cluster node.

Oracle evicts a cluster node as a last resort because it will first try killing the processes on the failed node that are capable of performing I/O, and then it will try stopping the clusterware on the failed node rather than simply rebooting the failed node.

Here is an example of the clusterware alter log from Oracle Grid Infrastructure showing how Oracle Clusterware is restarting the CSSD process on the problematic cluster node rather than rebooting the node:

https://t.me/learningnets

Technet24

```
CRS-1652:Starting clean up of CRSD resources.
...
CRS-1654:Clean up of CRSD resources finished successfully.
CRS-1655:CSSD on node racnode02 detected a problem and started to shutdown.
...
CRS-1713:CSSD daemon is started in clustered mode
```

Rebootless Node Fencing

In Oracle Grid Infrastructure 11.2.0.2, if one of the cluster nodes hangs or the cluster node does not perform the network or disk heartbeat on time, Oracle first tries killing the processes responsible for I/O operations on the failed node, such as the database writer, archiver, and so on. If Oracle succeeds in killing the processes responsible for performing I/O, it updates the ohasdrun cluster controlfile stored in the /etc/oracle/scls_scr/<hostname>/root directory and uses the restart flag; subsequently OHASD restarts Oracle Clusterware on the failed node.

Regardless of whether Oracle Clusterware reboots or just restarts the clusterware, the Oracle Clusterware alert log will record the important messages about the possible reason behind the Oracle Clusterware restart or node reboot. If the node is not able to complete the network heartbeat (NHB) on time, you must check the private network between the cluster nodes because sending and receiving packets over the private network performs the network heartbeat.

Oracle starts writing informational messages into the Clusterware alert log if it is not able to complete the network heartbeat, even after waiting for half of the time period specified by the **miscount** parameter of Oracle Clusterware—which is, by default, set to 30 seconds and must not be changed without consulting Oracle Support. You may observe the following messages in the Oracle Clusterware alert log when Oracle is not able to complete the network heartbeat:

```
Network communication with node rac1 (1) missing for 50% of timeout interval.
Removal of this node from cluster in 14.920 seconds
...
Network communication with node rac1 (1) missing for 90% of timeout interval.
Removal of this node from cluster in 2.900 seconds
This node is unable to communicate with other nodes in the cluster
and is going down to preserve cluster integrity
```

Oracle has continuously provided toolsets to proactively collect the diagnostics information to troubleshoot node evictions, because in the absence of historical diagnostics information, it is impossible to identify the cause of a node eviction. The Cluster Health Monitor collects diagnostics information to perform root cause analysis in such situations.

https://t.me/learningnets

Cluster Health Monitor

The Cluster Health Monitor (CHM) is an updated version of the Instantaneous Problem Detector for Clusters, or IPD/OS. Cluster Health Monitor is designed to proactively collect the diagnostics information for detecting and analyzing various problems in Oracle Grid Infrastructure, including node evictions. The Cluster Health Monitor continuously tracks the OS resource consumption at each node, process, and device level. You can configure the Cluster Health Monitor for real-time monitoring, and when thresholds are hit, an alert can be sent to the operator. For root-cause analysis, historical data can be replayed to understand what was happening at the time of failure.

We suggest using a divide-and-conquer policy for debugging node eviction issues because there could be various reasons for OS resource contention—from a faulty network to clock synchronization. You should analyze these possible reasons carefully, one by one. Cluster Health Monitor can be useful to provide the historical data from the time when a particular node was evicted. As mentioned, there could be various reasons for a node eviction, but the following items could provide a quick win:

- Check the clock synchronization between cluster nodes. Ensure that Network Time Protocol (NTP) is configured correctly on all cluster nodes.
- Check the network connectivity between cluster nodes. Use the following command to test the network connectivity between the cluster nodes on the public and private interface:

  ```
  ping -s <size of MTU> -c 10 -l <Public IP of local node>
  <Public IP of another cluster node>
  ping -s <size of MTU> -c 10 -l <Private IP of local node>
  <Private IP of another cluster node>
  ```

  If the cluster node is evicted because it could not access more than half of the number of voting disks or create and complete the disk heartbeat in time, then in a two-node cluster you may see the following messages in the Oracle Clusterware alert log file, which show that the cluster node is having some problems accessing the shared storage:

  ```
  No I/O has completed after 90% of the maximum interval.
  Voting file ORCL:OCR_VOTE2 will be considered not functional in 19750
  milliseconds
  ```

  The number of voting files available (1) is less than the minimum number of voting files required (2), resulting in CSSD termination to ensure data integrity.
- Analyze the output of the preceding test and ensure 100 percent of the packets are being received for the public and private interface for all cluster nodes.
- Check and ensure that all cluster nodes can access the shared storage where the OCR and voting disks are stored.

https://t.me/learningnets

Technet24

- Check that enough OS resources (such as CPU and memory) are available even when there is peak activity in the Oracle RAC database. Cluster Health Monitor will be helpful in analyzing the historical CPU data.

To get into deeper levels of the node-eviction process, you need to understand the basics of node membership and Instance Membership Recovery (IMR), also referred to as *Instance Membership Reconfiguration*.

Instance Membership Recovery

Instance Membership Recovery (IMR) is the basic framework that handles the split-brain resolution in the earlier versions, where the cluster foundation was provided by vendor clusterware. The functionality is still in the database kernel but is handled outside the RDBMS layer because the Oracle Grid Infrastructure stack is more tightly integrated with the hardware layer.

When a communication failure occurs between the instances, or when an instance is not able to issue the heartbeat information to the controlfile, the cluster group may be in danger of possible data corruption. In addition, when no mechanism is present to detect the failures, the entire cluster will hang. IMR removes the failed instance from the cluster group. When a subset of a cluster group survives during failures, IMR ensures that the larger partition group survives and kills all other smaller groups.

IMR is a part of the service offered by Cluster Group Services (CGS). LMON is the key process that handles many of the CGS functionalities. As you know, cluster software (known as Cluster Manager, or CM) can be a vendor-provided or Oracle-provided infrastructure tool. CM facilitates communication between all nodes of the cluster and provides information on the health of each node—the node state. It detects failures and manages the basic membership of nodes in the cluster. CM works at the cluster level and not at the database or instance level.

Inside Oracle RAC, the Node Monitor (NM) provides information about nodes and their health by registering and communicating with the CM. NM services are provided by LMON. Node membership is represented as a bitmap in the Global Resource Directory (GRD). A value of 0 denotes that a node is down, and a value of 1 denotes that the node is up. There is no value to indicate a “transition” period such as during bootup or shutdown. LMON uses the global notification mechanism to let others know of a change in the node membership. Every time a node joins or leaves a cluster, this bitmap in the GRD has to be rebuilt and communicated to all registered members in the cluster.

Node membership registration and deregistration is done in a series of synchronized steps—a topic beyond the scope of this chapter. Basically, cluster members register and deregister from a group. The important thing to remember is that NM always communicates with the other instances in the cluster about their health and status using the CM. In contrast, if LMON needs to send a message to LMON on another instance, it can do so directly without the help or involvement of CM. It is important to differentiate between cluster communication and Oracle RAC communication.

https://t.me/learningnets

A simple extract from the alert log file about member registration is provided here:

```
alter database mount
…
lmon registered with NM - instance id 1 (internal mem no 0)
…
Reconfiguration started
List of nodes: 0,
 Global Resource Directory frozen
```

Here you can see that this instance was the first to start up and that LMON registered itself with the NM interface, which is a part of the Oracle kernel.

When an instance joins or leaves the cluster, the LMON trace of another instance shows the reconfiguration of the GRD:

```
kjxgmpoll reconfig bitmap: 0 1 3
*** kjxgmrcfg: Reconfiguration started, reason 1
```

You may find these lines together with other lines asking SMON to perform instance recovery, as shown next. This happens when any instance crash occurs or when an instance departs the cluster without deregistering in a normal fashion.

```
Post SMON to start 1st pass IR
*** kjxgmpoll reconfig bitmap: 0 1 3
*** kjxgmrcfg: Reconfiguration started, reason 1
kjxgmcs: Setting state to 2 0.
***   Name Service frozen
```

The CGS is present primarily to provide a coherent and consistent view of the cluster from an OS perspective. It tells Oracle the number of nodes that are in the cluster. It is designed to provide a synchronized view of the cluster instance membership. Its main responsibility involves regular status checks of the members and measuring whether they are valid in the group—and very importantly, it detects split-brain scenarios in case of communication failures.

Specific rules bind together members within the cluster group, which keeps the cluster in a consistent state:

- Each member should be able to communicate without any problems with any other registered and valid member in the group.
- Members should see all other registered members in the cluster as valid and have a consistent view.
- All members must be able to read from and write to the controlfile.

So, when a communication failure occurs between the instances, or when an instance is not able to issue the heartbeat information to the voting disk, IMR is triggered. Other than IMR, there is no mechanism to detect failures that could result in

https://t.me/learningnets

Technet24

a hang of the entire cluster.

Member Voting

The CGS is responsible for checking whether members are valid. To determine periodically whether all members are alive, a voting mechanism is used to check the validity of each member. All members in the database group vote by providing details of what they presume the instance membership bitmap looks like. As mentioned, the bitmap is stored in the GRD. A predetermined master member tallies the status flag and communicates to the respective processes that the voting is done; then it waits for registration by all the members that have received the reconfigured bitmap.

How Voting Happens

The Checkpoint (CKPT) process updates the controlfile every three seconds in an operation known as the *heartbeat*. CKPT writes into a single block that is unique for each instance; therefore, intra-instance coordination is not required. This block is called the *checkpoint progress record*.

All members attempt to obtain a lock on a controlfile record (the *result record*) for updating. The instance that obtains the lock tallies the votes from all members. The group membership must conform to the *decided (voted)* membership before allowing the GCS/GES reconfiguration to proceed. The *controlfile vote result record* is stored in the same block as the heartbeat in the controlfile checkpoint progress record.

In this scenario of dead instances and member evictions, a potentially disastrous situation could arise if pending I/O from a dead instance is to be flushed to the I/O subsystem. This could lead to potential data corruption. I/O from an abnormally departing instance cannot be flushed to disk if database integrity is to be maintained. To "shield" the database from data corruption in this situation, a technique called *I/O fencing* is used. I/O fencing implementation is a function of CM and depends on the clusterware vendor.

I/O fencing is designed to guarantee data integrity in the case of faulty cluster communications causing a split-brain condition, which occurs when cluster nodes hang or node interconnects fail, and as a result, the nodes lose the communication link between them and the cluster. Split-brain is a problem in any clustered environment and is a symptom of clustering solutions—and not just for Oracle RAC. Split-brain conditions can cause database corruption when nodes become uncoordinated in their access to the shared datafiles.

For a two-node cluster, split-brain occurs when nodes in a cluster cannot talk to each other (the internode links fail), and each node assumes it is the only surviving member of the cluster. If the nodes in the cluster have uncoordinated access to the shared storage area, they can end up overwriting each other's data, causing data corruption because each node assumes ownership of shared data. To prevent data corruption, one node must be asked to leave the cluster or should be forced out immediately. This is where IMR comes in, as explained earlier in the chapter.

Many internal (hidden) parameters control IMR and determine when it should start.

https://t.me/learningnets

If a vendor clusterware is used, split-brain resolution is left to the vendor software, and Oracle would have to wait (10 minutes) for the clusterware to provide a consistent view of the cluster and resolve the split-brain issue. This can potentially cause a delay (and a hang in the whole cluster) because each node can potentially think it is the master and try to own all the shared resources. Still, Oracle relies on the clusterware for resolving these challenging issues.

Note that Oracle does not wait indefinitely for the clusterware to resolve a split-brain issue, and a timer is used to trigger an IMR-based node eviction. These internal timers are also controlled using hidden parameters. The default values of these hidden parameters are not to be touched because that can cause severe performance or operational issues with the cluster.

An obvious question that comes to mind is, why does a split-brain condition take place? You might be thinking, "Why does Oracle say a link is down, when my communication engineer says it's fine?" This question is easier asked than answered because the underlying hardware and software layers that support Oracle RAC are too complex, and it can be a nightmare trying to figure out why a cluster broke in two when things seemed to be normal. Usually, the configuration of communication links, temporary network glitches, and bugs in the clusterware can cause these issues.

As mentioned time and again, Oracle completely relies on the cluster software to provide cluster services, and if something is awry, Oracle, in its overzealous quest to protect data integrity, evicts nodes or aborts an instance and then assumes that something is wrong with the cluster.

Cluster reconfiguration is initiated when NM indicates a change in the database group or when IMR detects a problem. Reconfiguration is initially managed by the CGS, and after this is completed, IDLM (GES/GCS) reconfiguration starts.

Cluster Reconfiguration Steps

The cluster reconfiguration process triggers IMR, and a seven-step process ensures complete reconfiguration:

1. The name service is frozen. The CGS contains an internal database of all the members/instances in the cluster with all their configuration and servicing details. The name service provides a mechanism to address this configuration data in a structured and synchronized manner.
2. The Integrated Distributed Lock Manager (IDLM) is frozen. The lock database is frozen to prevent processes from obtaining locks on resources that were mastered by the departing/dead instance.
3. Membership, validation, and IMR are determined.
4. The bitmap is rebuilt, including instance name and uniqueness verification. CGS must synchronize the cluster to be sure that all members get the reconfiguration event and that they all see the same bitmap.
5. All dead instance entries are deleted and all newly configured names are republished.

https://t.me/learningnets

Technet24

6. The name service is unfrozen and released for use.
7. Reconfiguration is handed back over to GES/GCS.

Now that you know when IMR starts and node evictions take place, let's look at the corresponding messages in the alert log and LMON trace files to get a better picture. (The logs have been edited for brevity. Note that all the lines in boldface define the most important steps in IMR and the handoff to other recovery steps in CGS.)

Problem with a Node For this example, assume a four-node cluster (instances A, B, C, and D) in which instance C has a problem communicating with the other nodes because its private link is down. All other services on this node are assumed to be working normally.

Here's the relevant portion from the alter log on instance C:

```
ORA-29740: evicted by member 2, group incarnation 6
…
LMON: terminating instance due to error 29740
Instance terminated by LMON, pid = 692304
...
```

Here's the relevant portion from the alter log on instance A:

```
…
Communications reconfiguration: instance 2
Evicting instance 3 from cluster
…
Trace dumping is performing id=[50630091559]

…Waiting for instances to leave:
3
…Waiting for instances to leave:
3
…
Reconfiguration started
List of nodes: 0,1,3,
 Global Resource Directory frozen
 Communication channels reestablished
 Master broadcasted resource hash value bitmaps
…Reconfiguration started
```

Here's the relevant portion from the LMON trace file on instance A:

https://t.me/learningnets

```
***
kjxgrgetresults: Detect reconfig from 1, seq 12, reason 3
kjxgfipccb: msg 0x1113dcfa8, mbo 0x1113dcfa0, type 22, ack 0, ref 0,
stat 3
kjxgfipccb: Send timed out, stat 3 inst 2, type 22, tkt (10496,1496)
***
kjxgrcomerr: Communications reconfig: instance 2 (12,4)
Submitting asynchronized dump request [2]
kjxgfipccb: msg 0x1113d9498, mbo 0x1113d9490, type 22, ack 0, ref 0,
stat 6
kjxgfipccb: Send cancelled, stat 6 inst 2, type 22, tkt (10168,1496)
kjxgfipccb: msg 0x1113e54a8, mbo 0x1113e54a0, type 22, ack 0, ref 0,
stat 6
kjxgfipccb: Send cancelled, stat 6 inst 2, type 22, tkt (9840,1496)
```

Note that “Send timed out, stat 3 inst 2” is LMON trying to send messages to the broken instance.

Following is more content from the LMON trace file from instance A:

```
kjxgrrcfgchk: Initiating reconfig, reason 3 /* IMR Initiated */
***
kjxgmrcfg: Reconfiguration started, reason 3
kjxgmcs: Setting state to 12 0.
***

kjxgmcs: Setting state to 12 1.
***
Voting results, upd 1, seq 13, bitmap: 0 1 3
```

Note that instance A has not tallied the vote; hence, it has received only the voting results.

Here is an extract from the LMON trace file on instance B, which managed to tally the vote:

```
Obtained RR update lock for sequence 13, RR seq 13
*** Voting results, upd 0, seq 13, bitmap: 0 1 3
...
...
```

Here’s the LMON trace file on instance A:

```
Evicting mem 2, stat 0x0007 err 0x0002
kjxgmps: proposing substate 2
kjxgmcs: Setting state to 13 2.
Performed the unique instance identification check
kjxgmps: proposing substate 3
```

https://t.me/learningnets

Technet24

```
kjxgmcs: Setting state to 13 3.
  Name Service recovery started
  Deleted all dead-instance name entries
kjxgmps: proposing substate 4
kjxgmcs: Setting state to 13 4.
  Multicasted all local name entries for publish
  Replayed all pending requests
kjxgmps: proposing substate 5
kjxgmcs: Setting state to 13 5.
  Name Service normal
  Name Service recovery done
***
More content from the LMON trace file of instance A, showing that the Name Service is frozen:
kjxgmrcfg: Reconfiguration started, reason 1
kjxgmcs: Setting state to 13 0.
***  Name Service frozen
kjxgmcs: Setting state to 13 1.
```

GES/GCS recovery starts here:

```
Global Resource Directory frozen
node 0
node 1
node 3
res_master_weight for node 0 is 632960
res_master_weight for node 1 is 632960
res_master_weight for node 3 is 632960
...
```

Death of a Member Node For the same four-node cluster (A, B, C, and D), instance C has died unexpectedly:

https://t.me/learningnets

```
kjxgrnbrisalive: (3, 4) not beating, HB: 561027672, 561027672
***
kjxgrnbrdead: Detected death of 3, initiating reconfig
kjxgrrcfgchk: Initiating reconfig, reason 2
***
kjxgmrcfg: Reconfiguration started, reason 2
kjxgmcs: Setting state to 6 0.
***
  Name Service frozen
kjxgmcs: Setting state to 6 1.
***
Obtained RR update lock for sequence 6, RR seq 6
***
Voting results, upd 0, seq 7, bitmap: 0 2
Evicting mem 3, stat 0x0007 err 0x0001
kjxgmps: proposing substate 2
kjxgmcs: Setting state to 7 2.
  Performed the unique instance identification check
kjxgmps: proposing substate 3
kjxgmcs: Setting state to 7 3.
  Name Service recovery started
  Deleted all dead-instance name entries
kjxgmps: proposing substate 4
kjxgmps: proposing substate 4
kjxgmcs: Setting state to 7 4.
  Multicasted all local name entries for publish
  Replayed all pending requests
kjxgmps: proposing substate 5
kjxgmcs: Setting state to 7 5.
  Name Service normal
  Name Service recovery done
***
kjxgmps: proposing substate 6
...
...
...
kjxgmps: proposing substate 2
```

GES/GCS recovery starts here:

https://t.me/learningnets

```
Global Resource Directory frozen
node 0
node 2
res_master_weight for node 0 is 632960
res_master_weight for node 2 is 632960
 Total master weight = 1265920
 Dead  inst 3
Join  inst
 Exist inst 0 2
...
...
```

Advanced Debugging for Oracle Clusterware Modules

By default, Oracle records minimal debugging information in the diagnostic log files. There are situations where you require detailed information to diagnose the potential problems in Oracle Clusterware. You can instruct Oracle Clusterware to record more debugging information for different Oracle Clusterware processes and cluster resources. The CRSCTL utility is used to set the advanced debugging in Oracle Clusterware.

Oracle Clusterware services such as Cluster Synchronization Services, Cluster Ready Services, and so on, have multiple modules or functions in them for which advance debugging can be enabled to a debug level between 0 and 5. Here is an example of identifying the Oracle Clusterware services and modules on which advance debugging can be enabled.

Run the following command to display the services:

```
# ./crsctl get log -h
Usage:
  crsctl get {log|trace} {mdns|gpnp|css|crf|crs|ctss|evm|gipc} "<name1>,..."
 where
  mdns          multicast Domain Name Server
  gpnp          Grid Plug-n-Play Service
```

https://t.me/learningnets

```
  css            Cluster Synchronization Services
  crf            Cluster Health Monitor
  crs            Cluster Ready Services
  ctss           Cluster Time Synchronization Service
  evm            EventManager
  gipc           Grid Interprocess Communications
  <name1>, ...    Module names ("all" for all names)

 crsctl get log res <resname>
where
  <resname>      Resource name
```

The following command displays modules of Cluster Synchronization Services along with the current debug levels:

```
# ./crsctl get log css all
Get CSSD Module: CLSF  Log Level: 0
Get CSSD Module: CSSD  Log Level: 2
Get CSSD Module: GIPCCM  Log Level: 2
Get CSSD Module: GIPCGM  Log Level: 2
Get CSSD Module: GIPCNM  Log Level: 2
Get CSSD Module: GPNP  Log Level: 1
Get CSSD Module: OLR  Log Level: 0
Get CSSD Module: SKGFD  Log Level: 0
```

As shown here, Cluster Synchronization Services has eight different modules. Because setting advance debugging writes a lot of extra diagnostic information in the diagnostic file, so you must be very careful while setting advance debugging because it may adversely impact the performance of your Oracle cluster. You should only enable advance debugging on the required module rather than setting it on the whole service. For example, if you see some errors related to the Oracle Local Registry in the Oracle Clusterware alert log and you want to know more information about what's happening during the failure, you should enable advance debugging for the OLR module of the Cluster Synchronization Service, as shown here:

```
# ./crsctl set log css OLR:5
Set CSSD Module: OLR  Log Level: 5
```

Similarly, you can enable advance debugging for Oracle Cluster Resources as well. Suppose you are unable to start a virtual IP resource; it would not make sense to enable advance debugging for the whole of CRS. Rather, it is more efficient to just enable advance debugging for that failing cluster resource. Here is an example of enabling advance debugging for Oracle network resource ora.backup.vip:

```
#./crsctl set log res ora.backup.vip:3
Set Resource ora.alpha1.vip Log Level: 3
```

https://t.me/learningnets

Technet24

You can enable advanced debugging from level 0 to 5, where a higher level records more diagnostic information, with the possibility that the highest debug level may reveal some source code snippets, which must be analyzed by Oracle product support because end users may get confused by such raw diagnostic information. When advanced debugging is enabled, Oracle writes lots of diagnostic information and wraps the diagnostic files. Therefore, there's a chance that the interesting diagnostic file is wrapped. To avoid losing the required diagnostic data, it is recommended that you create a custom script to move the diagnostic files to a safer location before Oracle wraps them. Here is a sample script that can be executed on the cluster node to move log files of the CRSD daemon:

```
# Shell script to move diagnostic files of the CRSD daemon
# when advance debugging is enabled.
# Change the GRID_HOME variable to specify your Oracle Clusterware Home.
GRID_HOME=/u02/12.2.0/grid
# Specify the HOST variable to define your cluster node name.
# This should not contain domain name in it.
HOST=<add your cluster node name here>
# Change the DELAY variable to specify frequency time in seconds,
# this script will move the logfiles.
# By default script will move logfiles every 200 seconds.
DELAY=200
CRSDLOGFIR=$GRID_HOME/log/$HOST/crsd
while [ 2 -ne 3 ]; do
 COMPRESSFILE = $GRID_HOME/log/$HOST/`date +m%d%y%"-"%H%M`.tar
 tar -cf $COMPRESSFILE $CRSDLOGFIR/*
 sleep $DELAY
done
exit
```

Debugging Various Utilities in Oracle RAC

You can use the special environment variable SRVM_TRACE to trace the SRVM files and various Java-based utilities, such as DBCA, ASMCA, CLUVFY, DBUA, and SRVCTL. Oracle displays trace data on the standard output while tracing the SRVCTL utility, whereas trace files for others are written into the file system.

By default, tracing is enabled for the Database Configuration and Database Upgrade Assistants, and their trace files are located under $ORACLE_HOME/cfgtools/dbca and $ORACLE_HOME/cfgtools/dbua, respectively. The trace file for CLUVFY is stored under the $GRID_HOME/cv/log directory. There are situations where you enable this trace file for debugging. For example, if you are unable to start a database or database instance using the SRVCTL utility and the standard error reported by the SRVCTL utility is not providing enough information, setting the SRVM_TRACE environment variable to TRUE provides more information that is helpful in debugging the problem. Here is an example of enabling a Java trace for the SRVCTL utility,

https://t.me/learningnets

which is unable to start a single-instance database called SINGLEDB in the cluster:

```
$ srvctl start database id singledb
PRCR-1013 : Failed to start resource ora.singledb.db
PRCR-1064 : Failed to start resource ora.singledb.db on node racnode02
CRS-2674: Start of 'ora.singledb.db' on 'racnode02' failed
```

Setting the SRVM_TRACE environment variable to TRUE reveals more information, and it is found that the OCR is corrupt because it is showing SINGLEDB as a cluster database and reporting a database resource placement error as well:

```
…
[CRSNative.searchEntities:857]  found 1 entities
…
 [DatabaseImpl.getDBInstances:1185]
riID=ora.singledb.db 1 1 result={CLUSTER_DATABASE=true, LAST_SERVER=racnode02,
USR_ORA_INST_NAME=SINGLEDB, USR_ORA_INST_NAME@SERVERNAME(racnode02)=SINGLEDB}
[DatabaseImpl.getDBInstances:1322]  Instances: <SINGLEDB,racnode02>
[StartAction.executeDatabase:220]  starting db resource
…
[CRSNative.internalStartResource:339]  About to start resource:
Name: ora.singledb.db, node: null, options: 0, filter null
 [CRSNativeResult.addLine:106]  callback:
ora.oemrolt2.db false CRS-2672: Attempting to start 'ora.singledb.db' on 'racnode02'
…
[CRSNativeResult.addLine:106]  callback:
ora.oemrolt2.db true CRS-2674: Start of 'ora.singledb.db' on 'racnode02' failed
…
[CRSNativeResult.addLine:106]  callback:
ora.oemrolt2.db true CRS-2632: There are no more servers to try to place resource
'ora.singledb.db' on that

would satisfy its placement policy
[CRSNativeResult.addComp:162]  add comp: name
ora.singledb.db, rc 223, msg CRS-0223: Resource 'ora.singledb.db' has placement error.
…
[CRSNative.internalStartResource:352]  Failed to start resource:
Name: ora.singledb.db, node: null, filter: null, msg CRS-2674: Start of
'ora.singledb.db'

on 'racnode02' failed
CRS-2632: There are no more servers to try to place resource
'ora.singledb.db' on that would satisfy its placement policy
PRCR-1079 : Failed to start resource ora.singledb.db
CRS-2674: Start of 'ora.singledb.db' on 'racnode02' failed
CRS-2632: There are no more servers to try to place resource
'ora.singledb.db' on that would satisfy its placement policy
```

The issue is resolved after re-registering the database SINGLEDB in the OCR using the SRVCTL utility.

https://t.me/learningnets

Technet24

Using ORAchk to Troubleshoot RAC

Oracle offers a powerful and effective troubleshooting tool named ORAchk that replaces the earlier well-known tool RACcheck. The best of Oracle RAC professionals have poured thousands of hours and great effort into fine-tuning this tool over the years. Using ORAchk, you can proactively search for potential problems, not only in an Oracle RAC database but also in an Oracle E-Business Suite installation and in Enterprise Manager Grid Control (12*c*).

ORAchk offers the following features to help you troubleshoot your RAC installations:

- Proactive scanning of the critical problems across all the layers of the stack
- Simplifying and prioritizing the known problems in the stack
- Doesn't require you upload data to Oracle Support
- Provides reports that let you investigate specific problems and speed up their resolution
- Provides a dashboard of diagnostic data collection across the enterprise, with the help of the Collection Manager, a web application that comes with the tool

You can conduct an ORAchk health assessment for the Oracle database servers, the Grid Infrastructure, hardware, operating systems, as well as RAC software. Using ORAchk 2.2.0, you can also check the health of single-instance databases as well as RAC one-node configurations. Oracle Support's document titled "Health Checks for the Oracle Stack" (Doc ID 1268927.2) shows how to use the ORAchk tool.

Summary

This chapter provides a solid foundation for troubleshooting Oracle Grid Infrastructure —from troubleshooting a failed install to troubleshooting node evictions. Like Oracle RDBMS, Oracle Grid Infrastructure comes with a detailed diagnostic framework that helps you to quickly identify the root cause of any failures. Additional logging can be enabled whenever required, and this chapter covered the methods to enable further diagnostic logging.

Oracle Grid Infrastructure evicts the node in question when there is a communication failure either between the nodes or I/O path or between the nodes to the voting files. A node eviction or reboot is a mandatory evil to protect data integrity because whenever there is a question between database availability and avoiding database corruption, the vote goes to avoiding database corruption, which in turn triggers the node eviction.

In this chapter, you saw the basic and advanced methods of collecting diagnostic information when a hang situation occurs. You also studied the steps in node reconfiguration and IMR internals. Most of the complex problems may require assistance from Oracle Support, and this chapter will help you when dealing with

https://t.me/learningnets

Oracle Support personnel.

Finally, this chapter covered Oracle's ORAchk tool, which enables you to collect RAC diagnostic data unobtrusively with low overhead, to help you troubleshoot your RAC environment.

https://t.me/learningnets

Technet24

PART V

Deploying Oracle RAC

https://t.me/learningnets

CHAPTER 15

Extending Oracle RAC for Maximum Availability

Oracle Real Application Clusters is a popular solution for scalability across enterprises. Oracle designed Oracle RAC primarily as a scalability and availability solution that resides in a single datacenter. However, it's possible to build and deploy an Oracle RAC system where the nodes in the cluster are separated by great distances. Having nodes in separate locations provides continuous data availability and limited disaster recovery in case of site failures.

The choice of extending Oracle RAC depends on the failure it's designed to address. To protect against multiple points of failure, a cluster must be geographically dispersed: nodes can be put in different rooms, on different floors of a building, or even in separate buildings or separate cities. Your decision as to the distance between the nodes rests on the types of disasters from which you need protection and the technology used to replicate data between the storage systems.

Modern datacenter clusters are built around high-performance storage area networks (SANs) that provide secure, reliable, and scalable data storage facilities. An effective business continuance and disaster recovery plan mandates the deployment of multiple datacenters located at optimal distances to protect against regional power

https://t.me/learningnets

Technet24

failures or disasters, yet close enough for synchronous data replication without affecting the application performance. Achieving this balance for successful business continuance poses a significant challenge.

The time within which a business needs to recover during a disaster determines the type of data replication you need. Synchronous data replication provides the least amount of downtime but requires the datacenters to be close enough so that the application performance is not affected by the latencies introduced for every I/O operation. Asynchronous or semi-synchronous replication allows for greater distance, but the datacenters are in lockstep with each other, with the secondary datacenter lagging behind the primary by a fixed amount of time. This, in turn, implies that a loss of data occurs for that period of time. An optimal solution is to increase the distance between the datacenters without introducing additional latency for I/O operations.

For example, if a business has a corporate campus, the individual Oracle RAC nodes could be placed in separate buildings. This configuration provides a degree of disaster tolerance in addition to the normal Oracle RAC high availability, because a fire in one building would not bring the database down. Extended Oracle RAC, if properly set up, provides continuous data availability even during regional disasters. Figure 15-1 shows the basic architecture of an extended Oracle RAC configuration. The components used in Extended Clusters are discussed later in the chapter.

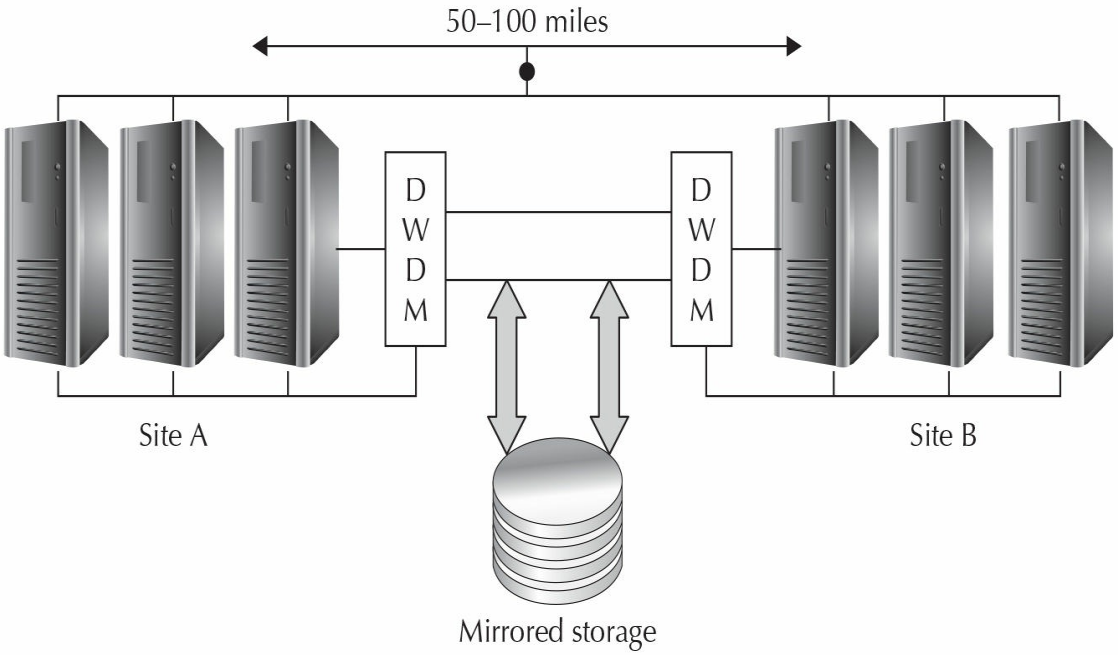


FIGURE 15-1. *Extended Oracle RAC architecture*

Implementing Oracle RAC on an extended cluster provides double advantages. Being able to distribute any and all work across all nodes, including running as a single workload across the whole cluster, allows for the greatest flexibility in usage of resources. Should one site fail—for example, because of a fire at a site—all work can be routed to the remaining site, which can rapidly (in less than a minute) take over the

https://t.me/learningnets

processing.

However, various other Oracle data replication technologies (such as Oracle Data Guard, Oracle Streams, and Oracle GoldenGate) work in tandem with Oracle RAC to build a *maximum availability architecture*. They require potential changes at the application level, and involve additional layers of software components to work synchronously. Moreover, they work on a primary/secondary (or primary/standby) method, which involves complex failover and failback procedures.

Extending Oracle RAC to maximum availability works transparently with the existing application because networking complexities are hidden in the Oracle RAC layers and because clients continue to work with available services defined in the Oracle RAC as if they were working in the regular datacenter instances.

Benefits of Extended RAC Clusters

Other than providing the usual maximum availability during the most common failures, an extended Oracle RAC configuration allows these two important benefits:

- Full utilization of resources
- Extremely rapid recovery

Full Utilization of Resources

In the extended Oracle RAC configuration, all cluster members are available for processing all the time—unlike in usual primary/standby configurations, where only one half of resources is used and the other half is kept idle or near idle, expecting a disaster at the primary. This provides a tremendous advantage for customers because the resources are fully utilized, and it allows greater flexibility of scheduling the resources across the cluster nodes.

Extremely Rapid Recovery

The key advantage of extended Oracle RAC over any other failover technologies is the ability to provide extremely rapid recovery. The mean time to recover (MTTR—in simple terms, the recovery time during failures) for most commonly available failover technologies is between just a few minutes to hours. However, in extended Oracle RAC, should there be any disaster in one datacenter, the existing and incoming workload can be routed to the other node in a rapid manner—in most cases, within seconds.

If the application layer is aware of Oracle RAC in terms of multiple addresses or is configured for Transparent Application Failovers (TAFs), the failover is totally transparent to the end users. No other high availability or disaster recovery technologies currently on the market provide this amount of flexibility.

https://t.me/learningnets

Technet24

Design Considerations

Before going into the design and implementation of extended clusters, let's take a look at the key points to consider when designing a solution that involves "geo-clusters." Network infrastructure plays a major role in extending Oracle RAC, because performance is greatly impacted by round-trip latency. An optimal solution is to increase the distance between the datacenters without introducing additional latency for I/O operations.

Speed of Light

In a vacuum, light travels at 186,282 miles per second. To make the math simpler, we can round that up to 200,000 miles per second, or 200 miles per millisecond. In any computer communications, we require confirmation for any message, so we must use round-trip distances for our calculations. Light can travel up to 100 miles away and back in 1 millisecond (ms), but the actual transit time is longer. This is in part because the speed of light is slowed by about 30 percent through an optical fiber and because straight lines rarely occur in global communications situations, but also because delays are introduced when the signal passes through an electronic switch or signal regenerator.

In considering all the factors, such as the inevitable switch delays, the conservative rule of thumb is that it takes 1 millisecond for every 50 miles of distance round trip. Therefore, 500 miles adds 10 milliseconds to the latency of a disk access. Given the normal disk access latency of 8 to 10 milliseconds, this distance merely doubles the total latency; however, the shadowing software layer can cope with that. But if the latency is more than double, the software might think the disk at the other end has gone offline and will incorrectly break the shadow set. Therefore, the speed of light becomes the limiting factor due to the latency caused by the distance between the datacenters, even with the use of dark fiber.

Network Connectivity

Network interconnect has a great impact on performance in a normal Oracle RAC environment. It has even greater impact in an extended Oracle RAC environment. Planned redundancy should be introduced at every component level to avoid failures and interconnect. In addition, SAN and IP networking need to be kept on separate dedicated channels.

A scalable and reliable lightweight and protocol-independent network is required for extended Oracle RAC. Traditional networks are limited to about six miles before repeaters are needed. Repeaters and any other intermediate switches introduce inevitable delays via latency between the switches. Dark fiber networks allow the communication to occur without these repeaters across a much greater distance. Dark fiber networks with dense wavelength-division multiplexing (an optical technology used to increase bandwidth over existing fiber-optic backbones) provide low-

https://t.me/learningnets

latency/high-bandwidth communication over longer distances.

Dense Wavelength-Division Multiplexing

In optical communications, wavelength-division multiplexing (WDM) technology multiplexes multiple optical carrier signals on a single optical fiber by using different wavelengths of laser light to carry different signals. This allows for a multiplication in capacity and makes it possible to perform bidirectional communications over one strand of fiber.

WDM technology is further divided into two market segments—dense and coarse WDM. Systems with more than eight active wavelengths per fiber are generally considered dense WDM (DWDM) systems, whereas those with fewer than eight active wavelengths are classed as coarse WDM (CWDM).

CWDM and DWDM technologies are based on the same concept of using multiple wavelengths of light on a single fiber, but the two technologies differ in the spacing of the wavelengths, the number of channels, and the ability to amplify signals in the optical space.

CWDM is also used in cable television networks, where different wavelengths are used for the downstream and upstream signals. DWDM works by combining and transmitting multiple signals simultaneously at different wavelengths on the same fiber. In effect, one fiber is transformed into multiple virtual fibers that provide the extended bandwidth. With this over-bandwidth extension, single fibers have been able to transmit data at speeds up to 400 Gbps (gigabits per second).

A key advantage to DWDM is that it is protocol and bit-rate independent. DWDM-based networks can transmit data in IP, ATM, SONET/SDH, and Ethernet and can handle bit rates between 100 Mbps (megabits per second) and 2.5 Gbps. Therefore, DWDM-based networks can carry different types of traffic at different speeds over an optical channel.

Cache Fusion Performance

As you have seen, the secret recipe behind the extreme performance of Oracle Real Application Clusters is Cache Fusion. Cache Fusion is built under the assumption that the nodes are connected through the high-speed local network. With extended Oracle RAC separating the nodes by miles, a great penalty is introduced to Cache Fusion performance. The local interconnect latencies are always on the order of one to two milliseconds, and any additional latency beyond this provides a great threat to the application’s scalability.

Oracle Corporation, along with various hardware partners, tested the extended Oracle RAC setup with different configurations and observed the preceding behaviors and illustrated them in the following figure. Figure 15-2 shows the effects of distance on I/O latency for reading from a physical disk.

https://t.me/learningnets

Technet24

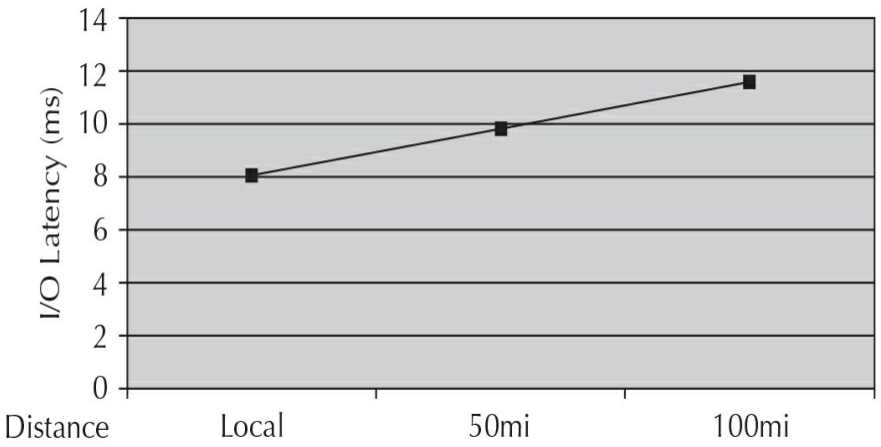


FIGURE 15-2. *Distance effect on I/O latency*

Data Storage

The next important factor in designing an extended Oracle RAC solution is the storage for the datafiles across the clusters. The storage should be concurrently available with both the instances, yet at the same time should be continuously synchronized in real time. *Synchronous replication* means that I/O is not complete until both sides acknowledge its completion.

Although it is possible to implement Oracle RAC on extended-distance clusters with storage at only one site, if the site with the storage should fail, the storage is no longer available to any surviving nodes and the whole cluster becomes unavailable. This defeats the purpose of having had Oracle RAC nodes at different locations. Hence, we mirror the storage in both the sites, and nodes use the storage at their respective sites transparently.

In addition, to maintain consistency of data, every write to the primary storage is written over to the secondary storage in the same sequence before the application writes are considered successful. This method helps ensure that data stored at both sites is always consistent; the only data that could be lost in the event of a failure is the data that was not committed or was being transmitted at the time of the failure.

Figure 15-3 shows the components involved in the physical implementation of the extended Oracle RAC cluster.

https://t.me/learningnets

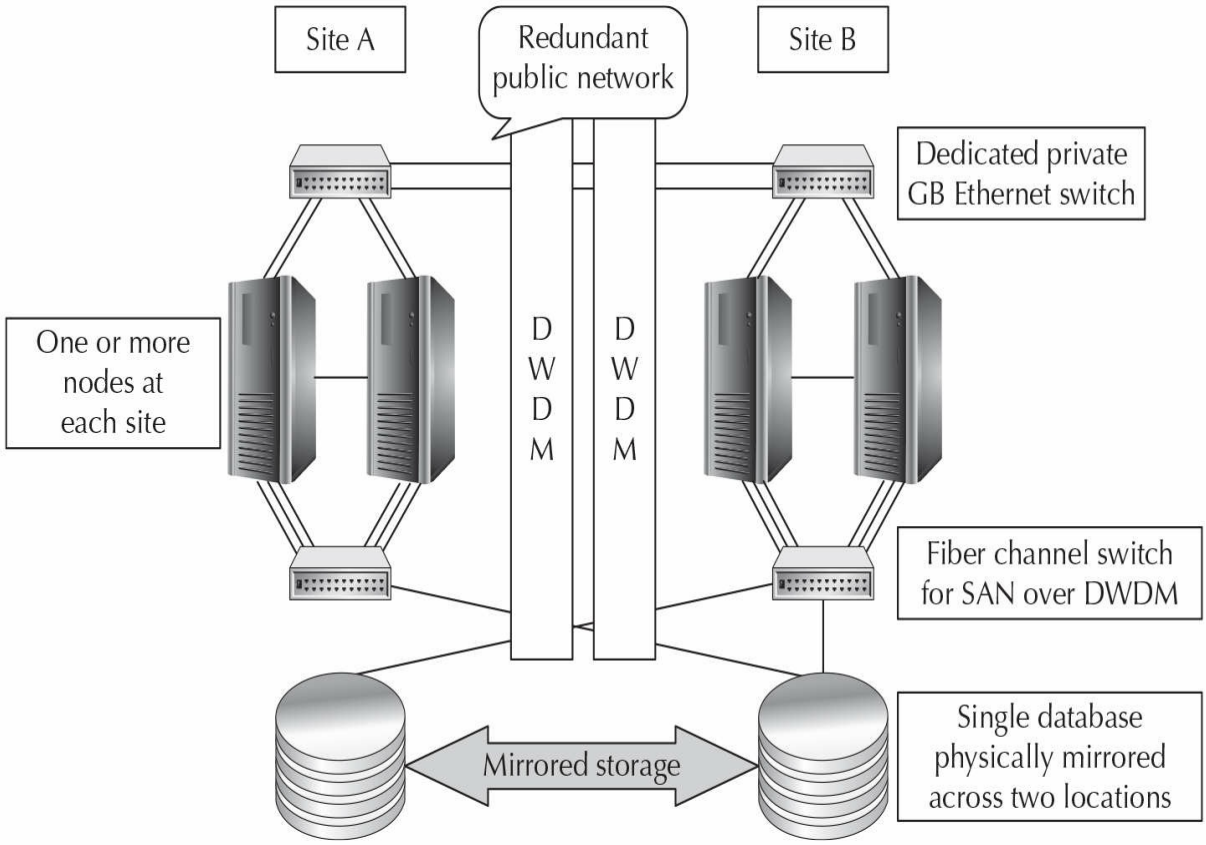


FIGURE 15-3. *Extended Oracle RAC component architecture*

Common Techniques for Data Mirroring

A couple of mirroring techniques are most commonly used in the industry. However, the choice of storage is totally dependent on the hardware—some hardware configurations support array-based mirroring and a few support host-based mirroring.

Array-Based Mirroring

In array-based mirroring, all the writes are forwarded to one site and then mirrored to the other node using the disk technologies. Array-based mirroring means having a primary/secondary site setup. Only one set of disks is always used, and all the reads and writes are serviced by that storage. If the node or site fails, all the instances will crash and need to be restarted after the secondary site is brought to life.

Array-based mirroring is quite simple to implement, because the mirroring is done by the storage system and is transparent to the database applications. If the primary site fails, all access to the primary disks is lost. An outage may be incurred before the system can switch to the secondary site. Figure 15-4 explains the architecture of array-based mirroring.

https://t.me/learningnets

Technet24

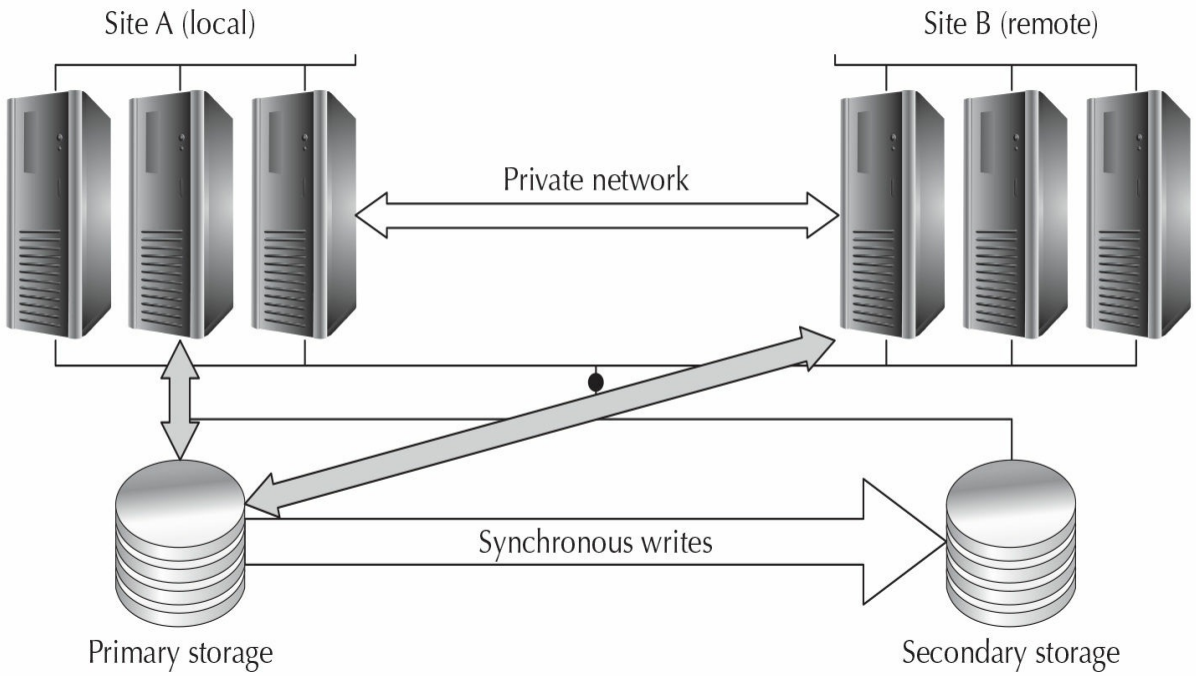
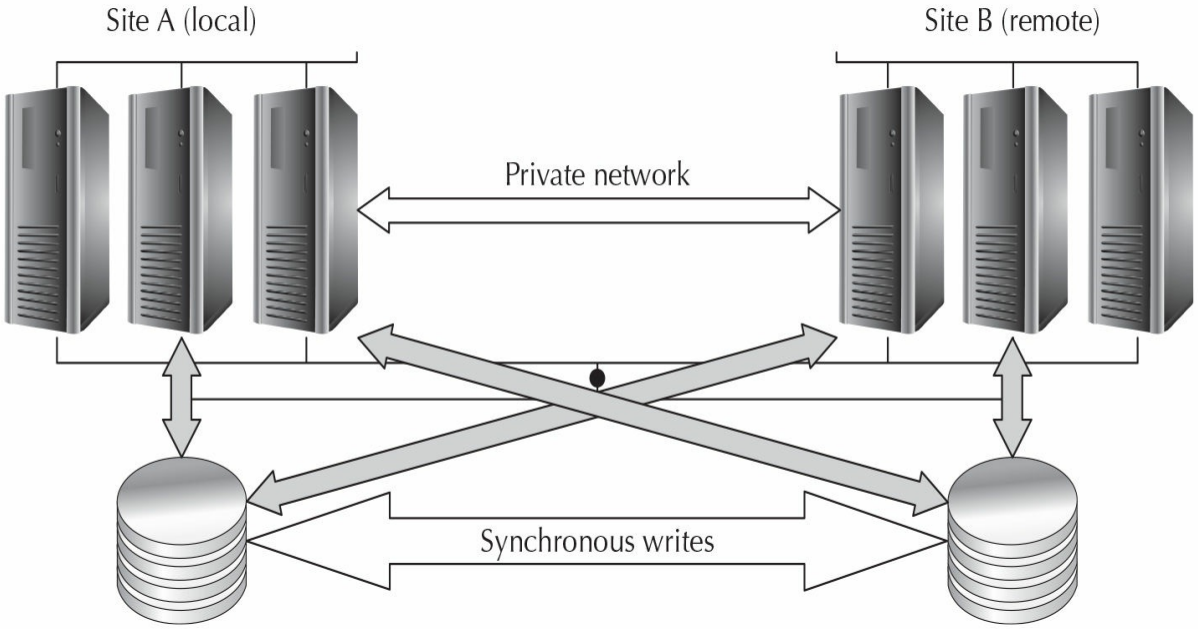


FIGURE 15-4. *Array-based mirroring*

Host-Based Mirroring

Host-based mirroring, as the name suggests, mirrors at the host level. It requires closely integrated clusterware and Logical Volume Manager (LVM). The underlying Cluster Logical Volume Manager (CLVM) mirrors the writes to both nodes synchronously. But from the database or operating system point of view, only one set of disks exists. The disk mirroring is done transparently to the database.

Figure 15-5 explains the architecture of host-based mirroring.



https://t.me/learningnets

FIGURE 15-5. *Host-based mirroring*

Table 15-1 summarizes the pros and cons of array-based mirroring and host-based mirroring.

Description	Array-Based Mirroring	Host-Based Mirroring
Write propagation	Disk writes are always sent to one site and changes are synchronously propagated to another node using disk-mirroring techniques.	Writes are always sent to both nodes. Both nodes service user requests all the time.
Primary/ secondary setup	Requires primary/secondary site setup. If the primary site fails, all access to primary disks is lost. An outage may be incurred before one can switch to secondary.	No primary/secondary setup is required, because writes are propagated on both sides. Similarly, reads also can be serviced from both nodes.
Storage appearance to the nodes	Storage system appears as two sets, and at any point in time only one set of storage is being used.	Storage is transparent to the application. The OS or the storage software (ASM) hides the underlying storage structure, and the application does not know the dual nature of the storage.
Failover during failures	Delay in recovery. The secondary site should be manually switched to active status during failures. This includes downtime.	Instant failure to surviving arrays. The surviving instance continues to run without downtime.

https://t.me/learningnets

Technet24

Advantage	Data mirroring is done at the storage (disk array) level. The host CPU is not used for data mirroring.	Both nodes can be used concurrently all the time.
Disadvantage	Only one set of mirroring is used at a time. Failover is not instant.	The nodes do mirroring, and overhead is passed to the CPU.
Uniform performance	A node close to the primary storage always performs better than the secondary one.	Because the I/O requests are serviced from all the nodes, close to uniform performance is expected in all the nodes. Also, when ASM with the "preferred read" option is used, the reads are serviced by the physically closest node.
Single-vendor storage requirement	Both the sites should have the same storage (physical, architectural, and logical) because proprietary block mirroring techniques are implemented at the storage layer.	Storage can be from different vendors/ different types because the mirroring is handled by the application.
Example	EMC (SRDF) Symmetric Remote Data Facility	Oracle ASM, HP OpenView storage mirroring, IBM HACMP

TABLE 15-1. *Array-Based Mirroring vs. Host-Based Mirroring*

ASM Preferred Read

Features such as "ASM preferred read" allow you to choose the extents physically closest to the node, even if that extent is a secondary extent for reads.

You can configure ASM to read from a secondary extent if that extent is closer to the node instead of ASM reading from the primary copy, which might be farther from the node. Using preferred read failure groups is most beneficial in an extended cluster.

You can configure the initialization parameter ASM_PREFERRED_READ_FAILURE_GROUPS to specify a list of failure group names as preferred read disks. It is recommended that you configure at least one mirrored extent copy from a disk that is local to a node in an extended cluster. However, a failure group that is preferred for one instance might be remote to another instance.

Challenges in Extended Clusters

You'll probably encounter some challenges in an extended cluster that you normally wouldn't expect to see in normal clusters. The first important thing is the complexity of the network connectivity between the nodes. The private interconnect plays a more

https://t.me/learningnets

important role in the extended clusters than in normal clusters because the distance between the nodes adds an important and expensive component to the overall performance—network latency. Network latency should be taken into consideration because every message and every block transfer includes this additional delta time.

The second important thing is the mirroring technique used for the storage replication. The choice of mirroring technique makes a difference, because from the application layer, the Oracle kernel should see the storage as a single unit and the abstraction should be provided from the lower layers. The choice of mirroring also plays an important role in manual restart during node failures.

The third important thing to be considered during the design is the quorum. A third site for the quorum device is recommended to maximize availability in an extended cluster environment. The third site acts as arbitrator as to which site will survive during communication failures. This is worthwhile only when third-party clusterware is used. Oracle's clusterware uses the quorum device during regular operations (versus only during a cluster reconfig event) and therefore no advantage is gained by having it at a third location.

Normally, with the support of mirrored voting disks, you can overcome this problem in regular clusters. However, for the stretch clusters, another level of redundancy can be set up using a third site as a quorum site. This will enable the cluster to be functional if a disturbance occurs between the communications channels from the nodes. The choice of quorum plays an important role in tie-breaking during split-brain situations. The third site can be a simple workstation to act as an arbitrator in case of failures, as shown in Figure 15-6.

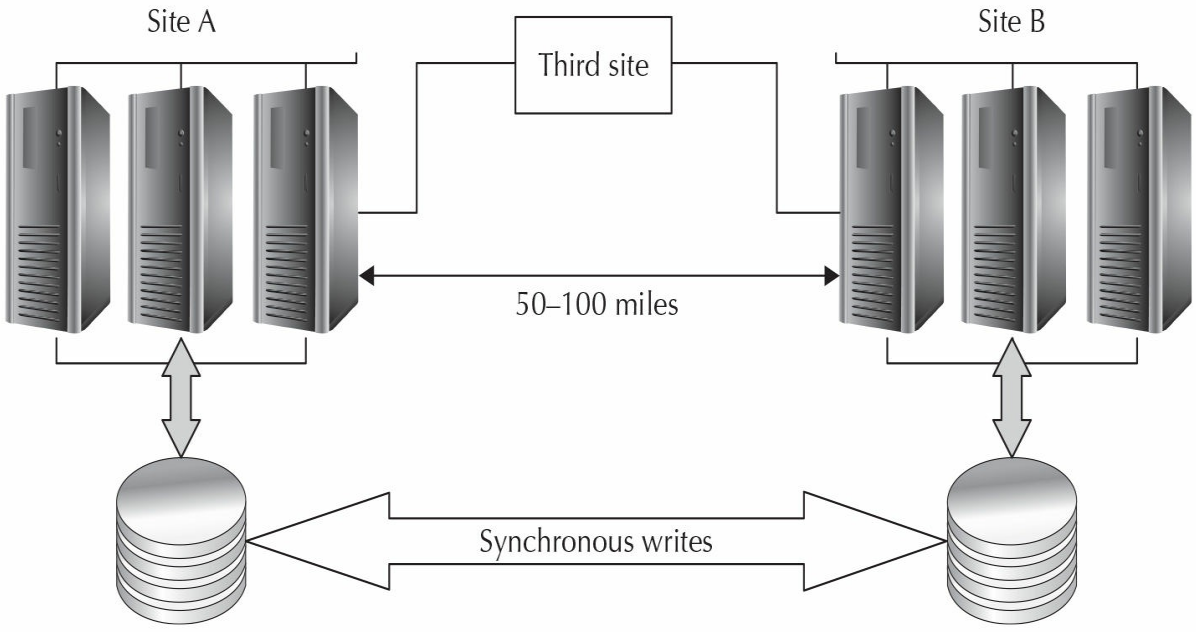


FIGURE 15-6. *Third site as quorum in a three-node structure*

https://t.me/learningnets
Technet24

Extended Oracle RAC Limitations

Oracle RAC on extended clusters is a very specific solution for customers who want to build business applications on extended clusters. However, it does not work well for long distances, due to high latencies. If the distance between the nodes is greater than 100 miles, you should carefully test the system and prototype it before implementing it.

Extended clusters *require* a dedicated point-to-point high-speed network. They cannot use the public network, and setting up a DWDM or using dark fiber may cost hundreds of thousands of dollars. Extended Oracle RAC clusters aren't suitable for unstable networks. A slight interruption in network service will cause node eviction and failures.

Extended Oracle RAC vs. Oracle Data Guard

You can use Oracle's Extended Oracle RAC high availability solution as a limited disaster recovery solution. Extended Oracle RAC doesn't replace the functionality of Oracle Data Guard, which is a true disaster recovery solution. If you're looking for a true disaster recovery solution, you should consider Oracle Data Guard in conjunction with extended Oracle RAC.

You can use Oracle Data Guard to overcome the intrinsic limitations imposed by Extended Oracle RAC. Oracle Data Guard also offers the following advantages over Extended Oracle RAC:

- **Network efficiency** Oracle Data Guard doesn't require expensive DWDM or dark fiber technologies for mirroring. It uses normal TCP/IP for communication and does not require additional protocol converters. Oracle Data Guard transmits only redo data or incremental changes. It does not ship block images unless traditional hot backups are used.
- **No distance limitation** Fiber-based mirroring solutions have an intrinsic distance limitation and they can't be deployed over thousands of miles. They also require additional protocol converters that indirectly add to the cost, complexity, and latency.
- **Data protection** Oracle Data Guard protects from logical corruptions as the database is recovered from the primary. It validates the redo records from the archive logs of the primary database before applying to the secondary database. Both SQL APPLY and redo apply do not simply mirror the block images like storage mirroring, thus providing better protection against physical and logical corruptions.
- **Better flexibility and functionality** Oracle Data Guard uses commodity hardware and storage. It doesn't restrict you to specific storage vendors using

https://t.me/learningnets

proprietary mirroring technology. Remote mirroring solutions typically need identically configured storage from the same vendor. You cannot have a scaled-down version of the disaster recovery site. You can also deploy Oracle Data Guard in an asynchronous mode. Oracle Data Guard provides a variety of protection methods and a graceful handling of network connectivity issues.

Deploying Oracle RAC on extended clusters requires careful architectural planning and deployment practices. You can also use a correctly implemented Extended Oracle RAC cluster as a limited disaster recovery solution. Oracle Data Guard and Oracle RAC are designed for different kinds of failure scenarios. They don't compete but rather complement each other when you use them correctly.

Summary

Extended Oracle RAC configurations are very special within the clustering technologies, with the member nodes spread across geographic locations. The "speed of light" factor plays a key role in the latency between member nodes and puts a practical limitation on the distance between the nodes. Extending a cluster introduces about one millisecond of latency for every 50 miles, which limits the range of extended clusters to less than a few hundred miles apart.

You can mirror the storage for Extended Oracle RAC via two methods: host mirroring and storage mirroring. Oracle ASM uses the host-mirroring approach, and storage vendors use storage-mirroring techniques. The ASM preferred read feature takes advantage of local storage for reading, and it increases the performance for reads.

In summary, Oracle RAC on extended distance clusters is an attractive alternative architecture that allows scalability, rapid availability, and even some limited disaster recovery protection with all nodes fully active. This architecture can provide great value when you use it properly, but it's critical that you understand its limitations.

https://t.me/learningnets

Technet24

CHAPTER 16

Developing Applications for Oracle RAC

You can use Oracle Real Application Clusters to scale most database applications—from custom applications to commercial off-the-shelf (COTS) applications. An application that runs in a single-instance environment can be directly ported to the Oracle RAC platform without needing any major architecture or code changes when it is designed optimally, and almost all packaged applications can be ported to Oracle RAC when the scalability or availability of the application is in question.

In simplest terms, an application vendor could install Oracle RAC on a cluster, install the application on top of it, and run the application without any loss in business functionality due to changes in database status (single instance to multiple instances). This means that your order-entry form or balance sheet report will not go missing from the application menu just because your database has been clustered.

Although any application developed on a single-instance database can run against an Oracle RAC database without modification, the performance of some apps may be suboptimal in Oracle RAC as compared to a single instance. That might be true even for well-designed and well-performing applications (on a single instance) because even though Oracle RAC is scalable, that's not necessarily the case for every

https://t.me/learningnets

application. Scalability of the underlying application is critically important for scaling applications running on Oracle RAC.

With this background, many technologists, architects, and managers *assumed* that Oracle RAC would solve all their scalability and performance problems. Suddenly, Oracle RAC was the much-anticipated panacea for a badly performing application on a single node. This created a large potential market for Oracle RAC, from small shops to large enterprises. What it also created, eventually, was much chagrin among users because Oracle RAC did not perform as it was "supposed" to. Like many new software products, Oracle RAC encountered many challenges in the beginning, but its technical prowess, robust features, and performance have always been its greatest strengths.

Oracle RAC is part of a high availability option that customers can exercise when maximum availability is a primary criterion. Using Oracle RAC for the sole purpose of improving system scalability and performance may or may not pay off because such improvements depend on many factors. Key among these factors are application design and scalability.

A well-designed and well-tuned database that services an equally well-designed application could scale admirably on Oracle RAC with multiple nodes, as compared to a single symmetric multiprocessing (SMP) box. This scalability in terms of the number of concurrent users, workload achieved, and performance has been proven many times by hardware and application vendors on various platforms and different types of applications. Reliable facts and figures are available from individual companies and analysts to prove the scalability of Oracle RAC on multiple nodes. (For more information on the latest results, visit www.tpc.org. Oracle regularly publishes results of large TPC benchmarks on Oracle E-Business Suite and other well-known applications using Oracle RAC.)

Application performance in Oracle RAC depends on how well the application performs on a single instance. Even though no changes are required to run an application in Oracle RAC, specific issues must be considered when an application moves to Oracle RAC, because these issues could hinder the performance of the application or the overall system. More insight on these best practices is provided throughout this chapter.

Comparison of application and database performance and scalability on single-node SMP boxes versus scalability in clustered systems is tricky and sometimes inconclusive due to the complex nature of the various software layers involved and database/application characteristics. For example, a batch job to reconcile all accounts in a banking application may perform 25 transactions per second and take 4500 seconds to complete. Of the total 4500 seconds, 4000 seconds comprise the CPU time on an eight-CPU box, at 90 percent utilization (7.2 CPUs fully utilized). For this example, consider the remaining 500 seconds to be for latency-related issues such as I/O waits, locking, and so on. Also assume that no other job or process is running when this batch is executed.

If the job needs to be completed in half the time—that is, 2250 seconds—then a generic solution would be to double the CPU capacity on the same box. Theoretically, a 16-CPU box should complete the same job in 2250 seconds at 25 transactions per

https://t.me/learningnets

Technet24

second and 90 percent utilization (14.5 CPUs fully utilized). This is assuming that nothing much can be done about reducing the wait time of 500 seconds and that the application is well tuned.

Without going deeper into any scalability formulas, queuing models, theorems (which are critical, vital, and indispensable tools for architects, designers, and developers), and all the other technical complexities that accompany scalability challenges, can we provide a confident answer as to whether a single-instance database with higher CPU capacity would scale? Is it possible to achieve the same scalability (completing the job in 2250 seconds) if we deploy a two-node Oracle RAC with eight CPUs each?

A straightforward and simple answer would be difficult to provide without knowing factors such as these:

- How much time are we spending waiting on a single instance for I/O, locks, and so on? Can that be reduced?
- Can the application work exactly the same way, with the same throughput of 25 transactions per second, without modifications in Oracle RAC?
- Can the application perform the same without running into locking/concurrency issues when deployed on Oracle RAC?
- What would be the application throughput with lock mastering overhead when Oracle RAC is used?
- Do we have enough parallel streams/processes within the application logic to take advantage of extra CPU resources?
- What about the cost factor? Which model is cheaper?

This is just the beginning of a long discussion that would take into consideration many more factors before a useful answer could be achieved. All that could prove useless, however, if the application testing results on both single-node and Oracle RAC are not considered.

Nevertheless, we have seen that with a given set of characteristics and metrics, significant trends can be observed and recorded in both single-node and Oracle RAC systems to provide effective and meaningful data for analysts and designers. This data can be used in conjunction with further testing results to make decisions or take positive actions. It would be rather naive to act solely upon individual data (such as just single-node SMP results) to make a decision on the best model for scaling and achieving the required performance.

This chapter is by no means a complete lesson on application scalability and performance, how to achieve it, and how not to. The aim is to provide some best-practice solutions and suggestions for an application running on an Oracle RAC database that can fit a wide variety of applications, from pure online transaction processing (OLTP) environments to a mix of data warehousing and online systems. Applications such as traditional enterprise resource planning, banking/insurance, general finance, customer relations, telecom billing, manufacturing, web applications,

https://t.me/learningnets

Internet stores, and data warehouses can apply these best practices according to their environments. As always, testing is mandatory because a generic recommendation that suits all requirements doesn't exist. The so-called "one size fits all" formula is a myth and fails more often than not.

These best practices and guidelines have been drawn from experience at various sites on different platforms and breeds of applications, from Oracle's own experiences and knowledge base, from debates and discussions with users, and from technologists and Oracle RAC enthusiasts across the globe. The guidelines are intended to reduce overall contention for resources such as data/index blocks, latches and locks, memory, and CPU cycles in an Oracle RAC environment.

Let's now examine some of the factors that affect application performance and scalability on an Oracle RAC database. There's no established order of priority, but all the factors deserve equal attention, and probably more so in certain situations, depending on the complexity and urgency of the problems/challenges you face.

Even though this chapter is not an exhaustive list of all the factors that need careful consideration and analysis, it includes some of the most contentious issues that affect Oracle RAC databases of all versions and sizes.

Application Partitioning

To start, consider reading "Oracle 8*i* Parallel Server Administration, Deployment, and Performance" for Release 2 (8.1.6), Part No. A76970-01, available online on the Oracle Technology Network site at https://docs.oracle.com/cd/A87860_01/doc/paraserv.817/a76970/toc.htm.

Although it's a couple of decades old, this manual is an excellent primer about the basic problems applications could face in a parallel-server environment and the types of partitioning schemes that can be adopted to overcome them. Subsequent releases of Oracle documentation have eliminated much of this foundational material, which was deemed unnecessary from Oracle 9*i* RAC onward. Yet a review of the book, especially Chapters 5 and 6, would do you a lot of good, especially if you're new to Oracle RAC and its traditional problem areas.

In Oracle Parallel Server (OPS), application partitioning was necessary to scale applications. This scheme had to be adopted to avoid block pings across all participating instances in the parallel database. One of the stumbling blocks in OPS 8*i* in certain implementations was the disk pinging effect and its need for a *forced block write* of a dirty block to the disk by a holding instance when another instance requested the same block for modification. Remember that Cache Fusion I (aka Consistent Read Server) minimized the need to write blocks to disk, even for a consistent read (CR) block. But when multiple instances required the same block for modification, application scalability was limited because increased requests for the same block caused an increase in disk writes. This also increased global locking activity—interconnect traffic as well as heavy CPU usage. A lot of special care and attention was needed to tune such parameters as GC_FILES_TO_LOCK, which was almost the only way to control locking at the block level.

https://t.me/learningnets

Technet24

Contention occurs when the same application with the same transaction pattern is run on multiple nodes at the same time. Hence, the application needs the same set of blocks for Data Manipulation Language (DML) operations as they touch or operate on the same set of tables. This is called *table overlap* and occurs when the same table or set of tables is accessed by two or more application components concurrently. For example, a transaction table could be simultaneously queried and updated by both the Receivables module and the Payments module. This creates more chances of contention for a block or set of blocks, depending on the type of transaction.

A high INSERT rate into the same table from different nodes has the potential to cause contention for the block being affected as well as for the bitmapped blocks (BMBs; in the ASSM tablespace). If indexes are present for that table, this may also cause contention for the index branch/leaf blocks. The use of database sequences for the generation of key values increases the chances of buffer contention.

Cache Fusion II (aka Write/Write Cache Fusion) minimized some of the forced disk write effect by directly transmitting a copy of the current block across the high-speed interconnect to the requesting instance in some circumstances. However, a copy of the buffer has to be shipped to the requesting instance through the interconnect constantly between the instances.

Best Practice: Application Partitioning Schemes

To reduce these effects, application-partitioning schemes were introduced in OPS 8*i* and continued to be used until Oracle 11*g*. These schemes are definitely not obsolete in Oracle 12*c* and can still provide increased performance and scalability. However, application partitioning is not a requirement for Oracle RAC database installations, although many sites are known to use it effectively.

Application partitioning refers to the following:

- Identification of the main functional business components of the application and segregation
- Scheduling an application or a component of the application to run on a specific node (or nodes) only, thereby restricting access to that application through a particular node (or nodes)

For example, a production manufacturing application could run Inventory on node 1, BOM on node 2, Payments on node 3, and Receivables on node 4. It’s difficult to avoid table overlaps in most of these types of applications, in which multiple modules share the same set of tables. By running an application module on a single node or a maximum of two, you can minimize block pinging effects. This is a partial solution when tables can’t be split (so that each application module uses its own tables and indexes) and when data partitioning is not possible. In the next section, you’ll see how you can use data partitioning to overcome this problem.

Application partitioning can reduce the chances of block pinging. The fewer the number of common tables across modules, the better the application performance.

https://t.me/learningnets

Oracle 12*c* RAC can also leverage this basic type of partitioning—in OLTP as well as batch/decision support system (DSS) types of environments.

Read-Mostly Locking

Oracle uses a locking mechanism called *read-mostly locking*. The database implements this mechanism in the form of a new locking element called *anti-lock*. The anti-lock element controls the behavior of the "read mostly" locking scheme by granting Shared read (S affinity lock) globally to all the nodes.

With global S access to an object, all nodes are pre-granted read access to all the blocks for the object, which eliminates the locking overhead associated with read access. Any read access on the objects covered by the read-mostly locking scheme can bypass the standard Cache Fusion protocol for block transfers between instances. However, the write access on the objects strictly follows the regular Cache Fusion mechanism and might come with a higher cost than reads.

Oracle optimizes this locking protocol further for application-partitioning schemes, especially in a carefully partitioned application with very little read-write contention between objects, and if most of the access is for executing SELECT statements.

Data Partitioning

The benefits of data partitioning are well known to users, and many large databases have reaped rich rewards by implementing the many schemes introduced in each successive release of the Oracle server. (Note that partitioning requires Oracle Enterprise Edition, which is an extra cost option.)

Partitioning data as well as indexes needs abundant consideration and a well-planned approach in Oracle RAC environments. Partitioning can help minimize a lot of contention for data and index blocks, provided the right partitioning strategy is chosen. Data partitions can have a much bigger impact on performance in Oracle RAC since, starting from 10*g*, lock mastering (and dynamic remastering) is handled at the segment level.

Interval Partitioning

Oracle 12*c* offers a partitioning strategy called *interval partitioning*. Interval partitioning overcomes the limitations built into range partitioning, where a specific range value or limit is specified by the DBA creating the partitions for the table. Interval partitioning is an extension of range partitioning in which Oracle creates a partition automatically when the inserted value does not meet the insertion criteria for all other existing partition ranges within the table.

Interval partitioning requires you to specify at least one range partition during

https://t.me/learningnets

Technet24

table creation. Once you provide the range partitioning key for the range partition, the database uses this transition point as the baseline to create interval partitions compared to this point.

The type of environment/application is a crucial factor in choosing the type of partitions to create. This means that the application could be a straightforward OLTP system or a mix of OLTP and batch processing. Some environments are pure batch or pure DSS in nature, and sometimes referred to as *data warehouses*. Each of these environments has a predominant "type" or "nature" of transaction. It could be INSERT intensive, UPDATE intensive, or SELECT intensive, or a mixture of these. It is common to see OLTP applications that do a lot of all three types of activities, as well as DELETE operations.

Some applications also have lots of Data Definition Language (DDL) embedded in their systems. Transaction volume (rate of inserts, updates, and deletes) is another critical factor that you need to consider while choosing the correct partitioning strategy.

Most of the discussion concerning data partitioning and buffer-busy problems is restricted to large INSERT-related issues because they are the most common.

Best Practice: Guidance Systems

You can use workload characterization in conjunction with application partitioning to improve performance. *Workload characterization* refers to cases where data is guided or directed into each partition based on the dataset that will be used for processing on that node/instance.

For example, the data is inserted using a guidance system to ensure that the data is processed from a particular node only. Assume a four-node Oracle RAC database with each node processing data for one region only. When data arrives for insert, the guidance system routes the data such that node 1 inserts Western region data only, node 2 inserts Eastern region data, and so on.

The guidance system can be application logic or a combination of software (for example, Data Dependent Routing in Tuxedo) and intelligent devices with algorithms built in. Data is usually routed through these systems. Telecom applications make extensive use of guidance systems, although the usage is far more complicated and extensive.

In certain cases, the data arriving (from an external system, flat files, and so on) could be presorted and guided to an instance such that no intelligence or guidance is required at the application end during the insert into a node. Once the data is in, the application on node 1 will process data from partition 1 to partition 25. Node 2 (running the same or another module) will process data from partition 26 to 50, and so on. User data is not shared between the nodes during processing, except in certain unavoidable circumstances. This creates an affinity between nodes and data/index blocks. Therefore, the application is more likely to find a block or set of blocks in its

https://t.me/learningnets

local cache, thereby effectively reducing cross-instance communication and block transfers.

In typical OLTP environments, the number of blocks generally visited per query or DML operation is small. This is because data selection is driven mostly by primary or unique key lookups. Large-range scans are less likely, unless the application requires them, is not well designed, or suffers from various types of "over-indexing" and architectural issues.

In contrast, batch operations that select lots of data (data mining, reporting, DML) need hundreds of thousands of blocks to be visited to satisfy the query. Without partitioning and proper guidance, intense competition for data/index blocks among instances will cause block contention, which is visible as one or a combination of the following wait events:

- buffer busy waits
- read by other session
- global cache cr request
- gc cr buffer busy
- gc current block busy

Application changes may be required to guide this data using a key value such as instance ID or the equivalent. Data structure changes are also required because a column will be added to the table(s). This column identifies the instance from which this row was inserted (or in some cases, last updated). For example, if the table is to be range partitioned on order_date, the partition key can be **(order_date,instance_id)**.

Some designers like to use a different column name that identifies each guided row and binds it to a particular instance or table partition. As mentioned before, guidance into each instance/partition is done using some predetermined algorithm. This column will also form a part of the partition key **(order_date, <column name>)**. Further, we could also have index partitions based on this column.

Using this key value, data is inserted or updated into the correct partition, which will later be processed by the application on a particular node only. This scheme can significantly reduce contention for data blocks, bitmap blocks, and index leaf blocks (when indexes are partitioned).

The following extract from an old Oracle documentation, "Oracle 9*i* Real Application Clusters Deployment and Performance" (Release 2 [9.2], Part No. A96598-01), summarizes this strategy in a very lucid manner:

> Cache Fusion eliminates most of the costs associated with globally shared database partitions by efficiently synchronizing this data across the cluster. However, object partitioning, without changing your application, can sometimes improve performance for hot blocks in tables and indexes. This is done by re-creating objects as hash or composite partitioned objects.
>
> For example, consider a table that has a high insert rate which also uses a sequence number as the primary key of its index. All sessions on all nodes access the right-

https://t.me/learningnets

Technet24

most index leaf block. Therefore, unavoidable index block splits can create a serialization point that results in a bottleneck. To resolve this, rebuild the table and its index, for example, as a 16-way hash partitioned object. This evenly distributes the load among 16 index leaf blocks.

Buffer Busy Waits/Block Contention

Almost every Oracle tuning book covers the topic of buffer busy waits. In Oracle 8*i* OPS days, reducing data block contention (mainly during inserts) was possible using FREELISTS and FREELIST GROUPS. If application designers, developers, and DBAs are able to work out the correct settings for these two parameters, it's still a valid and effective approach to reduce contention for data blocks significantly as well as segment header waits in Oracle RAC environments when concurrent inserts are used.

Performance tests with large concurrent inserts have shown that using these parameters can provide the same or better benefits when compared to using Automatic Segment Space Management (ASSM) tablespaces. Needless to say, end users would have to use locally managed tablespaces in Oracle 10*g* and above, along with accurately sized extents and other segment management clauses. Free space management is done using FREELISTS and FREELIST GROUPS in the pre–Oracle 10*g* manual segment space management-based tablespaces, and through the use of bitmap blocks in ASSM tablespaces.

The segment header block can be a point of great contention between instances. Freelist groups alleviate the contention for the segment header block because they move the point of contention away from the segment header to dedicated blocks of master freelists. Each freelist group can be associated with an instance, provided the numbers of freelist groups is equal to or greater than the number of instances.

Hot Blocks and X$KSLHOT

Starting from Oracle 10g, you no longer have to write multiple and repeated queries on V$BH or X$BH or dump data/index blocks to find hot blocks that are generally chased by applications. A new view called X$KSLHOT has been added to simplify the identification of hot blocks. To use this feature, you need to set the little-known hidden parameter **_db_block_hot_tracking** to true.

This table has a sample of the RDBAs that are responsible for contention. The kslhot_id column is the RDBA, and the kslhot_ref column is a count of the relative number of times it was encountered. A higher kslhot_ref implies the corresponding kslhot_id is a bright candidate for a hot buffer.

Be forewarned, though, that the clustering_factor of indexes is influenced by using multiple freelists (not freelist groups) as data gets scattered among more blocks. This sometimes increases the I/O cost of the table access by index because the index keys

https://t.me/learningnets

are widely scattered among more blocks.

Oracle 9*i* introduced ASSM, which is an effective "auto" method of reducing block contention, because it randomizes inserts by trying to pick a different block for insertion for each process. There is no guarantee that each time it will behave in the same way, but the chances that the same block will be picked by two or more inserting processes are low in ASSM. Hence, this random factor tilts the balance in favor of ASSM, especially in Oracle RAC.

When the FREELIST parameter is used, Oracle maps a process to a freelist using the following formula:

```
mod(oracle process_id, no. of process freelists) + 1
```

In ASSM, it uses the process ID to guide itself to a space map block, and even choosing a block is based on the process ID. As we have seen earlier, one of the key benefits of using ASSM tablespaces is that Level 2 BMBs have an affinity to an instance. Hence, they usually aren't sent across the interconnect. In addition, the blocks that they manage also have an affinity to the same instance as the Level 2 BMBs.

Using ASSM can also have an extreme effect on the clustering_factor of an index (although it is not a symptom that is repeated every time) because the data is distributed across many blocks. Application development teams should test their applications on ASSM with high inserts and analyze the trend because no silver-bullet solution exists for all situations. Also, using ASSM can cause performance problems during full table scans because the tables are generally larger than non-ASSM tables.

Index Partitioning

One of the most glaring mistakes made at many sites is to partition the tables, but not partition-related indexes. Index partitioning is equally or more important than table partitioning—index blocks have a tendency to get hotter than the table blocks because of high concurrency at the block level due to the smaller row length of index keys. The following techniques can be implemented to alleviate index block contention issues.

Buffer Busy Waits: Index Branch/Leaf Blocks Contention

In an application where the loading or batch processing of data is a dominant business function, there are more chances of performance issues affecting response times due to the overhead of index maintenance during large volume inserts. Depending on the access frequency and the number of processes concurrently inserting data, indexes can become hot spots and contention can be exacerbated due to frequent leaf block splits, ordered and monotonically increasing key values in the index (usually generated using a sequence), and a low tree depth.

https://t.me/learningnets

Technet24

A leaf or branch block split can become an important serialization point if the particular leaf block or branch of the tree is concurrently accessed. The following wait events are typical indicators of this kind of block contention issue:

- gc buffer busy
- gc current block busy
- gc current split

In an Oracle RAC environment, the transaction splitting a leaf block and the transactions waiting for the split to complete are affected if the data required for the operation is not locally cached. Before and during a split operation, the transaction performing the split will have to execute a series of global locking operations before a pin is acquired on the target block. These operations are expensive, especially in Oracle RAC, where the latencies adversely affect application performance during large inserts. *Blocking asynchronous trap (BAST)* processing during a branch/leaf split has additional impact on the time taken to complete the split.

Note that periodic index rebuilding for so-called "performance improvement" in the Oracle RAC environment could be counterproductive. As a matter of fact, Oracle B-tree indexes are balanced by design, and under normal conditions, rebuilding the indexes at periodic intervals is not required. Rebuilding indexes makes the index blocks more compact and reduces the free space. In other words, we need to consider Oracle RAC implications not just for design or development issues, but for maintenance activities, too.

The challenge is to minimize the chances of multiple instances inserting (data and index) into the same block concurrently. This can be done by using hash or range partitions with appropriate changes in the application code or database structure. The partition scheme is dependent on the suitability to the application.

A right-growing index experiences increased contention for the leaf blocks mostly due to the fact that keys are generated in order of increasing sequence. Therefore, the leaf blocks tend to heat up quite often. This is known as *buffer busy wait*. Moreover, frequent splitting of indexes may introduce further serialization points.

Distributing the access to the leaves over multiple index partitions alleviates the hot spots. Apart from a significant reduction of contention, local cache affinity improves because leaf blocks are retained in a local cache longer and are more available for local users.

Best Practice: Index Partitioning

Partitioning of indexes provides huge improvements in OLTP as well as batch-type environments. The choice between local or global indexes is highly dependent on the application and how the data is queried, whether index lookups are unique key (or primary key) based or index range scans. Partition maintenance and performance are also key points to consider while choosing an index partition type.

Global indexes are good for OLTP environments, where performance is generally

https://t.me/learningnets

high on the wish list, and for applications that like to perform unique key lookups. Oracle 10*g* R2 and onward support hash partitioning of global indexes, which provides more flexibility for handling partitioned indexes.

Oracle allows you to partition only the index independent of the table, and its performance tuning becomes simpler if you can thus limit contention just to the index blocks.

It's worth mentioning at this point, however redundant it might be, that SQL statements must make use of the partition key in their WHERE clause for partition elimination/pruning to happen. Note that partition elimination works only for range or list partitions. Index range scans can be disastrously slow without restrictive predicates based on partitioned and indexed columns. These columns should be the partition key or part of it to leverage partition pruning. The type of predicate (=, >, <) has the potential to make or break the query.

Reducing Leaf Block Contention with Reverse Key Indexes

Some applications benefit from reverse key indexes (high INSERT environments) in terms of performance, but they are notorious for generating high I/O. Because the database doesn't store the actual (nonreversed) values of the index in the index, the database can't perform index range scans.

Generally speaking, the database can perform fetch-by-key and full index scans on a reverse key index. If the reverse key index is built on a single column, the Cost Based Optimizer (CBO) won't use an index range scan. If the reverse key index is built on multiple columns, it will be used by CBO for a range scan operation when the equality predicate is used on the leading columns of the index.

Note that the clustering_factor of a reverse key index will lean more toward the number of rows because the index points to table rows that are scattered over different or numerous blocks. This will influence the CBO toward choosing an execution plan without the reverse key index.

Sometimes, a combination of reverse key indexes and an adequately cached database sequence that generates the key values can provide immense relief for databases with acute index leaf block contention.

You can attain great benefits (which are offset at times due to side effects) with the usage of reverse key indexes, but be careful while implementing reverse key indexes where the application performs index range scans. Designers and developers should also consider these options while choosing indexing and partitioning strategies.

One of the undesirable side effects of using reverse key indexes is that they must be constantly rebuilt with 50 percent or more free space to be beneficial, because a large number of block splits can and will take place in an active table.

Global Cache Defers and Reverse Key Indexes

Although reverse key indexes and hash partitioning help greatly in combating the waits for the objects with high concurrency, they require an additional tweak to work flawlessly. Oracle provides multiple optimization techniques to overcome a problem,

https://t.me/learningnets

Technet24

and some of them are counterproductive while working together. Reverse key indexes is one of the features that needs careful execution.

When there is high concurrency at the block level, it is often more beneficial to keep the blocks in a holding instance for an extended time and allow the lock processes to continue the work than to repeatedly send and receive the blocks. When high concurrency is observed at the block level, the new optimization technique called *Global Cache Defer* is implicitly invoked and suspends all the send requests for some time. This has an adverse effect on performance when hash partitioning or reverse key indexes are used to cure the same issue.

The Global Cache Defer mechanism used to be very aggressive in earlier versions of Oracle. The Global Cache Defer time is controlled by the parameter **_gc_defer_time**, which defaults to three centiseconds. Disabling Global Cache Defers should be carefully benchmarked before rolling out to production.

Sorted Hash Clusters

Oracle offers several optional methods for storing table data, including clusters and hash clusters. *Sorted hash clusters* are particularly suitable for systems requiring very high data insertion and retrieval rates.

This table structure better supports first-in first-out (FIFO) data processing applications, where data is processed in the order it was inserted. Such applications are typically found in the telecommunications and manufacturing environments. For example, consider the call detail record (CDR) data structure that is common in telecommunication environments. For billing and auditing purposes, each call is to be recorded and associated with its origin. Usually, calls are stored as they arrive from the switch. This data can be retrieved later in FIFO order or a customized order when customer bills are generated.

Although it’s common practice to capture this data in two standard tables, a sorted hash cluster is a better solution because it allows inexpensive access to the sorted list of call records for any given subscriber and allows the billing application reading the call records to walk through the sorted list in a FIFO manner. Any application in which data is always consumed in the order in which it is inserted will see tremendous performance improvements by using this type of optimized table structure.

The key is to partition the data and indexes as required by the application. Store your data the way the application needs it and queries it. If the algorithm selects customers in a descending order, explore the possibilities of storing customers in a descending order and use an index created in descending order of customers. There might be potential overhead in using this method because it would require constant table/index rebuilding, loading of data, and other associated administrative work. But with automated tasks, the benefits of using methods like these can sometimes outweigh the overhead.

In Oracle RAC environments, the application must build the intelligence to process data from a defined set of data/index partitions only. This reduces the chances of block contention and the “ping-pong” effect of repeated blocks traveling between instances.

https://t.me/learningnets

Working with Sequences

Database sequences are useful for generating numbers that grow or increase sequentially in an order that is configurable. Application designers need not use "home-grown" sequence generators using a database table and update it using local logic. Such sequences are not very scalable, especially in Oracle RAC environments with high inserts, when they're used with specific clauses. Yet, in certain cases, these customized and non-Oracle sequences are the only way to achieve certain goals that you can't achieve with Oracle's own sequences.

A select from a sequence is an update to the data dictionary when it isn't cached. So, every select internally invokes a DML and commit, which is expensive in Oracle RAC, because system commit numbers (SCNs) must be synchronized across the cluster when sequences are used extensively.

In Oracle RAC environments, performance degradation due to heavy use (or abuse) of sequences and a high insert rate can manifest in different ways:

- Contention for the SV enqueue. If you see contention for the SV enqueue, the problem is complicated and Oracle Support Services might be required.
- Buffer busy waits for various classes of blocks such as data, index leaves, and segment headers.
- TX enqueue waits on the index blocks may also be visible.
- In certain cases, ITL waits are possible when a sufficient number of transaction slots is not reserved or available. This could be due to a very low setting for PCTFREE combined with a default value for INITRANS or artificially limiting transaction slots using MAXTRANS.

CACHE and NOORDER

The CACHE option speeds up selection and processing of sequences and minimizes the writes to SEQ$. Note that the CACHE option is usable only if an application is able to accept gaps in the sequence, irrespective of whether you use Oracle RAC.

From a performance and scalability perspective, the best way to use sequences is to specify the CACHE and NOORDER options together. The CACHE option ensures that a distinct set of numbers is cached in each instance after the first use of the sequence. The NOORDER option specifies that the sequence numbers may not be in order. For applications that perform large concurrent inserts from multiple instances and generate a key value such as the primary key using the sequence, using the NOORDER option along with the CACHE option provides the best throughput.

In a high insert environment, tables that have their key values generated from sequences tend to have hot blocks for both data and indexes, because the block where inserts are going is wanted by every instance. Without partitioning of the table and the applicable indexes, these blocks would become a major point of contention. Hence, partitioning of data and indexes should be one of the first steps in reducing this type of

https://t.me/learningnets

Technet24

contention. Along with partitioning, using an additional column along with the key value is critical to avoid contention.

These hot blocks are visible as “buffer busy waits” or “read by other session,” depending on the mode being waited for. There would also be an increase in the number of CR and CUR mode blocks transferred between instances, sometimes even with partitioning implemented.

One of the disadvantages of the NOORDER option is that the key values may not be in order. So if an invoice number, customer number, or account number is generated using this sequence and regulatory norms require you to maintain the order of numbers, NOORDER cannot be used. An abnormal instance shutdown or a crash will introduce gaps in sequences. Cached sequences may be lost during a shared pool flush or rollback. Sequences do not support rollback because a sequence, once fetched, remains fetched and the number is deemed consumed forever.

CACHE and ORDER

If you need sequences for which the ordering of numbers must be guaranteed, you can specify the CACHE option with the ORDER clause. This ensures the ordering of numbers, because each instance caches the same set of numbers, unlike with the NOORDER option.

The CACHE and ORDER options may not be as scalable as the CACHE and NOORDER method, but they ensure that sequences are in order, although gaps can still occur across instance startup and shutdown.

NOCACHE and ORDER

The NOCACHE and ORDER option is used when an application requires all sequence values in order and without gaps. Using this model has the biggest performance impact on Oracle RAC environments, and designers must understand the inherent scalability limitations before choosing this model.

To overcome buffer busy waits for data blocks while using sequences to generate the key value, specify FREELISTS if segment management is manual or use ASSM tablespaces. In addition, the segment header block is also subject to heavy contention in Oracle RAC environments. To reduce that, use FREELIST GROUPS.

Best Practice: Use Different Sequences for Each Instance

A good practice in Oracle RAC is to use different sequences for each instance to avoid a single point of contention. Each inserting process on a particular node uses its own sequence to select monotonically increasing key values. Selecting from a different sequence for high inserts can be a great solution, but it requires a change in the application.

https://t.me/learningnets

In Oracle RAC environments, to overcome buffer busy waits for index leaves where the key values are generated using sequences, do the following:

- Partition the table and its relevant indexes. Guide the data into the table partitions using additional columns, if necessary. Remember to analyze the effects on the clustering_factor of your indexes.
- Cache large values of the sequence and use NOORDER if possible.

With caching, the difference between sequence values generated by different instances increases; thus, newer index block splits would create an effect called an *instance affinity* to index leaf blocks. This seems to work only when the database uses the CACHE and NOORDER options together.

For example, if we use a sequence cache of 20, instance 1 would insert values 1, 2, 3, and so on, and instance 2 would concurrently insert 20, 21, 22, and so on. If the difference between the values is a lot smaller than the number of rows in the block, the two instances will continue to modify the same index block as sequence values increase.

If we set a value of 5000 for the sequence cache, instance 1 would insert values 1, 2, 3, and so on, while instance 2 concurrently inserts 5001, 5002, 5003, and so on. At the beginning of the inserts, both instances would write to the same leaf block, but subsequent block splits would cause the inserts to be distributed across different blocks because the difference in sequence values causes the inserts to map to different blocks.

There is no such thing as a best value for the CACHE option in sequences. Generally, developers should consider the rate of inserts per second/per instance as a guide to setting the cache value. Sequences may not be the best scalable solution, especially when an instance is used for the generation of ordered and gapless sequence numbers at a very high rate.

Connection Management

Applications that don’t use transaction monitors are generally guilty of poor database connection management. Efficient management of database connections is critical in single-instance databases and in Oracle RAC databases.

Some applications disconnect from the database and then reconnect repeatedly. This is a common problem with stateless middleware applications such as Shell/Perl scripts that exit on error and tend to use the AUTO_CONNECT option to reestablish the connection.

Connecting to the database incurs operating system costs associated with creating a new process or thread. Many recursive SQL statements also need to be executed during the process of establishing a new connection to the database. Because the session is new, these recursive SQL statements require a soft parse. This adds to the load on the library cache latches. Data file operations require operating system coordination, and this adds to the overhead of maintaining the process.

All these types of overhead can be avoided by retaining database connections as

https://t.me/learningnets

Technet24

long as they are required. Usually, applications built with a transaction monitor such as BEA Tuxedo do not suffer from these issues because they consistently maintain a database connection.

Stateless connection pooling improves the scalability of the mid-tier applications by multiplexing the connections.

Full Table Scans

Repetitive full table scans in Oracle RAC can impede performance and scalability as the private interconnect is used for transferring blocks, and that can eat into network bandwidth. This would cause a delay for other required and important blocks such as index blocks, undo blocks, header blocks, and any messages. The goal is to reduce interconnect traffic and bring down latency. With a lot of full table scans, interconnects could get overloaded, thereby flooding the private network.

Another very important factor to be considered with full table scans is an appropriate value for DB_FILE_MULTIBLOCK_READ_COUNT, also known as MBRC. Not only does this parameter have a positive influence on the optimizer to lean toward a full table scan (irrespective of Oracle RAC or non–Oracle RAC), but achieving interconnect saturation is easy if high values are used for this parameter.

Experience has shown that 16 can be a high MBRC value for an application such as the Oracle E-Business Suite. Many such systems experience severe performance degradation with an MBRC of 16 while running in Oracle RAC mode. Lowering the value to 8 immediately brings the performance back to acceptable or normal levels.

With a high MBRC value in Oracle RAC, when Oracle performs multiblock reads and the data is transferred across instances, the interconnect tends to get flooded with a large number of blocks flowing from one end to the other. Using a lower value for MBRC (such as 8) is a good starting point for applications on Oracle RAC databases.

Full table scans are not necessarily evil, and in some cases they are the best data access method for a particular process. But they are a common issue in many Oracle RAC performance problems, especially with custom applications. You can easily identify the full table scans in the database using any of the following techniques.

Identifying Full Table Scans

The Automatic Workload Repository (AWR) is one of the most common tools people use to monitor Oracle database performance. You can easily tell from looking at the top wait events whether full table scans are the problem. Existence of the “db file scattered read” wait event in the top five wait events is a quick and early indicator of a full table scan problem.

You can also query the V$SYSSTAT view for the “table scan” statistic. You can run the following query on V$SYSSTAT to check the number of times data is accessed using the full table scan method:

https://t.me/learningnets

```
select name, value from v$sysstat
where name like 'table scan%'
```

A review of the AWR report followed by a quick query on the V$SYSSTAT view can confirm whether full table scans are a problem.

Normally, performance monitoring tools or scripts built on using v$session_wait will quickly detect the full scans if they find the excessive waits on the “db file scattered read” wait event. A “db file scattered read” wait event lists the **file#** and **block#** along with the number of blocks it is reading (typically less than or equal to the MBRC value) during the full scan. These are displayed as p1, p2, and p3 values for the “db file scattered read” wait event. For a known **file#** and **block#**, you can use the following query to get the name of the table that is being accessed via full table scans:

```
select distinct owner, segment_name,segment_type
from dba_extents where file_id=<file#> and
<block#> between  block_id and block_id+blocks -1;
```

Besides querying the v$sysstat and the v$session_wait views, you can implement numerous techniques to detect full table scans if you suspect they are a problem. For example, if PHYSBLKRDS is significantly higher than PHYRDS in V$FILESTAT, there’s a good chance of full scans on the segment residing in that datafile. Also, blocks read by full table scans aren’t subject to normal LRU (least recently used) processing in the buffer cache. A special flag is set in the buffer cache (X$BH.FLAG), and if the state is 0×80000, you know the block is read using the full scans.

After identifying the table scans, you need to confirm whether they are expected or unreasonable. You can query either the v$session_longops or v$sql_plan view to confirm that. You can use the hash_value column to identify and tune the offending SQL queries for efficiency:

```
select sid,username,opname,target,totalwork,sofar,sql_hash_value.
from v$session_longops
where opname='table scan'
and totalwork>sofar

select object_owner,object_name,hash_value
from v$sql_plan
where operation='TABLE ACCESS'
and options='FULL'
```

A distinct advantage of these views over v$session_wait is that the values are not quickly overwritten and they are cached for an extended period of time. You can even query at a later time and confirm the activity. The v$session_longops view is quite

https://t.me/learningnets

Technet24

important here, because it also lists the other expensive operations in the database, such as hash joins, sorting, and RMAN operations.

Interconnect Protocol

Choosing the appropriate protocol to carry your blocks across the high-speed interconnect is a crucial decision. So are the configuration and settings for the protocol.

Oracle 9*i* made the first definitive shift toward the User Datagram Protocol (UDP), a connectionless protocol. Starting with 10*g* R1, Oracle has actively suggested the use of UDP on all platforms, even though hardware vendors have developed some exciting and lighting-fast technologies for transfer of data across high-speed interconnects. Prominent among them are Digital Equipment's (aka Compaq, aka HP) Reliable DataGram (RDG), HP's Hyper Messaging Protocol (HMP), and Veritas' Low Latency Transport (LLT).

Each of these protocols uses its own proprietary hardware interconnects based on fiber-optic or Gigabit Ethernet technology with very low network latency. Yet, each one of them has its own problems with Oracle RAC—except for RDG, which seems to perform and scale wonderfully well on Tru64 systems with minimal fuss.

What makes the big difference in interconnect performance (and subsequently, application response time and scalability) is choosing the right values for the internal layers of each of these protocols, especially UDP. The receive and send buffer size is usually left at default values on many installation sites. The Oracle kernel uses its own default values (128KB for UDP on all UNIX platforms). If the OS tunable parameters for UDP have been configured, the Oracle kernel uses those values to set its buffer sizes.

Default values are seldom adequate for applications that support hundreds or thousands of users. The number of connections and concurrent users may not always be a good yardstick for workload measurement. Some applications have a lower user count but a large transaction volume and cause a lot of internode traffic. Using the default UDP settings, Oracle RAC databases/applications that tend to have highly active internode block transfers may not scale when the workload increases.

You can specify a value of 64KB (65,536 bytes) for the **_sendspace** parameter while tuning this layer. Any value above that is most likely useless because the lower layer (IP) cannot process more than 64KB. A value that is 10 times that of udp_sendspace is a good setting for udp_recvspace. Usually, recommendations suggest setting both these parameters to 1MB each, which is also valid in many cases, yet 1MB is more of a generic tuning guideline as compared to the 64KB practice, which is derived and calculated based on UDP and IP behavior.

An interesting relationship that UDP shares with FTS is that low or "untuned" values for UDP buffer sizes can cause timeouts when CR blocks are requested by remote instances. Timeouts are visible in the "gc cr multi block" wait event or in a trace file generated via SQL Trace or a 10046 event setting. A look into the V$SYSTEM_EVENT view can provide the ratio of timeouts versus waits for the CR

https://t.me/learningnets

request event, as shown here:

```
                                                              Avg
                                                 Total Wait   wait    Waits
Event                            Waits   Timeouts  Time (s)  (ms)     /txn
------------------------ ------------ ---------- ---------- ------ ---
gc cr multi block             12,852      3,085     612     47      3.6
```

A large value for MBRC can exacerbate the situation. Hence, proper care must be taken while choosing values for UDP buffers as well as MBRC.

Ethernet Frame Size

Other than the protocol, the frame size also has a significant impact on the interconnect performance. The most common data block size in Oracle is 8192 bytes, and the default frame size for the Ethernet is 1500 bytes. A data block that is sent from one instance to another instance through the interconnect is shipped in at least four packets, and they are assembled back at the other instance. Although the default frame size is quite good for most of the Cache Fusion messages, for efficient block transfers it is strongly recommended that you use the jumbo frames that support a 9KB frame size.

While using jumbo frames, you need to ensure that the complete data transfer path —from the NIC to switches—supports jumbo frames. With jumbo frames, there is no fragmentation of packets during the data transfer and thus less latency and low CPU usage compared with smaller frame sizes.

Library Cache Effect in the Parsing

As in single-instance environments, it is best to avoid excessive and unnecessary parsing of SQL statements and PL/SQL code in Oracle RAC. Oracle RAC incurs additional overhead for parsing because many locks are global.

Library cache load and pin lock acquisition on a library object such as a PL/SQL procedure, package, or function requires global coordination by Global Enqueue Services (GES). Although parsing of SELECT statements does not require global lock coordination, Data Definition Language (DDL) statements do need messages to be sent across to master nodes for synchronization and lock grants.

Note that if the same SQL statement is sent for parsing on all nodes, these statements are individually parsed and executed on each node and probably with different execution plans, if there are differences in the SPFILE parameters among the nodes.

When many objects are compiled or parsed (even reparsed due to invalidations), it can give rise to delays and have significant impact on performance. Application developers must take care while using DDL statements, including GRANT and REVOKE commands, because these statements invalidate all objects that are referred to or by the PL/SQL object.

Repeated hard parsing and parse failures can severely limit scalability and harm the

https://t.me/learningnets

Technet24

application performance. Tuning methodologies that you use in a single-instance Oracle database are also applicable in Oracle RAC, and you should implement them wherever possible.

Commit Frequency

Certain wait events have multiple effects in Oracle RAC. For example, a “log file sync” wait event is associated with excessive commits, and application designers/developers usually are advised to reduce the frequency of their commits to reduce the burden on the LGWR (Log Writer) and reduce the number of LGWR writes to a log file. A close associate of “log file sync” is “log file parallel write.”

In Oracle RAC environments, log file sync has more implications because each commit or rollback needs SCN propagation and global synchronization, and may also require controlfile synchronization. Hence, you must exercise extra caution in Oracle RAC environments when you’re choosing a commit interval or frequency. Some business applications may need to commit more often than other applications require. Developers and users must be aware of the performance tradeoff that accompanies a high commit rate.

In a single-instance database, when a user commits or rolls back a transaction, the process waits on log file sync, and the background Oracle process (LGWR) will wait on a log file parallel write. On an Oracle RAC database, in addition to the latency due to these usual suspects, you must also consider the time consumed in SCN-related communication and controlfile writes while you’re calculating the cost of commit/rollback.

NOTE

Excessive cross-instance requests for changed or dirty blocks will cause more redo generation because the database needs to write out all changes made to a block before a copy of the block is transferred across to the requesting instance. In this case, LMS (Lock Manager Server process, also called the GCS, or Global Cache Services process) initiates the log synchronization and waits on a log file sync. Once log writing is complete, LMS can go ahead and ship the copy of the block to the requestor.

Summary

As we noted at the beginning of the book, you can compare Oracle RAC to a stereo amplifier. If the original music is good, you’ll hear good music with Oracle RAC. If the original music has a lot of noise, Oracle RAC will amplify that noise and the outcome sometimes might not be pleasant to the ears. A well-performing application in a single-instance environment will perform equally well or outperform when implemented on Oracle RAC.

https://t.me/learningnets

Developing applications for Oracle RAC doesn’t require any major changes other than the few considerations we’ve explained in this chapter. Other than a few exceptions, all single-instance application design best practices are applicable to Oracle RAC databases as well.

Good luck in designing and deploying a highly scalable application on Oracle RAC!

https://t.me/learningnets

Technet24

PART VI

Appendixes

https://t.me/learningnets

APPENDIX

A

Oracle RAC Reference

The Oracle database is very rich in terms of diagnostic data and statistics. It provides an enormous amount of statistics about most of the internal operations, which are exposed through a group of underlying views maintained by the database server. These views are normally accessible to the database administrator user *SYS*. They are called *dynamic performance views* because they are continuously updated while a database is open and in use, and their contents relate primarily to performance.

These views are built on the set of internal memory structures commonly known as *x$views*. These internal and undocumented views expose the runtime statistics of the relative data structures in a tabular format. Although these views appear to be regular database tables, they are not. You can select from these views, but you cannot update or alter them because the definitions are not stored in the dictionary.

The catalog.sql script (which is executed automatically by the Database Configuration Assistant during database creation) creates the public synonyms for the dynamic performance views. After installation, only user *SYS* or anyone with the DBA role has access to the dynamic performance tables.

In this appendix, we explore the important v$views used in Oracle Real Application Clusters for diagnostics and troubleshooting. These views contain the system-level statistics from the instance startup, and the contents are reset when the instance is shut

https://t.me/learningnets

Technet24

down because the data structures containing the data are reset during the instance shutdown. All of the information is cached in the System Global Area (SGA) of the respective instances.

For clusterwide information, you can query GV$ views. For every v$view, a corresponding GV$ view exists. When you query the GV$ view, the data from other instances are retrieved by parallel query mechanisms, and the initialization parameter parallel_max_servers should be set to a minimum of 2 for GV$ views to work.

Global Cache Services and Cache Fusion Diagnostics

The following views can be queried to get the details about the buffer cache contents and Global Cache Services (GCS) operations. Cache Fusion diagnostic information can also be obtained from these views.

V$CACHE

This view contains information about every cached block in the buffer cache. This view is a good source of information about cached blocks in local instances in Oracle RAC environments. Table A-1 shows information about the columns and data types in this view.

https://t.me/learningnets

Column	Data Type	Description
FILE#	Number	Datafile ID (FILE# from V$DATAFILE or FILE_ID in DBA_DATA_FILES).
BLOCK#	Number	Block number of the cached block.
CLASS#	Number	Class number (see Table A-2).
STATUS	Varchar2 (6)	Status of the block in SGA (see Table A-3).
XNC	Number	Number of times PCM locks were converted from X to null lock mode.
FORCED_READS	Number	Number of times the block was reread from cache because another instance forced it out by requesting this block in exclusive mode.
FORCED_WRITES	Number	Number of times GCS had to write the block to cache because this instance used the block and another instance requested a lock on the block in conflicting mode.
NAME	Varchar2 (30)	Name of the database object to which the block belongs.
PARTITION_NAME	Varchar2 (30)	Name of the partition. The column is null if the object is not partitioned.
KIND	Varchar2 (15)	Type of object (table, view procedure, and so on).
OWNER#	Number	Number of owner.
LOCK_ELEMENT_ADDR	Raw (4|8)	Address of lock element that contains the PCM lock covering the buffer.
LOCK_ELEMENT_NAME	Number	Name of lock element that contains the PCM lock covering the buffer.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-1. *V$CACHE Information*

Table A-2 shows block classes and their descriptions. Table A-3 defines the buffer states.

https://t.me/learningnets

Technet24

Class	Explanation
1	Data
2	Sort segment
3	Save undo block
4	Segment header
5	Save undo segment header
6	Freelist block
7	Extent map
8	Space management bitmap block
9	Space management index block

TABLE A-2. *Block Classes and Their Descriptions*

State	Explanation
0	Buffer free, unused
1	Buffer current, locked in X mode
2	Buffer current, locked in S mode
3	Consistent read buffer
4	Buffer being read
5	Buffer under media recovery
6	Buffer under instance recovery
7	Write clone buffer
8	Past Image buffer

TABLE A-3. *Buffer States*

V$CACHE_TRANSFER

This Oracle RAC view is similar to the V$CACHE view. V$CACHE_TRANSFER contains information from block headers in SGA that have been pinged at least once by the current instance relative to the database object. Table A-4 shows the contents of the V$CACHE_TRANSFER view.

https://t.me/learningnets

Column	Data Type	Description
FILE#	Number	Datafile ID (FILE# from V$DATAFILE or FILE_ID in DBA_DATA_FILES).
BLOCK#	Number	Block number.
CLASS#	Number	Class number (see Table A-2).
STATUS	Varchar2 (6)	Status of block in SGA (see Table A-3).
XNC	Number	Number of times PCM locks were converted from X to null lock mode.
FORCED_READS	Number	Number of times the block was reread from cache because another instance forced it out by requesting this block in exclusive mode.
FORCED_WRITES	Number	Number of times GCS had to write the block to cache because this instance had used the block and another instance had requested a lock on the block in a conflicting mode.
NAME	Varchar2 (30)	Name of the database object to which the block belongs.
PARTITION_NAME	Varchar2 (30)	Name of the partition. This is null if the object is not partitioned.
KIND	Varchar2 (15)	Type of object.
OWNER#	Number	Owner number.
LOCK_ELEMENT_ADDR	Raw (4|8)	Address of the lock element containing the PCM lock covering the buffer.
LOCK_ELEMENT_NAME	Number	Name of the lock element that contains the PCM lock covering the buffer.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-4. *V$CACHE_TRANSFER Information*

V$INSTANCE_CACHE_TRANSFER

This view keeps information about the transfer of cache blocks through the interconnect. These statistics can be used to find the number of blocks transferred from one instance to another instance using Cache Fusion. This view also shows how many block transfers incurred a delay or congestion. Table A-5 provides information about the columns and data types in this view.

https://t.me/learningnets

Technet24

Column	Data Type	Description
INSTANCE	Number	Instance number transferring the block.
CLASS	Varchar2 (18)	Class of cache block being transferred.
LOST	Number	Number of blocks that were sent by another instance but never arrived in this instance.
LOST_TIME	Number	Time waited for blocks that were sent by another instance but never arrived in this instance.
CR_BLOCK	Number	CR block transfers not affected by remote processing delays.
CR_BLOCK_TIME	Number	Total time waited for CR blocks from a particular instance.
CR_2HOP	Number	Number of CR blocks received by this instance from a particular instance after a two-way round-trip.
CR_2HOP_TIME	Number	Time waited for CR blocks received by this instance from a particular instance after a two-way round-trip.
CR_3HOP	Number	Number of CR blocks received by this instance from a particular instance after a three-way round-trip.
CR_3HOP_TIME	Number	Time waited for CR blocks received by this instance from a particular instance after a three-way round-trip.
CR_BUSY	Number	CR block transfers affected by remote contention.
CR_BUSY_TIME	Number	Time waited for CR blocks that were received by this instance from a particular instance and were delayed by a log flushed on the sending instance.
CR_CONGESTED	Number	CR block transfers affected by remote system load.
CR_CONGESTED_TIME	Number	Time waited for CR blocks that were received by this instance from a particular instance and were delayed because LMS was busy.
CURRENT_BLOCK	Number	Current block transfers not affected by remote system delays.
CURRENT_BLOCK_TIME	Number	Total time waited for current blocks from a particular instance.
CURRENT_2HOP	Number	Number of current blocks received by this instance from a particular instance after a two-way round-trip.
CURRENT_2HOP_TIME	Number	Time waited for current blocks received by this instance from a particular instance after a two-way round-trip.
CURRENT_3HOP	Number	Number of current blocks received by this instance from a particular instance after a three-way round-trip.
CURRENT_3HOP_TIME	Number	Time waited for current blocks received by this instance from a particular instance after a three-way round-trip.

https://t.me/learningnets

Column	Data Type	Description
CURRENT_BUSY	Number	Current block transfers affected by remote contention.
CURRENT_BUSY_TIME	Number	Time waited for current blocks that were received by this instance from a particular instance and were delayed by a log flushed on the sending instance.
CURRENT_CONGESTED	Number	Current block transfers affected by remote system load.
CURRENT_ CONGESTED_TIME	Number	Time waited for current blocks that were received by this instance from a particular instance and were delayed because LMS was busy.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-5. *V$INSTANCE_CACHE_TRANSFER Information*

V$CR_BLOCK_SERVER

This view stores statistics about CR block transfers across the instances (see Table A-6). The Global Cache Service process (LMS) from the holding instance constructs the CR block for the requesting instance and ships the CR version of the block using the interconnect. CR building is discussed in more detail in Chapter 11.

https://t.me/learningnets

Technet24

Column	Data Type	Description
CR_REQUESTS	Number	Number of CR blocks served due to remote CR block requests.
CURRENT_REQUEST	Number	Number of current blocks served due to remote CR block requests.
DATA_REQUEST	Number	Number of CR or current requests for data blocks.
UNDO_REQUESTS	Number	Number of CR requests for undo blocks.
TX_REQUESTS	Number	Number of CR requests for undo segment header blocks. Total number of requests would be equal to the sum of the DATA_REQUEST, UNDO_REQUESTS, and TX_REQUESTS columns.
OTHER_REQUESTS	Number	Number of CR requests for other types of blocks.
CURRENT_RESULTS	Number	Number of requests when no changes were rolled out of the block returned to the requesting instance.
PRIVATE_RESULTS	Number	Number of requests when changes were rolled out of the block returned to the requesting instance and only the requesting transaction can use the CR block.
ZERO_RESULTS	Number	Number of requests when changes were rolled out of the block returned to the requesting instance and only zero-XID transactions can use the block.
DISK_READ_RESULTS	Number	Number of requests when the requesting instance had to read the block from disk.

https://t.me/learningnets

Column	Data Type	Description
FAIL_RESULTS	Number	Number of requests failed and the requesting transaction reissued the request.
STALE	Number	For internal use only.
FAIRNESS_DOWN_CONVERTS	Number	Number of times the instance receiving the request has down-converted an X lock on a block because it was not modifying the block.
FAIRNESS_CLEARS	Number	Number of times the "fairness counter" was cleared. This counter tracks the number of times a block was modified after it was served.
FREE_GC_ELEMENTS	Number	Number of times a request was received from another instance and X lock had no buffers.
FLUSHES	Number	Number of times LMS flushed the logs.
FLUSHES_QUEUED	Number	Number of flushes queued by LMS.
FLUSH_QUEUE_FULL	Number	Number of occasions the flush queue was full.
FLUSH_MAX_TIME	Number	Maximum flush time.
LIGHT_WORKS	Number	Number of times the light-work rule was evoked (see Chapter 11).
ERRORS	Number	Number of occasions an error was signaled by the LMS process.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-6. *V$CR_BLOCK_SERVER Information*

V$CURRENT_BLOCK_SERVER

This view keeps statistics about the current block transfer across the instances (see Table A-7). The GCS process (LMS) from the holding instance ships the current block to the requesting instance after flushing the required recovery information to the redo log buffer. This view provides the most important information about how long the holding instance waited to flush the information before sending the block to the requestor.

https://t.me/learningnets

Technet24

Column	Data Type	Description
PIN0	Number	Number of pins taking less than 100 microseconds (ms).
PIN1	Number	Number of pins taking less than 1 ms.
PIN10	Number	Number of pins taking 1 to 10 ms.
PIN100	Number	Number of pins taking 10 to 100 ms.
PIN1000	Number	Number of pins taking 100 to 1000 ms.
PIN10000	Number	Number of pins taking 1000 to 10,000 ms.
FLUSH1	Number	Number of flushes taking less than 1 ms.
FLUSH10	Number	Number of flushes taking 1 to 10 ms.
FLUSH100	Number	Number of flushes taking 10 to 100 ms.
FLUSH1000	Number	Number of flushes taking 100 to 1000 ms.
FLUSH10000	Number	Number of flushes taking 1000 to 10,000 ms.
WRITE1	Number	Number of writes taking less than 1 ms.
WRITE10	Number	Number of writes taking 1 to 10 ms.
WRITE100	Number	Number of writes taking 10 to 100 ms.
WRITE1000	Number	Number of writes taking 100 to 1000 ms.
WRITE10000	Number	Number of writes taking 1000 to 10,000 ms.
CLEANDC	Number	Reserved for internal use.
RCVDC	Number	Number of lock down-converts to S (shared) caused by instance recovery.
QUEUEDC	Number	Number of queued lock down-converts to NULL.
EVICTDC	Number	Number of lock down-converts to NULL caused by an SGA shrink.
WRITEDC	Number	Number of dirty blocks in read-mostly objects that were written and the X (exclusive) lock down-converts to S (shared locks).
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-7. *V$CURRENT_BLOCK_SERVER Information*

V$GC_ELEMENT

This view displays one-to-one information for each global cache resource used by the

https://t.me/learningnets

buffer cache (see Table A-8). In this view, each global cache resource used by a buffer cache has an entry.

Column	Data Type	Description
INDX	Number	Platform-specific lock manager identifier; can be joined with the V$CACHE_LOCK view.
CLASS	Number	Platform-specific lock manager identifier; can be joined with the V$CACHE_LOCK view.
GC_ELEMENT_NAME	Number	Name of the lock element that contains the PCM lock covering the buffer; if more than one buffer has the same address, these buffers are covered by the same PCM lock.
MODE_HELD	Number	Value for the lock mode held. This is often 3 for share or 5 for exclusive.
BLOCK_COUNT	Number	Number of blocks covered by the PCM lock.
RELEASING	Number	Nonzero if the PCM lock is downgraded.
ACQUIRING	Number	Nonzero if the PCM lock is upgraded.
WRITING	Number	If the GC_ELEMENT is being written, this is the write status.
RECOVERING	Number	If the GC_ELEMENT is being recovered, this is the recovery status.
LOCAL	Number	A value of 0 if the GC_ELEMENT is local, 1 if it is global.
FLAGS	Number	Process-level flags for the lock element.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-8. *V$GC_ELEMENT Information*

Global Enqueue Services Diagnostics

The following views can be queried to get details about statistics for Global Enqueue Services. Few of these views are extensively used in single-instance environments to query enqueue statistics.

V$LOCK

This view maintains information about locks held within a database and outstanding requests for locks or latches (see Table A-9). Table A-10 shows the lock modes.

https://t.me/learningnets

Technet24

Column	Data Type	Description
ADDR	Raw (4|8)	Address of the lock state object.
KADDR	Raw (4|8)	Address of the lock.
SID	Number	Session identifier holding the lock.
TYPE	Varchar2 (2)	Type of resource, which could be user or system type.
ID1	Number	Resource identifier 1.
ID2	Number	Resource identifier 2.
LMODE	Number	Lock mode held (see Table A-10).
REQUEST	Number	Mode of lock requested (see Table A-10).
CTIME	Number	Time since the current mode was granted.
BLOCK	Number	A value of 1 if the lock is blocking another lock; 0 otherwise.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-9. *V$LOCK Information*

Lock Mode (LMODE)	Description
0	None
1	null (NULL)
2	row-S (SS)
3	row-X (SX)
4	share (S)
5	SRow-X (SSX)
6	exclusive (X)

TABLE A-10. *Lock Modes*

V$GES_BLOCKING_ENQUEUE

This view maintains information about locks that are being blocked or are blocking others as well as locks that are known to the lock manager (see Table A-11). The content of this view is a subset of V$GES_ENQUEUE because this view maintains information only about blocked locks, whereas V$GES_ENQUEUE maintains all locks known to the lock manager.

https://t.me/learningnets

Column	Data Type	Description
HANDLE	(Raw 4|8)	Pointer to the lock.
GRANT_LEVEL	Varchar2 (9)	Level granted to the lock.
REQUEST_LEVEL	Varchar2 (9)	Level requested for the lock.
RESOURCE_NAME1	Varchar2 (30)	Resource name of the lock.
RESOURCE_NAME2	Varchar2 (30)	Resource name of the lock.
PID	Number	Process identifier holding the lock.
TRANSACTION_ID0	Number	Lower 4 bytes of the transaction identifier the lock belongs to.
TRANSACTION_ID1	Number	Upper 4 bytes of the transaction identifier the lock belongs to.
GROUP_ID	Number	Group identifier of the lock.
OPEN_OPT_DEADLOCK	Number	A value of 1 if the DEADLOCK option is set; otherwise 0.
OPEN_OPT_PERSISTENT	Number	A value of 1 if the PERSISTENT option is set; otherwise 0.

https://t.me/learningnets

Technet24

Column	Data Type	Description
OPEN_OPT_PROCESS_OWNED	Number	A value of 1 if the PROCESS_OWNED option is set; otherwise 0.
OPEN_OPT_NO_XID	Number	A value of 1 if the NO_XID option is set; otherwise 0.
CONVERT_OPT_GETVALUE	Number	A value of 1 if the GETVALUE convert option is set; otherwise 0.
CONVERT_OPT_PUTVALUE	Number	A value of 1 if the PUTVALUE convert option is set; otherwise 0.
CONVERT_OPT_NOVALUE	Number	A value of 1 if the NOVALUE convert option is set; otherwise 0.
CONVERT_OPT_DUBVALUE	Number	A value of 1 if the DUBVALUE convert option is set; otherwise 0.
CONVERT_OPT_NOQUEUE	Number	A value of 1 if the NOQUEUE convert option is set; otherwise 0.
CONVERT_OPT_EXPRESS	Number	A value of 1 if the EXPRESS convert option is set; otherwise 0.
CONVERT_OPT_NODEADLOCKWAIT	Number	A value of 1 if the NODEADLOCKWAIT convert option is set; otherwise 0.
CONVERT_OPT_NODEADLOCKBLOCK	Number	A value of 1 if the NODEADLOCKBLOCK convert option is set; otherwise 0.
WHICH_QUEUE	Number	Which queue the lock is currently located in: 0 for Null Queue, 1 for Granted Queue, or 2 for Convert Queue.
STATE	Varchar2 (64)	State of the lock.
AST_EVENT0	Number	Last know AST event.
OWNED_NODE	Number	Node identifier owning the lock.
BLOCKED	Number	A value of 1 if the lock request is blocked by others; otherwise 0.
BLOCKER	Number	A value of 1 if the lock is blocking others; otherwise 0.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-11. *V$GES_BLOCKING_ENQUEUE Information*

V$ENQUEUE_STATISTICS

This view displays details about enqueue statistics in the instance (see Table A-12). Most of the enqueues are global in nature and are visible across instances. Table A-13

https://t.me/learningnets

shows enqueue types and descriptions.

Column	Data Type	Description
EQ_NAME	Varchar2 (64)	Name of the enqueue that the request is for.
EQ_TYPE	Varchar2 (2)	Type of enqueue (see Table A-13).
REQ_REASON	Varchar2 (64)	Reason for the enqueue request.
TOTAL_REQ#	Number	Total number of times the enqueue was requested or converted.
TOTAL_WAIT#	Number	Number of times the enqueue request or conversion resulted in a wait.
SUCC_REQ#	Number	Number of times the enqueue request or conversion was granted.
FAILED_REQ#	Number	Number of times the enqueue request or conversion failed.
CUM_WAIT_TIME	Number	Amount of time (in milliseconds) spent waiting for the enqueue or conversion.
REQ_DESCRIPTION	Varchar2 (4000)	Description of the enqueue request.
EVENT#	Number	Number for the event.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-12. *V$ENQUEUE_STATISTICS Information*

Name	Enqueue Type	Enqueue Description
AB	ABMR process initialized	Lock held to ensure that the ABMR process is initialized.
AB	ABMR process start/stop	Lock held to ensure that only one ABMR is started in the cluster.
AD	Allocate AU	Synchronizes accesses to a specific ASM disk AU.
AD	Deallocate AU	Synchronizes accesses to a specific ASM disk AU.
AD	Relocate AU	Synchronizes accesses to a specific ASM disk AU.
AE	Lock	Prevents dropping an edition in use.
AF	Task serialization	Serializes access to an advisor task.
AG	Contention	Synchronizes generation use of a particular workspace.
AM	ASM Password File Update	Allows one ASM password file update per cluster at a time.
AM	Client registration	Registers the DB instance to the ASM client state object hash.
AM	Shutdown	Prevents the DB instance registration during ASM instance shutdown.
AM	Rollback COD reservation	Reserves a rollback COD entry.

https://t.me/learningnets

Technet24

Name	Enqueue Type	Enqueue Description
AM	Background COD reservation	Reserves a background COD entry.
AM	ASM cache freeze	Starts the ASM cache freeze.
AM	ASM ACD Relocation	Blocks the ASM cache freeze.
AM	Group use	Client group use.
AM	Group block	ASM group block.
AM	ASM File Destroy	Prevents same file deletion race.
AM	ASM User	Prevents a user from being dropped if it owns any open files.
AM	ASM Amdu Dump	Allows only one AMDU dump during a block read failure.
AM	ASM file descriptor	Serializes access to the ASM file descriptors.
AM	ASM disk-based alloc/dealloc	Synchronizes disk-based allocations/deallocations.
AM	Block repair	Serializes block repairs.
AM	ASM reserved	Checks ID1 of a call for a specific purpose.
AM	Disk offline	Synchronizes disk offline operations.
AO	Contention	Synchronizes access to objects and scalar variables.
AS	Service activation	Synchronizes new service activation.
AT	Contention	Serializes alter tablespace operations.
AV	Volume relocate	Serializes relocating volume extents.
AV	AVD client registration	Serializes instance registration and first DG use.
AV	Add/enable first volume in DG	Serializes taking the AVD DG enqueue.
AV	Persistent DG number	Prevents DG number collisions.
AW	User access for AW	Synchronizes user accesses to a particular workspace.
AW	AW$ table lock	Global access synchronization to the AW$ table.
AW	AW state lock	Row lock synchronization for the AW$ table.
AW	AW generation lock	In-use generation state for a particular workspace.
AY	Contention	Affinity synchronization.
BB	2PC across RAC instances	2PC distributed transaction branch across Oracle RAC instances.
BF	PMON Join Filter cleanup	PMON bloom filter recovery.
BF	Allocation contention	Allocates a bloom filter in a parallel statement.
BR	Multisection restore section	Lock held to serialize section access during multisection restore.
BR	Proxy-copy	Lock held to allow cleanup from backup mode during an RMAN proxy-copy backup.
BR	File shrink	Lock held to prevent file from decreasing in physical size during RMAN backup.
BR	Space info datafile hdr update	Lock held to prevent multiple processes from updating the headers at the same time.
BR	Request autobackup	Lock held to request controlfile auto-backups.

https://t.me/learningnets

https://t.me/learningnets

Technet24

Name	Enqueue Type	Enqueue Description
BR	Perform autobackup	Lock held to perform a new controlfile auto-backup.
BR	Multisection restore header	Lock held to serialize file header access during multisection restore.
CA	Contention	Synchronizes various IO calibration runs.
CF	Contention	Synchronizes accesses to the controlfile.
CI	Contention	Coordinates cross-instance function invocations.
CL	Drop label	Synchronizes accesses to label cache when a label is being dropped.
CL	Compare labels	Synchronizes accesses to label cache for label comparison.
CM	Instance	Indicates the ASM disk group is mounted.
CM	Gate	Serializes access to the instance enqueue.
CM	Diskgroup dismount	Serializes an ASM disk group dismount.
CN	Race with init	During descriptor initialization.
CN	Race with txn	During registration.
CN	Race with reg	During a transaction commit to see concurrent registrations.
CO	Master slave det	Enqueue held by master in cleanout optimization.
CQ	Contention	Serializes access to cleanup client query cache registrations.
CR	Block range reuse ckpt	Coordinates fast block range reuse checkpoint.
CT	Reading	Lock held to ensure that change-tracking data remains in existence until a reader is done with it.
CT	Global space management	Lock held during change-tracking space management operations that affect the entire tracking file.
CT	Local space management	Lock held during change-tracking space management operations that affect just the data for one thread.
CT	CTWR process start/stop	Lock held to ensure that only one CTWR process is started in a single instance.
CT	State	Lock held while enabling or disabling change tracking, to ensure that it is only enabled or disabled by one user at a time.
CT	Change stream ownership	Lock held by one instance while change tracking is enabled, to guarantee access to thread-specific resources.
CT	State change gate 2	Lock held while enabling or disabling change tracking in Oracle RAC.
CT	State change gate 1	Lock held while enabling or disabling change tracking in Oracle RAC.
CU	Contention	Recovers cursors in case of death while compiling.
CX	Index Specific Lock	Index-specific lock on CTX index.
DB	Contention	Synchronizes the modification of database-wide supplemental logging attributes.
DD	Contention	Synchronizes local accesses to ASM disk groups.
DF	Contention	Enqueue held by foreground or DBWR when a datafile is brought online in Oracle RAC.
DG	Contention	Synchronizes accesses to ASM disk groups.

https://t.me/learningnets

https://t.me/learningnets

Technet24

Name	Enqueue Type	Enqueue Description
DL	Contention	Lock to prevent index DDL during direct load.
DM	Contention	Enqueue held by foreground or DBWR to synchronize database mount/open with other operations.
DN	Contention	Serializes group number generations.
DO	Startup of MARK process	Synchronizes startup of MARK process.
DO	Staleness Registry create	Synchronizes Staleness Registry creation.
DO	Disk online recovery	Synchronizes disk online operations and their recovery.
DO	Disk online operation	Represents an active disk online operation.
DO	Disk online	Synchronizes disk online operations and their recovery.
DP	Contention	Synchronizes access to LDAP parameters.
DR	Contention	Serializes the active distributed recovery operation.
DS	Contention	Prevents a database suspend during LMON reconfiguration.
DT	Contention	Serializes changing the default temporary tablespace and user creation.
DV	Contention	Synchronizes access to lower-version Diana (PL/SQL intermediate representation).
DW	Contention	Serializes in-memory dispenser operations.
DX	Contention	Serializes tightly coupled distributed transaction branches.
FA	Access file	Synchronizes accesses to open ASM files.
FB	Contention	Ensures that only one process can format data blocks in auto-segment-space-managed tablespaces.
FC	Open an ACD thread	LGWR opens an ACD thread (ACD is Active Change Directory, a data structure like redo logs for an ASM Instance).
FC	Recover an ACD thread	SMON recovers an ACD thread.
FD	Flashback on/off	Synchronization.
FD	Marker generation	Synchronization.
FD	Tablespace flashback on/off	Synchronization.
FD	Flashback coordinator	Synchronization.
FD	Restore point create/drop	Synchronization.
FD	Flashback logical operations	Synchronization.
FE	Contention	Serializes flashback archive recovery.
FG	Serialize ACD relocate	Only one process in the cluster may do ACD relocation in a disk group.
FG	LGWR redo generation enq race	Resolves a race condition to acquire Disk Group Redo Generation Enqueue.
FG	FG redo generation enq race	Resolves a race condition to acquire Disk Group Redo Generation Enqueue.
FL	Flashback db command	Enqueue used to synchronize Flashback Database and deletion of flashback logs.

https://t.me/learningnets

https://t.me/learningnets

Technet24

Name	Enqueue Type	Enqueue Description
FL	Flashback database log	Synchronization.
FM	Contention	Synchronizes access to global file mapping state.
FP	Global fob contention	Synchronizes various File Object (FOB) operations.
FR	Recover the thread	Waits for lock domain detach.
FR	Use the thread	Indicates this ACD thread is alive.
FR	Contention	Begins recovery of the disk group.
FS	Contention	Enqueue used to synchronize recovery and file operations or synchronize dictionary check.
FT	Disable LGWR writes	Prevents LGWR from generating a redo in this thread.
FT	Allow LGWR writes	Allows LGWR to generate a redo in this thread.
FU	Contention	Serializes the capture of the DB Feature Usage and High Watermark statistics.
FX	Issue ACD Xtnt Relocation CIC	ARB relocates ACD extent.
HD	Contention	Serializes accesses to ASM SGA data structures.
HP	Contention	Synchronizes accesses to queue pages.
HQ	Contention	Synchronizes the creation of new queue IDs.
HV	Contention	Lock used to broker the high-water mark during parallel inserts.
HW	Contention	Lock used to broker the high-water mark during parallel inserts.
ID	Contention	Lock held to prevent other processes from performing a controlfile transaction while NID is running.
IL	Contention	Synchronizes accesses to internal label data structures.
IM	Contention for blr	Serializes block recovery for In Memory Undo (IMU) transaction.
IR	Contention	Synchronizes instance recovery.
IR	Contention 2	Synchronizes parallel instance recovery and shutdown immediate.
IS	Contention	Enqueue used to synchronize instance state changes.
IT	Contention	Synchronizes accesses to a temp object's metadata.
JD	Contention	Synchronizes dates between the job queue coordinator and slave processes.
JI	Contention	Lock held during materialized view operations (such as refresh and alter) to prevent concurrent operations on the same materialized view.
JQ	Contention	Lock to prevent multiple instances from running a single job.
JS	wdw op	Lock got when doing a window open/close.
JS	job run lock – synchronize	Lock to prevent a job from running elsewhere.
JS	aq sync	Scheduler event code and AQ synchronization.
JS	evt notify	Lock got during event notification.
JS	Contention	Synchronizes accesses to the job cache.
JS	evtsub drop	Lock got when dropping subscriber to event queue.

https://t.me/learningnets

https://t.me/learningnets

Technet24

Name	Enqueue Type	Enqueue Description
JS	evtsub add	Lock got when adding subscriber to event queue.
JS	q mem clnup lck	Lock obtained when cleaning up queue memory.
JS	sch locl enqs	Scheduler of nonglobal enqueues.
JS	queue lock	Lock on internal scheduler queue.
JS	job recov lock	Lock to recover jobs running on crashed Oracle RAC instance.
JX	cleanup of queue	Releases SQL statement resources.
JX	SQL statement queue	Statement queue
KD	determine DBRM master	Determines DBRM master.
KM	Contention	Synchronizes various Resource Manager operations.
KO	fast object checkpoint	Coordinates fast object checkpoint.
KP	Contention	Synchronizes Data Pump process startup.
KQ	access ASM attribute	Synchronization of ASM cached attributes.
KT	Contention	Synchronizes accesses to the current Resource Manager plan.
MD	Contention	Lock held during materialized view log DDL statements.
MH	Contention	Lock used for recovery when setting Mail Host for AQ e-mail notifications.
MK	Contention	Changing values in enc$.
ML	Contention	Lock used for recovery when setting Mail Port for AQ e-mail notifications.
MN	Contention	Synchronizes updates to the LogMiner dictionary and prevents multiple instances from preparing the same LogMiner session.
MO	Contention	Serializes MMON operations for restricted sessions.
MR	Contention	Lock used to coordinate media recovery with other uses of datafiles.
MR	standby role transition	Lock used to disallow concurrent standby role transition attempt.
MS	Contention	Lock held during materialized view refresh to set up MV log.
MV	datafile move	Held during online datafile move operation or cleanup.
MW	Contention	Serializes the calibration of the manageability schedules with the Maintenance window.
MX	sync storage server info	Lock held to generate a response to the storage server information request when an instance is starting up.
OC	Contention	Synchronizes write accesses to the outline cache.
OD	Serializing DDLs	Lock to prevent concurrent online DDLs.
OL	Contention	Synchronizes accesses to a particular outline name.
OQ	xsoqhiFlush	Synchronizes access to OLAPI history flushing.
OQ	xsoq*histrecb	Synchronizes access to OLAPI history parameter CB.
OQ	xsoqhiAlloc	Synchronizes access to OLAPI history allocation.
OQ	Xsoqhistrecb	Synchronizes access to OLAPI history globals.

https://t.me/learningnets

https://t.me/learningnets

Technet24

Name	Enqueue Type	Enqueue Description
OQ	xsoqhiClose	Synchronizes access to OLAPI history closing.
OT	Generic Lock	CTX generic locks.
OW	Termination	Terminates the wallet context.
OW	Initialization	Initializes the wallet context.
PD	Contention	Prevents others from updating the same property.
PE	Contention	Synchronizes system parameter updates.
PF	Contention	Synchronizes accesses to the password file.
PG	Contention	Synchronizes global system parameter updates.
PH	Contention	Lock used for recovery when setting Proxy for AQ HTTP notifications.
PI	Contention	Communicates remote Parallel Execution Server Process creation status.
PL	Contention	Coordinates plug-in operation of transportable tablespaces.
PR	Contention	Synchronizes process startup.
PS	Contention	Parallel Execution Server Process reservation and synchronization.
PT	Contention	Synchronizes access to ASM Partner Status Table (PST) metadata.
PV	Syncshut	Synchronizes instance shutdown_slvstart.
PV	Syncstart	Synchronizes slave start_shutdown.
PW	perwarm status in dbw0	DBWR 0 holds enqueue indicating prewarmed buffers present in cache.
PW	flush prewarm buffers	Direct Load needs to flush prewarmed buffers if DBWR 0 holds enqueue.
RB	Contention	Serializes ASM rollback recovery operations.
RC	Result Cache: Contention	Coordinates access to a resultset.
RD	RAC load	Updates Oracle RAC load info.
RE	block repair contention	Synchronizes block repair/resilvering operations.
RF	DG Broker Current File ID	Identifies which configuration metadata file is current.
RF	FSFO Primary Shutdown suspended	Records when FSFO Primary Shutdown is suspended.
RF	FSFO Observer Heartbeat	Captures recent Fast-Start Failover Observer heartbeat information.
RF	RF – Database Automatic Disable	Means for detecting when the database is being automatically disabled.
RF	synchronization: critical AI	Synchronizes critical apply instance among primary instances.
RF	new AI	Synchronizes selection of the new apply instance.
RF	synchronization: AIFO master	Synchronizes apply instance failure detection and failover operation.
RF	Atomicity	Ensures atomicity of log transport setup.
RF	synch: DG Broker metadata	Ensures read/write atomicity of DG configuration metadata.
RK	set key	Wallet master key rekey.

https://t.me/learningnets

https://t.me/learningnets

Technet24

Name	Enqueue Type	Enqueue Description
RL	RAC wallet lock	Oracle RAC wallet lock.
RN	Contention	Coordinates Nab computations of online logs during recovery.
RO	Contention	Coordinates flushing of multiple objects.
RO	fast object reuse	Coordinates fast object reuse.
RP	Contention	Enqueue held when resilvering is needed or when data block is repaired from mirror.
RR	Contention	Concurrent invocation of DBMS_WORKLOAD_* package API.
RS	prevent file delete	Lock held to prevent deleting file to reclaim space.
RS	prevent aging list update	Lock held to prevent aging list update.
RS	record reuse	Lock held to prevent file from accessing while reusing circular record.
RS	file delete	Lock held to prevent file from accessing during space reclamation.
RS	write alert level	Lock held to write alert level.
RS	read alert level	Lock held to read alert level.
RS	persist alert level	Lock held to make alert level persistent.
RT	Contention	Thread locks held by LGWR, DBW0, and RVWR to indicate mounted or open status.
RT	thread internal enable/disable	Thread locks held by CKPT to synchronize thread enable and disable.
RU	Waiting	Results of rolling migration CIC.
RU	Contention	Serializes rolling migration operations.
RW	MV metadata contention	Lock held by CREATE/ALTER/DROP materialized view while updating materialized view flags in detail tables.
RX	relocate extent	Synchronizes relocating ASM extents.
SB	table instantiation	Synchronizes table instantiation and EDS operations.
SB	logical standby metadata	Synchronizes Logical Standby metadata operations.
SE	Contention	Synchronizes transparent session migration operations.
SF	Contention	Lock used for recovery when setting Sender for AQ e-mail notifications.
SH	Contention	Should seldom see this contention because this enqueue is always acquired in no-wait mode.
SI	Contention	Prevents multiple streams table instantiations.
SJ	Slave Task Cancel	Serializes cancelling a task executed by a slave process.
SK	Contention	Serializes shrink of a segment.
SL	get lock for undo	Sends lock request for undo to LCK0.
SL	get lock	Sends lock request to LCK0.
SL	escalate lock	Sends lock escalate to LCK0.
SO	Contention	Synchronizes access to Shared Object (PL/SQL Shared Object Manager).
SP	contention 1	(1) Due to one-off patch.

https://t.me/learningnets

https://t.me/learningnets

Technet24

Name	Enqueue Type	Enqueue Description
SP	contention 2	(2) Due to one-off patch.
SP	contention 3	(3) Due to one-off patch.
SP	contention 4	(4) Due to one-off patch.
SQ	Contention	Lock to ensure that only one process can replenish the sequence cache.
SR	Contention	Coordinates replication/streams operations.
SS	Contention	Ensures that sort segments created during parallel DML operations aren't prematurely cleaned up.
ST	Contention	Synchronizes space management activities in dictionary-managed tablespaces.
SU	Contention	Serializes access to SaveUndo segment.
SW	Contention	Coordinates the "alter system suspend" operation.
TA	Contention	Serializes operations on undo segments and undo tablespaces.
TB	SQL Tuning Base Cache Update	Synchronizes writes to the SQL Tuning Base Existence Cache.
TB	SQL Tuning Base Cache Load	Synchronizes writes to the SQL Tuning Base Existence Cache.
TC	contention 2	Lock of setup of a unique tablespace checkpoint in null mode.
TC	Contention	Lock held to guarantee uniqueness of a tablespace checkpoint.
TD	KTF dump entries	KTF dumping time/SCN mappings in SMON_SCN_TIME table.
TE	KTF broadcast	KTF broadcasting.
TF	Contention	Serializes dropping of a temporary file.
TH	metric threshold evaluation	Serializes threshold in-memory chain access.
TK	Auto Task Slave Lockout	Serializes spawned Autotask Slaves.
TK	Auto Task Serialization	Lock held by MMON to prevent other MMON spawning of Autotask Slave.
TL	Contention	Serializes threshold log table read and update.
TM	Contention	Synchronizes accesses to an object.
TO	Contention	Synchronizes DDL and DML operations on a temp object.
TP	Contention	Lock held during purge and dynamic reconfiguration of fixed tables.
TQ	TM contention	TM access to the queue table.
TQ	DDL-INI contention	Streams DDL on the queue table.
TQ	INI contention	TM access to the queue table.
TQ	DDL contention	TM access to the queue table.
TS	Contention	Serializes accesses to temp segments.
TT	Contention	Serializes DDL operations on tablespaces.
TW	Contention	Lock held by one instance to wait for transactions on all instances to finish.
TX	allocate ITL entry	Allocates an ITL entry in order to begin a transaction.

https://t.me/learningnets

https://t.me/learningnets

Technet24

Name	Enqueue Type	Enqueue Description
TX	index contention	Lock held on an index during a split to prevent other operations on it.
TX	row lock contention	Lock held on a particular row by a transaction to prevent other transactions from modifying it.
TX	Contention	Lock held by a transaction to allow other transactions to wait for it.
UL	Contention	Lock used by user applications.
US	Contention	Lock held to perform DDL on the undo segment.
WA	Contention	Lock used for recovery when setting the watermark for memory usage in AQ notifications.
WF	Contention	Serializes the flushing of snapshots.
WG	lock fso	Acquires lobid local enqueue when locking File System Object (FSO).
WG	delete fso	Acquires lobid local enqueue when deleting File System Object (FSO).
WL	RAC-wide SGA contention	Serializes access to Oracle RAC–wide SGA.
WL	RFS global state contention	Serializes access to RFS global state.
WL	Contention	Coordinates access to redo log files and archive logs.
WL	Test access/locking	Tests redo transport access/locking.
WM	WLM Plan activation	Synchronizes new WLM plan activation.
WP	Contention	Handles concurrency between purging and baselines.
WR	Contention	Coordinates access to logs by Async LNS and ARCH/FG.
XC	XDB Configuration	Lock obtained when incrementing XDB configuration version number.
XD	ASM disk OFFLINE	Serializes OFFLINE Exadata disk operations.
XD	ASM disk ONLINE	Serializes ONLINE Exadata disk operations.
XD	ASM disk drop/add	Serializes Auto Drop/Add Exadata disk operations.
XH	Contention	Lock used for recovery when setting No Proxy Domains for AQ HTTP notifications.
XL	fault extent map	Keeps multiple processes from faulting in the same extent chunk.
XQ	Recovery	Prevents relocation during _recovery_asserts checking.
XQ	Relocation	Waits for recovery before doing relocation.
XQ	Purification	Waits for relocation before doing block purification.
XR	database force logging	Lock held during database force logging mode.
XR	quiesce database	Lock held during database quiesce.
XY	Contention	Lock used for internal testing.
ZA	add std audit table partition	Lock held to add partition to standard audit table.
ZF	add fga audit table partition	Lock held to add partition to fine-grained access (FGA) audit table.
ZG	Contention	Coordinates file group operations.
ZH	compression analysis	Synchronizes analysis and inserts into compression$ as well as prevents multiple threads analyzing the same table during a load.
ZZ	update hash tables	Lock held for updating global context hash tables.

https://t.me/learningnets

TABLE A-13. *Enqueue Types and Descriptions*

V$LOCKED_OBJECT

This view displays information about DML locks acquired by different transactions in databases with their mode held (see Table A-14). It also keeps the transaction's ID (transaction ID is XIDUSN.XIDSLOT.XIDSQN).

Column	Data Type	Description
XIDUSN	Number	Number of the undo segment
XIDSLOT	Number	Number of the slot
XIDSQN	Number	Sequence number of the transaction
OBJECT_ID	Number	ID of object being locked by this transaction
SESSION_ID	Number	Session ID responsible for this transaction and lock hold
ORACLE_USERNAME	Varchar2 (30)	Oracle user name
OS_USER_NAME	Varchar2 (30)	OS user name
PROCESS	Varchar2 (14)	OS process ID
LOCKED_MODE	Number	Lock mode; same as LMODE column of V$LOCK

TABLE A-14. *V$LOCKED_OBJECT Information*

V$GES_STATISTICS

This view displays miscellaneous statistics for GES (see Table A-15).

Column	Data Type	Description
STATISTIC#	Number	Statistic ID.
NAME	Varchar2 (64)	Name of statistic.
VALUE	Number	Value attached with this statistic.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-15. *V$GES_STATISTICS Information*

https://t.me/learningnets

Technet24

V$GES_ENQUEUE

This view displays information about all locks known to the lock manager (see Table A-16). It also keeps the grant level, request level, and other related information.

https://t.me/learningnets

Column	Data Type	Description
HANDLE	(Raw 4 | 8)	Pointer to the lock.
GRANT_LEVEL	Varchar2 (9)	Level granted to the lock.
REQUEST_LEVEL	Varchar2 (9)	Level requested for the lock.
RESOURCE_NAME1	Varchar2 (30)	Resource name of the lock.
RESOURCE_NAME2	Varchar2 (30)	Resource name of the lock.
PID	Number	Process ID holding the lock.
TRANSACTION_ID0	Number	Lower 4 bytes of the transaction ID to which the lock belongs.
TRANSACTION_ID1	Number	Upper 4 bytes of the transaction ID to which the lock belongs.
GROUP_ID	Number	Group ID of the lock.
OPEN_OPT_DEADLOCK	Number	A value of 1 if the DEADLOCK option is set; otherwise 0.
OPEN_OPT_PERSISTENT	Number	A value of 1 if the PERSISTENT option is set; otherwise 0.
OPEN_OPT_PROCESS_OWNED	Number	A value of 1 if the PROCESS_OWNED option is set; otherwise 0.
OPEN_OPT_NO_XID	Number	A value of 1 if the NO_XID option is set; otherwise 0.
CONVERT_OPT_GETVALUE	Number	A value of 1 if the GETVALUE convert option is set; otherwise 0.
CONVERT_OPT_PUTVALUE	Number	A value of 1 if the PUTVALUE convert option is set; otherwise 0.
CONVERT_OPT_NOVALUE	Number	A value of 1 if the NOVALUE convert option is set; otherwise 0.
CONVERT_OPT_DUBVALUE	Number	A value of 1 if the DUBVALUE convert option is set; otherwise 0.
CONVERT_OPT_NOQUEUE	Number	A value of 1 if the NOQUEUE convert option is set; otherwise 0.
CONVERT_OPT_EXPRESS	Number	A value of 1 if the EXPRESS convert option is set; otherwise 0.
CONVERT_OPT_NODEADLOCKWAIT	Number	A value of 1 if the NODEADLOCKWAIT convert option is set; otherwise 0.
CONVERT_OPT_NODEADLOCKBLOCK	Number	A value of 1 if the NODEADLOCKBLOCK convert option is set; otherwise 0.
WHICH_QUEUE	Number	Tells in which queue the lock is currently located: 0 for Null Queue, 1 for Granted Queue, or 2 for Convert Queue.
STATE	Varchar2 (64)	State of the lock.
AST_EVENT0	Number	Last known AST event.
OWNER_NODE	Number	Node ID owning the lock.
BLOCKED	Number	A value of 1 if the lock request is blocked by others; otherwise 0.
BLOCKER	Number	A value of 1 if the lock is blocking others; otherwise 0.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

https://t.me/learningnets

Technet24

TABLE A-16. *V$GES_ENQUEUE Information*

V$GES_CONVERT_LOCAL

This view maintains information about all local GES operations (see Table A-17). It displays such information as average convert time, count, and number of conversions. Lock conversion types are listed in Table A-18.

Column	Data Type	Description
INST_ID	Number	Instance ID.
CONVERT_TYPE	Varchar2 (64)	Conversion type (see Table A-18).
AVERAGE_CONVERT_TIME	Number	Average conversion time for each lock in hundredths of a second.
CONVERT_COUNT	Number	Number of conversions.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-17. *V$GES_CONVERT_LOCAL Information*

https://t.me/learningnets

Type	Description
NULL -> SS	NULL mode to subshared mode
NULL-> SX	NULL mode to shared exclusive mode
NULL -> S	NULL mode to shared mode
NULL -> SSX	NULL mode to subshared exclusive mode
NULL -> X	NULL mode to exclusive mode
SS -> SX	Subshared mode to shared exclusive mode
SS -> S	Subshared mode to shared mode
SS -> SSX	Subshared mode to subshared exclusive mode
SS-> X	Subshared mode to exclusive mode
SX -> S	Shared exclusive mode to shared mode
SX -> SSX	Shared exclusive mode to subshared exclusive mode
SX -> X	Shared exclusive mode to exclusive mode
S -> SX	Shared mode to shared exclusive mode
S -> SSX	Shared mode to subshared exclusive mode
S -> X	Shared mode to exclusive mode
SSX -> X	Subshared mode exclusive to exclusive mode

TABLE A-18. *GES Lock Conversion Types*

V$GES_CONVERT_REMOTE

This view maintains information about all remote GES operations (see Table A-19). It displays such information as average convert time, count, and number of conversions.

Column	Data Type	Description
INST_ID	Number	Instance ID.
CONVERT_TYPE	Varchar2 (64)	Conversion type.
AVERAGE_CONVERT_TIME	Number	Average conversion time for each lock in hundredths of a second.
CONVERT_COUNT	Number	Number of conversions.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

https://t.me/learningnets

Technet24

TABLE A-19. *V$GES_CONVERT_REMOTE Information*

V$GES_RESOURCE

This view maintains information about all resources known to the lock manager (see Table A-20). It keeps the information about the master node for that particular resource.

Column	Data Type	Description
RESP	Raw (4 | 8)	Pointer to the resource.
RESOURCE_NAME	Varchar2 (30)	Resource name in hexadecimal format for the lock.
ON_CONVERT_Q	Number	A value of 1 if on convert queue; otherwise 0.
ON_GRANT_Q	Number	A value of 1 if on convert queue; otherwise 0.
PERSISTENT_RES	Number	A value of 1 if it is a persistent resource; otherwise 0.
MASTER_NODE	Number	Node ID mastering this resource.
NEXT_CVT_LEVEL	Varchar2 (9)	Next lock level to be converted on global convert queue.
VALUE_BLK_STATE	Varchar2 (32)	State of value block.
VALUE_BLK	Varchar2 (64)	First 64 bytes of value block.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-20. *V$GES_RESOURCE Information*

Dynamic Resource Remastering Diagnostics

Dynamic resource remastering is a key feature of Oracle RAC that masters the frequently accessed resources to the local node. More information on dynamic remastering is available in Chapter 11.

V$HVMASTER_INFO

This view maintains information about current and previous master instances of GES resources in relation to the hash value ID of the resource (see Table A-21).

https://t.me/learningnets

Column	Data Type	Description
HV_ID	Number	Hash value ID of the resource.
CURRENT_MASTER	Number	Instance currently mastering the resource.
PREVIOUS_MASTER	Number	Previous instance that mastered this resource.
REMASTER_CNT	Number	Number of times this resource has remastered.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-21. *V$HVMASTER_INFO Information*

V$GCSHVMASTER_INFO

This view displays the same kind of information for GCS resources that V$HVMASTER_INFO displays for GES resources (see Table A-22). Both views display information about the number of times remastering has occurred for resources. This view does not display information about resources belonging to files mapped to a particular master.

Column	Data Type	Description
HV_ID	Number	Hash value ID of the resource.
CURRENT_MASTER	Number	Instance currently mastering this PCM resource.
PREVIOUS_MASTER	Number	Previous instance that mastered this PCM resource.
REMASTER_CNT	Number	Number of times this resource has remastered.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-22. *V$GCSHVMASTER_INFO Information*

V$GCSPFMASTER_INFO

This view displays information of the current and previous masters about GCS resources belonging to files mapped to a particular master, including the number of times the resource has remastered (see Table A-23). File-based remastering was introduced in Oracle 10*g* R1; starting with Oracle 10*g* R2, remastering is done by

https://t.me/learningnets

Technet24

object level, which allows fine-grained remastering.

Column	Data Type	Description
FILE_ID	Number	File ID.
DATA_OBJECT_ID	Number	Data object ID.
GC_MASTERING_POLICY	Number	Data object type, with two possible values: Affinity or Read-mostly.
CURRENT_MASTER	Number	Instance currently mastering this file.
PREVIOUS_MASTER	Number	Previous instance that mastered this file.
REMASTER_CNT	Number	Number of times this file has remastered.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-23. *V$GCSPFMASTER_INFO Information*

Cluster Interconnect Diagnostics

Starting from Oracle 10*g*, the cluster interconnect information is available from the v$views. In previous versions, you had to query the alert log or invoke an IPC dump to get details about the interconnects used. The following views provide information about the interconnects configured and used in Oracle RAC.

V$CLUSTER_INTERCONNECTS

This view displays information about interconnects being used for cluster communication (see Table A-24). It also lists details about the source of the information because interconnect information is also stored in the Oracle Cluster Registry (OCR) and can be configured using the initialization parameter.

https://t.me/learningnets

Column	Data Type	Description
NAME	Varchar2 (15)	Name of the interconnect—eth0, eth1, and so on.
IP_ADDRESS	Varchar2 (16)	IP address of the interconnect.
IS_PUBLIC	Varchar2 (4)	Yes if the interconnect is public. No if the interconnect is private. Could be null if the type of interconnect is unknown to the cluster.
SOURCE	Varchar2 (31)	Indicates where the interconnect was picked up; interconnect information is available from OCR, OSD software, or the CLUSTER_INTERCONNECTS parameter.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-24. *V$CLUSTER_INTERCONNECTS Information*

V$CONFIGURED_INTERCONNECTS

This view's display is the same as V$CLUSTER_INTERCONNECTS, but it displays information for all configured interconnects of which Oracle is aware, instead of just those that the database is using (see Table A-25).

Column	Data Type	Description
NAME	Varchar2 (15)	Name of the interconnect—eth0, eth1, and so on.
IP_ADDRESS	Varchar2 (16)	IP address of the interconnect.
IS_PUBLIC	Varchar2 (4)	Yes if the interconnect is public. No if the interconnect is private. Could be null if the type of interconnect is unknown to the cluster.
SOURCE	Varchar2 (31)	Indicates where the interconnect was picked up.
CON_ID	Number	The ID of the container to which the data pertains. Data that pertains to the entire CDB has the value 0. Data that pertains to only the root has the value 1. The value *n* is the applicable container ID for rows containing data.

TABLE A-25. *V$CONFIGURED_INTERCONNECTS Information*

https://t.me/learningnets

Technet24

APPENDIX

B

Adding and Removing Cluster Nodes

Oracle RAC is a highly scalable solution that allows users to add/remove nodes on demand in the existing cluster without affecting the availability of the active database services. Nodes should be homogenous and should have the same computing capacity. Though adding a node with a higher or lower capacity than the other members of the cluster is technically possible, it's usually not recommended because it creates a logical imbalance between the nodes.

Adding a Node

Adding a node to an existing cluster is similar to installing the Oracle RAC environment. You must complete all prerequisites explained in Chapters 4 and 5, which describe how to install Oracle RAC and Oracle Clusterware. You mainly have two different ways to add a new node into the existing cluster—you can either clone an existing grid home or use the addNode.sh script. You can clone an existing grid home either by executing the clone.pl script manually or by using Oracle Enterprise Manager. Although Oracle Enterprise Manager provides a graphical user interface to clone an existing grid home, underneath the hood it uses the clone.pl script. Even though you can add another node via cloning, most users use the addNode.sh script. We will also use the addNode.sh script to explain the procedure for adding a new node

https://t.me/learningnets

to the existing cluster.

Regardless of the option you choose to add a new node, typically you will need to complete the following prerequisites:

1. Configure the public and private networks and then ensure that multicasting is enabled for the private network. Also, make sure the network interface name is the same as that of the other nodes in the cluster.
2. Configure the operating system and make sure the system is configured exactly the same as the other nodes in the cluster. For example, you must ensure that the kernel parameters are set the same, that the same directory structure for storing software binaries is created, that users and groups are created with the same names and IDs, that shell limits are configured, and that user equivalences have been set up for software owners.
3. Configure shared storage as mentioned in Chapter 4. If you are using ASM, you install and configure ASM libraries on the new node but do not create ASM disks on the new node because ASM disks are already created. You will need to scan the ASM disks on the new node and make sure that ASM disks are detected on the new node.

In the following example, we add a new node to the cluster ORARAC, which has three nodes: RAC1, RAC2, and RAC3. The Oracle instances on these nodes will be called ORARAC1, ORARAC2, and ORARAC3, respectively. The newly added node will be called RAC4, and the Oracle instance on it will be named ORARAC4. The following walkthrough shows the process of adding the node RAC4 to the cluster ORARAC.

Performing Pre-Installation Checks

You can use the Cluster Verification utility to verify that the system has been configured properly for Oracle Grid Infrastructure, and you can use the following command to verify the readiness of the new node. You can run the Cluster Verification utility either from the staging directory or from any of the existing nodes. The following command in the node RAC1 invokes the Cluster Verification utility and performs the post-hardware and operating system check for Oracle Grid Infrastructure:

```
$ ./runcluvfy.sh stage - pre crsinst -n rac1,rac2,rac3,rac4 -r 12cR2
$ ./runcluvfy.sh stage -pos hwos -n rac4
```

Alternatively, the following command runs the Cluster Verification utility from the existing nodes:

```
$ cluvfy stage -pos hwos -n rac4
```

Once this post-hardware test has passed, you can compare the system configuration of the new node with any existing node in the cluster to ensure the new node has been

https://t.me/learningnets

Technet24

configured in the same way that other nodes in the cluster are configured. The following command in the node RAC1 invokes the Cluster Verification utility and compares the system configuration of the RAC1 and RAC4 nodes, assuming that the Oracle Inventory group is oinstall and that the OS dba group is asmdba:

```
$cluvfy comp peer -refnode rac1 -n rac4 -orainv oinstall -osdba asmdba -r 12cR2
```

The Cluster Verification utility provides another test to verify the integrity of the cluster and the new node being added. You should execute this stage test and then analyze its output carefully before executing the addNode.sh script. The following command invokes the Cluster Verification utility and executes the pre-“nodeadd” test:

```
$cluvfy stage -pre nodeadd -n rac4
```

The addNode.sh script performs this check internally, so make sure the preceding test succeeds prior to executing addNode.sh. The Cluster Verification utility should be successful before installation begins. If any error is reported during the verification, you must fix it before you proceed with the Cluster Ready Services installation.

Executing the addNode.sh Script

The addNode.sh script performs most of the work to add a node into the existing cluster by distributing the Oracle Grid Infrastructure binaries from the existing cluster node to the node being added to the cluster. This script also relinks the binaries on the new node being added. The addNode.sh script can be executed on the GUI in silent mode, whereas GUI mode would require the user to set the DISPLAY variable before invoking the addNode.sh script. Most users in large organizations execute this script in silent mode because doing so provides more options for automation by scripting the node-addition process. Once all the prerequisites mentioned in the previous section are met, execute the addNode.sh script located in the $GRID_HOME/oui/bin directory to add the new node:

```
$addNode.sh -silent "CLUSTER_NEW_NODES={rac4}" "CLUSTER_NEW_VIRTUAL_HOSTNAMES={rac4-vip}"
```

If you are using Grid Naming Services (GNS), you need not provide the virtual hostname for the new node because Oracle GNS will allocate the virtual hostname and IP address automatically. You will execute the following command to add a new node if GNS is configured:

```
$addNode.sh -silent "CLUSTER_NEW_NODES={rac4}"
```

The addNode.sh script internally performs various checks to verify the readiness of the node being added to the cluster and invokes the Oracle Universal Installer (OUI) in silent mode to distribute the Oracle Grid Infrastructure binaries.

https://t.me/learningnets

NOTE

The Cluster Verification utility has a historical trend of not verifying the shared storage properly, and you may find that the addNode.sh script fails, complaining about shared storage. If you are sure that the shared storage is configured properly, you can set the environment variable IGNORE_PREADDNODE_CHECKS to the value Y. This will avoid the addNode.sh script performing the pre-node check, which internally checks for the shared storage.

Installing the Oracle Database Software

After you have installed the Oracle Grid Infrastructure, you can install the Oracle software. Installation of the Oracle database software can also occur from any of the existing nodes. The shell script addNode.sh in the $ORACLE_HOME/oui/bin directory calls the OUI and copies the software to the new node. The following process installs the Oracle software to the newly added node.

Log in as the *oracle* user on any of the cluster nodes and set the environment variables. Invoke addNode.sh in silent mode from $ORACLE_HOME/oui/bin:

```
$./addNode.sh -silent "CLUSTER_NEW_NODES={rac4}"
```

This command executes the OUI in silent mode, copies the database software to the new node, and asks you to run the root.sh script at the end. Run the script on the newly added node as specified. If the existing cluster database is a policy-managed database, then the root.sh script will add this new node to the Free Pool, and whenever you increase the cardinality of the database server pool, Oracle will assign an Oracle RAC database instance on the RAC4 node and add this RAC4 node to the database server pool. For a policy-managed database, you need not follow any more steps to add a node to the existing Oracle RAC database; however, you should follow the remaining steps to create a database instance on the RAC4 node if you are adding this node to an administrator-managed database.

Creating a Database Instance

Now follow these steps on the first node to create the database instance in the new node:

1. Log in as the user *oracle* on RAC4, set the environment to the database home, and invoke the Database Configuration Assistant (DBCA):
   `$ORACLE_HOME/bin/dbca`
2. In the Welcome screen, choose Oracle Real Application Clusters Database to create the instance and then click Next.

https://t.me/learningnets

Technet24

3. Choose Instance Management and click Next.
4. Choose Add Instance and click Next.
5. Select ORARAC (or the cluster database name you have set up) as the database and enter the SYSDBA username and password at the bottom of the screen. Click Next.
6. You will see a list of the existing instances. Click Next. On the following screen, enter **ORARAC4** as the instance name and choose RAC4 as the node name.
7. This creates a database instance called ORARAC4 (on RAC4). Click Next in the Database Storage screen. During creation, you will be asked whether the ASM instance should be extended to RAC4. Choose Yes.

Removing a Node

Oracle stores vital information in the Oracle Inventory about Oracle products configured and running on the node, and it is very important to update the Oracle Inventory while removing a node so that Oracle Inventory contains current information. Removing a cluster is very easy because logically unpinning a cluster node from the cluster and shutting it down will separate the node from the cluster, and then you can use the unpinned node for any other purpose. However, if the cluster node being unpinned hosts the Oracle database running on 11*g* Release 1 or an earlier version and you want to completely remove the node from the cluster, you need to follow these steps to completely remove the node from the cluster:

1. Delete the database instance on the node to be deleted.
2. Remove the node from the database.
3. Remove the node from the clusterware.

Deleting the Instance on the Node to Be Deleted

You can delete the instance using the Database Configuration Assistant ($ORACLE_HOME/bin/dbca). Invoke the DBCA and choose the Oracle RAC database. From this screen, choose Instance Management and then choose Delete Instance. You will see a screen with the database name. At the bottom of the screen, enter the SYSDBA user and password. Then choose the instance to delete and confirm the deletion.

Alternatively, you can delete the database instance using Oracle Enterprise Manager. Navigate to the cluster database target page and then navigate to Server and click the Delete Instance link, which will take you to the Delete Instance page. Provide the database login information and follow the instructions to delete the database instance.

https://t.me/learningnets

Removing the Node from the Database

In Oracle Database 11*g* Release 2, Oracle introduced a new utility called Deinstall to uninstall the Oracle software. We highly recommend that you uninstall Oracle database or grid home using this utility. Refer to the Oracle database installation guide for complete and detailed instructions for using this utility.

Removing the RAC4 Node from the Clusterware

Follow these steps to remove the RAC4 node from the cluster, assuming you are not using GNS:

1. Run the following command as the *root* user from the RAC1 node (or any other node that will remain in the cluster) to end the lease of the RAC4 node:

   ```
   $cd $GRID_HOME/bin
   $./crsctl unpin css -n RAC4
   ```

   Make sure that the CSS (Cluster Synchronization Service) process is running on the RAC4 node before expiring the lease on this node because the preceding command will fail if CSS is not running on the node being deleted.

2. Execute the rootcrs.pl script as the root user on the RAC4 node from the $GRID_HOME/crs/install directory to disable the cluster resources. You run this only on the nodes to be deleted.

   ```
   $cd $GRID_HOME/crs/install
   $./rootcrs.pl -delete -force
   ```

   You must stop cluster resources manually if this script fails.

3. Execute the following command as the root user from any node in the cluster that will remain in the cluster:

   ```
   $cd $GRID_HOME/bin
   $./crsctl delete node -n RAC4
   ```

4. As the software owner, execute the following command to update the Oracle Inventory:

   ```
   $cd $GRID_HOME/oui/bin
   $./runInstaller -updateNodeList ORACLE_HOME=$GRID_HOME
   “CLUSTER_NODES={rac4}” CRS=TRUE -local
   ```

5. Execute the following command to deinstall the Oracle Grid Infrastructure home. You don’t delete the physical Oracle home from the node being deleted if it is a shared Oracle home, but you will update Oracle Inventory to detach the grid home from the node being deleted.

   If the RAC4 node is using the shared grid home, execute the following command from the $GRID_HOME/oui/bin directory to update the Oracle Inventory:

   ```
   $cd $GRID_HOME/oui/bin
   ```

https://t.me/learningnets

Technet24

```
$./runInstaller –detachHome ORACLE_HOME=$GRID_HOME
```

Otherwise, execute the following command as the software owner to delete the grid Oracle home:

```
$cd $GRID_HOME/deinstall
$./deinstall -local
```

6. Execute the following command as the software owner to update the Oracle Inventory on all the nodes remaining in the cluster:

```
cd $GRID_HOME/oui/bin
./runInstaller –updateNodeList ORACLE_HOME=$GRID_HOME
“CLUSTER_NODES={rac1,rac2,rac3}” CRS=TRUE
```

If you are using GNS, you only need to follow steps 2, 3, and 6 in this process.

https://t.me/learningnets

Index

Please note that index links point to page beginnings from the print edition. Locations are approximate in e-readers, and you may need to page down one or more times after clicking a link to get to the indexed material.

A

AASTs (acquisition asynchronous traps), 38, 346
access control
    disk groups, 169
    files, 169–170
ACCESS_CONTROL_ENABLED attribute, 169
access control lists (ACLs), 242–243
ACCESS_CONTROL_UMASK attribute, 169
ACD (Active Change Directory), 172–173
ACFS. *See* ASM Cluster File System (ACFS)
acfsutil registry command, 179
ACLs (access control lists), 242–243
acquisition asynchronous traps (AASTs), 38, 346
Active Change Directory (ACD), 172–173
Active Session History (ASH) reports, 330
add css votedisk command, 270–271, 292

https://t.me/learningnets

Technet24

add database command, 216
add instance command, 217
add listener command, 217
add serverpool command, 241
add service command, 220, 407, 412–413, 415–416
add srvpool command, 234–235
add votedisk command, 270
ADDM (Automatic Database Diagnostic and Monitor), 329
addnode.sh script, 541–542
Address Resolution Protocol (ARP), 61
administering
    Oracle Cluster Registry, 265
    Oracle Clusterware, 238
    Oracle Local Registry, 268–269
    services, 411–414
administrator-managed clusters, 238
administrator-managed databases
    converting to policy-managed databases, 235–236
    vs. policy-managed databases, 48–49, 146
administrator-managed deployments, 232
ADR (Automatic Diagnostic Repository), 450–451
ADR Command Interpreter (ADRCI), 451
Advanced configuration option, 142
ADVM (ASM Dynamic Volume Manager), 154, 160–161
affinity remastering, 286–292
affordability of OPS, 41
agility
    description, 16–17
    solutions, 17–18
alert directory, 452
alert logs, 450, 455–459
Alias Oracle ASM filenames, 162
allocation units, 156–157
ALTER DATABASE command, 226
ALTER DATABASE TEMPFILE command, 223
ALTER DISKGROUP command
    adding disks, 177
    alias names, 162
    attributes, 175
    directories, 164
    disk group scrubbing, 185

https://t.me/learningnets

disk rebalancing, 181–184
dropping disk group files, 163
dropping disks, 177
file access control, 169–170
file groups, 190
migrating to disk groups, 189
quota groups, 189
renaming disks, 178
replacing disks, 177
resizing disks, 177–178
undropping disks, 177
ALTER DISKGROUP MOUNT command, 166
ALTER DISKGROUP VOLUME command, 161
ALTER PLUGGABLE DATABASE command, 220
ALTER SYSTEM KILL SESSION command, 230
ALTER SYSTEM RELOCATE command, 188
ALTER SYSTEM SET command, 163, 208
ALTER TABLESPACE command, 223
anti-locks, 336, 358, 495
Application Continuity feature, 446–448
application VIPs (AVIPs), 61
applications
buffer busy waits and block contention, 498–502
commit frequency, 508
connection management, 504–507
contexts, 416
data partitioning, 496–498
failure issues, 439
full table scans, 504–507
guidance systems, 496–498
index partitioning, 499–502
library cache effect, 507–508
overview, 492–494
partitioning, 418, 494–495
sequences, 502–504
AQ_HA_NOTIFICATIONS view, 409, 416–417
ARB*n* background process, 171
architecture
administrator- vs. policy-managed databases, 48–49
extended clusters, 47
introduction, 44–45

https://t.me/learningnets
Technet24

kernel components, 67–73
key networking concepts, 60–62
networking stack components, 62–66
Oracle Clusterware, 51–60
Oracle Flex Clusters, 46–47
Oracle Multitenant option, 47–48
shared disk system, 49–51
single-instance environments vs. Oracle RAC, 45–46
archive log list command, 226
ARCHIVELOG mode, 279
archives
backups, 276–277
logs, 225–226
ARP (Address Resolution Protocol), 61
array-based mirroring, 485–487
ASH (Active Session History) reports, 330
ASM. *See* Automatic Storage Management (ASM)
ASM Cluster File System (ACFS)
administering, 178–179
description, 33, 154
setting up, 179
snapshots, 161–162, 180
ASM_DISK_REPAIR attribute, 180
ASM_DISKGROUPS parameter, 165–166, 174, 176
ASM_DISKSTRING parameter, 173–174, 176
ASM Dynamic Volume Manager (ADVM), 154, 160–161
ASM Filter Driver (ASMFD), 197–198
ASM_POWER_LIMIT parameter, 173, 181
ASM_PREFERRED_READ_FAILURE_GROUPS parameter, 174, 206, 487
ASMB background process, 172
ASMCA. *See* Automatic Storage Management Configuration Assistant (ASMCA)
ASMCMD utility
description, 192–194
flex clusters, 188
instance starting and stopping, 168
ASMFD (ASM Filter Driver), 197–198
ASMLib library
configuring, 196–197
devices, 95
Grid Infrastructure, 91–92
installing, 195–196

https://t.me/learningnets

ASMSNMP users, 112–113
ASSM (Automatic Segment Space Management) tablespaces, 498–499
asynchronous traps, 38, 346
AU_SIZE attribute, 175
AUTO_CONNECT option, 504
auto-startup for Oracle Clusterware, 252–253
AUTOEXTEND option, 224
Automatic Database Diagnostic and Monitor (ADDM), 329
Automatic Diagnostic Repository (ADR), 450–451
Automatic Memory Management, 145, 155
Automatic Segment Space Management (ASSM) tablespaces, 498–499
Automatic Storage Management (ASM)
    ACFS administration, 178–180
    ACFS snapshots, 161–162
    administration and management, 164–174
    ADVM, 160–161
    allocation units, 156–157
    ASMCA, 192
    ASMCMD, 192–194
    ASMFD, 197–198
    ASMLib, 195–197
    background processes, 170–172
    backups, 185–186, 277–278
    configuring, 93
    description, 51, 150
    disk failures, 159
    disk groups. *See* disk groups
    disk level protection, 10
    disk partners, 160
    disk rebalancing, 180–185
    drivers, 93
    dynamic performance views, 170
    extended disk groups, 191–192
    failure groups, 157–159
    filenames, 162
    files, 159–160, 163
    flex clusters, 186–191
    FTP, 194–195
    Grid Infrastructure, 78, 91–92
    initialization parameters, 173–174
    instances, 154–155, 165–169

https://t.me/learningnets
Technet24

introduction of, 32–33
migrating from OCR, 267–268
operation, 152–154
overview, 151
physical limits, 151–152
preferred read, 487
recovery, 185–186
Standard vs. Flex, 150–151
storage for voting disks, 269–270
Automatic Storage Management Configuration Assistant (ASMCA)
database creation, 140
description, 109, 192
disk groups, 81, 175, 177
Automatic Workload Repository (AWR)
description, 329
full table scans, 505
service level thresholds, 436
workload measurements, 435
Automatic Workload Repository (AWR) reports
GCS and GES messaging statistics sections, 327
GCS and GES workload characteristics section, 326
global cache efficiency percentages section, 326
global cache load profile section, 324–325
interpreting, 323–324
number of instances section, 324
overview, 322–323
service statistics section, 327
service wait class statistics section, 327–328
top 5 CR and current blocks segments sections, 328
auxiliary tablespace (SYSAUX), 329
availability. *See also* high availability (HA); maximum availability
network bonding, 65
OPS, 40–41
availability curve, 8
available instances for services, 407–408
AVIPs (application VIPs), 61
Awesome Storage Management, 150
AWR. *See* Automatic Workload Repository (AWR)

B

https://t.me/learningnets

background processes
ASM, 170–172
Cache Fusion, 71–72, 400–401
coordination, 68–72
instances, 45
overview, 68
BACKUP parameter, 429–430
backups
ASM, 185–186
basics, 275–278
options, 274–275
Oracle Cluster Registry, 56, 293–295
parameter files, 209
RMAN, 276–278
voting disks, 270, 292
BALANCE phase in disk rebalancing, 183
bandwidth scalability in network bonding, 65
base home directory for Grid Infrastructure, 90
BASIC policy for TAF, 427, 429
basic systems, level of availability in, 8
BASTs. *See* blocking asynchronous traps (BASTs)
big data, 30
block access in GCS, 352
block contention
applications, 498–499
overview, 39–40
block devices for ASM, 153
block-oriented waits, 306
Block Server Process (BSP), 39, 372
block size in databases, 302
block updates in Cache Fusion, 378–379, 391–395
Block Written Record (BWR)
GCS, 349
overview, 283
Past Image blocks, 368
blocking asynchronous traps (BASTs)
asynchronous traps, 346
branch/leaf splits, 500
Cache Fusion, 367
locking concepts, 38
messaging deadlocks, 347

https://t.me/learningnets
Technet24

three-way lock messages, 345–346
bounded recovery, 283
branch block splits, 499–502
BSP (Block Server Process), 39, 372
buffer busy global cr events, 313
buffer busy waits, 498–502
buffer caches in Cache Fusion, 67
buffer writes in Cache Fusion, 379–380
BWR (Block Written Record)
GCS, 349
overview, 283
Past Image blocks, 368

C

cache coherency, 68, 70, 339–340
Cache Fusion
background processes, 71–72, 400–401
buffer caches, 67
consistent read server, 369–373
contention, 369
deferred ping, 367–368
example, 382–399
forced block writes, 495
locks, 348, 368–369
operation, 375–383
overview, 366–367
Past Image, 284, 368
performance, 483–484
ping, 367
recovery, 285–286, 289–292
resource mastering and remastering, 399
Stage 1, 39–40
statistics, 303
views, 512–519
write/write cache fusion, 373–375
Cache Fusion I, 369–373
Cache Fusion II, 373–375
Cache Fusion Protocol, 366
CACHE option for sequences, 502–504
cache reads in Cache Fusion, 377–378, 390–391

https://t.me/learningnets

Call Data Records (CDR), 33, 501
callouts for FAN events, 444
cardinality of services, 408
Cassandra database, 28
catalog.sql script, 512
categorization for server pools, 233, 239
CBO (Cost Based Optimizer), 500–501
CDBs (container databases), 47–48, 220
CDR (Call Data Records), 33, 501
cdump directory, 451
CELL_SMART_SCAN_CAPABLE attribute, 175
cemutlo utility, 261–262
CF (control file) enqueues, 36, 321
CFS (cluster-wide file system), 29
CGS. *See* Cluster Group Services (CGS)
Change Discovery Path button, 112
change vectors, redo, 279
character devices in ASM, 153
check cluster command, 254–255
Checkpoint (CKPT) process
    heartbeats, 458
    instance membership recovery, 467–468
Checkpoint Queue (CKPTQ), 280
checkpoints
    Cache Fusion, 380–381, 396–397
    structures, 279–280
CHM (Cluster Health Monitor), 259, 465–466
CKPT (Checkpoint) process
    heartbeats, 458
    instance membership recovery, 467–468
CKPTQ (Checkpoint Queue), 280
CLB_GOAL value, 425
CLB_GOAL_LONG value, 425
CLB_GOAL_SHORT value, 425
clbgoal parameter, 448
client failover for Transaction Guard, 445
client-side load balancing, 409, 419–423
cloud computing for clusters, 21–23
clscfg utility, 260–261
cluster administrators, managing, 242
CLUSTER_DATABASE parameter, 207, 225–226

https://t.me/learningnets

Technet24

CLUSTER_DATABASE_INSTANCES parameter, 207–208
Cluster Group Services (CGS)
cache services, 72
description, 35
instance failures, 290
instance membership recovery, 466–467
Cluster Health Advisor, 79
Cluster Health Monitor (CHM), 259, 465–466
cluster interconnects
overview, 63–64
tuning, 330
views, 537–538
CLUSTER_INTERCONNECTS parameter, 205–206, 456–457
Cluster Logger Service (OLOGGERD), 59, 259
Cluster Logical Volume Manager (CLVM), 485
Cluster Managed Database Services page, 412
Cluster Manager (CM)
description, 34
instance membership recovery, 466
Cluster Node Information screen, 104–105
Cluster Ready Services (CRS)
disabling and enabling, 256
functionalities, 51–53
Grid Infrastructure, 99
OCR updates, 56
starting and stopping, 253–254
status, 255
Cluster Synchronization Services (CSS), 51
modules, 473
multicasting, 455
OCR updates, 55–56
overview, 57
Cluster Time Synchronization Services (CTSS), 59, 88
Cluster Verification Utility (CLUVFY)
adding nodes, 540–541
clusters, 129–130
Grid Infrastructure installation, 82, 124
multicast communication, 83–84
OCR validation, 294
pre–database installation check, 128–129
running, 96–100

https://t.me/learningnets

stages, 97–100
working with, 247–248
cluster-wide file system (CFS), 29
clusterized commands, 254–256
clusters
cloud computing, 21–23
description, 11
extended. *See* extended clusters
flex. *See* flex clusters
Hadoop, 30–32
installing, 129–140
killing sessions, 230
membership decisions, 53
nodes, 104–107
Oracle Clusterware. *See* Oracle Clusterware
overview, 20–21
reconfiguration in instance membership recovery, 468
shared storage, 25–30
smart scans, 23–25
virtual IPs, 60
CLUVFY. *See* Cluster Verification Utility (CLUVFY)
cluvfy comp baseline command, 247–248
CLVM (Cluster Logical Volume Manager), 485
CM (Cluster Manager)
description, 34
instance membership recovery, 466
coarse striping, 153
coarse WDM (CWDM) systems, 483
COD (Continuing Operations Directory), 172–173
coherency, cache, 68, 70, 339–340
cold failover, 12–14
commit_outcome parameter
Application Continuity, 446–447
Transaction Guard, 445
Commit SCNs, 459
commits
Cache Fusion, 379, 395–396
frequency in applications, 508
COMPACT phase in disk rebalancing, 183
compatibility matrix for lock modes, 348
COMPATIBLE.ADVM attribute, 175, 179

https://t.me/learningnets
Technet24

concurrency control, 36
config asm command, 188
config database commands, 215–217, 227–228
config listener command, 218
config scan command, 246
config service command, 414
config srvpool command, 234
configuration
    Application Continuity, 447
    ASM, 93
    ASMLib, 196–197
    databases, 216–217, 227–230
    Flex ASM, 150–151
    flex clusters, 187
    Grid Infrastructure, 83–87, 95
    operating system, 82–83
    Oracle Clusterware, 238
    server pools, 233–234
    services, 414
    shared storage, 90–95
    TAF, 428–430
Configure Security Updates screen, 130
connect time
    load balancing, 419
    workload balancing, 432
connections
    databases, 231–232
    load balancing, 409–410, 433–434
    management, 504–507
    pooling in workload balancing, 432–433
connectivity in maximum availability, 482
consistent read (CR) processing in GCS, 352–355
consistent read server for Cache Fusion, 369–373
consolidation of databases, 234
container databases (CDBs), 47–48, 220
contention
    block, 39–40, 498–502
    Cache Fusion, 369, 371
    MMON, 329
contention-oriented waits, 307
Continuing Operations Directory (COD), 172–173

https://t.me/learningnets

control file (CF) enqueues, 36, 321
CONTROL_FILES parameter, 163
controlfile vote result record, 468
convert queues, 336–338
converting
    administrator-managed databases to policy-managed databases, 235–236
    locks, 37, 336–337
coordination of background processes, 68–72
Cost Based Optimizer (CBO), 500–501
costs
    cache coherency, 70
    downtime, 7–9
CPU_COUNT parameter, 72
CPU time in ADDM, 329
CR block fabrication
    Cache Fusion, 371–372
    GCS, 352–354
CR (consistent read) processing in GCS, 352–355
crashes
    Cache Fusion, 381, 397–398
    instances, 288
    recovery, 186, 281–286
Create and configure a database option, 131
Create ASM Disk Group screen, 111
CREATE DATABASE command, 281
CREATE DISKGROUP command
    disk group attributes, 175
    extended disk groups, 192
    failure groups, 176
    flex disk groups, 189
Create Inventory screen, 114, 117
Create Service page, 412
CREATE TABLESPACE statement, 163
crossover cables, 66, 80
CRS. *See* Cluster Ready Services (CRS)
CRSCTL utility
    advanced debugging, 472–474
    cluster-aware commands, 254–256
    EVAL commands, 256–257
    Grid Infrastructure installation, 124–125
    server pools, 146

https://t.me/learningnets
Technet24

starting and stopping instances, 214
CRSd process startup, 250–251
CSS. *See* Cluster Synchronization Services (CSS)
CSSD process, 251–252, 464
CSSDAGENT process, 57, 463–464
CTSS (Cluster Time Synchronization Services), 59, 88
CTSSd daemon, 88
CURRENT_MASTER column, 361
cursor_sharing parameter, 299
Custom Database option, 144
cvuqdisk RPM, 91
CWDM (coarse WDM) systems, 483

D

DAS (directly attached storage), 50
data buffer cache blocks, 337
Data Definition Language (DDL) statements
disadvantages, 299
embedded in applications, 496
lock issues, 463, 507
DATA_OBJECT_ID column, 361
data partitioning for applications, 496–498
data protection, Extended Oracle RAC vs. Oracle Data Guard, 489
data selection in Cache Fusion, 398–399
data storage in maximum availability, 484
data unit size in ASM, 153
data warehouses, 496
database configuration
managing, 227–230
tracing, 474
Database Configuration Assistant (DBCA)
catalog.sql script, 512
database creation, 140–141
Grid Infrastructure, 80–81
services, 411
DATABASE events in FAN, 442
Database Resource Manager, 411, 434
Database Upgrade Assistants, 474
Database Writer (DBWR), 157, 280
databases

https://t.me/learningnets

adding, 216
alert log, 455–459
block size, 302
checkpoints, 280
configuration, 216–217
connections, 231–232
consolidation, 234
creating, 140–148
enabling and disabling, 217
events, 303
global locks, 341–345
hung, 460–461
modifying, 218
outages, 446–448
performance, 460–463
roles, 407
SRVCTL commands, 216–217
starting, 210–211
stopping, 212
structure, 329
datafile limits, 151
_DB_BLOCK_HOT_TRACKING parameter, 498
_DB_BLOCK_MAX_CR_DBA parameter, 353
DB_BLOCK_SIZE parameter, 302
DB_CREATE_FILE_DEST parameter, 163
DB_CREATE_ONLINE_LOG_DEST parameter, 163
DB_FILE_DIRECT_IO_COUNT parameter, 157
DB_FILE_MULTIBLOCK_READ_COUNT parameter, 504
DB_RECOVERY_FILE_DEST parameter, 146, 163, 226
DB_RECOVERY_FILE_DEST_SIZE parameter, 146, 226
DBA_EXTENTS view, 383
DBA_SERVICES view, 434
DBCA. *See* Database Configuration Assistant (DBCA)
DBMS_APPLICATION_INFO package, 416–417
DBMS_MONITOR package
service level thresholds, 437
workload measurements, 435
DBMS_RESOURCE_MANAGER package, 411
DBMS_SERVER_ALERT package, 437
DBMS_SERVICE package, 411, 426
DBMS_SERVICE.CREATE_SERVICE package, 425

https://t.me/learningnets
Technet24

DBMS_WORKLOAD_REPOSITORY package, 322
DBWR (Database Writer), 157, 280
DBWR (Dirty Buffer Writer), 39, 283
DBWR Fusion writes statistic in AWR report, 325
DDL (Data Definition Language) statements
    disadvantages, 299
    embedded in applications, 496
    lock issues, 463, 507
deadlocks, 347
debug log files, 450
debugging. *See also* troubleshooting
    node eviction issues, 463–465
    Oracle Clusterware modules, 472–474
    utilities, 474–475
default database services, 411–412
default file locations in ASM, 163
default server pools, 240
deferred ping in Cache Fusion, 367–368
delayed block cleanout, 344
delete css votedisk command, 270–271
delete serverpool command, 241
delete votedisk command, 270
deleting
    instances, 543
    server pools, 241
    voting disks, 271
dense wavelength division multiplexers (DWDMs), 10, 483
design considerations
    best practices, 298–300
    maximum availability, 482–484
Details window, 122
device discovery functions, 196–197
DFS (Distributed File System) enqueue locks, 36
DHB (disk heartbeat)
    node eviction issues, 463
    for split-brain, 57
DHCP (Dynamic Host Configuration Protocol), 62, 85
Diagnostic Daemon (DIAG), 401
DIAGNOSTIC_DEST parameter, 174, 450–451
dictionary cache, 71
directly attached storage (DAS), 50

https://t.me/learningnets

directories
disk groups, 164
validate global resource, 291
Dirty Buffer Writer (DBWR), 39, 283
disable crs command, 256
disable database command, 217
disable instance command, 217
disable listener command, 218
disabling
CRS, 256
databases, 217
instances, 217
listeners, 218
RAC options, 459–460
services, 414
disaster recovery (DR)
description, 5
vs. failover, 11–12
level of availability, 8
disaster tolerance (DT), 5
disk failures in ASM, 159
disk groups
access control, 169
adding disks to, 176–177
ADVM, 160–161
ASM, 110–112, 155–156
creating, 176
datafiles in, 151
directories, 164
disk rebalancing, 180–185
dropping disks from, 177
dropping files from, 163
even read feature, 159
extended, 191–192
flex clusters, 188–189
guidelines, 92
listing, 193
managing, 174–175
renaming, 178
replacing disks in, 177
resizing disks in, 177–178

https://t.me/learningnets
Technet24

scrubbing, 185
undropping disks from, 177
disk heartbeat (DHB)
node eviction issues, 463
for split-brain, 57
disk partners in ASM, 160
disk pings in Cache Fusion, 367
disk reads in Cache Fusion, 377, 388–390
disk rebalancing, 180–181
manual process, 182–183
monitoring, 184
phase options, 183–184
tuning, 184–185
DISK_REPAIR_TIME attribute, 175
disk resync enhancements, 152
disk system, shared, 49–51
disk writes in Cache Fusion, 396–397
displaying
SCAN status, 245–246
server pool configuration, 234
distance limitation, Extended Oracle RAC vs. Oracle Data Guard, 489
Distributed File System (DFS) enqueue locks, 36
Distributed Lock Manager (DLM)
cache coherency, 71
description, 35
instance recovery, 286
lock compatibility matrix, 36–37
resource sharing coordination, 45
table locks, 36
Distributed Transaction Processing (DTP), 415–416
Domain Name Service (DNS), 85
DOWN events in high availability, 440
downtime costs, 7–9
DR (disaster recovery)
description, 5
vs. failover, 11–12
level of availability, 8
drivers in ASM, 93
DROP_BASELINE command, 322
dropping
disk group directories, 164

https://t.me/learningnets

disk group disks, 177
disk group files, 163
DT (disaster tolerance), 5
DTP (Distributed Transaction Processing), 415–416
DUPLICATE DATABASE command, 278
DWDMs (dense wavelength division multiplexers), 10, 483
dynamic database services
characteristics, 406–410
overview, 404–406
Dynamic Host Configuration Protocol (DHCP), 62, 85
dynamic performance views, 170, 512
dynamic reconfiguration, 286–292
Dynamic Resource Mastering protocol, 358
dynamic resource remastering
read-mostly locking, 358–360
views, 536–537
dynamic services, 48, 423

E

EC2 (Elastic Cloud Computing), 17
editions of services, 407
efficiency, Extended Oracle RAC vs. Oracle Data Guard, 489
Elastic Cloud Computing (EC2), 17
Enable Archiving option, 145
enable crs command, 256
enable database command, 217
enable instance command, 217
enable listener command, 218
enable service command, 414
enabling
archive logs, 225
CRS, 256
databases, 217
flashback area, 225–226
instances, 217
listeners, 218
RAC options, 459–460
services, 414
enqueues. *See* Global Enqueue Services (GES)
enterprise database backups, 275

https://t.me/learningnets
Technet24

Enterprise Manager Agent, 81
Enterprise Manager Grid Control, 81
environment variables, 218
error and event logs for FAN events, 441
/etc/hosts file, 85–87
/etc/init.d/oracleasm utility, 196
/etc/inittab file, 54–55
/etc/oracle directory, 120
/etc/pam.d/login file, 96
/etc/security/limits.conf file, 96
/etc/sysctl.conf file, 96
Ethernet frame size, 507
eval add serverpool command, 241
EVAL commands, 256–257
eval modify resource command, 256–257
eval start resource command, 256
even read feature, 159
Event Management Logger, 58
Event Manager service (EVMd), 52, 58, 250
events
    event-based notification, 438–439
    FAN, 441–444
    wait. *See* wait events
EVMd (Event Manager service), 52, 58, 250
evmlogger process, 58
exclusive resources, 336
exclusive (X) lock mode
    Cache Fusion resource state, 376
    GCS, 349–350
    global locks, 342
Execute Configuration Scripts dialog box, 120–121, 139
EXPLAIN WORK statement, 184
extended clusters
    benefits, 481
    Grid Infrastructure, 79
    limitations, 488
    mirroring, 487–488
    overview, 47
extended disk groups, 191–192
Extended Oracle RAC vs. Oracle Data Guard, 489
extremely rapid recovery, 481

https://t.me/learningnets

EZconnect driver, 62

F

failed services, relocating, 414
failover
    cold, 12–14
    hot, 12, 14
    level of availability, 8
    none, 11–12
    TAF, 427–432
    Transaction Guard, 445
failover during failures in mirroring, 486
failoverdelay parameter, 448
failoverretry parameter, 448
failovertype parameter, 447
failure groups, 157–159
failure impact, 219–220
Failure Isolation Support screen, 113
failure issues in high availability, 440–444
fairness down-convert in consistent read mechanism, 354–355
false pings, 40
Fast Application Notification (FAN)
    events and notifications, 59, 441–444
    high availability, 438–444
    managing, 52
    service management policy, 409
fast commit process, 344
Fast Connection Failover (FCF)
    Application Continuity, 446
    service management policy, 409
fast reconfiguration, 287–289
Fast Recovery Area, 146, 277
fault tolerance, 5
FCF (Fast Connection Failover)
    Application Continuity, 446
    service management policy, 409
fencing
    I/O, 58, 468
    nodes, 464
FIFO (first-in first-out) data processing applications, 501

https://t.me/learningnets

Technet24

file groups in ASM, 190
File Transfer Protocol (FTP), 194–195
filenames in ASM, 162
files
    access control, 169–170
    ASM, 159–160, 163
fine striping, 153
Finish screen
    clusters, 138, 140
    database creation, 148
first-in first-out (FIFO) data processing applications, 501
first-pass redo log reads, 291
five nines term, 9
fix-up scripts
    clusters, 135
    Grid Infrastructure, 82
flashback area, 225–226
Flex ASM
    configuration, 150–151
    for redundancy, 10
flex clusters, 186
    configuring, 187
    disk groups, 188–189
    file groups, 190
    multitenant environments, 190
    quota groups, 189
    setting up, 187–188
flexibility
    Extended Oracle RAC vs. Oracle Data Guard, 489
    network bonding, 65
FORCE startup modes, 166
forced block writes, 494–495
frame size, 507
FREELIST GROUPS, 498, 503
FREELISTS, 498–499, 503
FTP (File Transfer Protocol), 194–195
full table scans, 504–507
fully qualified filenames, 162
fully scalable systems, 300–301

G

https://t.me/learningnets

_gc_affinity_locking parameter, 399
_gc_affinity_ratio parameter, 360
gc blocks corrupt statistic, 317
gc blocks lost statistic, 317
gc cr block 2-way events, 309–310
gc cr block 3-way events, 310–311
gc cr block build time statistic, 320
gc cr block busy events, 314
gc cr block congested events, 315
gc cr block flush time statistic, 319–320
gc cr block receive time statistic, 318–319
gc cr blocks received statistic, 316
gc cr blocks served statistic, 316
gc cr disk read events, 312–313
gc cr failure events, 313
gc cr request events, 304
gc current block 2-way events, 308–309
gc current block 3-way events, 309
gc current block busy events, 311–312
gc current block congested events, 315
gc current block flush time statistic, 319
gc current block pin time statistic, 319
gc current block receive time statistic, 317–318
gc current block send time statistic, 319
gc current blocks received statistic, 316
gc current blocks served statistic, 316
gc current buffer busy events, 315
gc current grant 2-way events, 304, 311
gc current request events, 304
_gc_defer_time parameter, 501
GC_FILES_TO_LOCK parameter, 494
GC_MASTERING_POLICY column, 361
gc_policy_minimum parameter, 360
gc prepare failures statistic, 316–317
_gc_read_mostly_locking parameter
    Cache Fusion, 399
    read-mostly locking, 358, 360
_gc_transfer_ratio parameter, 360
_gc_undo_affinity parameter, 399
GCS. *See* Global Cache Services (GCS)

https://t.me/learningnets

Technet24

GCS and GES messaging statistics sections in AWR reports, 327
GCS and GES workload characteristics section in AWR reports, 326
__GCS_FAST_RECONFIG parameter, 289
_GCS_RESOURCES parameter, 289
GCS_SERVER_PROCESSES parameter, 72, 204–205, 400
GDS (Global Data Services), 406
GES. *See* Global Enqueue Services (GES)
GET_BUFFER_STAT.SQL query, 384
GET_LTXID_OUTCOME procedure, 445
GET_SEGMENT_STATS query, 390
getenv command, 218
GIMR (Grid Infrastructure Management Repository), 79, 92–93, 109
GIPC*d* (Grid Interprocess Communication), 60
global cache, 336
Global Cache Defers, 501
global cache efficiency percentages section in AWR reports, 326
Global Cache Layer in read-mostly locking, 358
global cache load profile section in AWR reports, 324–325
Global Cache Services (GCS), 36
    consistent read mechanism, 352–355
    data sharing, 67
    description, 336
    instance coordination, 70
    lock modes and lock roles, 348–352
    overview, 348
    Past Image, 284, 368
    read-mostly locking, 357–362
    resource mastering, 355–357
    resource names, 385–386
    times, 317–320
    views, 512–519
global cache statistics, 315–318
global checkpoints, 280
Global Data Services (GDS), 406
Global Database Name, 144
Global Enqueue Services (GES), 36
    cache coherency, 71
    global locks database and structure, 341–345
    latches and enqueues, 340–341
    LMON monitoring of, 72
    messaging, 345–348

https://t.me/learningnets

overview, 67
shared resources, 321
views, 519–535
global locks
database and structure, 341–345
GCS, 349
Global Resource Directory (GRD)
cache coherency, 339–340
Cache Fusion, 71–72
Cache Fusion recovery, 285–286
description, 67
Global Cache Services, 348–362
Global Enqueue Services, 340–348
instance membership recovery, 466–467
introduction, 336
reconfiguration, 289–290
resource sharing coordination, 45
resources and enqueues, 336–339
Global Resource Manager (GRM), 71
global resources in cache coherency, 71
global roles in GCS, 349–350
Global Services Daemon (GSD), 230
global tablespaces, administering, 222–223
global transactions (XA transactions), 415
GLOBAL_TXN_PROCESSES parameter, 415
GMON background process, 171
GNS (Grid Naming Service), 59, 61
administering, 247
Grid Infrastructure, 84–85
specifying, 103
GNS VIP (Grid Naming Service virtual IP address), 85
GOAL_NONE setting, 433
GOAL_SERVICE_TIME setting, 425–426, 433–434
GOAL_THROUGHPUT setting, 425–426, 433–434
Goodness of service metric in load balancing, 433
GPNPd (Grid Plug and Play), 60, 103–104
Grant Q list, 36–37
grant queues, 336–338
granting locking concepts, 38–39
gratuitous ARP, 61
GRD. *See* Global Resource Directory (GRD)

https://t.me/learningnets
Technet24

GRID_HOME directory, 53, 452–453
Grid Infrastructure
cluster nodes and verifying SSH connectivity, 104–107
CUV, 96–100
groups and users, 88–89
installation, 100–104
installation process overview, 79–81
installation verification, 124–125
kernel parameters, 96
Linux directories, 89–90
logs, 452–453
management options and privileged OS system groups, 113–115
network configuration, 83–87
network interfaces, 107–108
NTP configuration, 88
operating system configuration, 82–83
Oracle Universal Installer, 82
overview, 78
preinstallation tasks overview, 81
prerequisite checks, 115–116
product installation, 122–123
root scripts, 116–122
secure shell and user limits configuration, 95
shared storage, 90–95
storage options, 108–113
user limits, 95–96
Grid Infrastructure Management Repository (GIMR), 79, 92–93, 109
Grid Interprocess Communication (GIPC*d*), 60
Grid Naming Service (GNS), 59, 61
administering, 247
Grid Infrastructure, 84–85
specifying, 103
Grid Naming Service virtual IP address (GNS VIP), 85
Grid Plug and Play (GPNPd), 60, 103–104
Grid Plug and Play profile management tool (GPNPTOOL), 258–259
GRM (Global Resource Manager), 71
group-based locking, 38
groups
disk. *See* disk groups
Grid Infrastructure, 88–89
GSD (Global Services Daemon), 230

https://t.me/learningnets

guidance systems for applications, 496–498
GV$ views
    introduction, 39, 302–303
    querying, 512
GV$CONFIGURED_INTERCONNECTS view, 456
GV$LOCK view, 36
GV$LOCK_ELEMENT view, 36
GV$RESOURCE_LIMIT view, 356
GV$SESSION view, 230
GV$SORT_SEGMENT view, 222–223
GV$TEMPSEG_USAGE view, 223

H

HA. *See* high availability (HA)
Hadoop clusters
    architecture, 31–32
    description, 30
Hadoop Distributed File System (HDFS), 31
HAIP (Highly Available IP), 80, 105–106, 455
hanganalyze utility, 461–463
hard pings, 367
hardware load balancing, 419
hash chains in GCS, 350–352
hash clusters, sorted, 501
HDFS (Hadoop Distributed File System), 31
heartbeats
    instance membership recovery, 467–468
    instances, 458
    node eviction issues, 463–464
    objectives, 57
high availability (HA). *See also* maximum availability
    downtime costs, 7–9
    end-to-end perspective, 6–7
    event-based notification, 438–439
    failover, 11–14
    failure issues, 440–444
    network bonding, 65
    notifications and FAN, 438
    option pros and cons, 14
    overview, 4–5, 437–438

https://t.me/learningnets
Technet24

planned and unplanned outages, 6
redundant components, 9–10
services, 407–408
solutions, 10
terminology, 5
Highly Available IP (HAIP), 80, 105–106, 455
historical background, 32
hm directory, 452
HMP (Hyper Messaging Protocol), 458
home directory for Grid Infrastructure, 89–90
homogenous servers, 20
horizontal scalability, 15
host-based mirroring, 485–487
host names in Grid Infrastructure, 86–87
hot blocks, 498
hot failover, 12, 14
Hot Standby Routing Protocol (HSRP), 10
hub nodes, 46–47, 186
hung databases, 460–461
hung sessions, 461–463
Hyper Messaging Protocol (HMP), 458

I

I/O fencing
instance membership recovery, 468
overview, 58
I/O processing functions, 196–197
IaaS (Infrastructure as a Service) model, 16–17
ICMP Route Discovery Protocol (IRDP), 10
identical parameters, 206–208
idle waits, 303
IDLM. *See* Integrated Distributed Lock Manager (IDLM)
IGMP (Internet Group Management Protocol), 84, 455
IGNORE_PREADDNODE_CHECKS parameter, 542
IMMEDIATE shutdown command, 167–168
IMPORTANCE attribute, 48, 233–234, 241
IMR. *See* Instance Membership Recovery (IMR) process
in-band notification for FAN events, 441
incident directory, 452
incremental checkpoints, 280

https://t.me/learningnets

index partitioning for applications, 499–502
indexes, managing, 231
Infrastructure as a Service (IaaS) model, 16–17
infrastructure installation, troubleshooting, 453–455
init.crsd wrapper script, 52–53, 248
init.cssd wrapper script, 52, 57, 248–252
init.evmd wrapper script, 52, 248, 250
initialization parameters
    ASM, 173–174
    backing up, 209
    identical parameters, 206–208
    managing, 208–209
    overview, 202–203
    search order, 209–210
    unique, 203–206
INITRANS parameter, 343
Install database software only option, 131
Install Product screen, 116, 121–122, 137, 139
installActions.log file, 120
installation
    ASMLib, 195–196
    clusters, 129–140
    Grid Infrastructure, 100–104
    log files, 450
    overview, 128–129
    software, 542–543
    verification, 124–125
instance crashes, 381, 397–398
INSTANCE events, 442
Instance Membership Recovery (IMR) process
    cluster reconfiguration, 469–472
    description, 66
    heartbeats, 458
    member voting, 467–468
    overview, 466–467
INSTANCE_NAME parameter, 204
INSTANCE_NUMBER parameter, 204
INSTANCE_TYPE parameter, 173
instances
    adding, 217
    administering, 165–169

https://t.me/learningnets
Technet24

AQ_HA_NOTIFICATIONS, 416–417
ASM, 154–155
AWR reports, 324
caging, 234
crashes, 288
creating, 542–543
deleting, 543
enabling and disabling, 217
heartbeats, 458
killing sessions, 230
MMON management, 329
modifying, 219
preferences, 407
recovery, 185, 278–283
SRVCTL commands, 217
starting, 210–211
status, 211–212
stopping, 213
System Global Area, 45
Instantaneous Problem Detector for Clusters (IPD/OS), 465
Integrated Distributed Lock Manager (IDLM)
Cache Fusion, 371
description, 286
instance membership recovery, 468
lock modes, 36
Oracle Parallel Storage Evaluation, 32–33
Intelligent Platform Management Interface (IPMI), 113
interconnect protocol, 506–507
interconnects
clusters, 63–64
issues, 314
latencies, 331–332
saturation checks, 332
switches, 65–66
interested transaction lists (ITLs), 343
Internet Group Management Protocol (IGMP), 84, 455
Interprocess Communications (IPC) layer, 38
interval partitioning, 496
inventory directory, 117
IP addresses
Grid Infrastructure, 85–86

https://t.me/learningnets

network, 63
SCAN, 62
virtual, 60–61
IPC (Interprocess Communications) layer, 38
IPD/OS (Instantaneous Problem Detector for Clusters), 465
IPMI (Intelligent Platform Management Interface), 113
IRDP (ICMP Route Discovery Protocol), 10
ITLs (interested transaction lists), 343

J

Java
ADDM, 329
Application Continuity, 446
FAN events, 409, 441
MMON, 329
OUI, 100, 453–454
TFA, 262
jdbc driver, 62
JDBC event-based applications, 439

K

kernel components
background processes, 68–73
global cache and global enqueue services, 67
Global Resource Directory, 67
parameter settings, 96
Kernel Sequential File I/O (ksfq), 157
key networking concepts, 60–62
key values, 497
killing sessions, 230
ksfq (Kernel Sequential File I/O), 157

L

latches
GES, 340–341
local locks, 36
latencies, interconnect, 331–332
lazy remastering, 286–287
LCK (Lock) process, 73, 401

https://t.me/learningnets
Technet24

LCKx process, 339
leaf block contention, 499–502
leaf nodes in Oracle Flex Clusters, 47
LEs (lock elements) in GCS, 348, 350–352
LGWR (Log Writer), 508
library cache effect, 507–508
light-work rule
    Cache Fusion, 373
    consistent read mechanism, 354–355
Linux
    kernel, 65
    required directories, 89–90
listener-connection load balancing, 423–426
listeners
    configuring, 218
    creating, 217
    enabling and disabling, 218
    SCAN, 246
    SRVCTL commands, 217–218
    starting, 210–211
    stopping, 213–214
_lm_drm_max_requests parameter, 360
_LM_MASTER_WEIGH parameter, 289
LMD (Lock Manager Daemon), 36, 72, 400–401
LMNON processes, 72
LMON. *See* Lock Monitor (LMON)
LMS processes, 72, 400
LOAD_BALANCE parameter, 419–420
load balancing
    client-side, 419–423
    connections, 409–410, 433–434
    FAN events, 442
    hardware and software, 419
    network bonding, 65
    overview, 417–418
    runtime, 434
    server-side, 423–426
    service time for, 434–435
Load Balancing Advisory, 410
    high availability, 440
    Oracle Notification Service, 52

https://t.me/learningnets

workload balancing, 432–433
load-oriented waits, 307–308
local archives with NFS scheme, 278
local checkpoints, 280
LOCAL FOR ALL option, 223
LOCAL FOR_RIM option, 223
LOCAL_LISTENER parameter, 422, 428
local locks, 36
local node for root scripts, 116
local resources in cache coherency, 71
local roles in GCS, 349–350
local temporary tablespaces, 223–224
lock elements (LEs) in GCS, 348, 350–352
Lock (LCK) process, 73, 401
lock manager, 32, 35
Lock Manager Daemon (LMD), 36, 72, 400–401
Lock Monitor (LMON)
Cache Fusion, 400
database alert log, 457–458
DLM implementation, 36
DLM reconfiguration, 290
GES management by, 72
instance membership recovery, 466–467
messaging, 345
locks and locking concepts
asynchronous traps, 38
cache coherency, 71
Cache Fusion, 368–369
convert queues, 338
DLM lock compatibility matrix, 36–37
enqueues, 338–339, 341
GCS, 348–352, 355–357
GES, 342
global, 341–345
granting locks, 38–39
lock acquisition and conversion, 37
lock mastering, 38, 355–357, 368–369
OPS, 35–36
processes and group-based locking, 38
read-mostly, 495
resources, 336

https://t.me/learningnets
Technet24

row lock independence, 36
three-way lock messages, 345–346
Log Writer (LGWR), 508
Logical Volume Manager (LVM)
Grid Infrastructure, 91
host-based mirroring, 485
logs and log files
backups, 277
database alert, 455–459
FAN events, 441
history, 280
installation, 450
merge operation, 285
online redo, 225
Oracle Grid Infrastructure, 452–453
LONG service goals, 410
lost block optimization, 355
lsnrctl services command, 422
lsop command, 182
lspwusr command, 166
LVM (Logical Volume Manager)
Grid Infrastructure, 91
host-based mirroring, 485

M

MAC addresses
ARP for, 61
network bonding, 65
Manageability Monitor (MMON), 322, 329
management commands and options
Grid Infrastructure, 113–115
SRVCTL, 215–218
management policies for services, 408–409
manual startup for Oracle Clusterware, 253
MAP function, 357
masking database outages, 446–448
master copies in cache coherency, 70
master instance crashes, 381, 397–398
master nodes in GCS, 355
mastering and remastering Cache Fusion, 399

https://t.me/learningnets

mastership distribution in GCS, 357
MAX_SIZE attribute, 233–234, 241
maximum availability. *See also* high availability (HA)
    data mirroring, 485–488
    design considerations, 482–484
    extended clusters, 481
    Extended Oracle RAC vs. Oracle Data Guard, 489
    overview, 480
maximum transmission unit (MTU) size, 314
MAXLOGHISTORY parameter, 281
MAXTRANS parameter, 343
MBRC value, 504–505
mDNS (multicast Domain Name Service), 85
mean time to recover (MTTR), 481
measuring workloads by service, 435–436
media recovery, 283–286
medium-to-large database backups, 275
member voting in instance membership recovery, 467–468
message-oriented waits, 306
messaging
    acquisition asynchronous traps, 346
    deadlocks, 347
    overview, 345
    three-way lock messages, 345–346
    traffic controller, 347–348
    TX enqueue affinity optimization, 346–347
metadata
    ASM instances, 154
    triple-mirrored, 185
metadata directory, 452
MGMTDB database, 59, 259
migrating
    to disk groups, 189
    OCR to ASM, 267–268
    voting disks, 271
MIN_SIZE attribute, 233–234, 240
mirroring
    array-based, 485–487
    ASM, 153, 157–159, 178
    ASM preferred reads, 487
    extended clusters, 487–488

https://t.me/learningnets
Technet24

host-based, 485–487
OCR, 267
miscount parameter, 464
mkdg command, 175
mkfs command, 179
MMON (Manageability Monitor), 322, 329
modes
Cache Fusion, 376
locks, 342, 348–352
modify database command, 218
converting administrator-managed database to a policy-managed database, 228
server pools, 234–235
modify instance command, 219
modify scan command, 246
modify service command, 413, 416
modify srvpool command, 234–235
modifying
databases, 218
instances, 219
server pools, 234–235
services, 413
monitoring disk rebalancing, 184
mount command, 179
MOUNT startup modes, 166
MTS (Multithreaded Server), 38
MTTR (mean time to recover), 481
MTU (maximum transmission unit) size, 314
multicast Domain Name Service (mDNS), 85
multicasting
enabling, 83–84
Grid Infrastructure, 455
multihosting, 9
multipathing, 9
multitenant environments, 190
Multithreaded Server (MTS), 38

N

names
clusters, 103
disk groups, 178

https://t.me/learningnets

locks, 37
services, 407
netstat -s command, 314
Network File System (NFS)
backups, 278
mount points across servers, 33
shared disk system, 50
network heartbeat (NHB), 57, 463–464
network interface cards, 62–63
Network Time Protocol (NTP)
configuring, 88
node time synchronization, 66
networking stack components
cluster interconnects, 63–64
interconnect switch, 65–66
IP addresses, 63
network bonding, 64–65
network interface cards, 62–63
node time synchronization, 66
networks
bonding, 64–65
concepts, 60–62
Grid Infrastructure, 83–87
interfaces, 84, 107–108
maximum availability connectivity, 482
NFS (Network File System)
backups, 278
mount points across servers, 33
shared disk system, 50
NG form for Cache Fusion resource state, 376
NHB (network heartbeat), 57, 463–464
NL form for Cache Fusion resource state, 376
NM (Node Monitor)
database alert log, 457
instance membership recovery, 466
NOCACHE option for sequences, 503
node awareness, 32
node eviction issues
Cluster Health Monitor, 465–466
debugging, 463–465
description, 66

https://t.me/learningnets
Technet24

instance membership recovery, 466–472
Node Monitor (NM)
database alert log, 457
instance membership recovery, 466
nodes
adding, 540–542
FAN events, 442
fencing, 464
Grid Infrastructure, 85–86, 104–107
instance membership recovery, 468–471
Oracle Flex Clusters, 46–47
removing, 543–545
time synchronization, 66
NOMOUNT startup modes, 166
Non-Parallel Cache Management (Non-PCM), 71, 341
Non-Stop Forwarding (NSF) protocol suites, 10
NONE value in TAF, 427, 429
NOORDER option for sequences, 502–503
NORMAL shutdown command, 167–168
Notification parameter for Application Continuity, 447
notifications
event-based, 438
FAN, 441–444
high availability, 438
NSF (Non-Stop Forwarding) protocol suites, 10
nslookup command, 87
NTP (Network Time Protocol)
configuring, 88
node time synchronization, 66
Null (N) lock mode
Cache Fusion resource state, 376
GCS, 349–350
global locks, 342
number of instances section in AWR reports, 324

O

object affinity statistics for read-mostly locking, 361–362
OCFS (Oracle Cluster File System), 32, 51, 247
OCI (Oracle Call Interface) callbacks, 427
OCLSOMON daemon, 251–252

https://t.me/learningnets

oclumon debug command, 259
oclumon dumpnodeview command, 260
oclumon manage command, 260
OCLUMON tool, 259–260
oclumon version command, 260
OCR. *See* Oracle Cluster Registry (OCR)
ocrcheck command, 265–266, 293
ocrconfig command, 56, 265, 267–268, 293–294
ocrdump command, 265–266, 269, 294
ocrmirror command, 267
ocssd.log file, 432, 454
OCSSd (Oracle Cluster Synchronization Services daemon), 57
OCSSD process, 464
ODM (Oracle Disk Manager), 32
OEM/GC (Oracle Enterprise Manager/Grid Control), 411
OEM (Oracle Enterprise Manager)
    event-based, 438
    service level thresholds, 436
    services, 412
OHAS (Oracle High Availability Services) stack, 51, 59–60
OHASd (Oracle High Availability Service daemon), 54–55
oifcfg utility, 105–106, 260–261
OINSTALL group, 88–89
OLOGGERD (Cluster Logger Service), 59, 259
OLR. *See* Oracle Local Registry (OLR)
OLSNODES utility, 257–258, 262
OLTP (online transaction processing) service, 28, 404, 500
OMF (Oracle Managed Files), 159
    ASM, 162
    datafile names, 145
OMOTION utility, 128
One Node database, 128
online block recovery, 292
online redo log files, 225
online transaction processing (OLTP) service, 28, 404, 500
ONS (Oracle Notification Service)
    event-based, 439
    FAN events, 441
    functionalities, 52
    overview, 59
ONSd daemon, 52

https://t.me/learningnets

Technet24

OPEN_CURSORS parameter, 203
open ITL, 344
OPEN startup modes, 166
OpenSSH, 82
operating system
    authenticating, 88–89
    configuring, 82–83
Operating System Dependent (OSD) layer, 32
OPS. *See* Oracle Parallel Server (OPS)
oraagent (Oracle Agent), 59–60
ORAchk utility
    Grid Infrastructure, 82
    for troubleshooting, 475–476
Oracle Agent (oraagent), 59–60
$ORACLE_BASE/admin directory, 146
ORACLE_BASE environment variable, 134, 451
Oracle Call Interface (OCI) callbacks, 427
Oracle Cluster File System (OCFS), 32, 51, 247
Oracle Cluster Manager (OraCM), 32
Oracle Cluster Registry (OCR)
    administering, 265
    backup and recovery, 293–295
    configuration information in, 53, 227
    Grid Infrastructure, 80
    information dumping, 266
    integrity, 266
    migrating to ASM, 267–268
    mirror copy, 267
    overview, 55–56, 109–111
    services, 411
Oracle Cluster Synchronization Services daemon (OCSSd), 57
Oracle Clusterware
    cemutlo utility, 261–262
    clscfg tool, 260
    Cluster Health Monitor, 259
    Cluster Ready Services stack, 51–53
    Cluster Synchronization Services, 57
    CLUVFY utility, 247–248
    configuring and administering, 238
    CRSCTL for, 253–257
    Event Management Logger, 58

https://t.me/learningnets

GPnP tool, 258–259
Grid Infrastructure, 78
I/O fencing, 58
module debugging, 472–474
OCLUMON, 259–260
OCRCHECK, 266
OCRCONFIG, 267
oifcfg tool, 260–261
olsnodes command, 257–258
Oracle Cluster Registry, 55–56, 265–268
Oracle High Availability Services, 59–60
Oracle Local Registry, 56, 268–269
Oracle Notification Services, 59
overview, 51, 238
resource management frameworks, 54
role-separated management, 242–243
SCAN administration, 244–247
server pools, 239–241
services, 411
starting and stopping, 54
startup process, 54–55, 248–253
Trace File Analyzer, 262–265
voting disks, 56–57, 269–271
weight-based server node eviction, 243–244
Oracle Data Guard
backups, 278
extended clusters, 47
vs. Extended Oracle RAC, 489
services, 407
Oracle Data Guard Broker for FAN events, 439
Oracle Disk Manager (ODM), 32
Oracle E-Business Suite, 33
Oracle Enterprise Manager (OEM)
event-based, 438
service level thresholds, 436
services, 412
Oracle Enterprise Manager/Grid Control (OEM/GC), 411
Oracle extended clusters. *See* extended clusters
Oracle Flex Clusters, 46–47, 79
Oracle Grid Infrastructure. *See* Grid Infrastructure
Oracle High Availability Service daemon (OHASd), 54–55

https://t.me/learningnets
Technet24

Oracle High Availability Services (OHAS) stack, 51, 59–60
ORACLE_HOME directory and variable, 53
    ASMCMD logins, 192
    database alert log, 455–459
    storage location for, 134
Oracle Local Registry (OLR)
    administering, 268–269
    backup and recovery, 293
    description, 56, 227
    Grid Infrastructure, 80
Oracle Managed Files (OMF), 159
    ASM, 162
    datafile names, 145
Oracle Multitenant option, 47–48
Oracle Notification Service (ONS)
    event-based, 439
    FAN events, 441
    functionalities, 52
    overview, 59
Oracle Parallel Server (OPS), 32
    affordability, 41
    application partitioning, 494
    architecture, 33–34
    availability, 40–41
    Cache Fusion Stage 1, 39–40
    components, 34
    dirty blocks, 285
    locking concepts, 35–39
    scalability, 41
Oracle RAC One Node database installation option, 132
Oracle Real Application Cluster database installation option, 132–133
Oracle Resource Manager, 406
Oracle Root Agent (orarootagent), 59–60
Oracle Scheduler, 411–414
ORACLE_SID environment variable, 165, 192, 204
Oracle Support Services (OSS), 261, 266
Oracle Universal Installer (OUI)
    clusters, 129–130, 135
    Grid Infrastructure, 82, 100–101
    logs, 453–454
    Oracle Cluster Registry for, 55

https://t.me/learningnets

Oracle virtual IPs, 60–61
Oracle XA client, 38
ORACLEASM_SCANEXCLUDE variable, 95
ORACLEASM_SCANORDER variable, 95
oracleasmlib RPM, 92
OraCM (Oracle Cluster Manager), 32
oradebug command
    system-state dumps, 461
    traffic controller, 348
orainstRoot.sh script, 117, 120
orarootagent (Oracle Root Agent), 59–60
ORDER option for sequences, 503
OSASM group, 88
OSBACKUPDBA group, 88
OSD (Operating System Dependent) layer, 32
OSDBA group, 88
OSDGDBA group, 88
OSKMDBA group, 88
OSOPER group, 88
OSS (Oracle Support Services), 261, 266
OSYSMOND (System Monitor Service), 259
OUI. *See* Oracle Universal Installer (OUI)
out-of-band notification for FAN events, 441
outages
    Application Continuity, 446–448
    planned and unplanned, 6
overhead for cache coherency, 70

P

PaaS (Platform as a Service) model, 16–17
packages for ADDM, 329
PAMs (Pluggable Authentication Modules), 96
Parallel Cache Management (PCM)
    cache coherency, 70
    GCS, 351
    lock space, 286
    locks, 341
    overview, 35–36
    resources, 337
parallel_max_servers parameter, 512

https://t.me/learningnets
Technet24

parallel query slaves, 303
parallel threads, 278
parallelism, 21
parsing, library cache effect in, 507–508
partitioning
    applications, 418, 494–495
    data, 496–498
    index, 499–502
    workloads, 300
passwords
    clusters, 145
    SYS and ASMSNMP users, 112–113
Past Image (PI)
    blocks in Cache Fusion, 368
    consistent read mechanism, 353
    GCS, 349–350
    overview, 284
PATH environment variable, 192
PCM. *See* Parallel Cache Management (PCM)
PDBs. *See* pluggable databases (PDBs)
Perform Prerequisite Checks screen
    errors reported on, 453
    Grid Infrastructure, 115–116, 119
    minimum requirements on, 135, 137
performance, 298
    ADDM, 329
    ASH reports, 330
    ASMLib functions for, 196
    AWR reports, 322–328
    block size, 302
    Cache Fusion, 483–484
    cluster interconnects, 330
    design considerations, 298–300
    enqueue tuning, 321
    Global Cache Service times, 317–320
    global cache statistics, 315–318
    index rebuilding for, 500
    interconnect latencies, 331–332
    mirroring, 487
    partitioning workloads, 300
    private interconnects, 330–331

https://t.me/learningnets

scalability, 300–302
troubleshooting, 460–463
views, 302–303
wait events. *See* wait events
persistent reservation (SCSI3) technique, 58
pfiles, 154
PHYS_META_REPLICATED attribute, 176
Physical writes statistics in AWR report, 325
PI. *See* Past Image (PI)
pings
Cache Fusion, 40, 367–368
Grid Infrastructure, 86–87
PL/SQL tuning, 329
planned and unplanned outages, 6
Platform as a Service (PaaS) model, 16–17
Pluggable Authentication Modules (PAMs), 96
pluggable databases (PDBs)
database status, 212
Grid Infrastructure, 79
managing, 220
services, 413
PMON process, 424–425
policy-based management for server pools, 239
policy-managed clusters, 238
policy-managed databases
vs. administrator-managed databases, 48–49, 146
converting from administrator-managed databases, 235–236
services, 410
pre-installation checks for adding nodes, 540–541
PRECONNECT policy, 427, 429
predict database command, 219
predict diskgroup command, 219
predict listener command, 220
predicting failure impact, 219–220
PREFER_LEAST_LOADED_NODE_ parameters, 426
preferred instances of services, 407–408
preferred reads in ASM, 487
Preinstallation RPMs, 82–83
PREPARE phase in disk rebalancing, 183
prerequisite checks for Grid Infrastructure, 115–116
PREVIOUS_MASTER column, 361

https://t.me/learningnets
Technet24

primary/secondary setup, mirroring, 486
private interconnects, verifying use of, 330–331
private network interfaces, 80
private node addresses, 86
Privileged Operating System Groups screen, 113, 115, 134, 136
privileged OS system groups, 113–115
PROCESS STATE dumps, 462
processes
    background. *See* background processes
    with locks, 38
public network interfaces, 80
public node addresses, 85–86

Q

Quality of Service (QoS) Management, 49, 244
queries
    Cache Fusion, 384–385
    parallel query slaves, 303
query css votedisk command, 270
quorums
    extended clusters, 488
    voting disks, 56
quota groups in ASM, 189

R

RAC ON and OFF, 459–460
rapid recovery, 481
RAW disk partitions, 32
RBAL background process, 171–172
RBAs (redo byte addresses), 279, 285
RDG (Reliable DataGram protocol), 458
read-mostly locking, 357–362, 495
read/read contention, 39–40, 369
read/write contention, 40
reader locks, 337
reader nodes, 47
REBALANCE POWER option, 182
rebalancing, disk. *See* disk rebalancing
rebootless node fencing, 464
REBULD operation in disk rebalancing, 183

https://t.me/learningnets

reconfiguration
cluster, 468
dynamic, 286–292
fast, 287–289
Global Resource Directory, 290
records
redo, 279
result, 468
RECOVER command, 278
recovery
ASM, 185–186
Cache Fusion, 285–286, 289–292
crash, 281–286
dynamic reconfiguration and affinity remastering, 286–292
instance membership recovery, 466–472
instances, 278–280, 282–283
lock claims, 291
Oracle Cluster Registry, 293–294
two-pass, 284–285
voting disks, 292
recovery sets, 284
redo and redo logs
administering, 225
backups, 277
Cache Fusion recovery, 291
records and change vectors, 279
threads and streams, 279
redo byte addresses (RBAs), 279, 285
redundancy
ASM, 157–159
availability level, 8
components for, 9–10
maximum availability, 482
REFER_LEAST_LOADED_NODE parameter, 425
registry files, 165
reliability, ASMLib functions for, 196
_RELIABLE_BLOCK_SENDS parameter, 458
Reliable DataGram protocol (RDG), 458
Reliable Shared Memory (RSM), 458
relocate database command, 219
relocate instance command, 219

https://t.me/learningnets
Technet24

relocate server command, 233, 235
relocate service command, 229, 414
relocating
    failed services, 414
    servers to server pools, 235
    services, 219
REMASTER_CNT column, 361
remastering
    affinity, 286–292
    Cache Fusion, 399
REMOTE_LISTENER parameter, 61–62, 424, 426, 428
REMOTE_LOGIN_PASSWORDFILE parameter, 166
remote operations, 254
Remote Slave Monitor (RSMN) process monitors, 73
remove database command, 219
remove instance command, 219
remove scan command, 246
remove srvpool command, 235
removing
    nodes, 543–545
    server pools, 235
renamedg command, 178
renaming disk groups, 178
replace votedisk command, 271
replacing disks in disk groups, 177
replay_init_time parameter, 447
replication in maximum availability, 484
reports
    ASH, 330
    AWR. *See* Automatic Workload Repository (AWR) reports
reserve/release (R/R) technique, 58
RESILVER operation, 183
RESIZE option, 224
resizing disk group disks, 177–178
resources
    cache coherency, 70–71, 339
    Cache Fusion, allocation, 386
    Cache Fusion, mastering and remastering, 399
    extended clusters, 481
    FAN events, 443
    GCS, mastering, 355–357

https://t.me/learningnets

GCS, names, 385–386
GCS, weight, 356–357
grants and conversions, 337–338
locks and enqueues, 338–339
Oracle Clusterware, 51, 54
overview, 336–337
read-mostly locking, 358–360
RESTORE phase in disk rebalancing, 183
restoring voting disks, 270
result records in instance membership recovery, 468
RESYNC operation in disk rebalancing, 183
Retention parameter
Application Continuity, 447
Transaction Guard, 445
reverse key indexes, 500–501
RLB (runtime load balancing), 434
RMAN in backups, 276–278
RMSn (Oracle RAC Management) processes, 73
role-separated management, 242–243
Root Script Execution Configuration screen, 118
root scripts for Grid Infrastructure, 116–122
root.sh script, 117–120, 123, 138, 454–455
rootcrs.pl script, 454–455
row cache, 71
row-level locks, 343–344
row lock independence, 36
rows in Cache Fusion
selection, 382, 395–396
updates, 391–395
RSM (Reliable Shared Memory), 458
RSMN (Remote Slave Monitor) process monitors, 73
run parameter, 250
Run queue length of the node metric, 433
run time load balancing, 432
runcluvfy.sh script, 97, 124
runtime load balancing (RLB), 434

S

SaaS (Software as a Service) model, 16–17
Sales Cloud 2, 17

https://t.me/learningnets
Technet24

SAME (Stripe and Mirror Everything) strategy, 153, 178
SANs (storage area networks), 153–154
saturation checks, interconnect, 332
scalability
    clusters, 20–21
    description, 15
    load balancing, 418
    network bonding, 65
    OPS, 41
    performance, 300–302
    solutions, 15–16
SCAN. *See* Single Client Access Name (SCAN)
scannumber parameter, 245–246
schedulers, Oracle Scheduler, 411–414
SCNs. *See* system commit numbers (SCNs)
scripts
    clusters, 135
    Grid Infrastructure, 82
scrubbing disk groups, 185
search order for parameter files, 209–210
second-pass log reads in recovery, 291
SECTOR_SIZE attribute, 176
secure shell (SSH)
    configuration, 95
    connectivity verification, 104–107
Select Cluster Configuration screen, 102–103
Select Database Creation Mode screen, 142
Select Database Creation Option screen, 145
Select Database Deployment Type screen, 142–143
Select Database Edition screen, 133–134
Select Database Installation Option screen, 132
Select Database Operation screen, 141
Select Database Storage Option screen, 145
Select Fast Recovery Option screen, 145
Select Installation Option screen, 101–102, 131
Select List of Nodes screen, 133, 144
Select Oracle Data Vault Config Option screen, 145
SELECT value in TAF, 429
sequences
    best practices, 503–504
    CACHE and NOORDER options, 502–503

https://t.me/learningnets

managing, 231
overview, 502
Server Control (SRVCTL) services, 411
database configuration, 227–230
debugging information, 474
management commands, 215–220
resource monitoring with, 62
server pools, 146–147
starting and stopping instances, 210–214
server pools
administering, 232–233
assigning, 409
benefits, 239
categorization, 239
configuring, 233–234
creating, 234–235, 240–241
deleting, 241
description, 48, 146
evaluating, 241
operation, 239–240
policy-based management, 239
removing, 235
status, 235
types, 240
server-side callouts for FAN events, 444
server-side load balancing, 409, 423–426
serverpool attribute, 233
servers attribute, 233
SERVICE events, 442
service level thresholds, 436–437
service statistics section in AWR reports, 327
service time for load balancing, 434–435
service wait class statistics section in AWR reports, 327–328
SERVICEMEMBER events, 442
SERVICEMETRICS events, 442
services
configuring, 414
DTP, 415–416
enabling and disabling, 414
FAN events, 442
management policy, 408–409

https://t.me/learningnets
Technet24

measuring workloads by, 435–436
modifying, 413
Oracle Scheduler, 411–414
policy-managed databases, 410
relocating, 219, 414
starting and stopping, 413
status, 414
views, 414–415
Session count by instance metric in load balancing, 433
SESSION value in TAF, 429
sessions, killing, 230
Set Up Software Only option, 102
setenv command, 218
setperm command, 242–243
setperm resource command, 243
SG form for Cache Fusion resource state, 376
SGA (System Global Area). *See* System Global Area (SGA)
shared disk architecture, 25–29
shared disk system, 49–51
shared everything architecture, 25–27, 29–30, 51
shared (S) lock mode
Cache Fusion resource state, 376
GCS, 349–350
global locks, 342
shared exclusive lock mode, 342
shared nothing architecture, 25–28
shared pool latches, 36
shared resources, 336
shared storage
clusters, 25–30
configuring, 90–95
Grid Infrastructure, 80
SHORT service goals, 410
Show Candidate/Provisioned Disks button, 112
show sga command, 166
showclustermode command, 188
SHRINK SPACE option, 224
SHUTDOWN command, 167–168
shutting down ASM instances, 167–169
SID Prefix values, 144
silent installation method for clusters, 136

https://t.me/learningnets

Single Client Access Name (SCAN)
administering, 244
client-side load balancing, 420
database connections, 231–232
Grid Infrastructure, 80, 84–87
overview, 61–62
starting and stopping, 244–245
status, 245–246
Single instance database installation option, 132
single-instance environments, 45–46
single point of failure (SPOF), 6
singleton applications
DTP service, 415
policy-managed deployment, 48, 232, 410
server pools, 239, 409
sites in Grid Infrastructure, 79
SL form for Cache Fusion resource state, 376
small, noncritical database backups, 274–275
smart scans, 23–25
SMON (System Monitor) process
Cache Fusion recovery, 285–286, 289–292
crash recovery, 282
SMP (symmetric multiprocessing) systems, 21–22, 345
Snap SCNs, 352–354
snapshots
ACFS, 161–162, 180
backups, 275
soft pings, 40
Software as a Service (SaaS) model, 16–17
software load balancing, 419
sorted hash clusters, 501
Specify ASM Password screen, 112
Specify Configuration Options screen, 145
Specify Database Identification Details screen, 144
Specify Database User Credentials screen, 145
Specify Fast Recovery Area option, 145
Specify Installation Location screen, 114, 116, 134–135
Specify Management Options screen, 113–114
Specify Network Interface Usage screen, 108
speed of light, 482
SPFILEs, 154, 202–203

https://t.me/learningnets
Technet24

split-brain
disk heartbeats for, 57
instance membership recovery, 468
resolution, 66
voting disks for, 110
split-mirror for backups, 275
SPOF (single point of failure), 6
SQL commands and interface
ASM, 153
MMON analysis of, 329
scope, 231
SQL*Plus for starting and stopping instances, 214–215
SRV_PRECONNECT events for FAN, 442
srvctl config database command, 209–210
SRVCTL utility. *See* Server Control (SRVCTL) services
SRVM_TRACE environment variable, 474–475
SSH (secure shell)
configuration, 95
connectivity verification, 104–107
SSYSKM administrative privilege, 89
Standard Oracle ASM configuration, 150–151
start asm command, 214
start cluster command, 254–255
start crs command, 253, 256
start database command, 210–211
start instance command, 212
start service command, 220
startcheck parameter, 250
starting
CRS, 253–254
instances, 210–215
Oracle Clusterware, 54–55, 248–253
SCAN, 244–245
services, 413
STARTUP command, 166
startup modes for ASM instances, 166
statistics
Cache Fusion, 303
global cache, 315–318
status
CRS, 255

https://t.me/learningnets

FAN events, 442
instances, 211–212
SCAN, 245–246
server pools, 235
services, 414
status asm command, 214
status database command, 211
status instance command, 211, 228
status listener command, 229
status nodeapps command, 229–230
status scan command, 245
status service command, 212, 229, 414
status srvpool command, 235
stop command, 212
stop asm command, 214
stop cluster command, 255
stop crs command, 253, 294
stop database command, 212
stop diskgroup command, 213
stop scan command, 244–245
stop service command, 220
stopping
CRS, 253–254
instances, 210–215
SCAN, 244–245
services, 413
storage area networks (SANs), 153–154
Storage Option Information screen, 108–109
storage options for Grid Infrastructure, 108–113
straight disks in ASM, 154
streams, redo, 279
Stripe and Mirror Everything (SAME) strategy, 153, 178
striping techniques in ASM, 153, 178
subshared (S) lock mode, 342
subshared exclusive (SSX) lock mode, 342
Summary screen
clusters, 136–138
database creation, 146–147
Grid Infrastructure, 115, 120
symmetric multiprocessing (SMP) systems, 21–22, 345
synchronization

https://t.me/learningnets
Technet24

description, 336
nodes, 66
synchronous replication for maximum availability, 484
SYS and ASMSNMP users, passwords for, 112–113
SYS$BACKGROUND service, 412
SYS$USERS service, 412
SYSAUX (auxiliary tablespace), 329
SYSBACKUP administrative privilege, 89
SYSDBA administrative privilege, 89
SYSRAC administrative privilege, 88
system capacity issues, 329
system commit numbers (SCNs)
Cache Fusion recovery, 291–292
checkpoints, 279–280
commit, 459
snap, 352–354
system failover level of availability, 8
System Global Area (SGA), 53
cache coherency, 70
GRD in, 339
instances, 33, 45, 67, 155
system-state dumps, 461
System Monitor Service (OSYSMOND), 259
System Monitor (SMON) process
Cache Fusion recovery, 285–286, 289–292
crash recovery, 282
system-state dumps
hung databases, 460–461
hung sessions, 461–463
SYSTEMSTATE dump
hung databases, 461
hung sessions, 461–463

T

table locks, 345
table overlap, 494
tables, managing, 231
TAF. *See* Transparent Application Failover (TAF)
temporary tablespaces
administering, 222–224

https://t.me/learningnets

hierarchy, 224
TFA (Trace File Analyzer), 262–265
tfactl utility, 262–265
THIN_PROVISIONED attribute, 175
THREAD parameter, 205
threads
checkpoints, 280
merge mechanism, 283–285
redo, 279
three-way events, 305–306
three-way lock messages, 345–346
tnslsnr process, 155
top 5 CR and current blocks segments sections in AWR reports, 328
Top Cluster Events section in ASH reports, 330
Top Remote Instance section in ASH reports, 330
trace directory, 451
Trace File Analyzer (TFA), 262–265
trace files
infrastructure installation, 454
location, 474
traffic controller (TRFC), 347–348
transaction enqueues. *See* TX (transaction) enqueues
Transaction Guard
Application Continuity, 446
client failover, 445
transaction IDs (XIDs)
global locks, 343
lock requests, 38
transaction locks, 343–344
transaction tables, 343
TRANSACTIONAL shutdown command, 167–168
Transparent Application Failover (TAF)
configuring, 428–430
considerations, 427–428
description, 60–61
extended clusters, 481
failover validation, 430–432
overview, 427
service management policy, 409
troubleshooting, 432
TRFC (traffic controller), 347–348

https://t.me/learningnets
Technet24

triple-mirrored metadata, 185
troubleshooting
    Cluster Health Monitor, 465–466
    database alert log, 455–459
    database performance, 460–463
    infrastructure installation, 453–455
    installation log files for, 450
    instance membership recovery, 466–472
    log directory structure, 451–453
    node eviction issues, 463–465
    ORAchk for, 475–476
    Oracle Clusterware modules, 472–474
    RAC ON and OFF, 459–460
    TAF, 432
    utilities, 474–475
24/7 forever term, 4–5
two-pass recovery, 284–285, 289
two-way events, 305–306
TX (transaction) enqueues
    affinity optimization, 346–347
    cache coherency, 339–340
    description, 36
    tuning, 321
TYPE parameter, 429
Typical configuration option, 142

U

UBA (undo byte address), 343
UDP (User Datagram Protocol), 314, 506–507
UFS (UNIX file systems), 50
Unbreakable Enterprise Kernel (UEK), 91–92
Unbreakable Linux Network (ULN), 82
undo byte address (UBA), 343
UNDO_MANAGEMENT parameter, 221
UNDO_RETENTION_TIME parameter, 221
UNDO_TABLESPACE parameter, 205, 221
undoing transactions, 221–222
undropping disks from disk groups, 177
unicast routing, 84
uniform applications, server pools for, 239

https://t.me/learningnets

UNIFORM services, 48, 410
unique parameters, 203–206
UNIX file systems (UFS), 50
unplumbing IP addresses, 86
unsetenv command, 218
UP events in high availability, 440
Upgrade an existing database option, 131
user-created server pools, 240
User Datagram Protocol (UDP), 314, 506–507
users in Grid Infrastructure, 88–89, 95–96
utilities, debugging, 474–475

V

V$ views, 302–303
V$ACTIVE_SERVICES view, 434
V$ACTIVE_SESSION_HISTORY view, 436
V$ASM views, 170
V$ASM_ACFSVOLUMES view, 179
V$ASM_DISK_STAT view, 184
V$ASM_DISKGROUP_STAT view, 184
V$ASM_ESTIMATE view, 184–185
V$ASM_FILESYSTEM view, 179
V$ASM_INSTANCES view, 215
V$ASM_OPERATION view, 181–184
V$BH view, 385
V$CACHE view, 512–514
V$CACHE_TRANSFER view, 514
V$CLIENT view, 170
V$CLUSTER_INTERCONNECTS view, 537–538
V$CONFIGURED_INTERCONNECTS view, 538
V$CR_BLOCK_SERVER view, 354–355, 516–517
V$CURRENT_BLOCK_SERVER view, 517–518
V$DLM_RESS view, 382
V$ENQUEUE_STATISTICS view, 522–531
V$ESTIMATE view, 170
V$GC_ELEMENT view, 518–519
V$GCSHVMASTER_INFO view, 536
V$GCSPFMASTER_INFO view, 361, 537
V$GES_BLOCKING_ENQUEUE view, 520–521
V$GES_CONVERT_LOCAL view, 534

https://t.me/learningnets
Technet24

V$GES_CONVERT_REMOTE view, 535
V$GES_ENQUEUE view, 382, 532–533
V$GES_MISC view, 345
V$GES_RESOURCE view, 535
V$GES_STATISTICS view, 532
V$GES_TRAFFIC_CONTROLLER view, 347
V$HVMASTER_INFO view, 536
V$INSTANCE_CACHE_TRANSFER view, 515–516
V$INSTANCE_RECOVERY view, 280, 459
V$LOCK view, 519–520
V$LOCKED_OBJECT view, 532
V$LOG view, 225
V$LOGFILE view, 225
V$PWFILE_USERS view, 166
V$RESOURCE_LIMIT view, 321
V$SEGMENT_STATISTICS view, 382, 384, 390
V$SERV_MOD_ACT_STATS view, 436
V$SERVICE_EVENT view
    dynamic database services, 406
    workload measurements, 436
V$SERVICE_METRICS view, 434
V$SERVICE_METRICS_HISTORY view, 434–435
V$SERVICE_STATS view
    dynamic database services, 405–406
    workload measurements, 436
V$SERVICE_WAIT_CLASS view, 436
V$SERVICEMETRIC view
    dynamic database services, 405–406
    workload measurements, 436
V$SERVICEMETRIC_HISTORY view, 436
V$SERVICES view
    dynamic database services, 406
    load balancing, 434
    service information, 414–415
V$SESSION view
    application contexts, 416–417
    processes, 38
    service level thresholds, 437
    wait events, 311
    workload measurements, 436
V$SESSION_EVENT view, 311

https://t.me/learningnets

V$SESSION_LONGOPS view, 506
V$SESSION_WAIT view, 505
V$SQL_PLAN view, 506
V$SYSSTAT view, 300, 505
V$SYSTEM_EVENT view, 507
V$TEMPFILE view, 223
validating
    global resource directory, 291
    OCR backups, 295
    TAF, 430–432
VBGn background process, 172
VBM background process, 171
VDBG background process, 171
verifying
    Grid Infrastructure installation, 124–125
    private interconnect use, 330–331
    SSH connectivity, 104–107
Veritas Cluster File Systems (VxFS), 32
vertical scalability, 15
very large databases (VLDBs), 156
views
    cluster interconnect, 537–538
    dynamic resource remastering, 536–537
    GCS operations, 512–519
    Global Enqueue Services, 519–535
    introduction, 302–303
    services, 414–415
virtual IPs, 60–61
Virtual Router Redundancy Protocol (VRRP), 10
VLDBs (very large databases), 156
volcreate command, 161, 179
voldelete command, 161
volume allocation units
    ASM, 157
    description, 178
Volume Table of Contents (VTOC), 94
voting disks
    adding and deleting, 271
    ASM storage for, 269–270
    backup and recovery, 270, 292
    description, 33

https://t.me/learningnets
Technet24

migrating, 271
overview, 56–57, 269
restoring, 270
voting in instance membership recovery, 467–468
VRRP (Virtual Router Redundancy Protocol), 10
VTOC (Volume Table of Contents), 94
VxFS (Veritas Cluster File Systems), 32

W

wait events
avoiding, 315
block-oriented, 306
buffer busy global cr, 313
contention-oriented, 307
gc cr block 2-way, 309–310
gc cr block 3-way, 310–311
gc cr block busy, 314
gc cr block congested, 315
gc cr disk read, 312–313
gc cr failure, 313
gc current block 2-way, 308–309
gc current block 3-way, 309
gc current block busy, 311–312
gc current block congested, 315
gc current buffer busy, 315
gc current grant 2-way, 311
interconnect-related issues, 314
load-oriented, 307–308
message-oriented, 306
overview, 303–304
two-way and three-way, 305–306
wait time in ADDM, 329
wavelength-division multiplexing (WDM) technology, 482–483
web scale, 30
weight-based server node eviction, 243–244
workload balancing. *See also* load balancing
connection load, 433–434
connection pooling, 432–433
RLB, 434
workload characterization, 496

https://t.me/learningnets

workloads
measuring by service, 435–436
partitioning, 300
wrapper scripts, 249
write propagation in mirroring, 486
write/read contention, 369, 371
write/write contention, 40, 369, 373–375

X

X (exclusive) lock mode
Cache Fusion resource state, 376
GCS, 349–350
global locks, 342
x$views, 512
X$BH view, 351, 382, 385
X$KCLCRST.LIGHT1, 354
X$KJBL view, 382, 386
X$KJMSDP view, 400
X$KSLHOT view, 498
X$KSXPCLIENT view, 332
X$KSXPIA view, 331
X$KSXPPING view, 331
X$LE view, 382
X$OBJECT_POLICY_STATISTICS view, 361
XA transactions (global transactions), 415
XDMG background process, 172
XG form for Cache Fusion resource state, 376
XIDs (transaction IDs)
global locks, 343
lock requests, 38
XL form for Cache Fusion resource state, 376

https://t.me/learningnets
Technet24

Beta Test Oracle Software

Get a first look at our newest products—and help perfect them. You must meet the following criteria:

- ✓ Licensed Oracle customer or Oracle PartnerNetwork member
- ✓ Oracle software expert
- ✓ Early adopter of Oracle products

Please apply at: pdpm.oracle.com/BPO/userprofile

ORACLE®

If your interests match upcoming activities, we'll contact you. Profiles are kept on file for 12 months.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates.

https://t.me/learningnets

Join the Largest Tech Community in the World

Download the latest software, tools, and developer templates

Get exclusive access to hands-on trainings and workshops

Grow your professional network through the Oracle ACE Program

Publish your technical articles – and get paid to share your expertise

Join the Oracle Technology Network
Membership is free. Visit community.oracle.com

@OracleOTN facebook.com/OracleTechnologyNetwork

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates.

https://t.me/learningnets

Technet24

ORACLE
CERTIFICATION PROGRAM

BOOST YOUR PROFESSIONAL IMAGE

The Value of Oracle Certification

Confidently display your globally recognized skills with an Oracle certification digital badge. Your digital badge can be added to all your on-line profiles and is displayed and validated immediately.

Advance Your Career with Oracle Certification

- Positive impact on professional image 60%
- Moved to a career in IT 26%
- Received a salary increase 20%
- Found a job 19%
- Got a promotion 14%

88% of IT professionals said they would recommend certification to a colleague when discussing a career or advancement in IT

Three Oracle Certified Professionals Tell us Their Stories

Saint-Paul, Software Architect and Developer, Oracle Certified Associate, Java SE 8 Programmer
"By becoming certified, I learned about new APIs which helped me update different parts of my programs and improve some of our software performances. It also adds credibility to my knowledge."

Adrienne, Data Administrator, Oracle Database 12c Certified SQL Associate
"Certifications are a way for me to stay current with the breadth and depth of the products."

Giovanni, Oracle Developer, Oracle Certified PL/SQL Associate
"But what the certifications have given me is more confidence and also complimentary attributes to my resume, thus allowing me to increase my salary."

Read more stories at
http://www.oracle.com/certstory

GET ORACLE CERTIFIED http://education.oracle.com/certification/

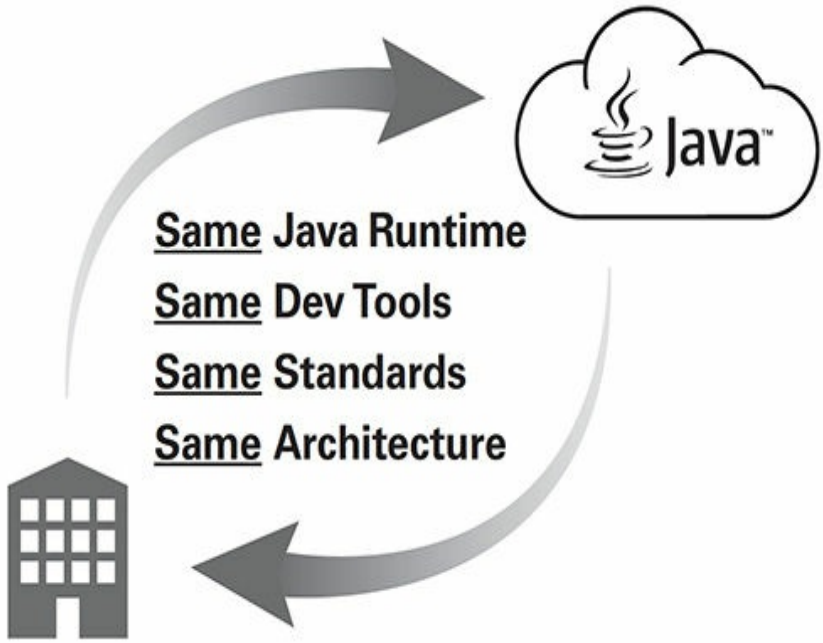
ORACLE Integrated Cloud Applications & Platform Services

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

https://t.me/learningnets

Push a Button

Move Your Java Apps to the Oracle Cloud



... or Back to Your Data Center

ORACLE®

cloud.oracle.com/java

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates.

https://t.me/learningnets

Technet24

Oracle Learning Library

Created by Oracle Experts
FREE for Oracle Users

- ✓ Vast array of learning aids
- ✓ Intuitive & powerful search
- ✓ Share content, events & saved searches
- ✓ Personalize your learning dashboard
- ✓ Find & register for training events

ORACLE®

oracle.com/oll

Copyright © 2017, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates.

https://t.me/learningnets

Reach More than 640,000 Oracle Customers with Oracle Publishing Group

Connect with the Audience that Matters Most to Your Business

ORACLE MAGAZINE

Oracle Magazine
The Largest IT Publication in the World
Circulation: 325,000
Audience: IT Managers, DBAs, Programmers, and Developers

PROFIT

Profit
Business Insight for Enterprise-Class Business Leaders to Help Them Build a Better Business Using Oracle Technology
Circulation: 90,000
Audience: Top Executives and Line of Business Managers

Java magazine

Java Magazine
The Essential Source on Java Technology, the Java Programming Language, and Java-Based Applications
Circulation: 225,00 and Growing Steady
Audience: Corporate and Independent Java Developers, Programmers, and Architects

For more information or to sign up for a FREE subscription: Scan the QR code to visit Oracle Publishing online.

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

ORACLE

https://t.me/learningnets

Technet24

Table of Contents

Title Page 3
Copyright Page 5
Dedication 7
Contents 10
Contents 10
Acknowledgments 22
Introduction 25
PART I High Availability Architecture and Clusters 30
1 Introduction to High Availability and Scalability 31
High Availability 31
HA Terminology 32
Planned and Unplanned Outages 34
An End-to-End Perspective 34
Cost of Downtime 35
Building Redundant Components 38
Common Solutions for HA 39
Cluster, Cold Failover, and Hot Failover 40
HA Option Pros and Cons 44
Scalability 45
The Oracle RAC Solution 45
Agility 46
The Oracle Database 12c RAC Solution 48
Summary 49
2 Oracle Database Clustering Basics and Its Evolution 50
Cloud Computing with Clusters 53
Shared Storage in Clustering 58
Types of Clustering Architectures 58
Hadoop Clusters 64
Architecture of Hadoop 64
Historical Background of Oracle RAC 66
Oracle Parallel Storage Evaluation 66
Oracle Parallel Server Architecture 67
Components of an OPS Database 68
Cluster Group Services (CGS) 69

https://t.me/learningnets

Distributed Lock Manager (DLM) 69
Locking Concepts in Oracle Parallel Server 70
Cache Fusion Stage 1, CR Server 74
The Oracle RAC Solution 76
Availability 76
Scalability 76
Affordability 77
Summary 77
3 Oracle RAC Architecture 79
Introduction to Oracle Real Application Clusters 79
Single-Instance Environment vs. Oracle RAC Environment 81
Oracle Flex Clusters 83
Oracle Extended Clusters 84
Oracle Multitenant and Oracle RAC 84
Administrator- vs. Policy-Managed Databases 84
Dynamic Services 85
Oracle Database Quality of Service Management 85
Oracle RAC Components 86
Shared Disk System 87
Oracle Clusterware 88
The Oracle High Availability Services Technology Stack 98
Oracle RAC Networking Concepts and Components 99
Key Networking Concepts 99
The Networking Stack Components 102
Oracle Kernel Components 107
Global Cache and Global Enqueue Services 107
Global Resource Directory 108
Oracle RAC Background Processes 108
Summary 115
PART II Installation, Configuration, and Storage 116
4 Oracle Grid Infrastructure Installation 117
An Overview of the Grid Infrastructure Installation Process 120
Preinstallation Tasks 122
Understanding the Installer, CVU, and ORAchk 122
Configuring the Operating System 123
Configuring the Network 124
Configuring NTP 129
Setting Up the Groups and Users 130

https://t.me/learningnets
Technet24

Creating the Required Linux Directories 131
Configuring Shared Storage 132
Secure Shell and User Limits Configuration 138
Setting User Limits 139
Configuring the Kernel Parameters 139
Running the Cluster Verification Utility 140
Installing Oracle Grid Infrastructure with OUI 144
Installing Oracle Grid Infrastructure 145
Choosing the Installation Options and Naming Your Cluster 146
Specifying the Cluster Nodes and Verifying SSH Connectivity 149
Specifying the Network Interfaces 153
Selecting the Storage Options 155
Specifying Management Options and Privileged OS System Groups 161
Performing the Prerequisite Checks 166
Running the root Scripts 169
The Product Installation 172
Verifying the Oracle Grid Infrastructure Installation 174
Summary 178
5 Installing Oracle RAC and Creating an Oracle RAC Database 179
Installing Oracle Real Application Clusters 181
Creating the Oracle RAC Database 194
Summary 205
6 Automatic Storage Management 206
Standard Oracle ASM and Oracle Flex ASM 207
Introduction to Automatic Storage Management 207
Physical Limits of ASM 208
ASM in Operation 209
ASM Building Blocks 211
Managing Oracle ASM Files and Directories 221
ASM Filenames 221
Creating and Referencing ASM Files 222
Managing Disk Group Directories 223
ASM Administration and Management 224
Managing an ASM Instance 225
ASM Initialization Parameters 234
Managing ASM Disk Groups 236
Creating a Disk Group 238

https://t.me/learningnets

Adding Disks to a Disk Group 238
Dropping, Undropping, Resizing, and Renaming Disks in a Disk Group 239
Administering ACFS 240
Setting Up ACFS 241
Creating an ACFS Snapshot 242
ASM Disk Rebalancing 243
Manually Rebalancing a Disk Group 245
Rebalancing Phase Options 246
Monitoring the Performance of Balancing Operations 247
Tuning Disk Rebalancing Operations 247
Backup and Recovery in ASM 249
ASM Flex Clusters 249
Configuration of Oracle ASM in Flex ASM 250
Setting Up Flex ASM 251
Managing ASM Flex Disk Groups 252
Understanding ASM File Groups and ASM Quota Groups 253
Oracle Extended Disk Groups 256
ASM Tools 256
ASMCA: The ASM Configuration Assistant 257
ASMCMD: The ASM Command-Line Utility 257
ASM FTP Utility 259
ASMLib 260
Installing ASMLib 261
Configuring ASMLib 261
Oracle ASM Filter Driver (Oracle ASMFD) 263
Summary 264
PART III Oracle RAC Administration and Management 265
7 Oracle RAC Basic Administration 266
Oracle RAC Initialization Parameters 267
Parameters That Are Unique to an Instance 268
Identical Parameters 271
Instance Parameters That “Should” Be the Same 273
Managing the Parameter File 274
Backing Up the Server Parameter File 275
Search Order for the Parameter Files in an Oracle RAC Database 275
Starting and Stopping Instances 276
Using SRVCTL to Start/Stop Databases and Instance(s) 277

https://t.me/learningnets
Technet24

Administering the Oracle ASM Instances 280
Using CRSCTL to Stop Databases and Instances 281
Using SQL*Plus to Start/Stop Instances 281
Common SRVCTL Management Commands 282
Database-Related SRVCTL Commands 283
Instance-Related SRVCTL Commands 285
Listener-Related SRVCTL Commands 285
Setting, Unsetting, and Displaying the Environment Variables 286
Changing the Configuration of Databases and Instances 286
Relocating Services 287
Removing the Configuration Information for Specific Targets 287
Predicting the Impact of Failures 287
Managing Pluggable Databases in a RAC Environment 288
Administering Undo in an Oracle RAC Database 289
Administering a Temporary Tablespace 290
Managing Traditional (Global) Tablespaces 291
Managing Local Temporary Tablespaces 292
Hierarchy of Temporary Tablespaces 293
Administering Online Redo Logs 294
Enabling Archive Logs in the Oracle RAC Environment 294
Enabling the Flashback Area 295
Managing Database Configuration with SRVCTL 296
Killing Sessions on Specific Instances of a Cluster 300
Managing Database Objects 301
Managing Sequences 301
Managing Tables 301
Managing Indexes 302
Scope of SQL Commands 302
Database Connections 302
Administering Server Pools 303
Configuring Server Pools 304
Consolidation of Databases 305
Creating a Server Pool 305
Converting an Administrator-Managed Database to a Policy-Managed Database 307
Summary 307
8 Managing Oracle Clusterware 309
Configuring and Administering Oracle Clusterware 310
https://t.me/learningnets

Benefits of Server Pools 310
Server Pools and Policy-Based Management 310
Server Pools and Categorization 311
How Server Pools Work 311
Types of Server Pools 312
Creating Server Pools 312
Evaluating the Addition of a Server Pool 313
Deleting a Server Pool 314
Role-Separated Management 314
Managing Cluster Administrators 314
Configuring Role Separation 314
Using the crsctl setperm Command 315
Weight-Based Server Node Eviction 316
Administering SCAN 317
Starting and Stopping SCAN 317
Displaying the SCAN Status 317
Administering the Grid Naming Service (GNS) 320
Using the CLUVFY Utility for Managing Oracle RAC 320
Clusterware Startup 322
The Clusterware Startup Process 322
Clusterware Starting Sequence 323
Oracle Clusterware Auto-Startup 327
Oracle Clusterware Manual Startup 328
Using CRSCTL to Manage the Clusterware 328
Starting and Stopping CRS 328
Clusterized (Cluster-Aware) CRSCTL Commands 329
Verifying the Status of CRS 330
Disabling and Enabling CRS 331
The CRSCTL EVAL Commands 332
Using Other Utilities to Manage Oracle Clusterware 333
Using the olsnodes Command 333
The GPnP Tool 334
The Cluster Health Monitor 335
The OCLUMON Tool 336
Oracle Interface Configuration: oifcfg 337
Cluster Configuration Utility: clscfg 338
The Cluster Name Check Utility: cemutlo 339

https://t.me/learningnets
Technet24

Oracle Trace File Analyzer 339
Administering the OCR 343
Checking OCR Integrity 344
Dumping OCR Information 344
Managing the OCR with the OCRCONFIG Utility 345
Maintaining a Mirror OCR 345
Migrating the OCR to ASM 346
Administering the Oracle Local Registry 347
Administering the Voting Disks 348
Using ASM Storage for Voting Files 348
Backing Up Voting Disks 349
Restoring Voting Disks 349
Adding and Deleting Voting Disks 350
Migrating Voting Disks 350
Summary 350
9 Oracle RAC Backup and Recovery 352
Introduction to Backups 353
Oracle Backup Options 353
Oracle Backup Basics 354
Performing Backups in Oracle 355
Instance Recovery in Oracle RAC 358
Redo Threads and Streams 358
Redo Records and Change Vectors 359
Checkpoints 359
Crash Recovery 361
Steps in Crash Recovery (Single Instance) 361
Crash Recovery in Oracle RAC 362
Instance Recovery 362
Crash Recovery and Media Recovery 363
Bounded Recovery 363
Block-Written Record (BWR) 363
Past Image (PI) 364
Two-Pass Recovery 364
Cache Fusion Recovery 366
Dynamic Reconfiguration and Affinity Remastering 366
Fast Reconfiguration in Oracle RAC 368
Internals of Cache Fusion Recovery 370

https://t.me/learningnets

Backup and Recovery of the Voting Disk and OCR 374
Backup and Recovery of Voting Disks 374
Backup and Recovery of OCR 374
Validating OCR Backups 377
Summary 377
10 Oracle RAC Performance Management 378
Oracle RAC Design Considerations 379
Oracle Design Best Practices 379
Oracle RAC–Specific Design Best Practices 380
Partitioning the Workload 381
Scalability and Performance 382
Choosing the Block Size for an Oracle RAC Database 383
Introduction to the V$ and GV$ Views 384
Parallel Query Slaves 384
V$ Views Containing Cache Fusion Statistics 384
Oracle RAC Wait Events 384
Understanding Cluster Waits 386
Global Cache Statistics 401
Global Cache Statistics Summary 402
Global Cache Service Times 403
Global Cache Service Times Summary 406
Enqueue Tuning in Oracle RAC 406
Oracle AWR Report 408
Interpreting the AWR Report 410
ADDM 416
ASH Reports 417
Tuning the Cluster Interconnect 417
Verifying That Private Interconnect Is Used 417
Interconnect Latencies 418
Verifying That Network Interconnect Is Not Saturated 419
Summary 420
PART IV Advanced Oracle RAC Concepts 421
11 Global Resource Directory 422
Resources and Enqueues 423
Grants and Conversions 424
Locks and Enqueues 426
Cache Coherency 426

https://t.me/learningnets
Technet24

Global Enqueue Services 427
Latches and Enqueues 427
Global Locks Database and Structure 430
Messaging in Oracle RAC 433
Global Cache Services 437
Lock Modes and Lock Roles 438
Consistent Read Processing 442
GCS Resource Mastering 447
Read-Mostly Locking 449
Summary 455
12 A Closer Look at Cache Fusion 456
Key Components in Cache Fusion 457
Ping 458
Deferred Ping 458
Past Image (PI) Blocks 459
Lock Mastering 460
Types of Contention 460
Cache Fusion I, or Consistent Read Server 461
Cache Fusion II, or Write/Write Cache Fusion 465
Cache Fusion in Operation 468
Cache Fusion Walkthrough 477
Resource Mastering and Remastering 497
Background Processes and Cache Fusion 498
LMON: Lock Monitor Process 498
LMS: Lock Manager Server 498
LMD: Lock Manager Daemon Process (LMDn) 499
LCKn: Lock Process (LCK0) 499
DIAG: Diagnostic Daemon (DIAG) 500
Summary 500
13 Workload Management, Connection Management, and Application Continuity 501
Understanding Dynamic Database Services 502
Service Characteristics 504
Services and Policy-Managed Databases 509
Resource Management and Services 510
Using Services with Oracle Scheduler 510
Administering Services 510
Using Views to Get Service Information 514

https://t.me/learningnets

Distributed Transaction Processing 515
AQ_HA_Notifications 516
Workload Distribution and Load Balancing 517
Hardware and Software Load Balancing 520
Client-Side Load Balancing 520
Server-Side Load Balancing 525
Transparent Application Failover 530
TAF Considerations 530
Workload Balancing 536
Measuring Workloads by Service 540
Using Service-Level Thresholds 541
Oracle RAC High Availability Features 543
High Availability, Notifications, and FAN 543
Event-Based Notification 544
Application Failure Issues 545
Using Transaction Guard for Efficient Client Failover 551
Application Continuity for Masking Database Outages 553
Summary 555
14 Oracle RAC Troubleshooting 557
Installation Log Files 558
Log Directory Structure in the Oracle RDBMS 558
Log Directory Structure in Oracle Grid Infrastructure 560
Troubleshooting a Failed Oracle Grid Infrastructure Installation 562
Inside the Database Alert Log 564
RAC ON and OFF 569
Database Performance Issues 570
Hung Database 570
Hanganalyze Utility 572
Debugging Node Eviction Issues 573
Cluster Health Monitor 576
Instance Membership Recovery 577
Advanced Debugging for Oracle Clusterware Modules 585
Debugging Various Utilities in Oracle RAC 587
Using ORAchk to Troubleshoot RAC 589
Summary 589
PART V Deploying Oracle RAC 591
15 Extending Oracle RAC for Maximum Availability 592

https://t.me/learningnets
Technet24

Benefits of Extended RAC Clusters 594
Full Utilization of Resources 594
Extremely Rapid Recovery 594
Design Considerations 595
Speed of Light 595
Network Connectivity 595
Cache Fusion Performance 596
Data Storage 597
Common Techniques for Data Mirroring 598
Array-Based Mirroring 598
Host-Based Mirroring 599
ASM Preferred Read 601
Challenges in Extended Clusters 601
Extended Oracle RAC Limitations 603
Extended Oracle RAC vs. Oracle Data Guard 603
Summary 604
16 Developing Applications for Oracle RAC 605
Application Partitioning 608
Best Practice: Application Partitioning Schemes 609
Data Partitioning 610
Best Practice: Guidance Systems 611
Buffer Busy Waits/Block Contention 613
Index Partitioning 614
Buffer Busy Waits: Index Branch/Leaf Blocks Contention 614
Sorted Hash Clusters 617
Working with Sequences 618
CACHE and NOORDER 618
CACHE and ORDER 619
NOCACHE and ORDER 619
Best Practice: Use Different Sequences for Each Instance 619
Connection Management 620
Full Table Scans 621
Identifying Full Table Scans 621
Interconnect Protocol 623
Ethernet Frame Size 624
Library Cache Effect in the Parsing 624
Commit Frequency 625

https://t.me/learningnets

Summary 625
PART VI Appendixes 627
A Oracle RAC Reference 628
Global Cache Services and Cache Fusion Diagnostics 629
V$CACHE 629
V$CACHE_TRANSFER 631
V$INSTANCE_CACHE_TRANSFER 632
V$CR_BLOCK_SERVER 634
V$CURRENT_BLOCK_SERVER 636
V$GC_ELEMENT 637
Global Enqueue Services Diagnostics 638
V$LOCK 638
V$GES_BLOCKING_ENQUEUE 639
V$ENQUEUE_STATISTICS 641
V$LOCKED_OBJECT 660
V$GES_STATISTICS 660
V$GES_ENQUEUE 661
V$GES_CONVERT_LOCAL 663
V$GES_CONVERT_REMOTE 664
V$GES_RESOURCE 665
Dynamic Resource Remastering Diagnostics 665
V$HVMASTER_INFO 665
V$GCSHVMASTER_INFO 666
V$GCSPFMASTER_INFO 666
Cluster Interconnect Diagnostics 667
V$CLUSTER_INTERCONNECTS 667
V$CONFIGURED_INTERCONNECTS 668
B Adding and Removing Cluster Nodes 669
Adding a Node 669
Performing Pre-Installation Checks 670
Executing the addNode.sh Script 671
Installing the Oracle Database Software 672
Creating a Database Instance 672
Removing a Node 673
Deleting the Instance on the Node to Be Deleted 673
Removing the Node from the Database 674
Removing the RAC4 Node from the Clusterware 674

https://t.me/learningnets
Technet24

https://t.me/learningnets