



Integrated Cloud Applications & Platform Services

Oracle Database 12c R2: RAC Administration

Student Guide

D81250GC20

Edition 2.0 | March 2018 | D103167

Learn more from Oracle University at education.oracle.com



ORACLE®

Author

Sean Kim

**Technical Contributors
and Reviewers**

Sean Kim

Jerry Lee

Joel Goodman

Frank Fu

James Womack

Graphic Designers

Seema Bopaiah

Kavya Bellur

Publishers

Raghunath M

Veena Narasimhan

Joseph Fernandez

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Contents

1 Grid Infrastructure: Overview

Objectives 1-2
Cluster 1-3
Clusterware 1-4
Oracle Clusterware 1-5
Oracle Flex Clusters 1-6
Cluster Configuration Options 1-7
Clusterware Architecture and Cluster Services 1-10
Features of Oracle Clusterware 1-11
Oracle Clusterware Networking 1-12
Oracle Clusterware Initialization 1-14
GPnP Architecture: Overview 1-15
How GPnP Works: Cluster Node Startup 1-17
Grid Naming Service (GNS) 1-18
Single-Client Access Name 1-19
Client Database Connections 1-20
Oracle ASM 1-21
Oracle ACFS 1-22
Oracle Flex ASM 1-23
ASM Features and Benefits 1-24
Quiz 1-25
Practice 1: Overview 1-28
Summary 1-29

2 RAC Databases Overview and Architecture

Objectives 2-2
Oracle RAC: Overview 2-3
Typical Oracle RAC Architecture 2-4
Oracle RAC One Node 2-5
Oracle RAC One Node and Oracle Clusterware 2-6
Cluster-Aware Storage Solutions 2-7
Oracle RAC and Network Connectivity 2-8
Benefits of Using RAC 2-9
Clusters and Scalability 2-10
Levels of Scalability 2-11
Scaleup and Speedup 2-12

Speedup/Scaleup and Workloads	2-13
I/O Throughput Balanced: Example	2-14
Necessity of Global Resources	2-15
Additional Memory Requirement for RAC	2-16
Parallel Execution with RAC	2-17
Summary	2-18

3 Installing and Configuring Oracle RAC

Objectives	3-2
Lesson Agenda	3-3
Installing the Oracle Database Software	3-4
Creating the Cluster Database	3-9
Database Type Selection	3-10
Database Identification	3-11
Storage Options	3-12
Database Content	3-13
Configuration Options	3-14
Cluster Database Management Options	3-15
Passwords for Database Schema Owners	3-16
Create the Database	3-17
Monitoring Progress	3-18
Post-Installation Tasks	3-19
Background Processes Specific to Oracle RAC	3-20
Lesson Agenda	3-22
Read Only Instances on Leaf Nodes	3-23
Oracle RAC Reader Nodes	3-24
Running RAC Instances on Leaf Nodes	3-25
Lesson Agenda	3-26
Single Instance-to-RAC Conversion	3-27
Considerations for Converting Single-Instance Databases to Oracle RAC	3-28
Scenario 1: Using DBCA	3-29
Step 1: Create an Image of the Single-Instance Database	3-30
Example: Result of Step 1	3-31
Step 2: Create an Oracle Cluster for RAC	3-32
Example: Result of Step 2	3-33
Step 3: Copy the Preconfigured Database Image	3-34
Example: Database Structure File (*.dbc)	3-35
Example: Result of Step 3	3-36
Step 4: Create an Oracle RAC Database	3-37
Scenario 2: Using rconfig	3-38
Step 1: Check the Database Type	3-39

Step 2: Modify the XML File for the rconfig Utility	3-40
Example: ConvertToRAC_AdminManaged.xml	3-41
Step 3: Perform Prerequisite Checks	3-42
Step 4: Convert to an Oracle RAC Database	3-43
Step 5: Verify the Conversion	3-45
Example: Result of Using rconfig	3-46
Quiz	3-47
Summary	3-50
Practice 3: Overview	3-51

4 Oracle RAC Administration

Objectives	4-2
Separation of Duty for Administering Oracle Real Application Clusters	4-3
Enterprise Manager Cloud Control Cluster Database Home Page	4-4
Cluster Database Home Page	4-5
Cluster Database Instance Home Page	4-6
Cluster Home Page	4-7
Topology Viewer	4-8
Enterprise Manager Alerts and RAC	4-9
Enterprise Manager Metrics and RAC	4-10
Enterprise Manager Blackouts and RAC	4-11
Enterprise Manager Database Express	4-12
Redo Log Files and RAC	4-13
Automatic Undo Management and RAC	4-15
Local Temporary Tablespaces	4-16
Local Temporary Tablespace Organization	4-17
Temporary Tablespace Hierarchy	4-18
Local Temporary Tablespace Considerations	4-19
Managing Local Temporary Tablespaces	4-20
Local Temporary Tablespace Dictionary Views	4-21
Starting and Stopping RAC Instances	4-22
Starting and Stopping RAC Instances with srvctl	4-23
Starting and Stopping RAC Instances with SQL*Plus	4-24
Starting and Stopping Pluggable Databases in Oracle RAC	4-25
Switch Between Automatic and Manual Policies	4-27
RAC Initialization Parameter Files	4-28
SPFILE Parameter Values and RAC	4-29
EM and SPFILE Parameter Values	4-30
RAC Initialization Parameters	4-31
Parameters That Require Identical Settings	4-33
Parameters That Require Unique Settings	4-34

Quiescing RAC Databases	4-35
Terminating Sessions on a Specific Instance	4-36
How SQL*Plus Commands Affect Instances	4-37
Transparent Data Encryption and Keystores in RAC	4-38
Quiz	4-40
Summary	4-43
Practice 4: Overview	4-44

5 Upgrading and Patching Oracle RAC

Objectives	5-2
Patch and Patch Set: Overview	5-3
Types of Patches	5-4
Obtaining Oracle RAC Patch Sets	5-5
Obtaining Oracle RAC Patches	5-7
Downloading Patches	5-9
RAC Patching methods	5-10
Out-of-Place Upgrades with OUI	5-11
Rolling Patches	5-14
OPatch: Overview	5-15
OPatch: General Usage	5-16
Before Patching with OPatch	5-17
Installing a Rolling Patch with OPatch	5-18
OPatch Automation	5-20
OPatch Automation: Examples	5-21
OPatch Log and Trace Files	5-22
Queryable Patch Inventory	5-23
Alternative Methods of Patching	5-24
Quiz	5-25
Summary	5-26

6 Managing Backup and Recovery for RAC

Objectives	6-2
Instance Recovery and RAC	6-3
Instance Recovery and Database Availability	6-5
Instance Recovery and RAC	6-6
Protecting Against Media Failure	6-8
Media Recovery in Oracle RAC	6-9
Parallel Recovery in RAC	6-10
RAC and the Fast Recovery Area	6-11
RAC Backup and Recovery Using EM	6-12
Configuring RAC Recovery Settings with EM	6-13

Archived Redo File Conventions in RAC	6-14
Configuring RAC Backup Settings with EM	6-15
Oracle Recovery Manager	6-16
Configuring RMAN Snapshot Control File Location	6-17
Configuring Control File and SPFILE Autobackup	6-18
Crosschecking on Multiple RAC Clusters Nodes	6-19
Channel Connections to Cluster Instances	6-20
RMAN Channel Support for the Grid	6-21
RMAN Default Autolocation	6-22
Distribution of Backups	6-23
Managing Archived Redo Logs Using RMAN	6-24
Noncluster File System Local Archiving Scheme	6-25
Configuring Non-Cluster, Local Archiving	6-26
ASM and Cluster File System Archiving Scheme	6-27
Configuring the CFS Archiving Scheme	6-28
Restoring and Recovering	6-29
Quiz	6-30
Summary	6-32
Practice 6: Overview	6-33

7 Global Resource Management Concepts

Objectives	7-2
Need for Global Concurrency Control	7-3
Global Resource Directory (GRD)	7-4
Global Resource Management	7-5
Global Resource Remastering	7-6
Global Resource Recovery	7-7
Global Resource Background Processes	7-8
Global Resource Access Coordination	7-9
Global Enqueues	7-10
Instance Locks	7-11
Global Cache Management: Overview	7-12
Global Cache Management Components	7-13
Global Cache Buffer States	7-14
Global Cache Management Scenarios for Single Block Reads	7-15
Global Cache Scenarios: Overview	7-16
Scenario 1: Read from Disk	7-17
Scenario 2: Read-Write Cache Fusion	7-21
Scenario 3: Write-Write Cache Fusion	7-25
Scenario 4: Write-Read Cache Fusion	7-29
Global Cache Management Scenarios for Multi-Block Reads	7-33

Useful Global Resource Management Views 7-34
Quiz 7-35
Summary 7-36

8 RAC Database Monitoring and Tuning

Objectives 8-2
CPU and Wait Time Tuning Dimensions 8-3
RAC-Specific Tuning 8-4
Analyzing Cache Fusion Impact in RAC 8-5
Typical Latencies for RAC Operations 8-6
Wait Events for RAC 8-7
Wait Event Views 8-8
Global Cache Wait Events: Overview 8-9
Global Enqueue Waits 8-10
Session and System Statistics 8-11
Most Common RAC Tuning Tips 8-12
Index Block Contention: Considerations 8-14
Oracle Sequences and Index Contention 8-15
Undo Block Considerations 8-16
High-Water Mark Considerations 8-17
Concurrent Cross-Instance Calls: Considerations 8-18
Monitoring RAC Database and Cluster Performance 8-19
Cluster Database Performance Page 8-20
Determining Cluster Host Load Average 8-21
Determining Global Cache Block Access Latency 8-22
Determining Average Active Sessions 8-23
Determining Database Throughput 8-24
Accessing the Cluster Cache Coherency Page 8-26
Viewing the Database Locks Page 8-28
AWR Snapshots in RAC 8-29
AWR Reports and RAC: Overview 8-30
Active Session History Reports for RAC 8-31
Automatic Database Diagnostic Monitor for RAC 8-32
What Does ADDM Diagnose for RAC? 8-34
EM Support for ADDM for RAC 8-35
EM Database Express Performance Hub 8-36
Monitoring RAC With Cluster Health Advisor (CHA) 8-37
Quiz 8-38
Summary 8-40
Practice 8: Overview 8-41

9 Managing High Availability of Services
Objectives 9-2
Oracle Services 9-3
Service Usage in an Oracle RAC Database 9-4
Parallel Operations and Services 9-5
Service-Oriented Buffer Cache Access 9-6
Service Characteristics 9-7
Default Service Connections 9-9
Restricted Service Registration 9-10
Creating Service with Enterprise Manager 9-11
Creating Services with SRVCTL 9-12
Managing Services with Enterprise Manager 9-13
Managing Services with EM 9-14
Managing Services with srvctl 9-15
Using Services with Client Applications 9-16
Services and Connection Load Balancing 9-17
Services and Transparent Application Failover 9-18
Using Services with the Resource Manager 9-19
Services and Resource Manager with EM 9-20
Using Services with the Scheduler 9-21
Services and the Scheduler with EM 9-22
Using Distributed Transactions with RAC 9-24
Distributed Transactions and Services 9-25
Service Thresholds and Alerts 9-27
Services and Thresholds Alerts: Example 9-28
Service Aggregation and Tracing 9-29
Top Services Performance Page 9-30
Service Aggregation Configuration 9-31
Service, Module, and Action Monitoring 9-32
Service Performance Views 9-34
Quiz 9-35
Summary 9-37
Practice 9: Overview 9-38

10 High Availability for Connections and Applications

Objectives 10-2
Types of Workload Distribution 10-3
Client-Side Connect-Time Load Balancing 10-4
Fast Application Notification (FAN): Overview 10-5
Fast Application Notification: Benefits 10-6
Implementing FAN Events 10-7

FAN and Oracle Integrated Clients	10-8
FAN-Supported Event Types	10-10
FAN Event Reasons	10-11
FAN Event Status	10-12
FAN Event Format	10-13
Load Balancing Advisory: FAN Event	10-14
Server-Side Callouts Implementation	10-15
Server-Side Callout Parse: Example	10-16
Server-Side Callout Filter: Example	10-17
Server-Side ONS	10-18
Optionally Configuring the Client-Side ONS	10-19
UCP JDBC Fast Connection Failover: Overview	10-20
JDBC/ODP.NET FCF Benefits	10-21
Load Balancing Advisory	10-22
UCP JDBC/ODP.NET Runtime Connection Load Balancing: Overview	10-23
Connection Load Balancing in RAC	10-24
Monitoring LBA FAN Events	10-25
Transparent Application Failover: Overview	10-26
TAF Basic Configuration on Server-Side: Example	10-27
TAF Basic Configuration on a Client-Side: Example	10-28
TAF Preconnect Configuration: Example	10-29
TAF Verification	10-30
FAN Connection Pools and TAF Considerations	10-31
Introducing Transaction Guard and Application Continuity	10-32
What Is Transaction Guard?	10-33
Benefits of Transaction Guard	10-34
How Transaction Guard Works	10-35
Using Transaction Guard	10-36
Creating Services for Transaction Guard	10-38
What Is Application Continuity?	10-39
Benefits of Application Continuity	10-40
How Does Application Continuity Work?	10-41
RAC and Application Continuity	10-42
Using Application Continuity	10-43
Creating Services for Application Continuity	10-45
Quiz	10-47
Summary	10-49
Practice 10 Overview: Using Application Continuity	10-50

11 Oracle RAC One Node

Objectives 11-2

Oracle RAC One Node	11-3
Creating an Oracle RAC One Node Database	11-4
Verifying an Existing RAC One Node Database	11-5
Oracle RAC One Node Online Relocation	11-6
Online Relocation Considerations	11-7
Performing an Online Relocation	11-8
Online Relocation Illustration	11-9
Online Maintenance: Rolling Patches	11-12
Adding an Oracle RAC One Node Database to an Existing Cluster	11-14
Converting a RAC One Node Database to RAC	11-15
Converting a Single Instance Database to RAC One Node	11-17
Converting a RAC Database to RAC One Node	11-18
Quiz	11-19
Summary	11-21
Practice 11: Overview	11-22

12 Oracle Database In-Memory in RAC

Objectives	12-2
In-Memory Column Store	12-3
Advantages of In-Memory Column Store	12-5
In-Memory Column Store Pools	12-6
Implementing In-Memory Column Store	12-7
In-Memory Column Store Population	12-8
Prioritization of In-Memory Population	12-9
In-Memory Column Store and Oracle RAC	12-10
In-Memory FastStart	12-11
In-Memory FastStart Architecture	12-12
Enabling In-Memory FastStart	12-13
FastStart Area in Oracle RAC	12-14
How the Database Reads from the FastStart Area	12-15
Summary	12-16
Practice 12: Overview	12-17

13 Multitenant Architecture and RAC

Objectives	13-2
Non-CDB Architecture	13-3
Multitenant Architecture: Benefits	13-4
CDB in a Non-RAC Environment	13-5
Containers	13-6
Terminology	13-7
Data Dictionary Views	13-8

Connection to a Non-RAC CDB	13-9
Switching Connection	13-10
Oracle RAC and Multitenant Configuration	13-11
Oracle RAC and Multitenant Architecture	13-12
Creating a RAC CDB	13-13
Hosting a RAC CDB in Server Pools	13-14
Creating a RAC CDB Including PDBs	13-15
After CDB Creation	13-16
Connecting Using CDB/PDB Services	13-17
Opening a PDB in a RAC CDB	13-18
Closing a PDB in a RAC CDB	13-19
Types of Services	13-20
Managing Services	13-21
Affinitizing PDB Services to Server Pools	13-22
Adding a PDB to a RAC CDB	13-23
Dropping a PDB from a RAC CDB	13-24
Quiz	13-25
Summary	13-26
Practice 13: Overview	13-27

14 Quality of Service Management

Objectives	14-2
QoS Management Background	14-3
QoS Management Overview	14-4
QoS Management and Exadata Database Machine	14-5
QoS Management Focus	14-6
QoS Management Benefits	14-7
QoS Management Functional Overview	14-8
QoS Management Policy Sets	14-10
Server Pools	14-11
Performance Classes	14-13
Classification and Tagging	14-15
Performance Policies	14-16
Performance Class Ranks	14-17
Performance Objectives	14-18
Performance Satisfaction Metrics	14-19
Server Pool Directive Overrides	14-20
Overview of Metrics	14-21
QoS Management Architecture	14-22
QoS Management Recommendations	14-23
Implementing Recommendations	14-24

QoS Support For Admin-Managed RAC Databases	14-25
Quiz	14-26
Summary	14-28

15 Oracle Database Exadata Cloud Service Overview

Objectives	15-2
Introducing Exadata Cloud Service	15-3
Service Configuration Options	15-5
Service Connection Options	15-7
Service Architecture	15-8
Service Availability	15-9
Management Responsibilities	15-10
Storage Configuration	15-11
Storage Management Details	15-13
Simple Web-Based Provisioning	15-14
Simple Web-Based Management	15-15
REST APIs	15-16
Migrating to Exadata Cloud Service	15-17
Summary	15-18

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Grid Infrastructure: Overview



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Explain the principles and purposes of clusters
- Describe the Oracle Clusterware architecture
- Describe how Grid Plug and Play affects Clusterware

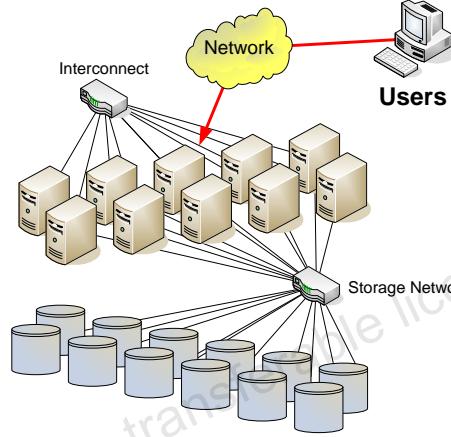


ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Cluster

- A group of independent, but interconnected, computers that act as a single system
- Usually deployed to increase availability and performance or to balance a dynamically changing workload



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

A cluster consists of a group of independent but interconnected computers whose combined resources can be applied to a processing task. A common cluster feature is that it should appear to an application as though it were a single server. Most cluster architectures use a dedicated network (interconnect) for communication and coordination between cluster nodes.

A common cluster architecture for data-intensive transactions and computations is built around shared disk storage. Shared-nothing clusters use an alternative architecture where storage is not shared and data must be either replicated or segmented across the cluster. Shared-nothing clusters are commonly used for workloads that can be easily and predictably divided into small units that can be spread across the cluster in parallel. Shared disk clusters can perform these tasks but also offer increased flexibility for varying workloads. Load balancing clusters allow a single application to balance its workload across the cluster. Alternatively, in a failover cluster, some nodes can be designated as the primary host for an application, whereas others act as the primary host for different applications. In a failover cluster, the failure of a node requires that the applications it supports be moved to a surviving node. Load balancing clusters can provide failover capabilities but they can also run a single application across multiple nodes providing greater flexibility for different workload requirements. Oracle supports a shared disk cluster architecture providing load balancing and failover capabilities. In an Oracle cluster, all nodes must share the same processor architecture and run the same operating system. With the release of Oracle Database 12c, Flex ASM allows nodes in the cluster access to shared storage indirectly through an ASM instance on another node in the cluster.

Clusterware

- Clusterware is a software that provides various interfaces and services for a cluster.
- Typically, this includes capabilities that:
 - Allow the cluster to be managed as a whole
 - Protect the integrity of the cluster
 - Maintain a registry of resources across the cluster
 - Deal with changes to the cluster
 - Provide a common view of resources



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Clusterware is a term used to describe the software that provides interfaces and services that enable and support a cluster.

Different cluster architectures require clusterware that delivers different services. For example, in a simple failover cluster, the clusterware may monitor the availability of applications and perform a failover operation if a cluster node becomes unavailable. In a load balancing cluster, different services are required to support workload concurrency and coordination.

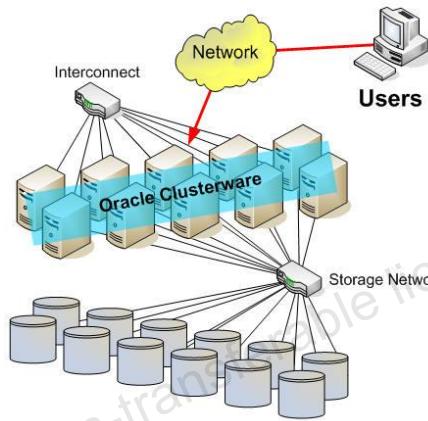
Typically, clusterware includes capabilities that:

- Allow the cluster to be managed as a single entity (not including OS requirements), if desired
- Protect the integrity of the cluster so that data is protected and the cluster continues to function even if communication with a cluster node is severed
- Maintain a registry of resources so that their location is known across the cluster and so that dependencies between resources is maintained
- Deal with changes to the cluster such as node additions, removals, or failures
- Provide a common view of resources such as network addresses and files in a file system

Oracle Clusterware

Oracle Clusterware is:

- A key part of Oracle Grid Infrastructure
- Integrated with Oracle Automatic Storage Management (ASM)
- The basis for Oracle ASM Cluster File System (ACFS)
- A foundation for Oracle Real Application Clusters (RAC)
- A generalized cluster infrastructure for all kinds of applications



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

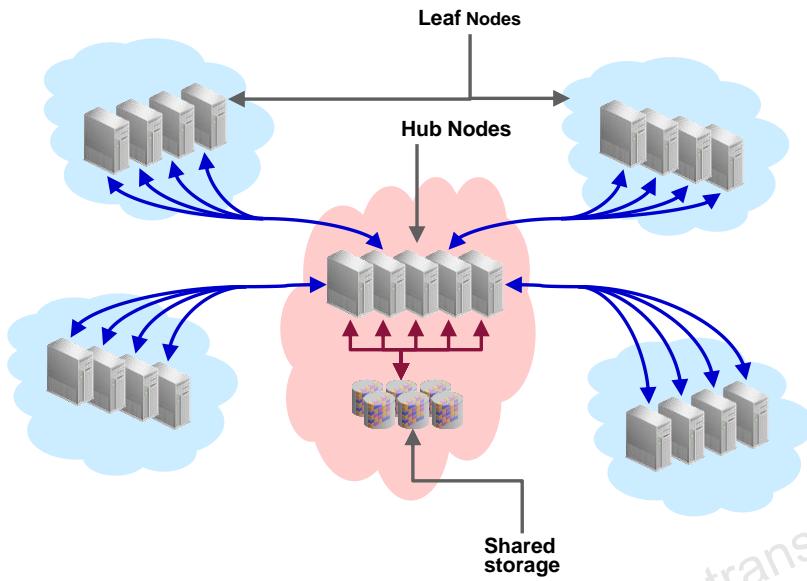
Oracle Clusterware is a key part of Oracle Grid Infrastructure, which also includes Automatic Storage Management (ASM) and the Oracle ASM Cluster File System. Oracle Clusterware can use ASM for all the shared files required by the cluster. Oracle Clusterware is also a foundation for the ASM Cluster File System, a generalized cluster file system that can be used for most file-based data such as documents, spreadsheets, and reports.

The combination of Oracle Clusterware, ASM, and ACFS provides administrators with a unified cluster solution that is not only the foundation for the RAC database, but can also be applied to all kinds of other applications. Oracle Clusterware also manages resources, such as virtual IP (VIP) addresses, databases, listeners, services, and so on.

Using Oracle Clusterware eliminates the need for proprietary vendor clusterware and provides the benefit of using only Oracle software. Oracle provides an entire software solution, including everything from disk management with Oracle Automatic Storage Management (Oracle ASM) to data management with Oracle Database and Oracle RAC. In addition, Oracle Database features, such as Oracle Services, provide advanced functionality when used with the underlying Oracle Clusterware high availability framework.

With the introduction of Oracle 12c Flex Clusters, pure shared disk clusters are not the only type of clustered hardware supported. The architecture has become hybrid with the introduction of hub and leaf nodes.

Oracle Flex Clusters



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Flex Clusters use a hub-and-spoke topology, as illustrated in the slide.

All nodes in an Oracle Flex Cluster belong to a single Oracle Grid Infrastructure cluster. This architecture centralizes policy decisions for deployment of resources based on application needs, to account for various service levels, loads, failure responses, and recovery.

The core of a Flex Cluster is a group of Hub Nodes. The group is essentially the same as a release 11.2 cluster, and can scale up to the size of an existing release 11.2 cluster. There must be one, and only one, group of Hub Nodes in a Flex Cluster deployment, and like a release 11.2 cluster, each Hub Node must be connected to storage that is shared across the group of Hub Nodes.

Zero or more Leaf Nodes may be connected to a Flex Cluster. Each Leaf Node is connected to the cluster through a Hub Node. Leaf Nodes do not require direct access to the shared storage connected to the Hub Nodes.

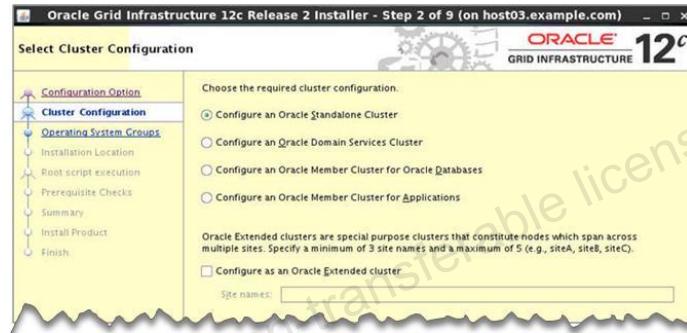
Starting with Oracle Grid Infrastructure 12c Release 2 (12.2), Oracle Grid Infrastructure cluster configurations are Oracle Flex Clusters deployments.

Oracle Grid Infrastructure installed in an Oracle Flex Cluster configuration is a scalable, dynamic, robust network of nodes. Oracle Flex Clusters provide a platform for Oracle Real Application Clusters databases with large numbers of nodes, to support massive parallel query operations. Oracle Flex Clusters also provide a platform for other service deployments that require coordination and automation for high availability.

Cluster Configuration Options

The cluster configuration options that are available in Oracle Grid Infrastructure 12c, Release 2 include:

- Oracle Standalone Clusters
- Oracle Domain Services Cluster
- Oracle Member Cluster for Oracle Databases
- Oracle Member Cluster for Applications
- Oracle Extended Clusters



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle Standalone Cluster

An Oracle Standalone (Flex) Cluster hosts all Oracle Grid Infrastructure services and Oracle ASM locally, and requires direct access to shared storage.

Oracle Standalone Clusters contain two types of nodes arranged in a hub-and-spoke architecture: Hub Nodes and Leaf Nodes.

- The number of Hub Nodes in an Oracle Standalone Cluster can be as many as 64. The number of Leaf Nodes can be many more.
- The Hub Nodes and Leaf Nodes can host different types of applications.
- The Oracle Standalone Cluster Hub Nodes are tightly connected, and have direct access to shared storage. The Leaf Nodes do not require direct access to shared storage.
- The Hub Nodes can run in an Oracle Standalone Cluster configuration without having any Leaf Nodes as cluster member nodes, but Leaf Nodes must be members of a cluster with a pool of Hub Nodes.
- Shared storage is locally mounted on each of the Hub Nodes, with an Oracle ASM instance that is available to all Hub Nodes.

Note: This class focuses on the RAC databases in the Oracle Standalone Cluster.

Oracle Cluster Domain

An Oracle Cluster Domain is a choice of deployment architecture for new clusters, introduced in Oracle Clusterware 12c, Release 2.

Oracle Cluster Domain enables you to standardize, centralize, and optimize your Oracle Real Application Clusters (Oracle RAC) deployment for the private database cloud. Multiple cluster configurations are grouped under an Oracle Cluster Domain for management purposes, and use the shared services that are available within that Oracle Cluster Domain. The cluster configurations within that Oracle Cluster Domain include Oracle Domain Services Cluster and Oracle Member Clusters.

Oracle Domain Services Cluster

The Oracle Domain Services Cluster provides centralized services to other clusters within the Oracle Cluster Domain. These services include:

- A centralized Grid Infrastructure Management Repository (housing the MGMTDB for each of the clusters within the Oracle Cluster Domain)
- Trace File Analyzer (TFA) services, for targeted diagnostic data collection for Oracle Clusterware and Oracle Database
- Consolidated Oracle ASM storage management service
- An optional Rapid Home Provisioning (RHP) service to install clusters, and provision, patch, and upgrade Oracle Grid Infrastructure and Oracle Database homes. When you configure the Oracle Domain Services Cluster, you can also choose to configure the Rapid Home Provisioning Server.

An Oracle Domain Services Cluster provides these centralized services to Oracle Member Clusters. Oracle Member Clusters use these services:

- For centralized and simplified management
- To reduce their local resource usage and dependence

Oracle Member Cluster for Oracle Databases

An Oracle Member Cluster for Oracle Databases supports Oracle Real Application Clusters (Oracle RAC) or Oracle RAC One Node database instances. This cluster registers with the management repository service and uses the centralized TFA service. It can use additional services as needed. An Oracle Member Cluster for Oracle Databases can be configured with local Oracle ASM storage management or use the consolidated Oracle ASM storage management service offered by the Oracle Domain Services Cluster.

An Oracle Member Cluster for Oracle Databases always uses the remote Grid Infrastructure Management Repository (GIMR) from its Oracle Domain Services Cluster. For two-node or four-node clusters, hosting the GIMR on a remote cluster reduces the overhead of running an extra infrastructure repository on a cluster.

Oracle Member Cluster for Applications

Oracle Member Cluster for Applications hosts applications other than Oracle Database, as part of an Oracle Cluster Domain. An Oracle Member Cluster requires connectivity to Oracle Cluster Domain Services for centralized management and resource efficiency. The Oracle Member Cluster uses remote Oracle ASM storage and does not require direct shared storage access. This cluster configuration enables high availability of any software application.

Oracle Extended Cluster

An Oracle Extended Cluster consists of nodes that are located in multiple locations called sites. When you deploy an Oracle Standalone Cluster, you can also choose to configure the cluster as an Oracle Extended Cluster. You can extend an Oracle RAC cluster across two, or more, geographically separate sites, each equipped with its own storage. In the event that one of the sites fails, the other site acts as an active standby.

Both Oracle ASM and the Oracle Database stack, in general, are designed to use enterprise-class shared storage in a data center. Fibre Channel technology, however, enables you to distribute compute and storage resources across two or more data centers, and connect them through Ethernet cables and Fibre Channel, for compute and storage needs, respectively.

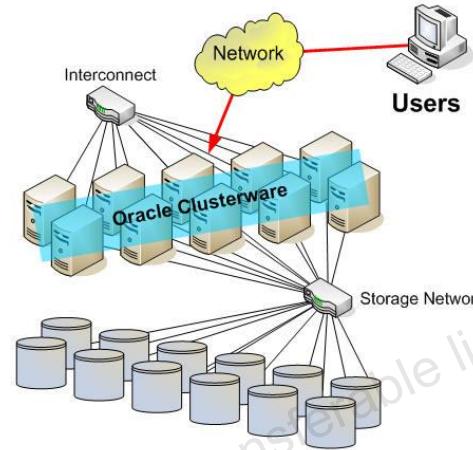
You can configure an Oracle Extended Cluster when you install Oracle Grid Infrastructure. You can also do so post-installation by using the `ConvertToExtended` script. You manage your Oracle Extended Cluster by using `crsctl`.

Oracle recommends that you deploy Oracle Extended Clusters with normal redundancy disk groups. You can assign nodes and failure groups to sites. Sites contain failure groups, and failure groups contain disks. For normal redundancy disk groups, a disk group provides one level of failure protection, and can tolerate the failure of either a site or a failure group.

For more information about the cluster configuration options available in Oracle Grid Infrastructure 12c Release 2, refer to *Oracle Grid Infrastructure Installation and Upgrade Guide*

Clusterware Architecture and Cluster Services

- Shared disk cluster architecture supporting application load balancing and failover
- Services include:
 - Cluster management
 - Node monitoring
 - Event services
 - Time synchronization
 - Network management
 - High availability
 - Cluster Interconnect Link Aggregation (HAIP)



ORACLE®

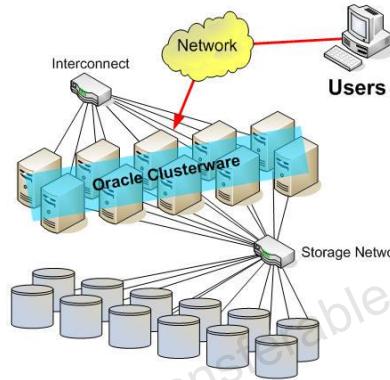
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle Clusterware provides a complete set of cluster services to support the shared disk, load balancing cluster architecture of the Oracle Real Application Cluster (RAC) database. Oracle Clusterware can also be used to provide failover clustering services for single-instance Oracle databases and other applications. The services provided by Oracle Clusterware include:

- Cluster management, which allows cluster services and application resources to be monitored and managed from any node in the cluster
- Node monitoring, which provides real-time information regarding which nodes are currently available and the resources they support. Cluster integrity is also protected by evicting or fencing unresponsive nodes.
- Event services, which publish cluster events so that applications are aware of changes in the cluster
- Time synchronization, which synchronizes the time on all nodes of the cluster
- Network management, which provisions and manages Virtual IP (VIP) addresses that are associated with cluster nodes or application resources to provide a consistent network identity regardless of which nodes are available. In addition, Grid Naming Service (GNS) manages network naming within the cluster.
- High availability, which services, monitors, and restarts all other resources as required
- Cluster Interconnect Link Aggregation (HAIP)

Features of Oracle Clusterware

- Easy installation
- Easy management
- Continuing tight integration with Oracle RAC
- ASM enhancements with benefits for all applications
- No additional clusterware required



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

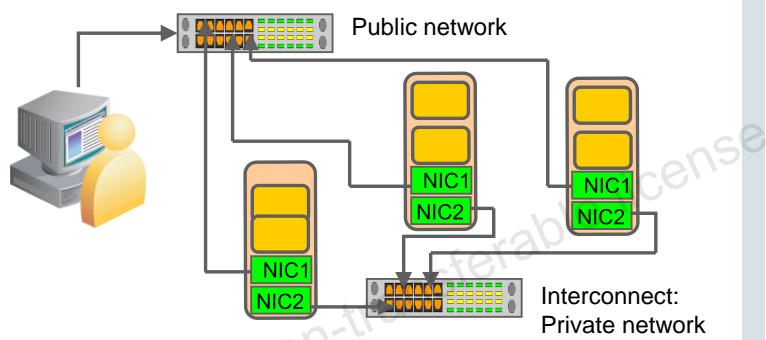
Oracle Clusterware has become the required clusterware for Oracle Real Application Clusters (RAC). Oracle Database 12c builds on the tight integration between Oracle Clusterware and RAC by extending the integration with Automatic Storage Management (ASM). The result is that now all the shared data in your cluster can be managed by using ASM. This includes the shared data required to run Oracle Clusterware, Oracle RAC, and any other applications you choose to deploy in your cluster.

In most cases, this capability removes the need to deploy additional clusterware from other sources, which also removes the potential for integration issues caused by running multiple clusterware software stacks. It also improves the overall manageability of the cluster.

Although most of the enhancements to ASM are the subject of later lessons, the next part of this lesson examines a series of additional Oracle Clusterware capabilities and the benefits they provide.

Oracle Clusterware Networking

- Each node must have at least two network adapters.
- Each public network adapter must support TCP/IP.
- The interconnect adapter must support:
 - User Datagram Protocol (UDP) or Reliable Data Socket (RDS) for UNIX and Linux for database communication
 - TCP for Windows platforms for database communication
- All platforms use Grid Interprocess Communication (GIPC).



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Each node must have at least two network adapters: one for the public network interface and the other for the private network interface or interconnect. In addition, the interface names associated with the network adapters for each network must be the same on all nodes. For example, in a two-node cluster, you cannot configure network adapters on node1 with eth0 as the public interface, but on node2 have eth1 as the public interface. Public interface names must be the same, so you must configure eth0 as public on both nodes. You should configure the private interfaces on the same network adapters as well. If eth1 is the private interface for node1, eth1 should be the private interface for node2.

Before starting the installation, on each node, you must have at least two interfaces to configure for the public and private IP addresses. You can configure IP addresses with one of the following options:

- Oracle Grid Naming Service (GNS) using one static address defined during installation, which dynamically allocates VIP addresses using Dynamic Host Configuration Protocol (DHCP), which must be running on the network. You must select the Advanced Oracle Clusterware installation option to use GNS.
- Static addresses that network administrators assign on a network domain name server (DNS) or each node. To use the Typical Oracle Clusterware installation option, you must use static addresses.

For the public network, each network adapter must support TCP/IP.

For the private network, the interconnect must support UDP or RDS (TCP for Windows) for communications to the database. Grid Interprocess Communication (GIPC) is used for Grid (Clusterware) interprocess communication. GIPC is a common communications infrastructure to replace CLSC/NS. It provides full control of the communications stack from the operating system up to whatever client library uses it. The dependency on network services (NS) before 11.2 is removed, but there is still backward compatibility with existing CLSC clients (primarily from 11.1). GIPC can support multiple communications types: CLSC, TCP, UDP, IPC, and of course, the communication type GIPC.

Use high-speed network adapters for the interconnects and switches that support TCP/IP. Gigabit Ethernet or an equivalent is recommended.

If you have multiple available network interfaces, Oracle recommends that you use the Redundant Interconnect Usage feature to make use of multiple interfaces for the private network. However, you can also use third-party technologies to provide redundancy for the private network.

Note: Cross-over cables are not supported for use with Oracle Clusterware interconnects.

Oracle Clusterware Initialization

- Oracle Clusterware is started by the OS init daemon calling the /etc/init.d/init.ohasd startup script.
- On OL6, Clusterware startup is controlled by Upstart via the /etc/init/oracle-ohasd.conf file.

```
# cat /etc/init/oracle-ohasd.conf
# Oracle OHASD startup

start on runlevel [35]
stop on runlevel [!35]
respawn
exec /etc/init.d/init.ohasd run >/dev/null 2>&1 </dev/null
```

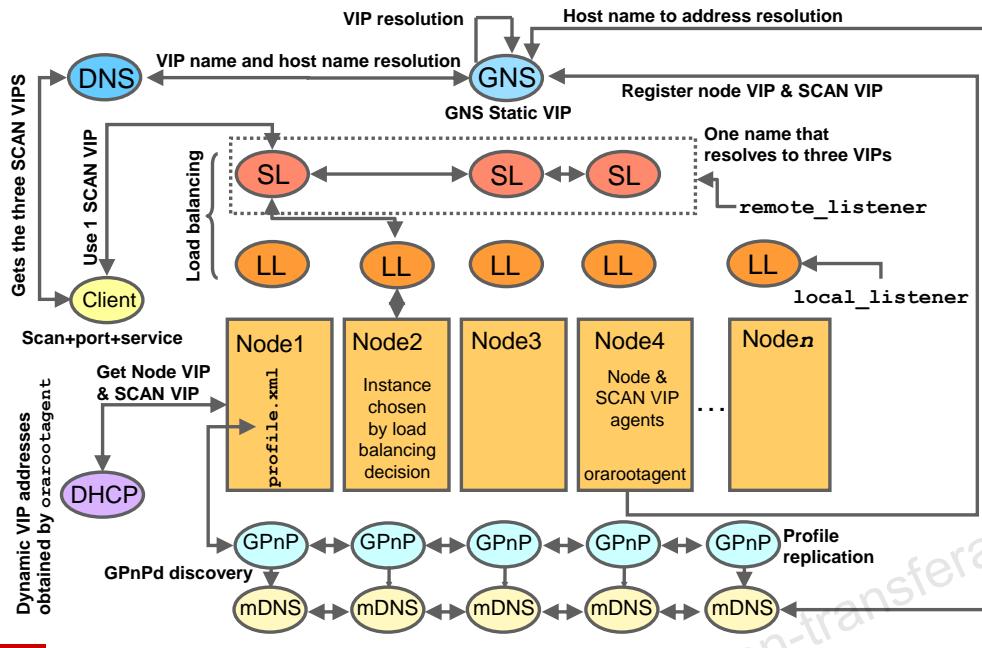
- On OL7, Clusterware startup is controlled by systemd to manage start/stop services (example: /etc/systemd/system/oracle-ohasd.service)



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle Linux 6 (OL6) or Red Hat Linux 6 (RHEL6) has deprecated inittab, rather, init.ohasd will be configured via upstart in /etc/init/oracle-ohasd.conf, however, the process /etc/init.d/init.ohasd run should still be up. Oracle Linux 7 (and Red Hat Linux 7) uses systemd to manage start/stop services (example: /etc/systemd/system/oracle-ohasd.service)

GPnP Architecture: Overview



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

GPnP Service

The GPnP service is collectively provided by all the GPnP agents. It is a distributed method of replicating profiles. The service is instantiated on each node in the domain as a GPnP agent. The service is peer-to-peer; there is no master process. This allows high availability because any GPnP agent can crash and new nodes will still be serviced. GPnP requires standard IP multicast protocol (provided by mDNS), to locate peer services. Using multicast discovery, GPnP locates peers without configuration. This is how a GPnP agent on a new node locates another agent that may have a profile it should use.

Name Resolution

A name defined within a GPnP domain is resolvable in the following cases:

- Hosts inside the GPnP domain use normal DNS to resolve the names of hosts outside of the GPnP domain. They contact the regular DNS service and proceed. They may get the address of the DNS server by global configuration or by having been told by DHCP.
- Within the GPnP domain, host names are resolved by using mDNS. This requires an mDNS responder on each node that knows the names and addresses used by this node, and operating system client library support for name resolution using this multicast protocol. Given a name, a client executes `gethostbyname`, resulting in an mDNS query. If the name exists, the responder on the node that owns the name will respond with the IP address.

The client software may cache the resolution for the given time-to-live value.

- Machines outside the GPnP domain cannot resolve names in the GPnP domain by using multicast. To resolve these names, they use their regular DNS. The provisioning authority arranges the global DNS to delegate a subdomain (zone) to a known address that is in the GPnP domain. GPnP creates a service called GNS to resolve the GPnP names on that fixed address.

The node on which the GNS server is running listens for DNS requests. On receipt, they translate and forward to mDNS, collect responses, translate, and send back to the outside client. GNS is “virtual” because it is stateless. Any node in the multicast domain may host the server. The only GNS configuration is global:

- The address on which to listen on standard DNS port 53
- The names of the domains to serviced

There may be as many GNS entities as needed for availability reasons. Oracle-provided GNS may use CRS to ensure availability of a single GNS provider.

SCAN and Local Listeners

When a client submits a connection request, the SCAN listener listening on a SCAN IP address and the SCAN port are contacted on the client’s behalf. Because all services on the cluster are registered with the SCAN listener, the SCAN listener replies with the address of the local listener on the least-loaded node where the service is currently being offered. Finally, the client establishes a connection to the service through the listener on the node where service is offered. All these actions take place transparently to the client without any explicit configuration required in the client.

During installation, listeners are created on nodes for the SCAN IP addresses. Oracle Net Services routes application requests to the least loaded instance providing the service. Because the SCAN addresses resolve to the cluster, rather than to a node address in the cluster, nodes can be added to or removed from the cluster without affecting the SCAN address configuration.

Static Configuration

With static configurations, no subdomain is delegated. A DNS administrator configures the GNS VIP to resolve to a name and address configured on the DNS, and a DNS administrator configures a SCAN name to resolve to three static addresses for the cluster. A DNS administrator also configures a static public IP name and address, and virtual IP name and address for each cluster member node. A DNS administrator must also configure new public and virtual IP names and addresses for each node added to the cluster. All names and addresses are resolved by DNS.

How GPnP Works: Cluster Node Startup

1. IP addresses are negotiated for public interfaces using DHCP:
 - Node VIPs
 - SCAN VIPs
2. A GPnP agent is started from the nodes Clusterware home.
3. The GPnP agent either gets its profile locally or from one of the peer GPnP agents that responds.
4. Shared storage is configured to match profile requirements.
5. Service startup is specified in the profile, which includes:
 - Grid Naming Service for external names resolution
 - Single-client access name (SCAN) listener



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When a node is started in a GPnP environment:

- Network addresses are negotiated for all interfaces using DHCP
- The Clusterware software on the starting node starts a GPnP agent
- The GPnP agent on the starting node gets its profile locally or uses resource discovery (RD) to discover the peer GPnP agents in the grid. If RD is used, it gets the profile from one of the GPnP peers that responds.

The GPnP agent acquires the desired network configuration from the profile. This includes creation of reasonable host names. If there are static configurations, they are used in preference to the dynamic mechanisms. Network interfaces may be reconfigured to match the profile requirements.

- Shared storage is configured to match the profile requirements.
- System and service startup is done as configured in the image. In the case of RAC, the CSS and CRS systems will then be started, which will form the cluster and bring up appropriate database instances. The startup of services may run down their own placeholder values, or may dynamically negotiate values rather than rely on fixed-up configurations. One of the services likely to be started somewhere is the GNS system for external name resolution. Another of the services likely to be started is an Oracle SCAN listener.

Grid Naming Service (GNS)

- The only static IP address required for the cluster is the GNS virtual IP address.
- The cluster subdomain is defined as a delegated domain.

```
[root@my-dns-server ~]# cat /etc/named.conf
// Default initial "Caching Only" name server configuration
...
# Delegate to gns on cluster01
cluster01.example.com #cluster sub-domain# NS cluster01-gns.example.com
# Let the world know to go to the GNS vip
cluster01-gns.example.com 192.0.2.155 #cluster GNS Address
```

- A request to resolve cluster01-scan.cluster01.example.com would be forwarded to the GNS on 192.0.2.155.
- Each cluster node runs a multicast DNS (mDNS) process.
- You cannot use GNS with another multicast DNS.
 - If you want to use GNS, disable any third-party mDNS daemons on your system.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Employing Grid Naming Service (GNS) assumes that there is a DHCP server running on the public network with enough addresses to assign to the VIPs and single-client access name (SCAN) VIPs. With GNS, only one static IP address is required for the cluster, the GNS virtual IP address. This address should be defined in the DNS domain. GNS sets up a multicast DNS (mDNS) server within the cluster, which resolves names in the cluster without static configuration of the DNS server for other node IP addresses.

The mDNS server works as follows: Within GNS, node names are resolved by using link-local multicast name resolution (LLMNR). It does this by translating the LLMNR “.local” domain used by the multicast resolution to the subdomain specified in the DNS query. When you select GNS, an mDNS server is configured on each host in the cluster. LLMNR relies on the mDNS that Oracle Clusterware manages to resolve names that are being served by that host.

To use GNS, before installation, the DNS administrator must establish domain delegation to the subdomain for the cluster. Queries to the cluster are sent to the GNS listener on the GNS virtual IP address. When a request comes to the domain, GNS resolves it by using its internal mDNS and responds to the query.

Note: You cannot use GNS with another multicast DNS. If you want to use GNS, disable any third-party mDNS daemons on your system.

Single-Client Access Name

- The single-client access name (SCAN) is the address used by clients connecting to the cluster.
- The SCAN is a fully qualified host name located in the GNS subdomain registered to three IP addresses.

```
$ nslookup cluster01-scan.cluster01.example.com
Server:      192.0.2.1
Address:     192.0.2.1#53

Non-authoritative answer:
Name:   cluster01-scan.cluster01.example.com
Address: 192.0.2.243
Name:   cluster01-scan.cluster01.example.com
Address: 192.0.2.244
Name:   cluster01-scan.cluster01.example.com
Address: 192.0.2.245
```

- The SCAN provides a stable, highly available name for clients to use, independent of the nodes that make up the cluster.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The single-client access name (SCAN) is the address used by clients connecting to the cluster. The SCAN is a fully qualified host name (host name + domain) registered to three IP addresses. If you use GNS, and you have DHCP support, then the GNS will assign addresses dynamically to the SCAN.

If you do not use GNS, the SCAN should be defined in the DNS to resolve to the three addresses assigned to that name. This should be done before you install Oracle Grid Infrastructure. The SCAN and its associated IP addresses provide a stable name for clients to use for connections, independent of the nodes that make up the cluster.

SCANS function like a cluster alias. However, SCANS are resolved on any node in the cluster, so unlike a VIP address for a node, clients connecting to the SCAN no longer require updated VIP addresses as nodes are added to or removed from the cluster. Because the SCAN addresses resolve to the cluster, rather than to a node address in the cluster, nodes can be added to or removed from the cluster without affecting the SCAN address configuration.

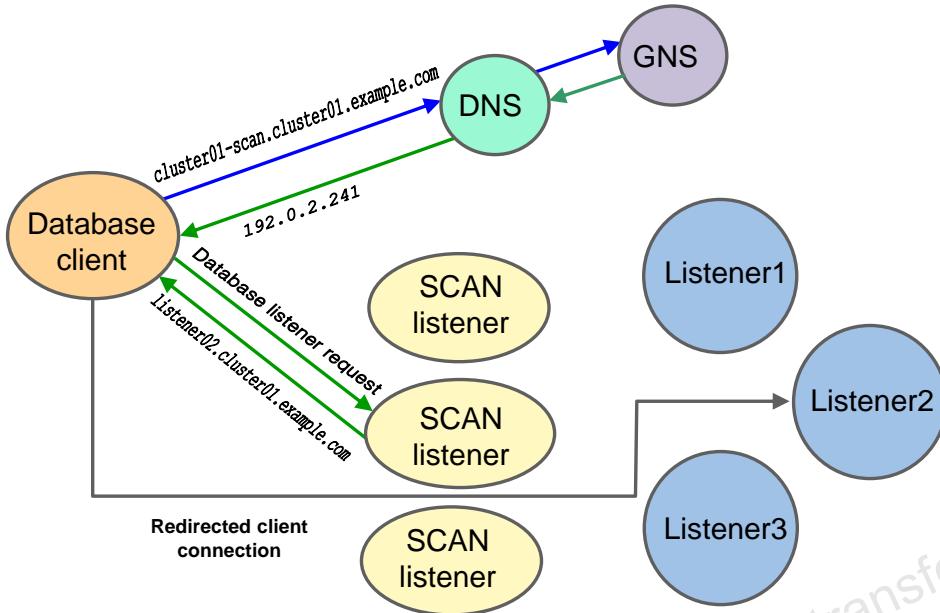
During installation, listeners are created on each node for the SCAN IP addresses. Oracle Clusterware routes application requests to the cluster SCAN to the least loaded instance providing the service.

SCAN listeners can run on any node in the cluster. SCANS provide location independence for databases so that the client configuration does not have to depend on which nodes run a particular database.

Instances register with SCAN listeners only as remote listeners. Upgraded databases register with SCAN listeners as remote listeners, and also continue to register with all other listeners.

If you specify a GNS domain during installation, the SCAN defaults to *clusternname-scan.GNS_domain*. If a GNS domain is not specified at installation, the SCAN defaults to *clusternname-scan.current_domain*.

Client Database Connections



ORACLE®

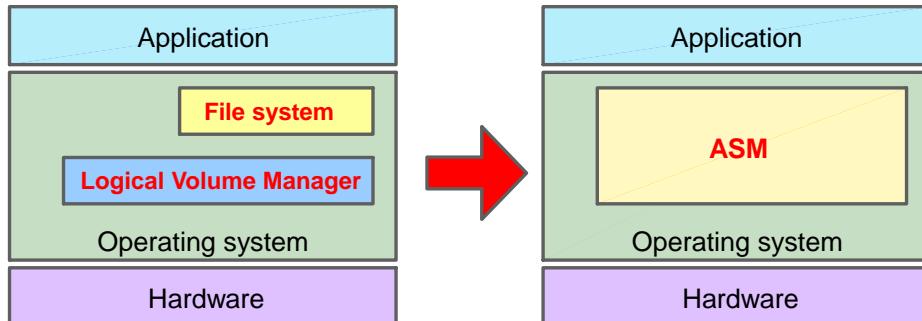
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In a GPNP environment, the database client no longer has to use the TNS address to contact the listener on a target node. Instead, it can use the EZConnect method to connect to the database. When resolving the address listed in the connect string, the DNS will forward the resolution request to the GNS with the SCAN VIP address for the chosen SCAN listener and the name of the database service that is desired. In EZConnect syntax, this would look like:

`scan-name.cluster-name.company.com/ServiceName`, where the service name might be the database name. The GNS will respond to the DNS server with the IP address matching the name given; this address is then used by the client to contact the SCAN listener. The SCAN listener uses its connection load balancing system to pick an appropriate listener, whose name it returns to the client in an OracleNet Redirect message. The client reconnects to the selected listener, resolving the name through a call to the GNS.

The SCAN listeners must be known to all the database listener nodes and clients. The database instance nodes cross-register only with known SCAN listeners, also sending them per-service connection metrics. The SCAN known to the database servers may be profile data or stored in OCR.

Oracle ASM



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle ASM is a volume manager and a file system for Oracle Database files that supports single-instance Oracle Database and Oracle Real Application Clusters (Oracle RAC) configurations. ASM has been specifically engineered to provide the best performance for both single instance and RAC databases. Oracle ASM is Oracle's recommended storage management solution that provides an alternative to conventional volume managers, file systems, and raw devices.

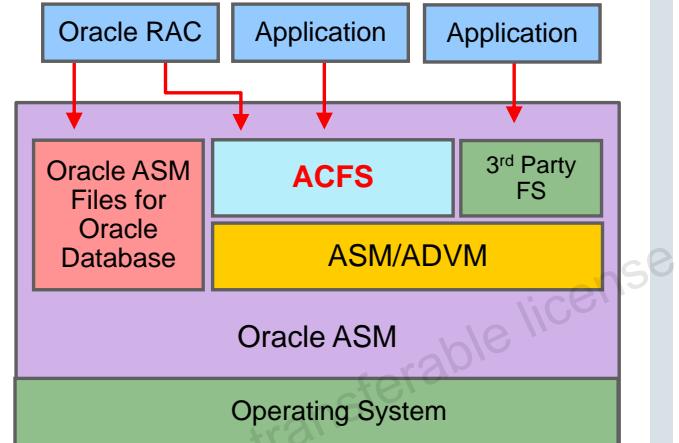
Combining volume management functions with a file system allows a level of integration and efficiency that would not otherwise be possible. For example, ASM is able to avoid the overhead associated with a conventional file system and achieve native raw disk performance for Oracle data files and other file types supported by ASM.

ASM is engineered to operate efficiently in both clustered and nonclustered environments.

Oracle ASM files can coexist with other storage management options such as raw disks and third-party file systems. This capability simplifies the integration of Oracle ASM into pre-existing environments.

Oracle ACFS

- Multi-platform, Scalable file system, and storage management technology that extends Oracle ASM functionality
- Support Oracle database files and other files.
- Spreads data across disks to balance load.
- Provides integrated mirroring across disks.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle ACFS is a multi-platform, scalable file system, and storage management technology that extends Oracle Automatic Storage Management (Oracle ASM) functionality to support all customer files. Oracle ACFS supports Oracle Database files and application files, including executables, database data files, database trace files, database alert logs, application reports, BFILEs, and configuration files. Other supported files are video, audio, text, images, engineering drawings, and other general-purpose application file data. Oracle ACFS conforms to POSIX standards for Linux and UNIX, and to Windows standards for Windows.

An Oracle ACFS file system communicates with Oracle ASM and is configured with Oracle ASM storage, as shown previously. Oracle ACFS leverages Oracle ASM functionality that enables:

- Oracle ACFS dynamic file system resizing
- Maximized performance through direct access to Oracle ASM disk group storage
- Balanced distribution of Oracle ACFS across Oracle ASM disk group storage for increased I/O parallelism
- Data reliability through Oracle ASM mirroring protection mechanisms

Oracle ACFS is tightly coupled with Oracle Clusterware technology, participating directly in Clusterware cluster membership state transitions and in Oracle Clusterware resource-based high availability (HA) management. In addition, Oracle installation, configuration, verification, and management tools have been updated to support Oracle ACFS.

Oracle Flex ASM

- Oracle Flex ASM enables an Oracle ASM instance to run on a separate physical server from the database servers.
- Larger clusters of ASM instances can support more clients while reducing the ASM footprint for the overall system.
- With Flex ASM, you can consolidate all the storage requirements into a single set of disk groups.
 - These disk groups are mounted by and managed by a small set of Oracle ASM instances running in a single cluster.
- ASM clients can be configured with direct access to storage or the I/Os can be sent through a pool of I/O servers.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle Flex ASM enables an Oracle ASM instance to run on a separate physical server from the database servers. With this deployment, larger clusters of Oracle ASM instances can support more database clients while reducing the Oracle ASM footprint for the overall system.

With Oracle Flex ASM, you can consolidate all the storage requirements into a single set of disk groups. All these disk groups are mounted by and managed by a small set of Oracle ASM instances running in a single cluster. You can specify the number of Oracle ASM instances with a cardinality setting. The default is three instances.

When using Oracle Flex ASM, you can configure Oracle ASM clients with direct access to storage or the I/Os can be sent through a pool of I/O servers.

A cluster is a set of nodes that provide group membership services. Each cluster has a name that is globally unique. Every cluster has one or more Hub nodes. The Hub nodes have access to Oracle ASM disks. Every cluster has at least one private network and one public network. If the cluster is going to use Oracle ASM for storage, it has at least one Oracle ASM network. A single network can be used as both a private and an Oracle ASM network. For security reasons, an Oracle ASM network should never be public. There can be only one Oracle Flex ASM configuration running within a cluster.

ASM Features and Benefits

- Stripes files rather than logical volumes
- Provides redundancy on a file basis
- Enables online disk reconfiguration and dynamic rebalancing
- Reduces the time significantly to resynchronize a transient failure by tracking changes while the disk is offline
- Provides adjustable rebalancing speed
- Is cluster-aware
- Supports reading from mirrored copy instead of primary copy for extended clusters
- Is automatically installed as part of the Grid Infrastructure



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

ASM provides striping and mirroring without the need to purchase a third-party Logical Volume Manager. ASM divides a file into pieces and spreads them evenly across all the disks. ASM uses an index technique to track the placement of each piece. Traditional striping techniques use mathematical functions to stripe complete logical volumes. ASM is unique in that it applies mirroring on a file basis, rather than on a volume basis. Therefore, the same disk group can contain a combination of files protected by mirroring or not protected at all.

When your storage capacity changes, ASM does not restripe all the data. However, in an online operation, ASM moves data proportional to the amount of storage added or removed to evenly redistribute the files and maintain a balanced I/O load across the disks. You can adjust the speed of rebalance operations to increase or decrease the speed and adjust the impact on the I/O subsystem. This capability also enables the fast resynchronization of disks that may suffer a transient failure.

ASM supports all Oracle database file types. It supports Real Application Clusters (RAC) and eliminates the need for a cluster Logical Volume Manager or a cluster file system. In extended clusters, you can set a preferred read copy.

ASM is included in the Grid Infrastructure installation. It is available for both the Enterprise Edition and Standard Edition installations.



Quiz

Which of the following statements regarding Grid Naming Service is *not* true?

- a. GNS is an integral component of Grid Plug and Play.
- b. Each node in the cluster runs a multicast DNS (mDNS) process.
- c. The GNS virtual IP address must be assigned by DHCP.
- d. The cluster subdomain is defined as a delegated domain.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.



Quiz

Each cluster node's public Ethernet adapter must support UDP or RDS.

- a. True
- b. False



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.



Quiz

Which of the following cluster types are available in Oracle Clusterware 12.2?

- a. Standalone Cluster
- b. Oracle Domain Services Cluster
- c. Oracle Leaf Cluster
- d. Oracle Member Cluster for Oracle Databases
- e. Oracle Member Cluster for Applications
- f. Oracle Extended Clusters



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Practice 1: Overview

This practice covers Configuring a Standalone Flex Cluster.

Summary

In this lesson, you should have learned to:

- Explain the principles and purposes of clusters
- Describe the Oracle Clusterware architecture
- Describe how Grid Plug and Play affects Clusterware



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

RAC Databases Overview and Architecture



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the benefits of Oracle RAC
- Explain the necessity of global resources
- Describe global cache coordination



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle RAC: Overview

- A cluster comprises multiple interconnected servers that appear as one server to end users and applications.
- With Oracle Clusterware, Oracle RAC enables you to cluster an Oracle database.
 - Oracle Clusterware enables nonclustered and RAC databases to use the Oracle high-availability infrastructure.
 - Oracle Clusterware enables you to create a clustered pool of storage to be used by any combination of nonclustered and Oracle RAC databases.
- Noncluster Oracle databases have a one-to-one relationship between the database and the instance.
- Oracle RAC environments have a one-to-many relationship between the database and instances.
 - An Oracle RAC database can have up to 100 instances.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

A cluster comprises multiple interconnected computers or servers that appear as if they were one server to end users and applications. Oracle RAC enables you to cluster an Oracle database. Oracle RAC uses Oracle Clusterware for the infrastructure to bind multiple servers so they operate as a single system.

Oracle Clusterware is a portable cluster management solution that is integrated with Oracle Database. Oracle Clusterware is also a required component for using Oracle RAC. In addition, Oracle Clusterware enables both noncluster Oracle databases and Oracle RAC databases to use the Oracle high-availability infrastructure. Oracle Clusterware enables you to create a clustered pool of storage to be used by any combination of noncluster and Oracle RAC databases.

Oracle Clusterware is the only clusterware that you need for most platforms on which Oracle RAC operates. You can also use clusterware from other vendors if the clusterware is certified for Oracle RAC.

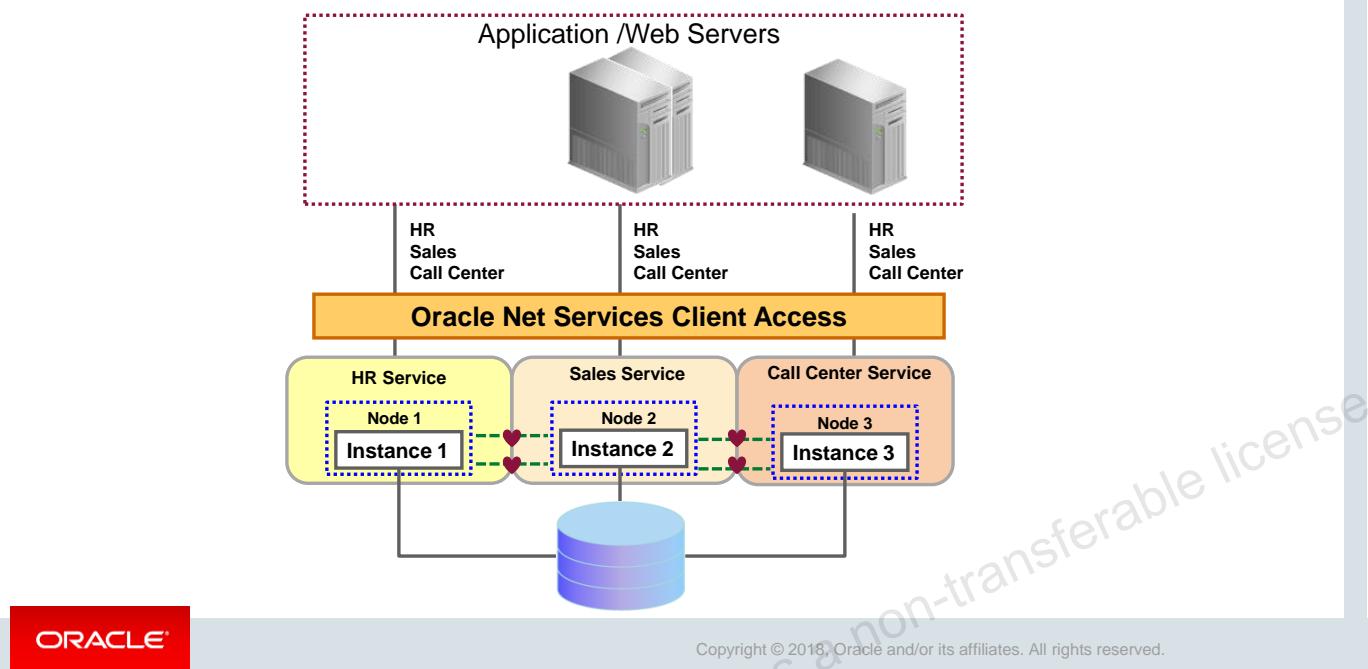
Noncluster Oracle databases have a one-to-one relationship between the Oracle database and the instance. Oracle RAC environments, however, have a one-to-many relationship between the database and instances. An Oracle RAC database can have up to 100 instances, all of which access one database. All database instances must use the same interconnect, which can also be used by Oracle Clusterware. You can also scale up to 64 reader nodes per Hub Node. The Reader Node topic is covered in later lessons.

Oracle RAC databases differ architecturally from noncluster Oracle databases in that each Oracle RAC database instance also has:

- At least one additional thread of redo for each instance
- An instance-specific undo tablespace

The combined processing power of the multiple servers can provide greater throughput and Oracle RAC scalability than is available from a single server.

Typical Oracle RAC Architecture



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The graphic in the slide shows how Oracle RAC is the Oracle Database option that provides a single system image for multiple servers to access one Oracle database. In Oracle RAC, each Oracle instance must run on a separate server.

Traditionally, an Oracle RAC environment is located in one data center. However, you can configure Oracle RAC on an extended distance cluster, which is an architecture that provides extremely fast recovery from a site failure and allows for all nodes, at all sites, to actively process transactions as part of a single database cluster. In an extended cluster, the nodes in the cluster are typically dispersed, geographically, such as between two fire cells, between two rooms or buildings, or between two different data centers or cities. For availability reasons, the data must be located at both sites, thus requiring the implementation of disk mirroring technology for storage.

If you choose to implement this architecture, you must assess whether this architecture is a good solution for your business, especially considering distance, latency, and the degree of protection it provides. Oracle RAC on extended clusters provides higher availability than is possible with local Oracle RAC configurations, but an extended cluster may not fulfill all of the disaster-recovery requirements of your organization. A feasible separation provides great protection for some disasters (for example, local power outage or server room flooding) but it cannot provide protection against all types of outages. For comprehensive protection against disasters, including protection against corruptions and regional disasters, Oracle recommends the use of Oracle Data Guard with Oracle RAC.

Oracle RAC One Node

- With online database relocation, a RAC One Node instance can be relocated to another server. This can be useful if:
 - The current server is running short on resources
 - The current server requires maintenance operations, such as operating system patches
- The same technique can be used to relocate RAC One Node instances to high-capacity servers to accommodate changes in workload.
- Single Client Access Name (SCAN) allows clients to connect to the database regardless of where the service is located.
- An Oracle RAC One Node database can be easily scaled up to a full Oracle RAC database if conditions demand it.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

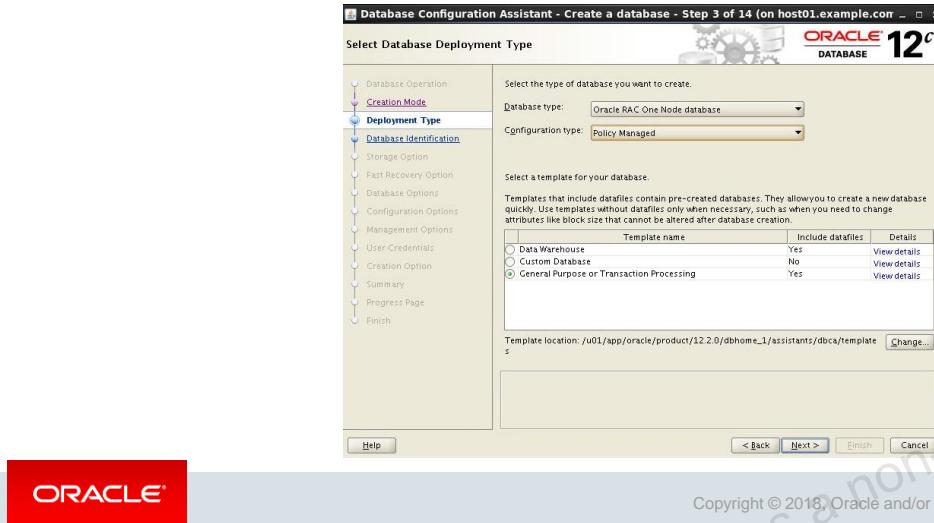
Using Oracle RAC One Node online database relocation, you can relocate the Oracle RAC One Node instance to another server, if the current server is running short on resources or requires maintenance operations, such as operating system patches. You can use the same technique to relocate Oracle RAC One Node instances to high-capacity servers (for example, to accommodate changes in workload), depending on the resources available in the cluster. In addition, Resource Manager Instance Caging or memory optimization parameters can be set dynamically to further optimize the placement of the Oracle RAC One Node instance on the new server.

By using the Single Client Access Name (SCAN) to connect to the database, clients can locate the service independently of the node on which it is running. Relocating an Oracle RAC One Node instance is, therefore, mostly transparent to the client, depending on the client connection.

For policy-managed Oracle RAC One Node databases, you must ensure that the server pools are configured such that a server will be available for the database to fail over to in case its current node becomes unavailable. In this case, the destination node for online database relocation must be located in the server pool in which the database is located. Alternatively, you can use a server pool of size 1 (one server in the server pool), setting the minimum size to 1 and the importance high enough in relation to all other server pools used in the cluster, to ensure that, upon failure of the one server used in the server pool, a new server from another server pool or the Free server pool is relocated into the server pool, as required.

Oracle RAC One Node and Oracle Clusterware

- Being closely related to Oracle RAC, Oracle RAC One Node requires Oracle Clusterware.
- You can create an Oracle RAC One Node database with DBCA.

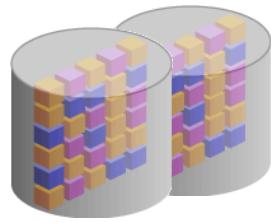


You can create Oracle RAC One Node databases by using the Database Configuration Assistant (DBCA), as with any other Oracle database (manually created scripts are also a valid alternative). Oracle RAC One Node databases may also be the result of a conversion from either a single-instance Oracle database (using `rconfig`, for example) or an Oracle RAC database. Typically, Oracle-provided tools register the Oracle RAC One Node database with Oracle Clusterware.

For Oracle RAC One Node databases, you must configure at least one dynamic database service (in addition to and opposed to the default database service). When using an administrator-managed Oracle RAC One Node database, service registration is performed as with any other Oracle RAC database. When you add services to a policy-managed Oracle RAC One Node database, SRVCTL does not accept any placement information, but instead configures those services using the value of the `SERVER_POOLS` attribute.

Cluster-Aware Storage Solutions

- RAC databases use a shared-everything architecture and require cluster-aware storage for all database files.
- The Oracle RAC database software manages disk access and is certified for use on a variety of storage architectures.
- Supported Storage Options for Oracle RAC:
 - Review Grid Infrastructure Installation and Upgrade Guide
 - Review RAC Technologies Matrix for Linux Platforms
 - Review Certification matrices located on My Oracle Support
- Use of ASM is the current and future direction for storage on a RAC system and is a highly recommended best practice.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

An Oracle RAC database is a shared-everything database. All data files, control files, PFILEs, and redo log files in Oracle RAC environments must reside on cluster-aware shared disks, so that all of the cluster database instances can access these storage components. Because Oracle RAC databases use a shared-everything architecture, Oracle RAC requires cluster-aware storage for all database files.

In Oracle RAC, the Oracle Database software manages disk access and is certified for use on a variety of storage architectures. It is your choice how to configure your storage, but you must use a supported cluster-aware storage solution. Oracle Database provides several supported storage options for Oracle RAC.

Review the following references.

- Grid Infrastructure Installation and Upgrade Guide (12.2): 6.1 Supported Storage Options for Oracle Grid Infrastructure and Oracle RAC
- RAC Technologies Matrix for Linux Platforms:
<http://www.oracle.com/technetwork/database/clustering/tech-generic-linux-new-086754.html>
- Certification matrices located on My Oracle Support

Use of ASM is the current and future direction for storage on a RAC system and is a highly recommended best practice.

Oracle RAC and Network Connectivity

- All nodes in an RAC environment must connect to at least one Local Area Network, referred to as the public network.
- RAC requires private network connectivity used exclusively for communication between the cluster nodes.
 - The interconnect network is a private network that connects all of the servers in the cluster.
- You must configure UDP for the cluster interconnect on Linux and Unix platforms. Windows clusters use TCP.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

All nodes in an Oracle RAC environment must connect to at least one Local Area Network (LAN) (commonly referred to as the public network) to enable users and applications to access the database. In addition to the public network, Oracle RAC requires private network connectivity used exclusively for communication between the nodes and database instances running on those nodes. This network is commonly referred to as the interconnect.

The interconnect network is a private network that connects all of the servers in the cluster. The interconnect network must use at least one switch and a Gigabit Ethernet adapter.

You must configure User Datagram Protocol (UDP) for the cluster interconnect, except in a Windows cluster. Windows clusters use the TCP protocol. On Linux and UNIX systems, you can configure Oracle RAC to use either the UDP or Reliable Data Socket (RDS) protocols for inter-instance communication on the interconnect. Oracle Clusterware uses the same interconnect using the UDP protocol, but cannot be configured to use RDS.

An additional network connectivity is required when using Network Attached Storage (NAS). NAS can be typical NAS devices, such as NFS filers, or can be storage that is connected by using Fibre Channel over IP, for example. This additional network communication channel should be independent of the other communication channels used by Oracle RAC (the public and private network communication). If the storage network communication needs to be converged with one of the other network communication channels, you must ensure that storage-related communication gets first priority.

Benefits of Using RAC

- High availability: Surviving node and instance failures
- Scalability: Adding more nodes as you need them in the future
- Pay as you grow: Paying for only what you need today
- Key grid computing features:
 - Growth and shrinkage on demand
 - Single-button addition of servers
 - Automatic workload management for services



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

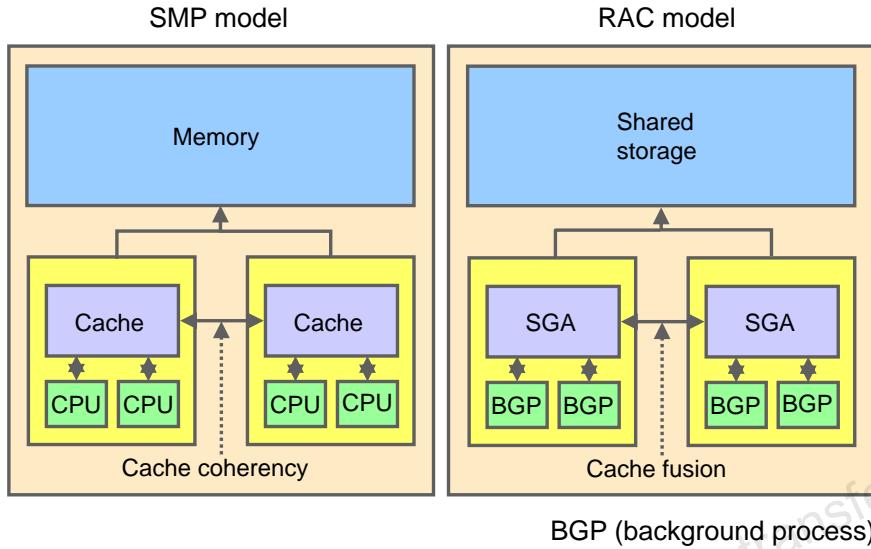
Oracle Real Application Clusters (RAC) enables high utilization of a cluster of standard, low-cost modular servers such as blades.

RAC offers automatic workload management for services. Services are groups or classifications of applications that comprise business components corresponding to application workloads. Services in RAC enable continuous, uninterrupted database operations and provide support for multiple services on multiple instances. You assign services to run on one or more instances, and alternate instances can serve as backup instances. If a primary instance fails, then Clusterware moves the services from the failed instance to a surviving alternate instance. The Oracle server also automatically load-balances connections across instances hosting a service.

RAC harnesses the power of multiple low-cost computers to serve as a single large computer for database processing, and provides the only viable alternative to large-scale symmetric multiprocessing (SMP) for all types of applications.

RAC, which is based on a shared-disk architecture, can grow and shrink on demand without the need to artificially partition data among the servers of your cluster. RAC also offers a single-button addition of servers to a cluster. Thus, you can easily provide or remove a server to or from the database.

Clusters and Scalability



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

If your application scales transparently on SMP machines, then it is realistic to expect it to scale well on RAC, without having to make any changes to the application code.

RAC eliminates the database instance, and the node itself, as a single point of failure, and ensures database integrity in the case of such failures.

The following are some scalability examples:

- Allow more simultaneous batch processes
- Allow larger degrees of parallelism and more parallel executions to occur
- Allow large increases in the number of connected users in online transaction processing (OLTP) systems

With Oracle RAC, you can build a cluster that fits your needs, whether the cluster is made up of servers where each server is a two-CPU commodity server or clusters where the servers have 32 or 64 CPUs in each server. The Oracle parallel execution feature allows a single SQL statement to be divided up into multiple processes, where each process completes a subset of work. In an Oracle RAC environment, you can define the parallel processes to run only on the instance where the user is connected to run across multiple instances in the cluster.

Levels of Scalability

- Hardware: Disk input/output (I/O)
- Internode communication: High bandwidth and low latency
- Operating system: Number of CPUs
- Database management system: Synchronization
- Application: Design



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

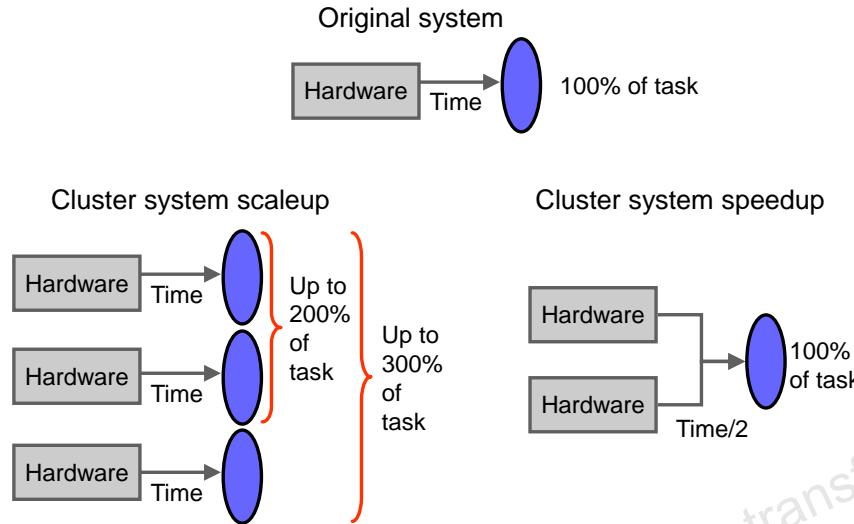
Successful implementation of cluster databases requires optimal scalability on four levels:

- **Hardware scalability:** Interconnectivity is the key to hardware scalability, which greatly depends on high bandwidth and low latency.
- **Operating system scalability:** Methods of synchronization in the operating system can determine the scalability of the system. In some cases, potential scalability of the hardware is lost because of the operating system's inability to handle multiple resource requests simultaneously.
- **Database management system scalability:** A key factor in parallel architectures is whether the parallelism is affected internally or by external processes. The answer to this question affects the synchronization mechanism.
- **Application scalability:** Applications must be specifically designed to be scalable. A bottleneck occurs in systems in which every session is updating the same data most of the time. Note that this is not RAC-specific and is true on single-instance systems, too.

It is important to remember that if any of the preceding areas are not scalable (no matter how scalable the other areas are), then parallel cluster processing may not be successful. A typical cause for the lack of scalability is one common shared resource that must be accessed often.

This causes the otherwise parallel operations to serialize on this bottleneck. High latency in the synchronization increases the cost of synchronization, thereby counteracting the benefits of parallelization. This is a general limitation and not a RAC-specific limitation.

Scaleup and Speedup



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Scaleup is the ability to sustain the same performance levels (response time) when both workload and resources increase proportionally:

$$\text{Scaleup} = (\text{volume parallel}) / (\text{volume original})$$

For example, if 30 users consume close to 100 percent of the CPU during normal processing, then adding more users would cause the system to slow down due to contention for limited CPU cycles. However, by adding CPUs, you can support extra users without degrading performance.

Speedup is the effect of applying an increasing number of resources to a fixed amount of work to achieve a proportional reduction in execution times:

$$\text{Speedup} = (\text{time original}) / (\text{time parallel})$$

Speedup results in resource availability for other tasks. For example, if queries usually take 10 minutes to process, and running in parallel reduces the time to 5 minutes, then additional queries can run without introducing the contention that might occur if they were to run concurrently.

Speedup/Scaleup and Workloads

Workload	Speedup	Scaleup
OLTP and Internet	No	Yes
DSS with parallel query	Yes	Yes
Batch (mixed)	Possible	Yes



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

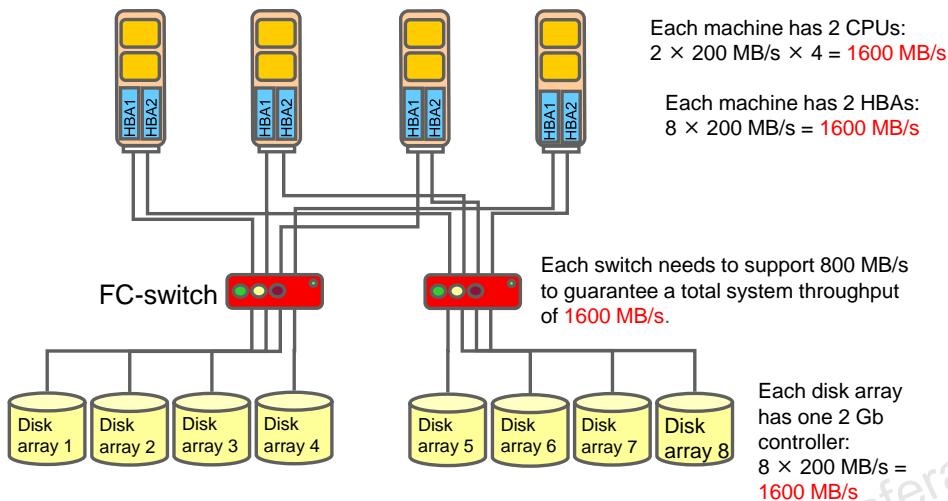
The type of workload determines whether scaleup or speedup capabilities can be achieved using parallel processing.

Online transaction processing (OLTP) and Internet application environments are characterized by short transactions that cannot be further broken down and, therefore, no speedup can be achieved. However, by deploying greater amounts of resources, a larger volume of transactions can be supported without compromising the response.

Decision support systems (DSS) and parallel query options can attain speedup, as well as scaleup, because they essentially support large tasks without conflicting demands on resources. The parallel query capability within the Oracle database can also be leveraged to decrease overall processing time of long-running queries and to increase the number of such queries that can be run concurrently.

In an environment with a mixed workload of DSS, OLTP, and reporting applications, scaleup can be achieved by running different programs on different hardware. Speedup is possible in a batch environment, but may involve rewriting programs to use the parallel processing capabilities.

I/O Throughput Balanced: Example



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

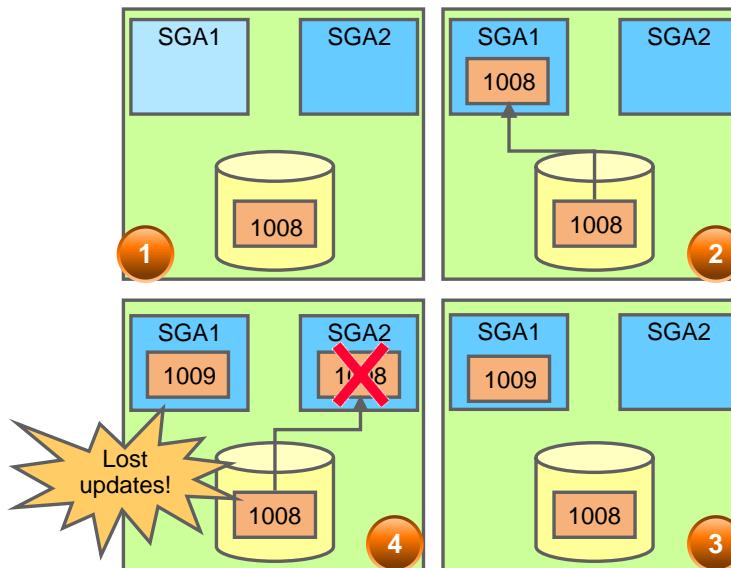
To make sure that a system delivers the I/O demand that is required, all system components on the I/O path need to be orchestrated to work together.

The weakest link determines the I/O throughput.

On the left, you see a high-level picture of a system. This is a system with four nodes, two Host Bus Adapters (HBAs) per node, two Fibre Channel switches, which are attached to four disk arrays each. The components on the I/O path are the HBAs, cables, switches, and disk arrays. Performance depends on the number and speed of the HBAs, switch speed, controller quantity, and speed of disks. If any one of these components is undersized, the system throughput is determined by this component. Assuming you have a 2 GB HBA, the nodes can read about $8 \times 200 \text{ MB/s} = 1.6 \text{ GB/s}$. However, assuming that each disk array has one controller, all eight arrays can also do $8 \times 200 \text{ MB/s} = 1.6 \text{ GB/s}$. Therefore, each of the Fibre Channel switches also needs to deliver at least 2 GB/s per port, to a total of 800 MB/s throughput. The two switches will then deliver the needed 1.6 GB/s.

Note: When sizing a system, also take the system limits into consideration. For instance, the number of bus slots per node is limited and may need to be shared between HBAs and network cards. In some cases, dual port cards exist if the number of slots is exhausted. The number of HBAs per node determines the maximal number of Fibre Channel switches. The total number of ports on a switch limits the number of HBAs and disk controllers.

Necessity of Global Resources



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Additional Memory Requirement for RAC

- Heuristics for scalability cases:
 - 15% more shared pool
 - 10% more buffer cache
- Smaller buffer cache per instance in the case of single-instance workload distributed across multiple instances
- Current values:

```
SELECT resource_name,
       current_utilization,max_utilization
  FROM v$resource_limit
 WHERE resource_name like 'g%s_%';
```

```
SELECT * FROM v$sgastat
 WHERE name like 'g_s%' or name like 'KCL%';
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

RAC-specific memory is mostly allocated in the shared pool at SGA creation time. Because blocks may be cached across instances, you must also account for bigger buffer caches.

Therefore, when migrating your Oracle Database from single instance to RAC, keeping the workload requirements per instance the same as with the single-instance case, about 10% more buffer cache and 15% more shared pool are needed to run on RAC. These values are heuristics, based on RAC sizing experience. However, these values are mostly upper bounds.

If you use the recommended automatic memory management feature as a starting point, then you can reflect these values in your `SGA_TARGET` initialization parameter.

However, consider that memory requirements per instance are reduced when the same user population is distributed over multiple nodes.

Actual resource usage can be monitored by querying the `CURRENT_UTILIZATION` and `MAX_UTILIZATION` columns for the Global Cache Services (GCS) and Global Enqueue Services (GES) entries in the `V$RESOURCE_LIMIT` view of each instance. You can monitor the exact RAC memory resource usage of the shared pool by querying `V$SGASTAT` as shown in the slide.

Parallel Execution with RAC

- In a RAC environment, a SQL statement executed in parallel can run across all of the nodes in the cluster.
- For effective inter-node parallel execution, the interconnect must be size appropriately.
 - Inter-node parallel execution may increase interconnect traffic.
 - If the interconnect bandwidth is less than the disk subsystem, parallel execution should be limited to a smaller set of nodes.
- The `PARALLEL_FORCE_LOCAL` parameter is used to limit inter-node parallel execution.
 - When set to `TRUE`, parallel server processes can execute only on the same node where the SQL statement was started.
- Services can be used to limit the number of instances that participate in a parallel SQL operation.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

By default, in an Oracle RAC environment, a SQL statement executed in parallel can run across all of the nodes in the cluster. For this cross-node or inter-node parallel execution to perform, the interconnection in the Oracle RAC environment must be size appropriately because inter-node parallel execution may result in a lot of interconnect traffic. If the interconnection has a considerably lower bandwidth in comparison to the I/O bandwidth from the server to the storage subsystem, it may be better to restrict the parallel execution to a single node or to a limited number of nodes. Inter-node parallel execution does not scale with an undersized interconnection.

To limit inter-node parallel execution, you can control parallel execution in an Oracle RAC environment using the `PARALLEL_FORCE_LOCAL` initialization parameter. By setting this parameter to `TRUE`, the parallel server processes can only execute on the same Oracle RAC node where the SQL statement was started.

In Oracle Real Application Clusters, services are used to limit the number of instances that participate in a parallel SQL operation. The default service includes all available instances. You can create any number of services, each consisting of one or more instances. Parallel execution servers are to be used only on instances that are members of the specified service. When the parameter `PARALLEL_DEGREE_POLICY` is set to `AUTO`, Oracle Database automatically decides if a statement should execute in parallel or not and what degree of parallelism (DOP) it should use.

Oracle RAC can also take advantage of In-Memory Parallel Query. In-Memory Parallel Query loads a large amount of data into the database cache on the physical memory of the servers and execute data processing by using multiple CPU resources.

Oracle Database also determines if the statement can be executed immediately or if it is queued until more system resources are available. Finally, Oracle Database decides if the statement can take advantage of the aggregated cluster memory or not.

Summary

In this lesson, you should have learned how to:

- Describe the benefits of Oracle RAC
- Explain the necessity of global resources
- Describe global cache coordination



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Installing and Configuring Oracle RAC

ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Install the Oracle database software
- Create a cluster database
- Configure Oracle RAC Reader Nodes
- Convert a single-instance Oracle database to RAC



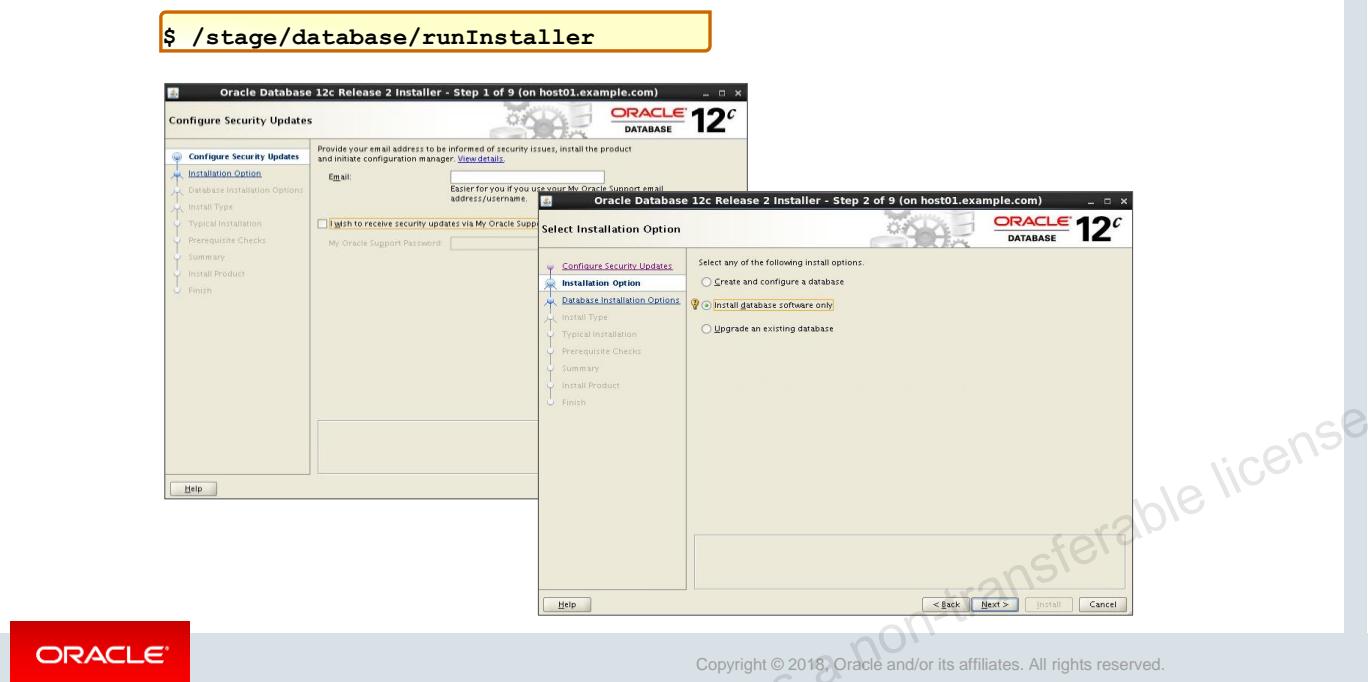
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Install the Oracle database software
- Create a cluster database
- Configure Oracle RAC Reader Nodes
- Convert a single-instance Oracle database to RAC



Installing the Oracle Database Software



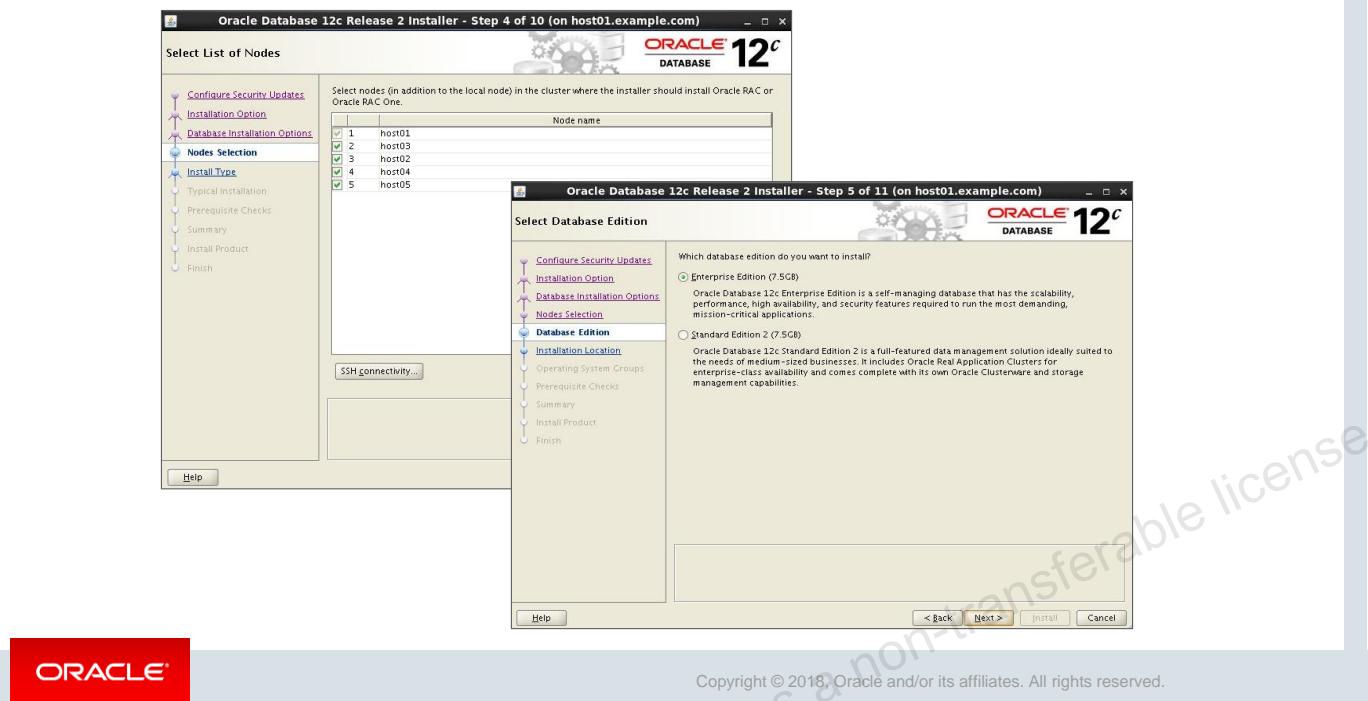
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The Oracle Universal Installer (OUI) is used to install the Oracle Database 12c software. Start the OUI by executing the `runInstaller` command from the `root` directory of the Oracle Database 12c CD-ROM or from the software staging location. You can use the Configure Security Updates window to specify an email address to receive security updates directly from Oracle Support as they occur. Alternatively, you can elect to opt out of these alerts. If you want to receive them, supply your email address and your Oracle Support password, and click Next.

The Download Software Updates page allows you to include database software updates or patches in the installation. The page allows you to either download them directly or use pre-downloaded updates. Alternatively, you can choose to bypass software updates entirely.

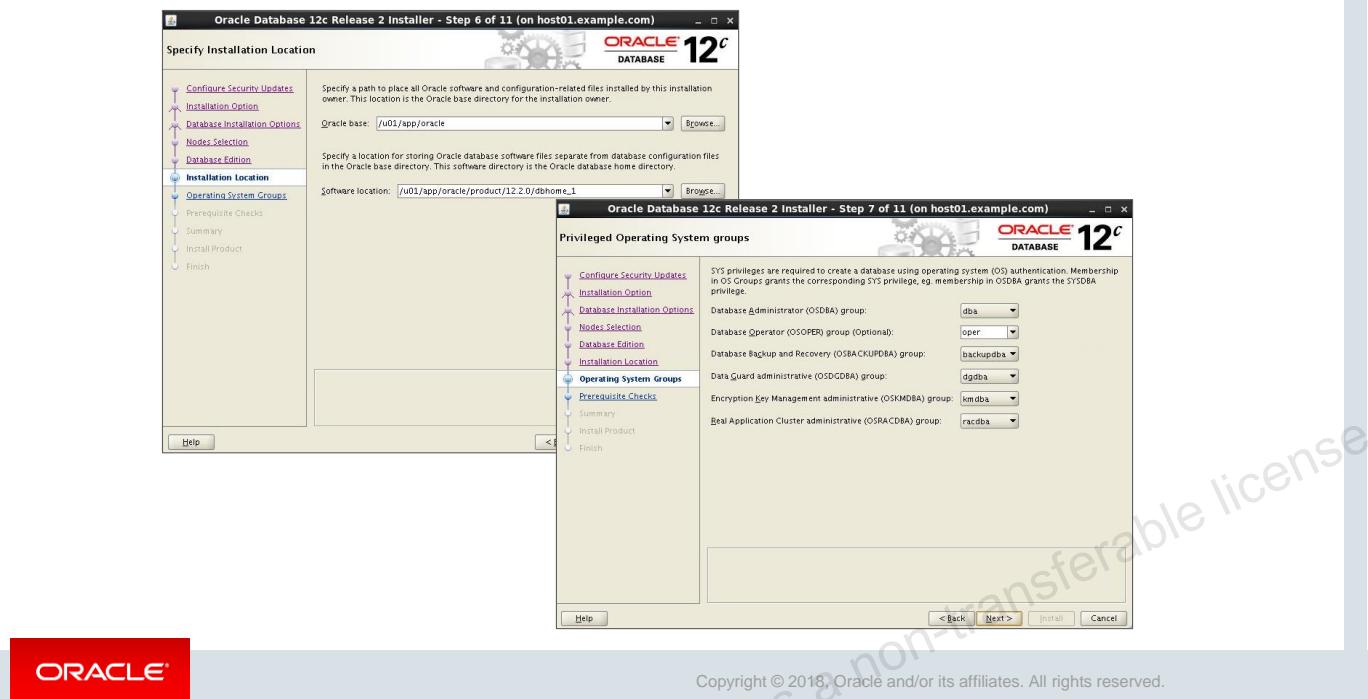
The Select Installation Option window enables you to create and configure a database, install database software only, or upgrade an existing database. Select the “Install database software only” option and click Next.

Installing the Oracle Database Software



In the Grid Installation Options window, select “Real Application Clusters database installation” and select all nodes in your cluster on which the software should be installed. If SSH for the `oracle` user has not been set up, click SSH Connectivity, then provide the `oracle` user’s password and click Setup. If SSH has already been set up, click Test. When finished, click Next to continue. In the “Select Database Edition” window, you select whether to install the Enterprise Edition or the Standard Edition. Select the Enterprise Edition option and click Next to continue.

Installing the Oracle Database Software

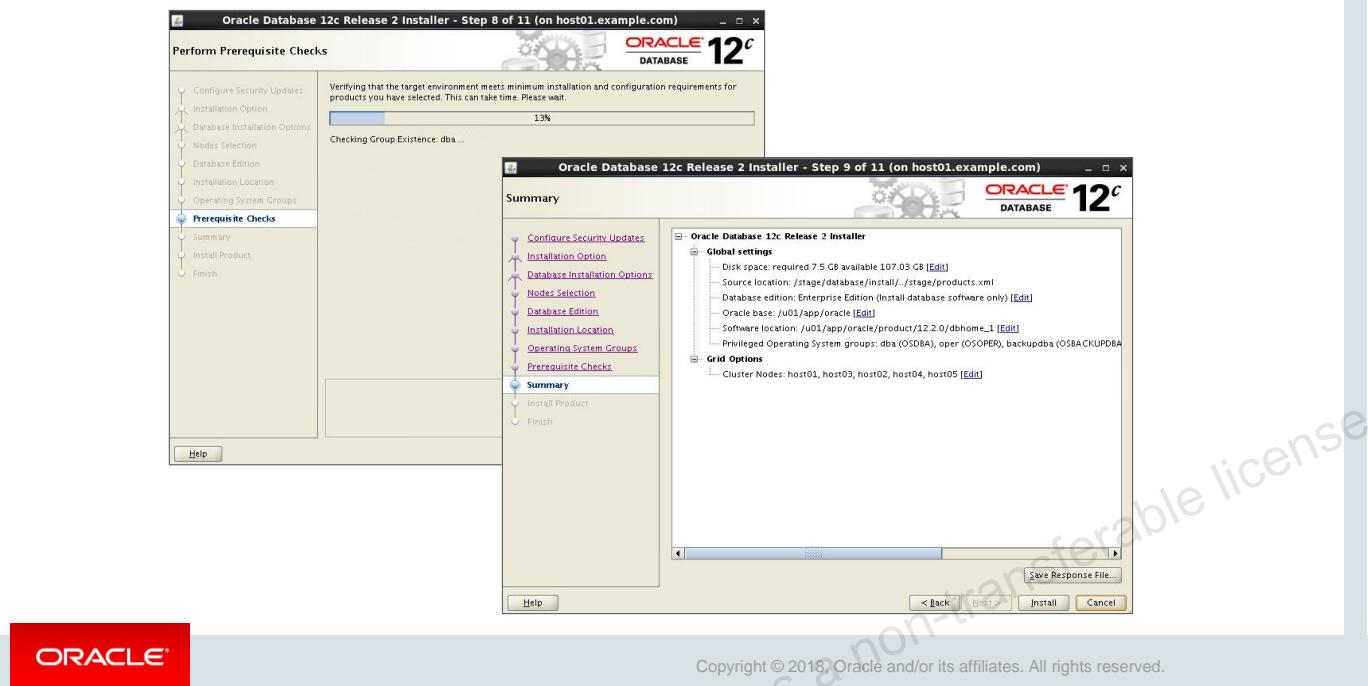


ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In the Specify Installation Location window, provide a value for `ORACLE_BASE` if you have not already done so. The default `ORACLE_BASE` location is `/u01/app/oracle`, provided the RDBMS software is being installed by the `oracle` account. The Software Location section of the window enables you to specify a value for the `ORACLE_HOME` location. The default `ORACLE_HOME` location is `/u01/app/oracle/product/12.2.0/dbhome_1`. Accept the suggested path or enter your own location. After entering the information, review it for accuracy, and click Next to continue. In the “Privileged operating system groups” window, select the operating system groups that will act as the `OSDBA` group and the `OSOPER` group. In addition, you can assign operating system groups to the database backup and recovery group, the Data Guard administrative group, the encryption key management administrative group, and Real Application Cluster administrative group. Click Next to continue.

Installing the Oracle Database Software



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

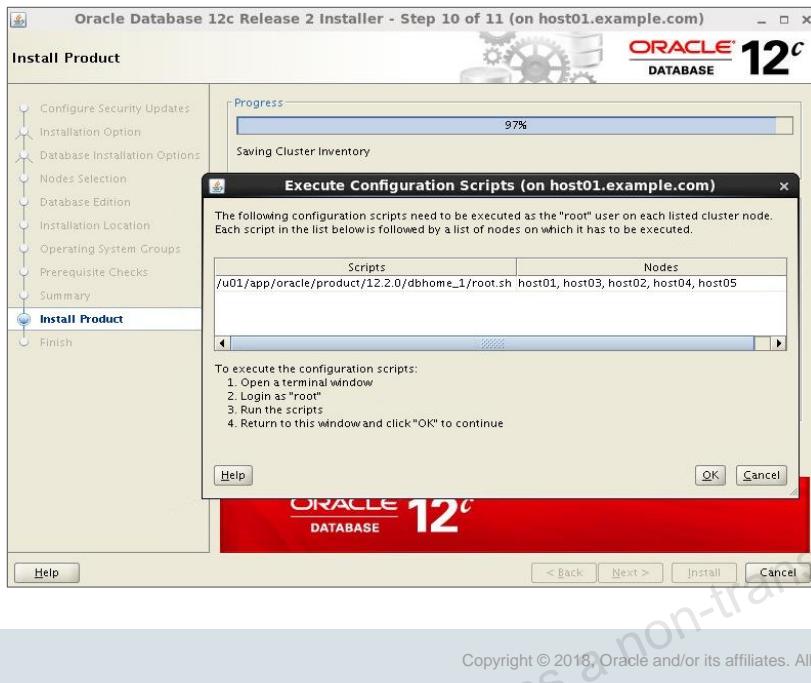
ORACLE®

The “Perform Prerequisite Checks” window verifies the operating system requirements that must be met for the installation to be successful. These requirements include:

- Certified operating system check
- Kernel parameters as required by the Oracle Database software
- Required operating system packages and correct revisions

After each successful check, the Status for that check will indicate Succeeded. Any tests that fail are also reported here. If any tests fail, click the “Fix & Check Again” button. The Installer will generate fix-up scripts to correct the system deficiencies if possible. Execute the scripts as directed by the Installer. The tests will be run again after completing the script executions. When all tests have succeeded, click Next to continue. In the Summary window, review the Global settings and click Finish.

Installing the Oracle Database Software



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

At the end of the installation, the OUI will display another window, prompting you to run the `root.sh` scripts on the nodes you chose for the installation. Follow the instructions to run the scripts. When finished, click the OK button to close the Execute Configuration Scripts window and return to the Finish screen. Click Close to complete the installation and close the OUI.

Creating the Cluster Database



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

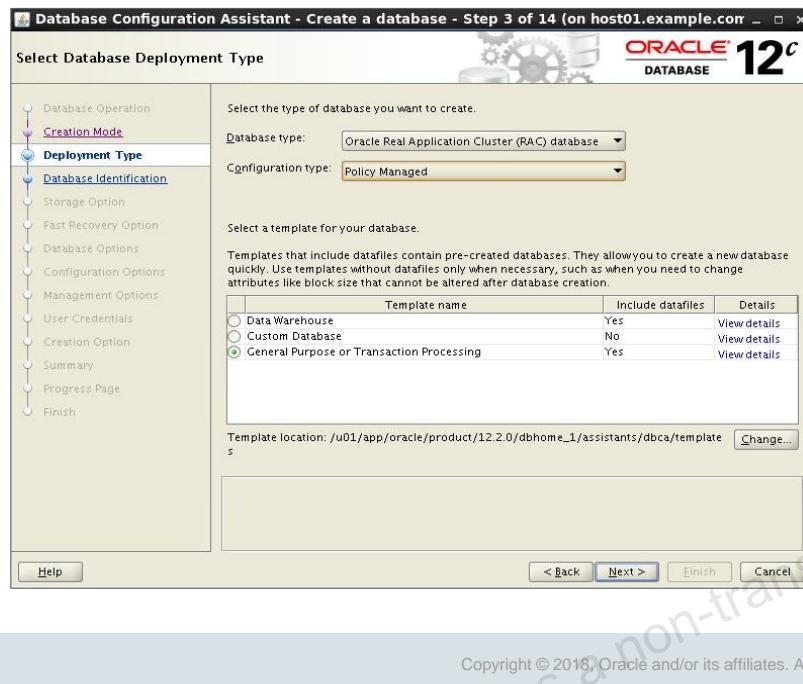
ORACLE®

To create the cluster database, change the directory to `$ORACLE_HOME/bin` on the installing node and execute the database configuration assistant (DBCA) utility as follows:

```
$ cd /u01/app/oracle/product/12.2.0/dbhome_1/bin
$ ./dbca
```

The Database Operation window appears first. You must select the database operation to perform. Select Create Database and then click Next. The Creation Mode window appears. You can choose to create a database by using the default configuration or you can choose Advanced Mode. Select Advanced Mode and click Next to continue.

Database Type Selection



The Database Templates window appears next. The Database Type pull-down menu allows you to choose to create an Oracle RAC database, an Oracle RAC One Node database, or a single instance database. The Configuration Type pull-down menu allows you to choose between Policy Managed and Administration Managed database configurations.

Administrator-managed RAC databases specify a list of cluster nodes where RAC instances will run. Services may also be specified and associated with preferred and alternative nodes. There is an explicit association between database services, instances, and cluster nodes.

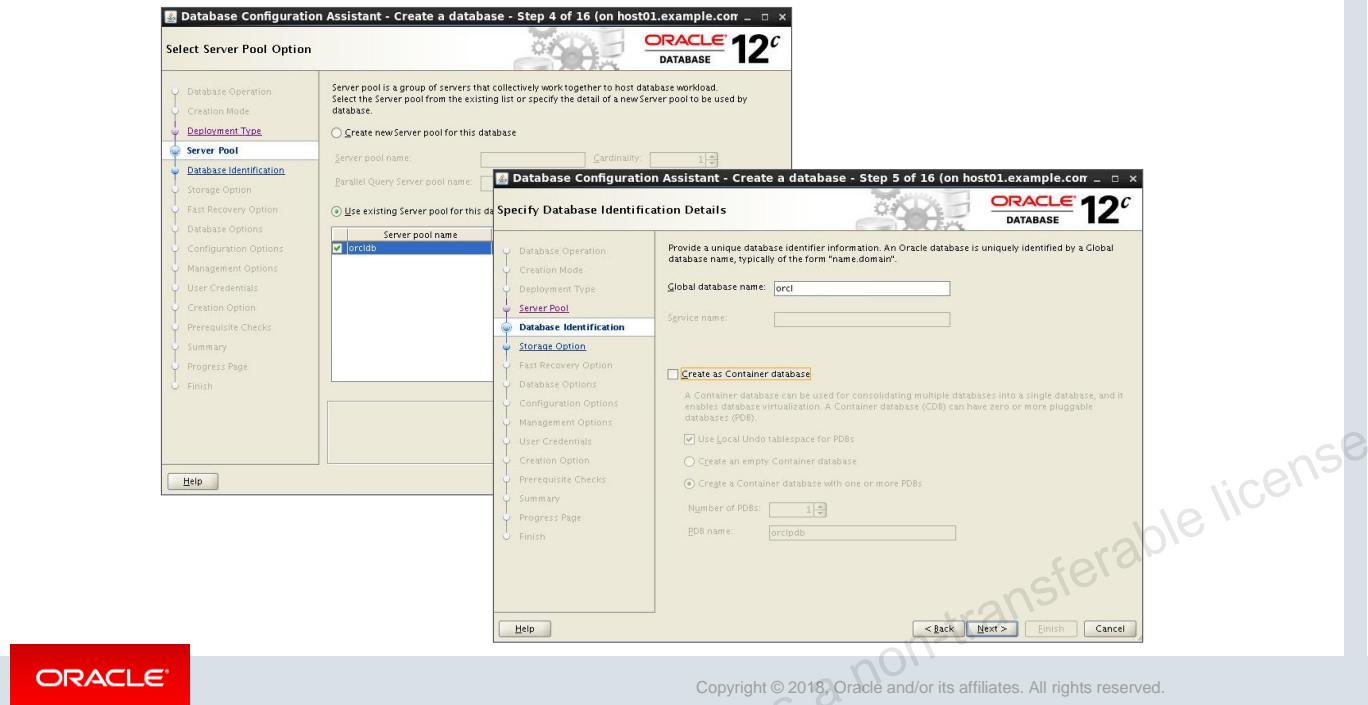
Policy-based management breaks the explicit association between services, instances, and cluster nodes. Policy-based management employs server pools, which are logical divisions of a cluster that are dynamically allocated based on relative importance. Database services are associated with server pools, and RAC instances are automatically started to satisfy the service to server pool associations. You specify in which server pool the database resource will run and the number of instances needed (cardinality). Oracle Clusterware is responsible for placing the database resource on a server. Server pools are logical divisions of a cluster into pools of servers that are allocated to host databases or other applications. Server pools are managed by using the `crsctl` and `srvctl` commands. Names must be unique within the resources defined for the cluster.

The DBCA tool provides several predefined database types to choose from, depending on your needs. The templates include:

- Data Warehouse
- Custom Database
- General Purpose or Transaction Processing

In the example in the slide, the “General Purpose or Transaction Processing” option is chosen. Click Next to continue.

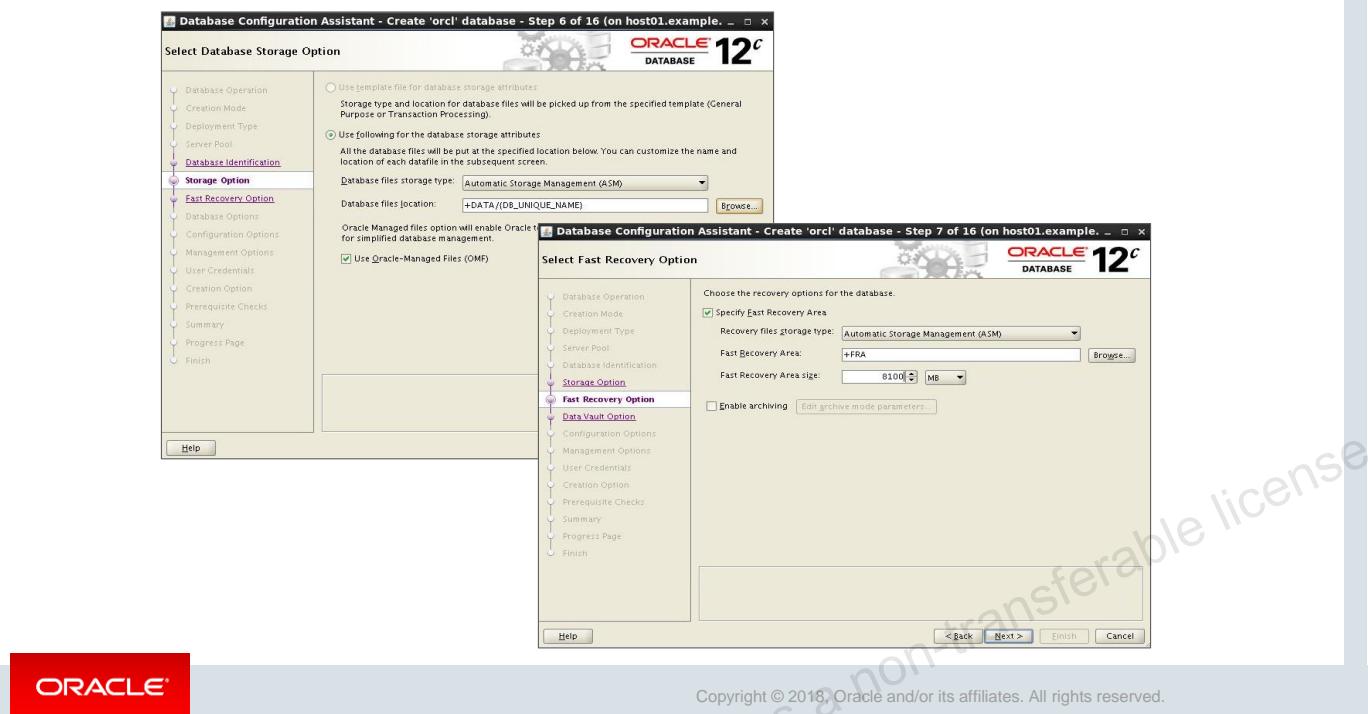
Database Identification



On the Server Pool Option page, you can choose to create a new server pool name for your database or you can elect to use an existing pool to place the database. If you create a new server pool, you will be required to set the cardinality. Click Next to continue.

In the Database Identification Details window, you must choose the global database name. The global database name can be up to 30 characters in length and must begin with an alphabetical character. The global database name is of the form `database_name.database_domain`. In addition, you can choose to create a clustered container database by selecting the Create as a Container Database check box. You can choose to create an empty container database or create it with one or more pluggable databases. If you elect to create a pluggable database, you will be required to provide a unique service name. When you have finished, click Next to continue.

Storage Options

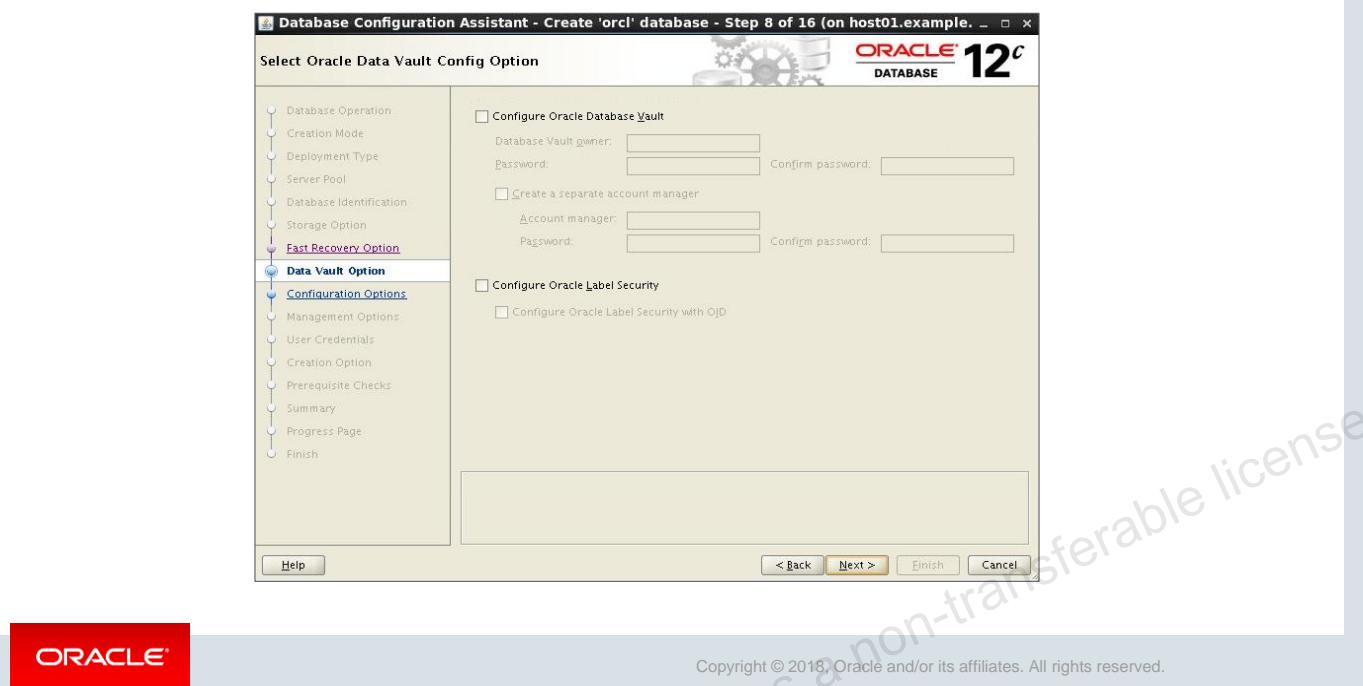


Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In the Database Storage option window, you must indicate where the database files are to be stored. You can choose your storage type from the drop-down list. Detected (and supported) shared storage types are available here. You can choose to use a standard template for file locations, one common location, or Oracle-Managed Files (OMF). This cluster database uses ASM and OMF. Therefore, select the Use Oracle-Managed Files option, and enter the disk group name in the Database Area field. Alternatively, you can click the Browse button to indicate the location where the database files are to be created.

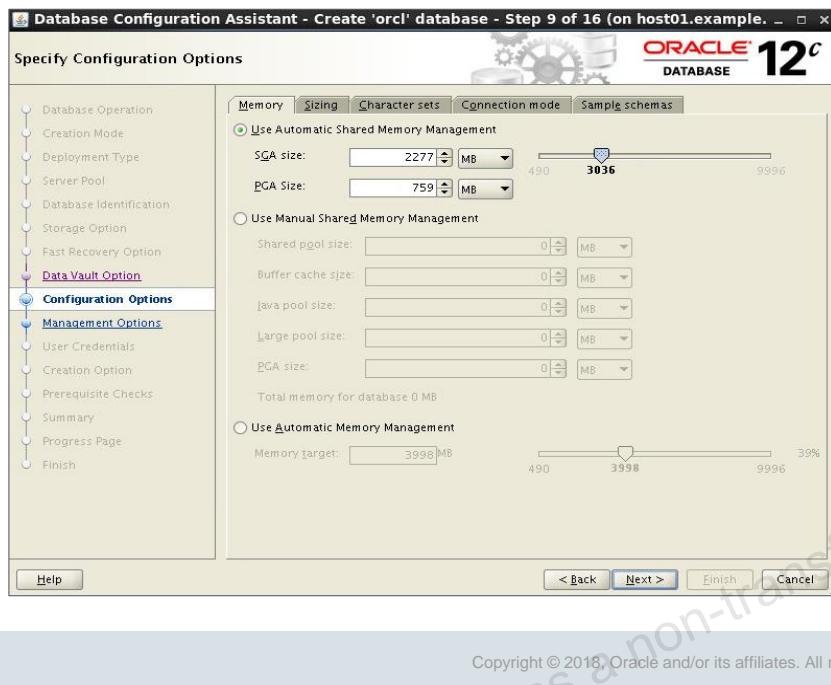
In the Fast Recovery option page, you can select redo log archiving by selecting Enable Archiving. If you are using ASM or cluster file system storages, you can also select the fast recovery area size. The size of the area defaults to 8016 megabytes., but you can change this figure if it is not suitable for your requirements. If you are using ASM and a single disk group, the fast recovery area defaults to the ASM Disk Group. If more than one disk group has been created, you can specify it here. When you have completed your entries, click Next, and the Database Content window is displayed.

Database Content



In the Oracle Database Vault Configure page, you can choose the Database Vault and Label Security option. If you choose to configure these, you must provide login credentials for the Database Vault owner. You can also choose to create a separate account manager here. Select the Configure Label Security check box to configure Label Security. When you have finished, click Next to continue to the next window.

Configuration Options



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

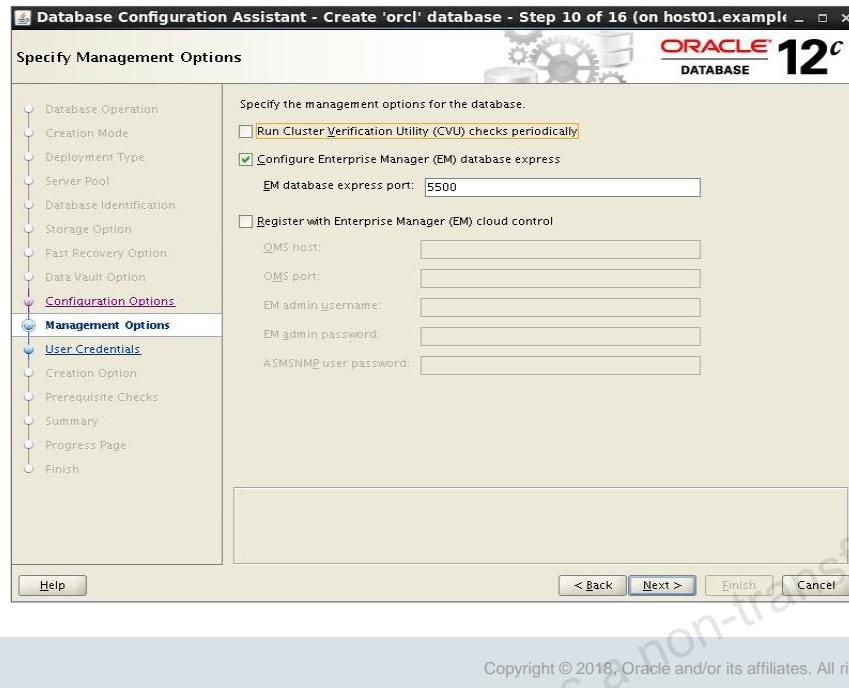
In the Configuration option window, you can set important database parameters. The parameters are grouped under five tabs:

- Memory
- Sizing
- Character Sets
- Connection Mode
- Sample Schemas

On the Memory tabbed page, you can set parameters that deal with memory allocation, including shared pool, buffer cache, Java pool, large pool, and PGA size. Automatic Memory Management is the preferred memory management method and can be selected here. On the Sizing tabbed page, you can adjust the database block size. In addition, you can set the number of processes that can connect simultaneously to the database.

By clicking the Character Sets tab, you can change the database character set. You can also select the default language and the date format. On the Connection Mode tabbed page, you can choose the connection type that clients use to connect to the database. The default type is Dedicated Server Mode. If you want to use Oracle Shared Server, click the Shared Server Mode button. On the Sample schemas page, you can choose to install the Sample Schemas included with the database distribution. If you want to review the parameters that are not found on the four tabs, click the All Initialization Parameters button. Click the Use Automatic Shared Memory Management button and click Next to continue.

Cluster Database Management Options



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

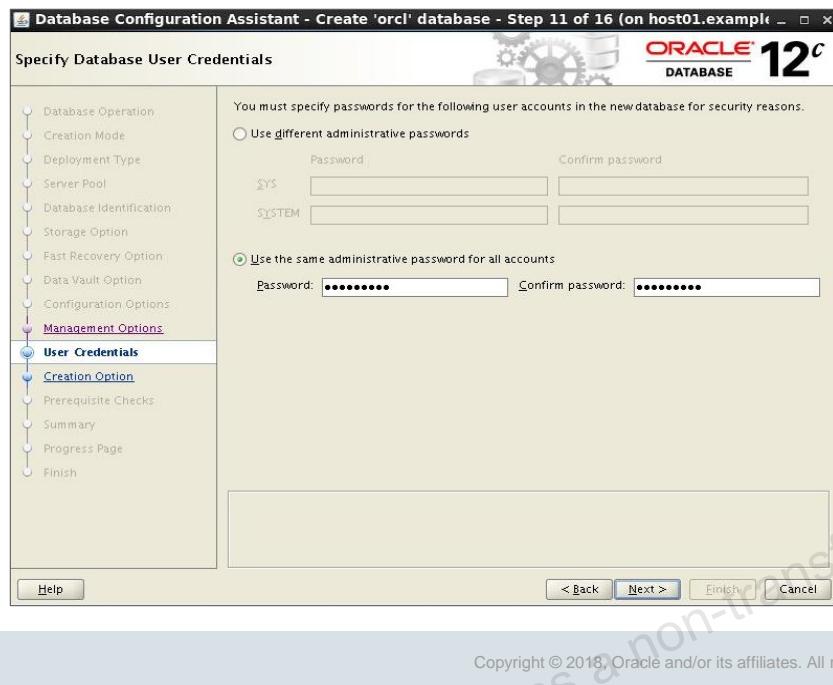
ORACLE®

The Management Options window is displayed. For small cluster environments, you may choose to manage your cluster with Enterprise Manager Database Express. To do this, select the “Configure Enterprise Manager (EM) database Express” check box. You can also elect to run CVU checks periodically.

If you have Grid Control installed somewhere on your network, you can select the “Register with Enterprise Manager (EM) Cloud Control” option. EM Cloud Control can simplify database management in large, enterprise deployments. If you select Enterprise Manager, you must provide the OMS host name and the OMS port number. You must also provide login credentials for the EM administrative user.

If you want to use Grid Control to manage your database, but have not yet installed and configured a Grid Control server, do not click either of the management methods. When you have made your choices, click Next to continue.

Passwords for Database Schema Owners



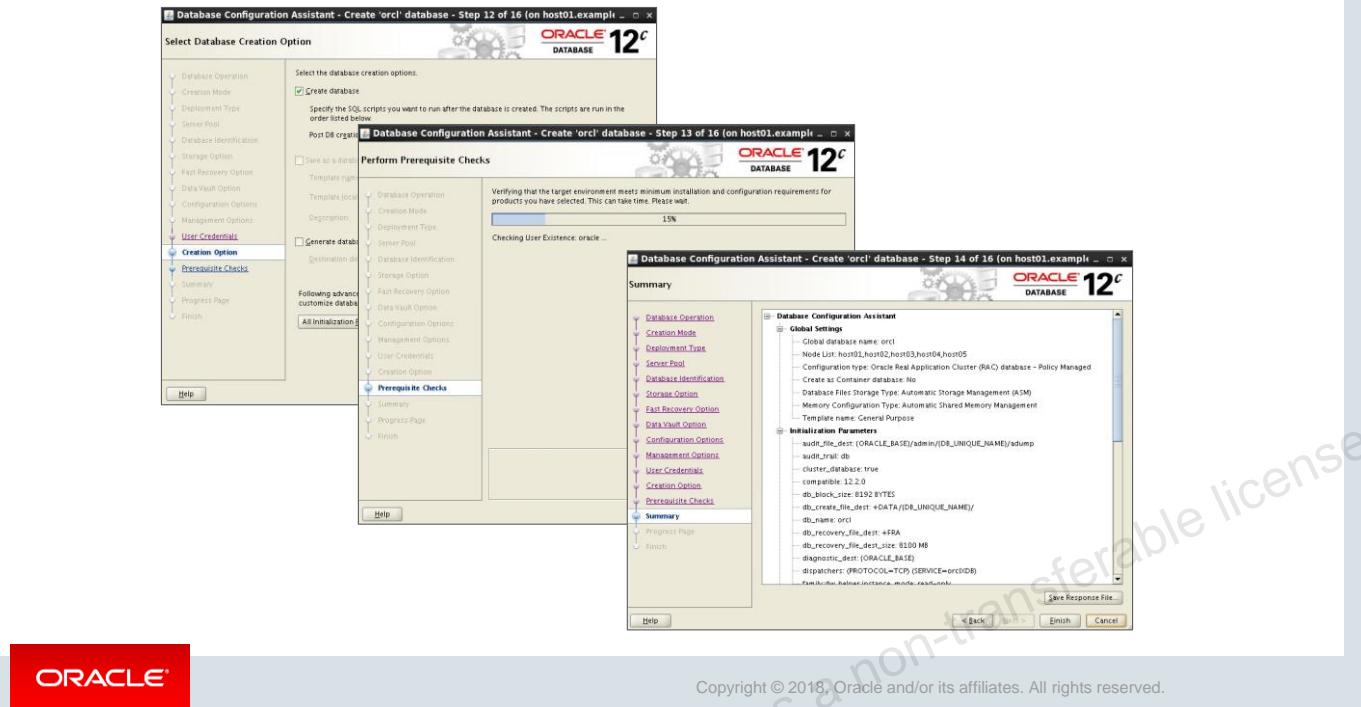
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

ORACLE®

The Database Credentials window appears next. You must supply passwords for the user accounts created by the DBCA when configuring your database. You can use the same password for all of these privileged accounts by selecting the “Use the Same Administrative Password for All Accounts” option. Enter your password in the Password field, and then enter it again in the Confirm Password field.

Alternatively, you may choose to set different passwords for the privileged users. To do this, select the “Use Different Administrative Passwords” option, enter your password in the Password field, and then enter it again in the Confirm Password field. Repeat this for each user listed in the User Name column. Click Next to continue.

Create the Database

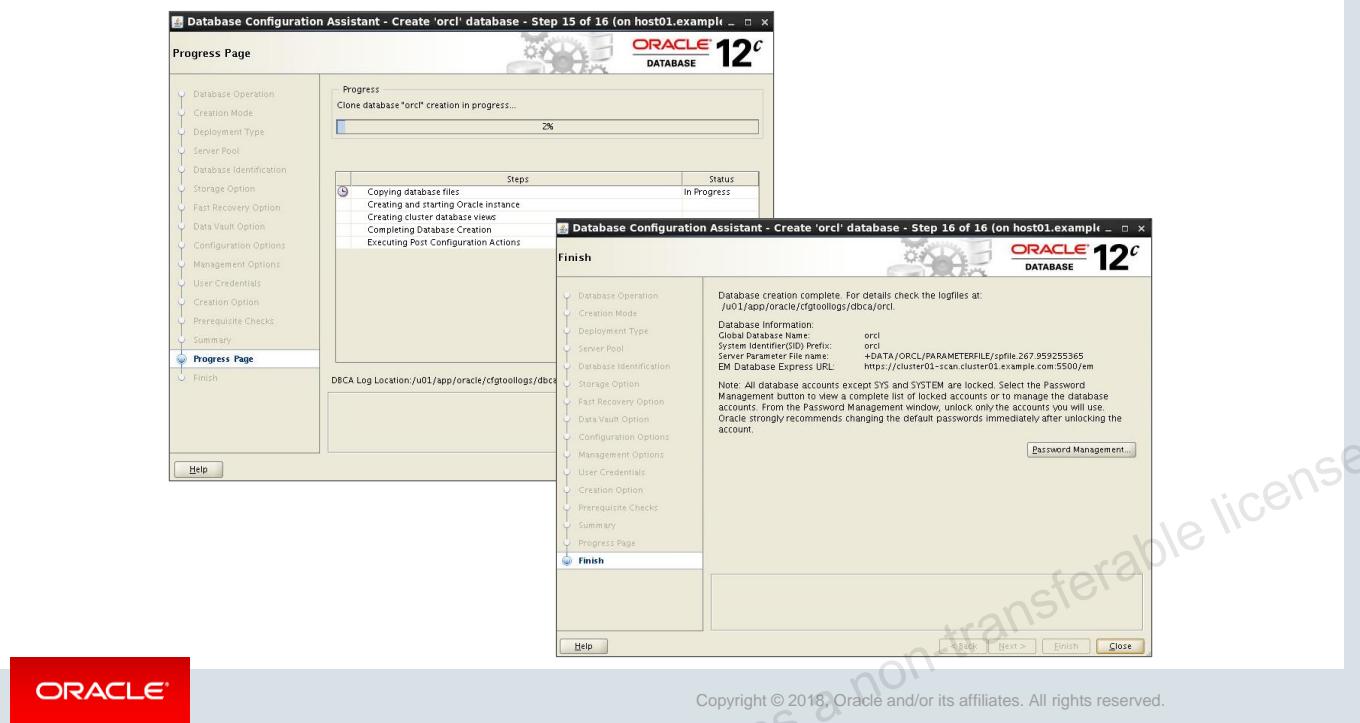


Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The Creation Options window appears. You can choose to create the database, or save your DBCA session as a database creation script by selecting the corresponding check box. Select the Create Database check box, and then click the Finish button. The DBCA displays the Summary screen, giving you a last chance to review all options, parameters, and so on that have been chosen for your database creation.

Review the summary data. When you are ready to proceed, close the Summary window by clicking Finish.

Monitoring Progress



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The Progress Monitor window appears next. In addition to informing you about how fast the database creation is taking place, it also informs you about the specific tasks being performed by the DBCA in real time. When the database creation progress reaches 100 percent, the DBCA displays the Finish page, announcing the completion of the creation process. It also directs you to the installation log file location, parameter file location, and Enterprise Manager URL. By clicking the Password Management button, you can manage the database accounts created by the DBCA.

Post-Installation Tasks

- Download and install the required patch updates.
- Verify the cluster database configuration.

```
$ 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.289.863634609
Password file: +DATA/ORCL/PASSWORD/pwdorcl.276.863633507
Domain: cluster01.example.com
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools: orcldb
Disk Groups: DATA
Mount point paths:
Services:
Type: RAC
Start concurrency:
Stop concurrency:
...
Database instances:
Configured nodes:
Database is policy managed
```

ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

After the cluster database has been successfully created, run the following command to verify the Oracle Cluster Registry configuration in your newly installed RAC environment:

```
$ srvctl config database -db db_name
```

Background Processes Specific to Oracle RAC

- **ACMS**: Atomic Control File to Memory Service
- **GTX[0-j]**: Global Transaction Process
- **LMON**: Global Enqueue Service Monitor
- **LMD**: Global Enqueue Service Daemon
- **LMS**: Global Cache Service Process
- **LCK0**: Instance Enqueue Process
- **LMHB**: Global Cache/Enqueue Service Heartbeat Monitor
- **PING**: Interconnect Latency Measurement Process
- **RCBG**: Result Cache Background Process
- **RMSn**: Oracle RAC Management Processes
- **RSMN**: Remote Slave Monitor



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

An Oracle RAC database has the same processes and memory structures as a single-instance Oracle database and additional process and memory structures that are specific to Oracle RAC. The global cache service and global enqueue service processes, and the global resource directory (GRD) collaborate to enable cache fusion. The Oracle RAC processes and their identifiers are as follows:

- **Atomic Control File to Memory Service (ACMS)**: In a RAC environment, the ACMS per-instance process is an agent that contributes to ensuring a distributed SGA memory update is either globally committed if successful, or globally aborted if a failure occurs.
- **Global Transaction Process (GTX[0-j])**: The GTX[0-j] processes provide transparent support for XA global transactions in a RAC environment. The database automatically tunes the number of these processes based on the workload of XA global transactions.
- **Global Enqueue Service Monitor (LMON)**: The LMON process monitors global enqueues and resources across the cluster and performs global enqueue recovery operations.
- **Global Enqueue Service Daemon (LMD)**: The LMD process manages incoming remote resource requests within each instance.
- **Global Cache Service Process (LMS)**: The LMS process maintains records of the data file statuses and each cached block by recording information in the GRD. The LMS process also controls the flow of messages to remote instances and manages global data block access and transmits block images between the buffer caches of different instances. This processing is part of the cache fusion feature.
- **Instance Enqueue Process (LCK0)**: The LCK0 process manages noncache fusion resource requests such as library and row cache requests.
- **Global Cache/Enqueue Service Heartbeat Monitor (LMHB)**: LMHB monitors LMON, LMD, and RMSn processes to ensure they are running normally without blocking or spinning.

- **Interconnect Latency Measurement Process (`PING`):** Every few seconds, the process in one instance sends messages to each instance. The message is received by `PING` on the target instance. The time for the round trip is measured and collected.
- **Result Cache Background Process (`RCBG`):** This process is used for handling invalidation and other messages generated by server processes attached to other instances in Oracle RAC.
- **Oracle RAC Management Processes (`RMSn`):** The `RMSn` processes perform manageability tasks for Oracle RAC. Tasks accomplished by an `RMSn` process include creation of resources related to Oracle RAC when new instances are added to the clusters.
- **Remote Slave Monitor (`RSMN`):** The `RSMN` process manages background slave process creation and communication on remote instances. These background slave processes perform tasks on behalf of a coordinating process running in another instance.

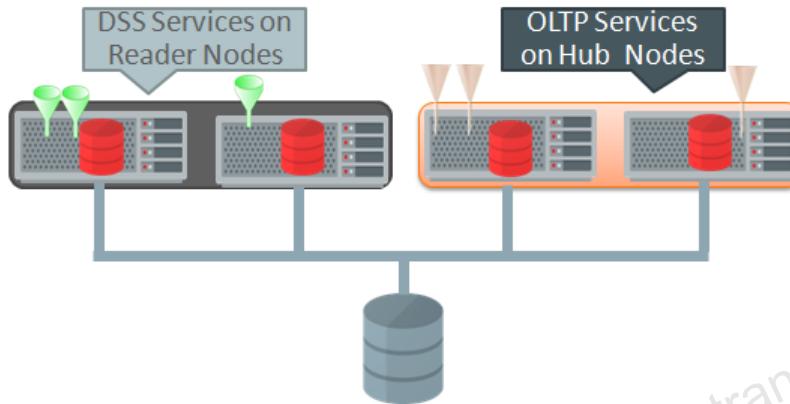
Lesson Agenda

- Install the Oracle database software
- Create a cluster database
- **Configure Oracle RAC Reader Nodes**
- Convert a single-instance Oracle database to RAC



Read Only Instances on Leaf Nodes

- Starting with Oracle Grid Infrastructure 12c Release 2, it is now possible to run read-only instances on Leaf nodes.



Oracle Grid Infrastructure 12c introduced the concept of Oracle Flex cluster in which nodes can be classified as a Hub node or a Leaf node. In Oracle Grid Infrastructure 12c Release 1, database instances could only be installed on Hub nodes and Leaf nodes were used to host applications. Starting with Oracle Grid Infrastructure 12c Release 2, it is now possible to run read-only instances on Leaf nodes.

Customers can configure their OLTP services to connect to the instances running on Hub nodes while DSS or read sessions can be directed to the instances running on Leaf nodes.

The graphic in the slide shows a four node RAC instance with two Hub nodes and two Leaf nodes with services configured.

Note: Running database instances on Leaf Nodes requires that they have direct storage access.

Oracle RAC Reader Nodes

- Reader nodes are instances of an Oracle RAC database that run on Leaf Nodes.
- Database instances on reader nodes should run in read-only mode.

```
alter system set instance_mode='read-only' family='rf_node'
scope=spfile;
```

- These instances are not affected if you need to reconfigure a hub node.
- Parallel queries running on reader nodes are not subject to brownouts that can occur when a hub node is reconfigured.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Reader nodes are instances of an Oracle RAC database that run on Leaf Nodes in an Oracle Flex Cluster, a feature that was unavailable in previous Oracle Clusterware releases.

An advantage of using reader nodes is that these instances are not affected if you need to reconfigure a hub node. Parallel query jobs running on reader nodes are not subject to the brownout times that can occur when you reconfigure a hub node, and can continue to service clients that are connected to reader nodes, as long as the hub node to which it is connected is not evicted from the cluster. You can scale up to 64 reader nodes per hub node, and reader nodes can utilize massive parallel query to speed up operations on large data sets.

Database instances running on reader nodes have different characteristics than database instances running on Hub Nodes. Hub Node instances are similar to previous versions of Oracle RAC database instances, whereas instances running on reader nodes have the following notable differences:

- Database instances run in read-only mode.
- Database instances are unaffected if you reconfigure Hub Node Clusterware. Database instances running on reader nodes are not subject to the brownout times and can continue to service the clients connected to the reader nodes, as long as the Hub Node to which it is connected is not evicted from the cluster. Using reader nodes, you can scale up to 64 reader nodes per hub.
- Because database instances running on the reader nodes are read only, you will receive an error if you attempt any DDL or DML operations.

You can create services to direct queries to read-only instances running on reader nodes. These services can use parallel query to further speed up performance. Oracle recommends that you size the memory in these reader nodes as high as possible so that parallel queries can use the memory for best performance.

You can use Leaf Nodes to host RAC database instances that run on reader nodes in read-only mode. You can optimize these nodes for parallel query by provisioning nodes with a large amount of memory so that data is cached on the Leaf Node.

In this architecture, updates to the read-write instances are immediately propagated to the read-only instances on the Leaf Nodes, where they can be used for online reporting or instantaneous queries.

Running RAC Instances on Leaf Nodes

To run Oracle RAC database instances on reader nodes:

1. Connect Leaf Nodes to storage.
2. Install Oracle Database home on all required nodes.
3. Extend public network to Leaf Nodes:

```
# srvctl modify network -netnum 1 -extendtoleaf YES  
# srvctl start nodeapps
```

4. Create a policy-managed RAC database using DBCA.
5. Extend listeners to the Leaf Nodes:

```
# srvctl modify listener -listener LISTENER -extendtoleaf yes
```

6. Add a Reader Farm server pool to the system:

```
$ srvctl add svrpool -serverpool RF1POOL -category LEAF  
$ srvctl add service -database rfdb -service RFWL -rfpool RF1POOL  
$ srvctl start service -database rfdb -service RFWL
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To run Oracle RAC database instances on reader nodes:

1. Connect Leaf Nodes to storage, if not already connected. Running database instances on Leaf Nodes requires that they have direct storage access.
2. Install Oracle Database home on all required nodes, and at least one Hub Node. To run a database instance on a Leaf Node, you must install a database home like any other node.
3. Extend public network to Leaf Nodes with the `srvctl modify network` command:
4. Create a policy-managed RAC database using DBCA. RAC Reader Nodes and Massive Parallel Query Oracle RAC require a policy-managed database. You cannot extend admin-managed databases to Leaf Nodes.
 - For Massive Parallel Query RAC, create new server pools along with the database. Ensure that you create one Parallel Query server pool.
 - For RAC Reader Nodes, create databases on Hub Nodes. The addition of database instances on Leaf nodes is dynamic and is managed from the command line.
5. Extend listeners to the Leaf Nodes using the `srvctl modify listener` command.
6. For Oracle RAC Reader Nodes, add a Reader Farm server pool to the system using the `srvctl add service` command.

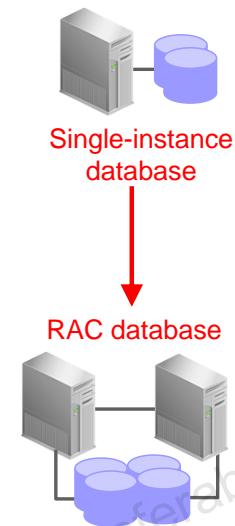
Lesson Agenda

- Install the Oracle database software
- Create a cluster database
- Configure Oracle RAC Reader Nodes
- Convert a single-instance Oracle database to RAC



Single Instance–to-RAC Conversion

- Single-instance databases can be converted to RAC using:
 - DBCA
 - Enterprise Manager
 - RCONFIG utility
- Before conversion, ensure that:
 - Your hardware and operating system are supported
 - Your cluster nodes have access to shared storage



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

A single-instance Oracle database can be converted using several methods, which include Database Configuration Assistant (DBCA), Enterprise Manager, and the `RCONFIG` utility. You can use one of these tools based on your needs.

Before converting a single-instance database to a RAC database, ensure that your system meets the following conditions:

- It has a supported hardware and operating system configuration.
- It has shared storage. A supported Cluster File System, network file system (NFS) mount, or ASM is available and accessible from all nodes.
- Your applications have no design characteristics that preclude their use with cluster database processing.

Considerations for Converting Single-Instance Databases to Oracle RAC

- Backup procedures should be available before conversion takes place.
- Archiving in Oracle RAC environments requires a thread number in the archive file format.
- The archived logs from all instances of an Oracle RAC database are required for media recovery.
- By default, all database files are migrated to Oracle Managed Files (OMF).



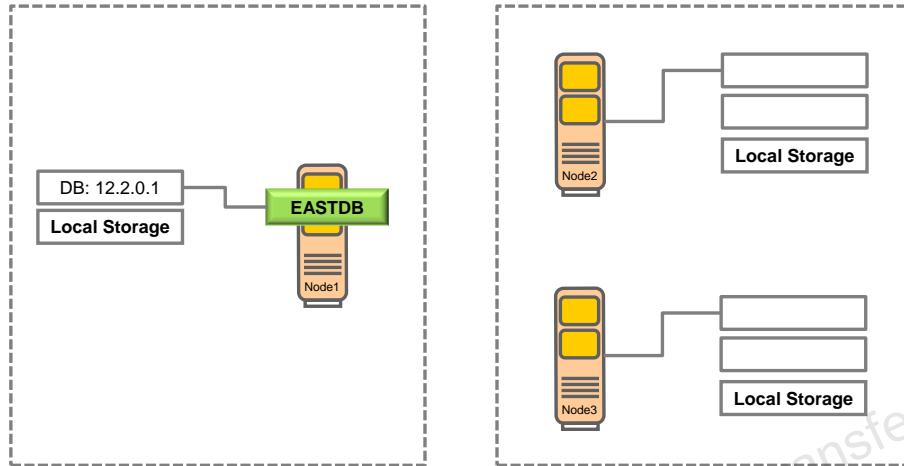
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Whatever method you choose to use for the conversion, note the following administrative considerations before converting single-instance databases to Oracle RAC:

- Backup procedures should be available before converting from a single-instance Oracle database to Oracle RAC. You should take a backup of your database before starting any major change.
- For archiving with Oracle RAC environments, the archive file format requires a thread number.
- The archived logs from all instances of an Oracle RAC database are required for media recovery. Because of this, if you archive to a file and you do not use a cluster file system, or some other means to provide shared file systems, then you require a method of accessing the archive logs from all nodes on which the cluster database has instances.
- By default, all database files are migrated to Oracle Managed Files (OMF). This feature simplifies tablespace creation, ensures data file location consistency and compliance with Optimal Flexible Architecture (OFA) rules, and reduces human error with data file management.

Scenario 1: Using DBCA

Conversion steps for a single-instance database **in a *non-clustered* environment:**



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can use DBCA to convert from single-instance Oracle databases to Oracle RAC or Oracle RAC One Node databases. DBCA automates the configuration of the control file attributes, creates the undo tablespaces and the redo logs, and creates the initialization parameter file entries for cluster-enabled environments. DBCA also configures Oracle Net Services, Oracle Clusterware resources, and the configuration for Oracle RAC database management using Oracle Enterprise Manager or the Server Control utility (SRVCTL).

To convert from a single-instance Oracle database that is on a non-cluster computer to a RAC database, perform the following steps:

1. Create an image of the single-instance database using DBCA.
2. Create an Oracle Cluster for RAC.
3. Copy the preconfigured database image files.
4. Create an Oracle RAC database using DBCA.

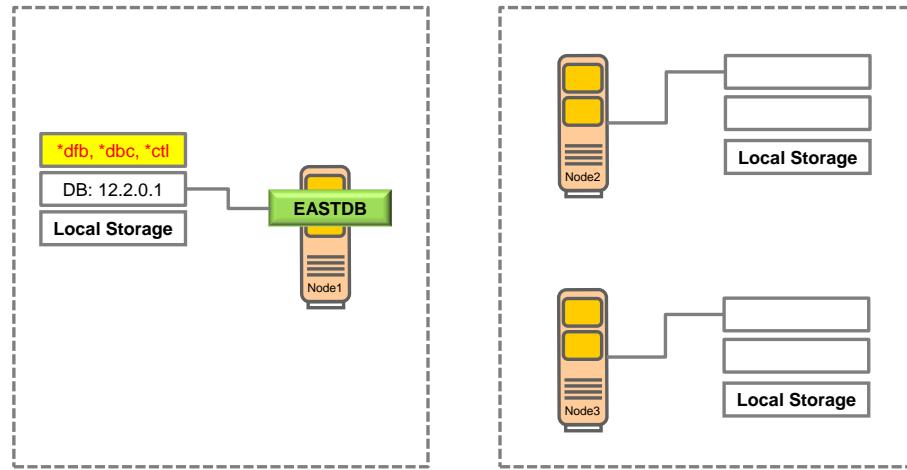
Step 1: Create an Image of the Single-Instance Database



Use the DBCA to create a preconfigured image of your single-instance database by using the following procedure:

1. Navigate to the `bin` directory in `$ORACLE_HOME`, and start the DBCA.
2. In the Welcome window, click Next.
3. In the Operations window, select Manage Templates, and click Next.
4. In the Template Management window, select “Create a database” template and “From an existing database (structure as well as data),” and click Next.
5. In the Source Database window, enter the database name in the Database instance field, and click Next.
6. In the Template Properties window, enter a template name in the Name field. By default, the template files are generated in `$ORACLE_HOME/assistants/dbca/templates`. Enter a description of the file in the Description field, and change the template file location in the Template data file field if you want. When you have finished, click Next.
7. In the Location of Database Related Files window, select “Maintain the file locations” so that you can restore the database to the current directory structure, and click Finish. The DBCA generates two files: a database structure file (`*.dbc`) and a database preconfigured image file (`*.dfb`).

Example: Result of Step 1



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- The DBCA generated three files: a database structure file (*.dbc), a control file (*.ctl), and a database preconfigured image file (*.dfb).
- The default location of these files is \$ORACLE_HOME/assistants/dbca/templates.

Step 2: Create an Oracle Cluster for RAC

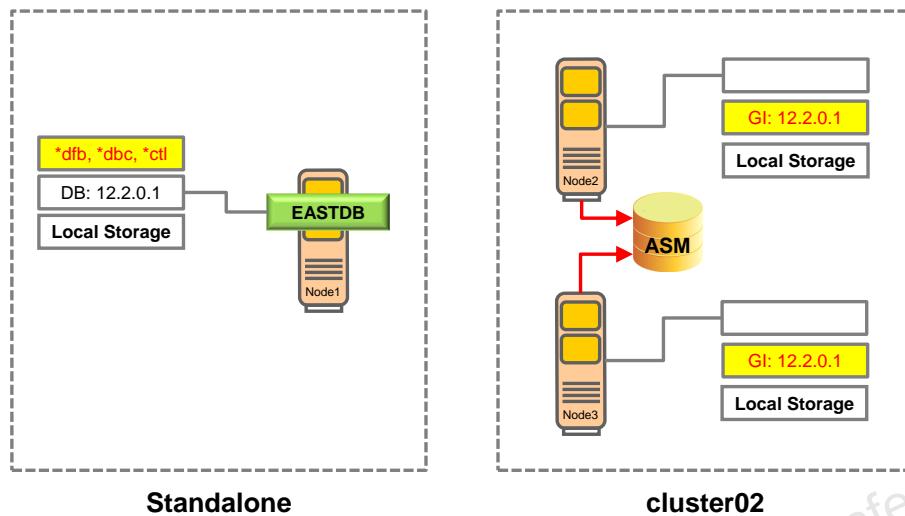
- Perform the pre-installation steps.
 - Tasks include kernel parameter configuration, hardware setup, network configuration, and shared storage setup.
- Set up and validate the cluster.
 - Create a cluster with the required number of nodes according to your hardware vendor's documentation.
 - Validate cluster components before installation.
 - Install Oracle Clusterware.
 - Validate the completed cluster installation by using `cluvfy`.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- Perform the Pre-installation Steps:
Complete the Oracle Clusterware installation, as described in the *Oracle Grid Infrastructure Installation Guide* for your platform.
- Set Up and Validate the Cluster:
Form a cluster with the required number of nodes according to your business needs. When you have configured all the nodes in your cluster, validate cluster components by using the Cluster Verification Utility, and then install Oracle Clusterware. When the clusterware is installed, validate the completed cluster installation and configuration by using the Cluster Verification Utility.

Example: Result of Step 2



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Created a 2-node Oracle Cluster called `cluster02` for RAC by installing the Grid Infrastructure software on node 2 and node 3

Step 3: Copy the Preconfigured Database Image

- Copy the preconfigured database image to a temporary location in one of the cluster nodes.
 - The database structure *.dbc file
 - The preconfigured database image *.dfb file
 - The control file image *.ctl file
- Move the database structure *.dbc file to the \$ORACLE_HOMEassistants/dbca/templates directory.
- Modify the *.dbc file to point to the preconfigured database image file in the temporary location.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- This includes copying the database structure *.dbc file and the database preconfigured image *.dfb file that the DBCA created in step one (Back Up the Original Single-Instance Database) to a temporary location on the node in the cluster from which you plan to run the DBCA.
- Move the database structure *.dbc file to the \$ORACLE_HOMEassistants/dbca/templates directory.
- Modify the *.dbc file to point to the preconfigured database image file in the temporary location.

Example: Database Structure File (*.dbc)

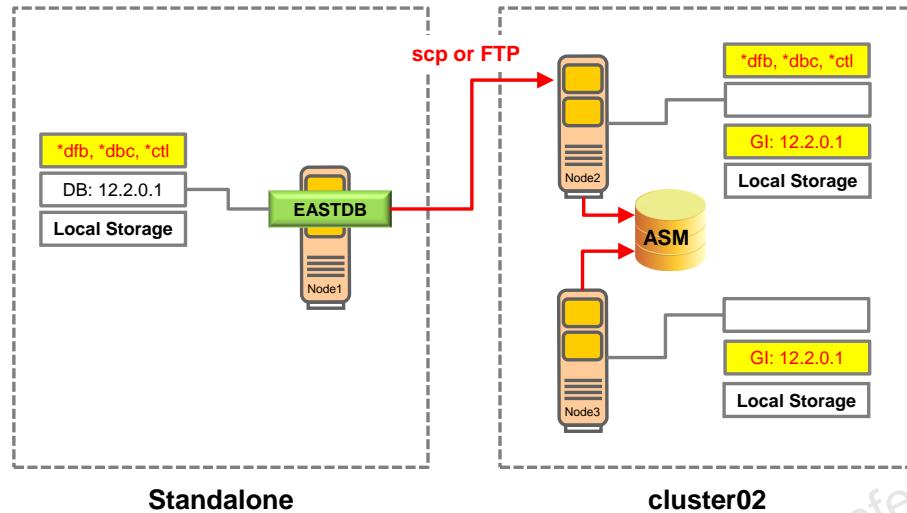
```
$ cd $ORACLE_HOME/assistants/dbca/templates  
$ vi template_name.dbc  
  
----- The Output Truncated-----  
  
<DataFiles>  
  <Location>  
    {ORACLE_HOME}/assistants/dbca/templates/template_name.dfb  
  </Location>  
  <SourceDBName cdb="false">eastdb</SourceDBName>  
  <Name id="1" Tablespace="SYSTEM" ... system01.dbf</Name>  
  <Name id="3" Tablespace="SYSAUX" ... sysaux01.dbf</Name>  
  <Name id="4" Tablespace="UNDOTBS1" ... undotbs01.dbf</Name>  
  <Name id="6" Tablespace="USERS" ...users01.dbf</Name>  
</DataFiles>  
<TempFiles>  
  <Name id="1" Tablespace="TEMP" ... temp01.dbf</Name>  
</TempFiles>  
  
----- The Output Truncated-----
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The slide shows the content of the database structure file (*.dbc). You need to modify this file if the location of the preconfigured database image (*.dfb) is different from the path specified in the file.

Example: Result of Step 3



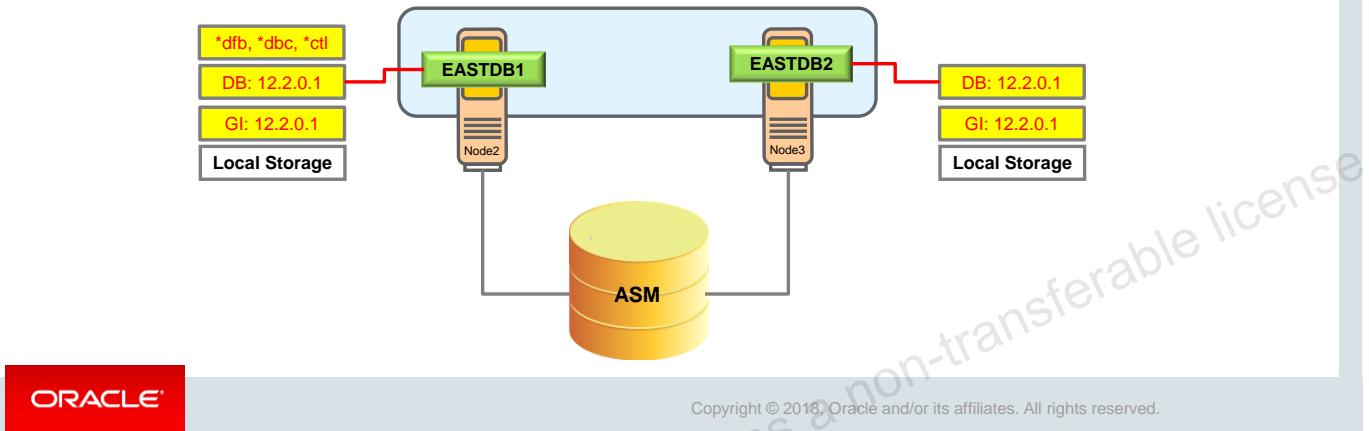
ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- Copied the preconfigured image files to the local storage of node 2
- Moved the template file (*.dbc) to \$ORACLE_HOME/assistants/dbca/templates directory
- Modified the template file (*.dbc) to point to the location where the preconfigured database image (*.dfb) was copied

Step 4: Create an Oracle RAC Database

- Install the Oracle Database 12c software.
- Create an Oracle RAC database using the preconfigured database images.

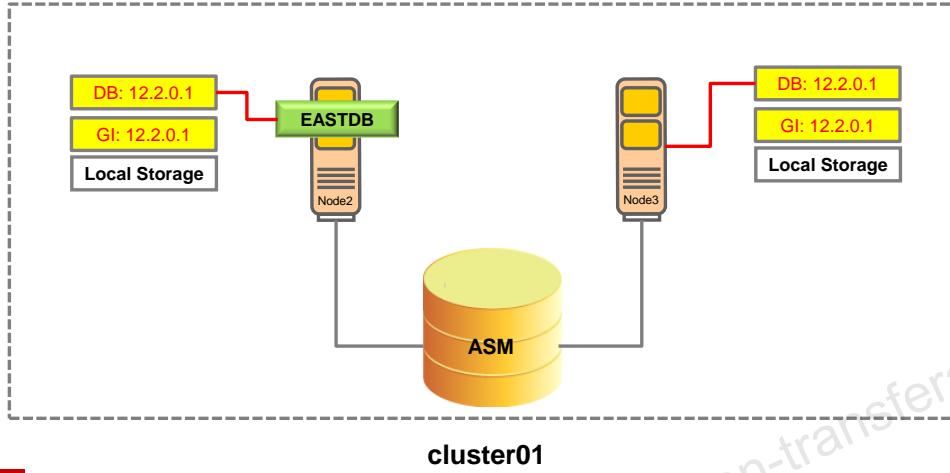


Install the Oracle Database 12c Software with RAC:

1. Run the Oracle Universal Installer (OUI) to perform an Oracle database installation with RAC. Select Cluster Installation Mode and select the nodes to include in your RAC database.
2. On the Oracle Universal Installer Database Configuration Types page, select the Advanced installation type. After installing the software, the OUI runs post-installation tools such as NETCA, DBCA, and so on.
3. In the DBCA Template Selection window, use the template that you copied to a temporary location in the “Copy the Preconfigured Database Image” step. Use the Browse option to select the template location.
4. After creating the RAC database, the DBCA displays the Password Management page in which you must change the passwords for database privileged users. When the DBCA exits, the conversion is complete.

Scenario 2: Using rconfig

- Conversion steps for a single-instance database **in a clustered environment**:



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can use the `rconfig` utility to convert from single-instance Oracle databases to Oracle RAC or Oracle RAC One Node databases. The `rconfig` utility automates many conversion steps.

To convert from a single-instance Oracle database that is on a cluster computer to a RAC database, perform the following steps:

1. Check the database type.
2. Modify the XML file for the `rconfig` utility.
3. Perform prerequisite checks.
4. Convert to an Oracle RAC database.
5. Verify the conversion.

Step 1: Check the Database Type

```
$ srvctl config database -db eastdb
Database unique name: eastdb
Database name: eastdb
Oracle home: /u01/app/oracle/product/12.2.0/dbhome_1
Oracle user: oracle
Spfile: +DATA/eastdb/spfileeastdb.ora
Password file: +DATA/eastdb/orapweastdb
Domain: cluster01.example.com
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools: eastdb
Database instances: eastdb
Disk Groups: DATA, FRA
Mount point paths:
Services:
Type: SINGLE
...
Database is administrator managed
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The database type is **SINGLE**, which indicates a single-instance database.

Step 2: Modify the XML File for the rconfig Utility

- Locate the appropriate .xml file located in the \$ORACLE_HOME/assistants/rconfig/sampleXMLs directory.
- Modify the ConvertToRAC_AdminManaged.xml or ConvertToRAC_PolicyManaged.xml file as required for your system.
- Save the file under a different name.

```
$ cd $ORACLE_HOME/assistants/rconfig/sampleXMLs  
$ cp ConvertToRAC_AdminManaged.xml ConvertToRAC_AdminManaged.xml.bkp  
$ vi ConvertToRAC_AdminManaged.xml
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Before running the rconfig utility, modify the XML file as required for your system.

- Go to the \$ORACLE_HOME/assistants/rconfig/sampleXMLs directory as the oracle user and open the ConvertToRAC_*.xml file using a text editor, such as vi.
- Review the XML file, and modify the parameters as required for your system. The XML sample file contains comment lines that provide instructions about how to configure the file.
- When you have finished making changes, save the file with the syntax filename.xml. Make a note of the name you select.

Example: ConvertToRAC_AdminManaged.xml

- Modify the XML file as required for your system:

```
$ vi ConvertToRAC_AdminManaged.xml

<n:Convert verify="ONLY">
<n:SourceDBHome>/u01/app/oracle/product/12.2.0/dbhome_1</n:SourceDBHome>
<n:TargetDBHome>/u01/app/oracle/product/12.2.0/dbhome_1</n:TargetDBHome>
<n:SourceDBInfo SID="eastdb">
<n:Node name="enode01"/>
<n:Node name="enode02"/>
<n:InstancePrefix>eastdb</n:InstancePrefix>
<n:Password>oracle_4U</n:Password>
<n:TargetDatabaseArea></n:TargetDatabaseArea>
<n:TargetFlashRecoveryArea></n:TargetFlashRecoveryArea>
```

- Change the password for sys user using SQL*Plus to match the Password parameter in the XML file.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The slide shows an example of the input parameters for the rconfig XML file.

- SourceDBHome and TargetDBHome are same if the single-instance database is already running out of the RAC-enabled Oracle Home.
- Leave the TargetDatabaseArea and TargetFlashRecoveryArea parameters as empty if the database is already in the desired shared storage.
- Change the password of sys user to match <n:Password>oracle_4U</n:Password> in the XML file.

Note: The Convert verify option in the .xml file has three options:

- Convert verify="YES": rconfig performs checks to ensure that the prerequisites for single-instance to RAC conversion have been met before it starts conversion.
- Convert verify="NO": rconfig does not perform prerequisite checks, and starts conversion.
- Convert verify="ONLY": rconfig performs only prerequisite checks; it does not start conversion after completing the prerequisite checks.

Step 3: Perform Prerequisite Checks

- Run rconfig with Convert verify="ONLY":

```
$ rconfig ConvertToRAC_AdminManaged.xml

<?xml version="1.0" ?>
<RConfig version="1.1" >
<ConvertToRAC>
  <Convert>
    <Response>
      <Result code="0" >
        Operation Succeeded
      </Result>
    </Response>
    <ReturnValue type="object">
      There is no return value for this step </ReturnValue>
    </Convert>
  </ConvertToRAC></RConfig>
```

- The default listener must be configured in Grid Infrastructure Home.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- Run rconfig with Convert verify="ONLY" to perform a test conversion to ensure that a conversion can be completed successfully.
- rconfig with Convert verify="ONLY" does not check if the default listener is configured in Grid Infrastructure Home or not. However, it is checked when running rconfig with Convert verify="YES". If the default listener is configured in Oracle Database Home, you will receive the following messages:

```
[oracle@enode01 sampleXMLs]$ rconfig ConvertToRAC_AdminManaged.xml
...
oracle.sysman.assistants.rconfig.engine.InvalidConfigurationException:
oracle.sysman.assistants.rconfig.engine.InvalidConfigurationException: Default
Listener is not configured in Grid Infrastructure Home.
Operation Failed. Refer logs at
/u01/app/oracle/cfgtoollogs/rconfig/rconfig_09_18_15_17_28_33.log for more
details.
...
```

Step 4: Convert to an Oracle RAC Database

- Run rconfig with Convert verify= "YES":

```
$ rconfig ConvertToRAC_AdminManaged.xml

Converting Database "eastdb" to Cluster Database. Target Oracle Home:
/u01/app/oracle/product/12.2.0/dbhome_1. Database Role: PRIMARY.
Setting Data Files and Control Files
Adding Database Instances
Adding Redo Logs
Enabling threads for all Database Instances
Setting TEMP tablespace
Adding UNDO tablespaces
Adding Trace files
Setting Fast Recovery Area
Updating Oratab
Creating Password file(s)
Configuring Listeners
Configuring related CRS resources
Starting Cluster Database

----- The Output Continued next page -----
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Step 4: Convert a Single Instance to an Oracle RAC Database

- Run rconfig with Convert verify="YES" to start a conversion.

During the conversion, the rconfig utility performs the following operations:

- Setting data files and control files
- Adding database instances
- Adding redo logs
- Enabling threads for all database instances
- Setting TEMP tablespace
- Adding UNDO tablespaces
- Adding Trace files
- Setting Fast Recovery Area
- Updating Oratab
- Creating password file(s)
- Configuring listeners
- Configuring related CRS resources
- Starting cluster database

Step 4: Convert to an Oracle RAC Database

```
<?xml version="1.0" ?>
<RConfig version="1.1" >
<ConvertToRAC>
<Convert>
<Response>
<Result code="0" >
Operation Succeeded
</Result>
</Response>
<ReturnValue type="object">
<Oracle_Home>
/u01/app/oracle/product/12.2.0/dbhome_1
</Oracle_Home>
<Database type="ADMIN_MANAGED" >
<InstanceList>
<Instance SID="eastdb1" Node="enode01" >
</Instance>
<Instance SID="eastdb2" Node="enode02" >
</Instance>
</InstanceList>
</Database> </ReturnValue>
</Convert>
</ConvertToRAC></RConfig>
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Step 4: Convert a Single Instance to an Oracle RAC Database

- The output shows that the conversion has been completed successfully.

Step 5: Verify the Conversion

```
$ srvctl config database -db eastdb
Database unique name: eastdb
Database name: eastdb
Oracle home: /u01/app/oracle/product/12.2.0/dbhome_1
Oracle user: oracle
Spfile: +DATA/eastdb/spfileeastdb.ora
Password file: +DATA/eastdb/orapweastdb
Domain: cluster01.example.com
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools: eastdb
Database instances: eastdb1,eastdb2
Disk Groups: DATA,FRA
Mount point paths:
Type: RAC
...
Database is administrator managed
```

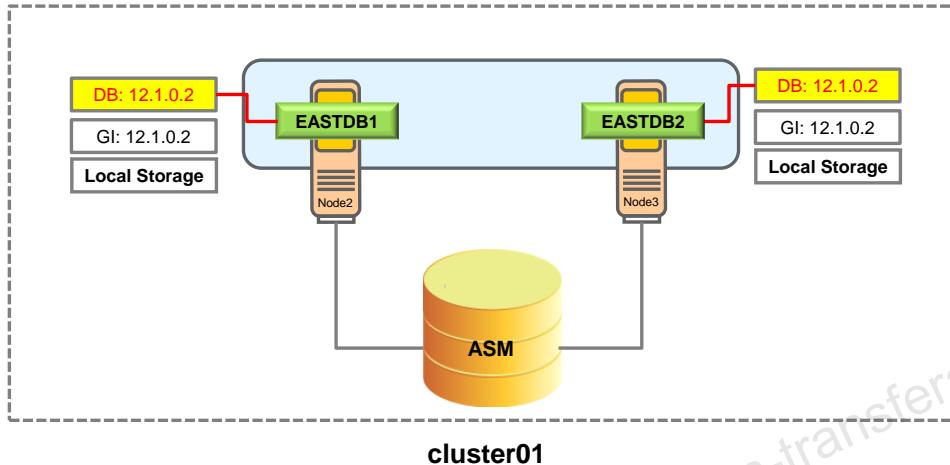


Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The database type is now RAC, which indicates a RAC database.

Example: Result of Using rconfig

A single-instance database in a *clustered environment* has been successfully converted using the `rconfig` utility.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

A single-instance database in the clustered environment has been successfully converted to an Oracle RAC database without having to move the database files.



Quiz

The RAC database software installation is initiated by executing `runInstaller` from the root directory of the Oracle Database 12c CD-ROM or from the software staging location.

- a. True
- b. False



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.



Quiz

Which of the following statements regarding:

- a. Reader nodes are instances of an Oracle RAC database that run on Leaf Nodes.
- b. Reader nodes can run on either Hub or Leaf nodes.
- c. Parallel queries running on reader nodes are not subject to brownouts that can occur when a hub node is reconfigured.
- d. Database instances on reader nodes run in read-only mode.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.



Quiz

A single-instance database can be converted to a RAC database by using (choose the correct options):

- a. rconfig
- b. netca
- c. dbca



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Install the Oracle database software
- Create a cluster database
- Configure Oracle RAC Reader Nodes
- Convert a single-instance Oracle database to RAC



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Practice 3: Overview

This practice covers the following topics:

- Installing the Oracle database software
- Creating a RAC database



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Oracle RAC Administration



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Use Enterprise Manager Cluster Database pages
- Define redo log files in a RAC environment
- Define undo tablespaces in a RAC environment
- Explain the benefits of Local Temporary Tablespaces
- Start and stop RAC databases and instances
- Modify initialization parameters in a RAC environment



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Separation of Duty for Administering Oracle Real Application Clusters

- Separation of duty best practices when administering Oracle RAC is supported by the SYSRAC administrative privilege.
- This feature eliminates the need to use the powerful SYSDBA administrative privilege for Oracle RAC.
- SYSRAC is the default mode for connecting to the database by the clusterware agent on behalf of RAC utilities, like SRVCTL.
- No SYSDBA connections to the database are necessary for everyday administration of Oracle RAC database clusters.
- Connecting as SYSRAC:

```
SQL> CONNECT / AS SYSRAC  
SQL> CONNECT /@db1 as SYSRAC  
SQL> CONNECT /@db2 as SYSRAC
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

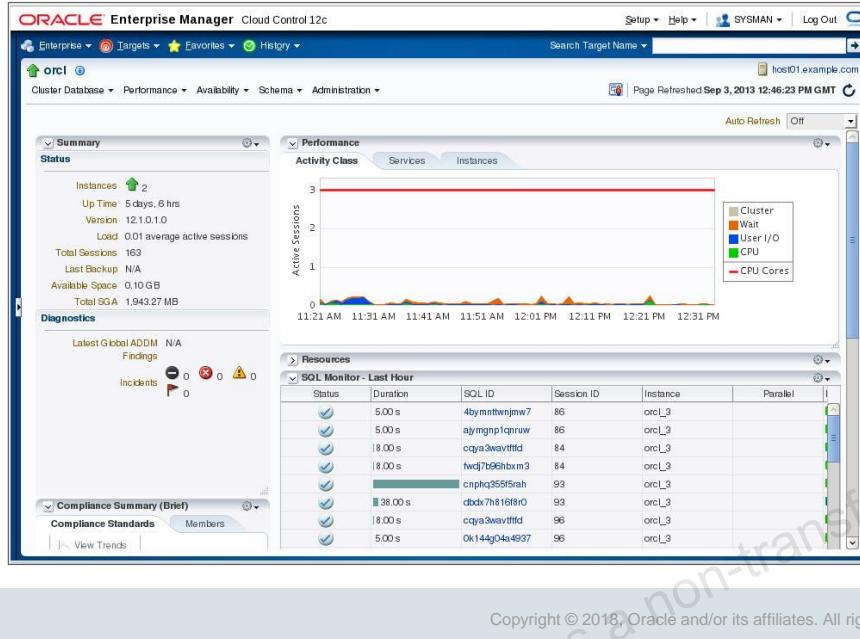
Starting with Oracle Database 12c release 2, Oracle Database provides support for separation of duty best practices when administering Oracle RAC by introducing the SYSRAC administrative privilege for the clusterware agent. This feature removes the need to use the powerful SYSDBA administrative privilege for Oracle RAC.

SYSRAC, like SYSDG, SYSBACKUP and SYSKM, helps enforce separation of duties and reduce reliance on the use of SYSDBA on production systems. This administrative privilege is the default mode for connecting to the database by the clusterware agent on behalf of the Oracle RAC utilities, such as SRVCTL.

The SYSRAC administrative privilege is the default mode of connecting to the database by the Oracle Clusterware agent on behalf of Oracle RAC utilities, such as SRVCTL, meaning that no SYSDBA connections to the database are necessary for everyday administration of Oracle RAC database clusters.

Note: The SYSRAC privilege only allows OS authentication by the Oracle agent of Oracle Clusterware. Password files and strong authentication cannot be used with the SYSRAC privilege.

Enterprise Manager Cloud Control Cluster Database Home Page



The Enterprise Manager Cloud Control (EMCC) Cluster Database Home page serves as a crossroad for managing and monitoring all aspects of your RAC database. On this page, you find Summary, Performance, Diagnostics, Compliance, Instances, and Incidents sections for information that pertains to your cluster database as a whole. Here, you can see the Summary section showing number of instances up, uptime, software version, and so on. Below the Summary section, you can find the Diagnostics pane with a link to the latest ADDM results. In the Performance section, you can browse session statistics by service or instance. In the Resources section, you can view resource usage grouped by Database or Instance. The SQL Monitor section provides a performance summary of SQL statements executed in the last hour.

The Incidents and Problems table shows all the recent alerts that are open. Click the alert message in the Message column for more information about the alert. When an alert is triggered, the name of the metric for which the alert was triggered is displayed in the Name column.

Cluster Database Home Page

The screenshot shows the Oracle Enterprise Manager Cloud Control 12c interface for a cluster database named 'orcl_3'. The top navigation bar includes links for Enterprise, Targets, Favorites, History, Setup, Help, SYSMAN, and Log Out. The page is refreshed on Sep 3, 2013 at 12:57:03 PM GMT. The main content area is divided into several sections:

- Compliance Standards**: Shows a table with columns Name and Average Score, indicating 'No data to display'.
- Members**: Displays three database members with their names, response times (8.00 s, 5.00 s, 5.00 s), and scores (96, 96, 92).
- Instances**: Lists two instances: 'orc1_orcl_3' (hosted on host01.example.com) and 'orc1_orcl_1' (hosted on host02.example.com). It also lists an ASM instance '+ASM1_host01.example.com'.
- Incidents and Problems**: A table showing zero incidents or problems found.
- Jobs Running**: A table showing zero jobs running.

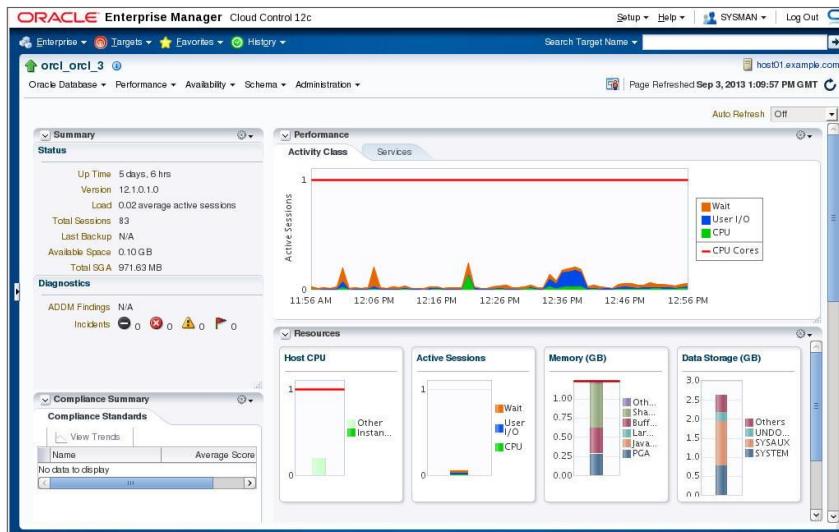
ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Scanning farther down the cluster database home page, the number of instances is displayed for the RAC database, in addition to their status. A RAC database is considered to be up if at least one instance has the database open. If your cluster database uses ASM for shared storage, the ASM instances will be listed also. Drilling down on any of the instance links will allow you to access summary information for that instance only. Clicking the hostname links will display summary information such as CPU, memory, file system, and network utilization. A hardware configuration summary for the selected host is also displayed.

A Compliance summary is displayed in the left pane of the cluster database home page. Below that section is a Jobs Running summary, allowing you to monitor running jobs at a glance.

Cluster Database Instance Home Page



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The Cluster Database Instance Home page enables you to view the current state of the instance by displaying a series of metrics that portray its overall health. This page provides a launch point for the performance, administration, and maintenance of the instance environment.

You can access the Cluster Database Instance Home page by clicking one of the instance names from the Instances section of the Cluster Database Home page. This page has basically the same sections as the Cluster Database Home page. The difference is that tasks and monitored activities from these pages apply primarily to a specific instance.

Cluster Home Page

The screenshot shows the Oracle Enterprise Manager Cluster Control 12c Cluster Home Page. At the top, there's a navigation bar with links for Enterprise, Targets, Favorites, History, Setup, Help, SYSMAN, and Log Out. A search bar for 'Search Target Name' is also present. The main content area is titled 'cluster01' under 'Cluster Administration'. It includes several sections:

- General:** Shows the cluster is Up with 3 hosts (2 green, 1 yellow), availability at 100.0% (last 24 hours), Cluster Name (cluster01), Clusterware Status (Up), Clusterware Version (12.1.0.1.0), Oracle Home (/u01/app/12.1.0/grid), Cluster Mode (Flex Cluster), and Reconfiguration Activities (11).
- Status Summary:** Details Hub Nodes (3), Clusterware on Hub Nodes (3), Listeners on Hub Nodes (1), and SCAN Listeners (7).
- Diagnostic Summary:** Shows 0 Interconnect Events.
- Resource Summary:** Shows 0 Problem Resources.
- Cluster Databases:** A table showing one database 'orcl' with status Up, 0 incidents, and 100% compliance.
- Cluster ASM:** A table showing 'No Cluster ASM.' with 0 ASM Instances and 0 ASM IO Servers.
- Incidents:** A table showing 'No Incidents.'
- Hosts:** A table listing three hosts: host01.example.com, host02.example.com, and host03.example.com, each with their status, Clusterware Status, Incidents, Compliance Score (%), and ASMI names.

ORACLE®

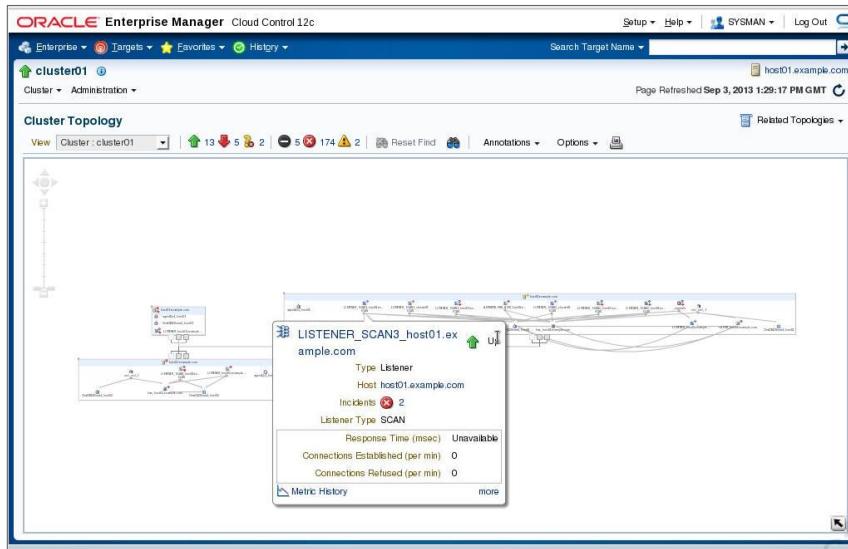
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The slide shows you the Cluster Home page, which can be accessed by clicking the Cluster link located in the Targets pull-down menu. Even if the database is down, the Cluster page is available to manage resources. The cluster is represented as a composite target composed of nodes and cluster databases. An overall summary of the cluster is provided here. The Cluster Home page displays several sections, including General, Status Summary, Diagnostic Summary, Cluster Databases, Incidents, and Hosts. The General section provides a quick view of the status of the cluster, providing basic information such as current Status, Availability, Up nodes, Clusterware Version, and Oracle Home.

The Cluster Databases table displays the cluster databases (optionally associated with corresponding services) associated with this cluster, their availability, and any alerts on those databases. The Incidents table provides information about any alerts that have been issued along with the severity rating of each.

The Hosts table displays the hosts for the cluster, availability, corresponding alerts, and CPU and memory utilization percentage.

Topology Viewer



The Topology Viewer enables you to visually see the relationships between target types for each host of your cluster database. You can zoom in or out, pan, and see selection details. These views can also be used to launch various administration functions.

The Topology Viewer populates icons on the basis of your system configuration. If a listener is serving an instance, a line connects the listener icon and the instance icon. Possible target types include:

- Interface
- Listener
- ASM Instance
- Database Instance

You can click an icon and then right-click to display a menu of available actions.

Enterprise Manager Alerts and RAC

The screenshot shows the Oracle Enterprise Manager Cloud Control 12c interface. The main window displays a grid of 'Incident Manager: All open incidents'. The columns include Severity, Summary, Target, Priority, Status, Last Updated, Owner, Ackno, Escal, Type, and Category. Most incidents are categorized as 'Error' and involve TNS-1199 or CRS-2412 errors related to listener or host issues. A sidebar on the left provides links for 'Getting Started' (Key Concepts, Setting up Notifications, Common Tasks and How To), and a footer at the bottom right contains the Oracle logo and copyright information.

Severity	Summary	Target	Priority	Status	Last Updated	Owner	Ackno	Escal	Type	Category
✖	[crs@1162]CRS-2412:The Cluster Time Synchro has _host02:None	New	Sep 3, 2013 11:41:41 PM -	No	No	Incident	Error			
✖	TNS-1199. Please check log for details.	LISTENER_1:None	New	Sep 3, 2013 12:54:19 PM -	No	No	Incident	Error		
✖	[crs@1162]CRS-2412:The Cluster Time Synchro has _host02:None	New	Sep 3, 2013 12:41:36 PM -	No	No	Incident	Error			
✖	[crs@1162]CRS-2412:The Cluster Time Synchro has _host02:None	New	Sep 3, 2013 12:11:31 PM -	No	No	Incident	Error			
✖	TNS-1199. Please check log for details.	LISTENER_1:None	New	Sep 3, 2013 11:54:02 AM -	No	No	Incident	Error		
✖	[crs@1162]CRS-2412:The Cluster Time Synchro has _host02:None	New	Sep 3, 2013 11:41:26 AM -	No	No	Incident	Error			
✖	[crs@1162]CRS-2412:The Cluster Time Synchro has _host02:None	New	Sep 3, 2013 11:11:20 AM -	No	No	Incident	Error			
✖	TNS-1199. Please check log for details.	LISTENER_1:None	New	Sep 3, 2013 11:04:53 AM -	No	No	Incident	Error		
✖	TNS-1199. Please check log for details.	LISTENER_1:None	New	Sep 3, 2013 10:54:00 AM -	No	No	Incident	Error		
✖	TNS-1199. Please check log for details.	LISTENER_1:None	New	Sep 3, 2013 10:53:57 AM -	No	No	Incident	Error		
✖	[crs@1162]CRS-2412:The Cluster Time Synchro has _host02:None	New	Sep 3, 2013 10:41:14 AM -	No	No	Incident	Error			
✖	TNS-1199. Please check log for details.	LISTENER_1:None	New	Sep 3, 2013 10:39:28 AM -	No	No	Incident	Error		

ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can use Enterprise Manager to administer alerts for RAC environments. Enterprise Manager distinguishes between database and instance-level alerts in RAC environments.

Enterprise Manager also responds to metrics from across the entire RAC database and publishes alerts when thresholds are exceeded. Enterprise Manager interprets both predefined and customized metrics. You can also copy customized metrics from one cluster database instance to another, or from one RAC database to another. A recent alert summary can be found on the Cluster Database Home page. Notice that alerts are sorted by relative time and target name.

Enterprise Manager Metrics and RAC

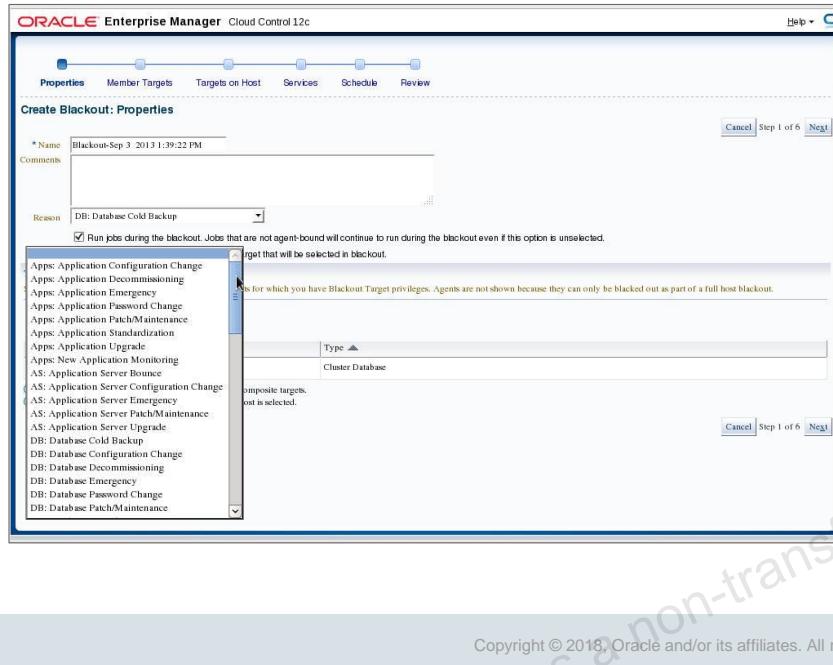
The screenshot shows the Oracle Enterprise Manager interface. In the top left, there's a navigation bar with 'Enterprise', 'Targets', and 'Favorites'. Below it, a cluster database named 'orcl' is selected. The main area is titled 'Metric and Collection Settings' for 'Database Instance: orcl_oracl_3'. It has tabs for 'Metrics' (selected) and 'Other Collected Items'. Under 'Metrics', there are two sections: 'Metrics with thresholds' and 'Metrics without thresholds'. The 'Metrics with thresholds' section is expanded, showing several alert log metrics for 'orcl_oracl_3' with their respective warning and critical thresholds, corrective actions, and collection schedules. The 'Metrics without thresholds' section is collapsed. On the right side of the interface, there are 'Cancel' and 'OK' buttons.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Alert thresholds for instance-level alerts, such as archive log alerts, can be set at the instance target level. This enables you to receive alerts for the specific instance if performance exceeds your threshold. You can also configure alerts at the database level, such as setting alerts for tablespaces. This enables you to avoid receiving duplicate alerts at each instance.

Enterprise Manager Blackouts and RAC



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

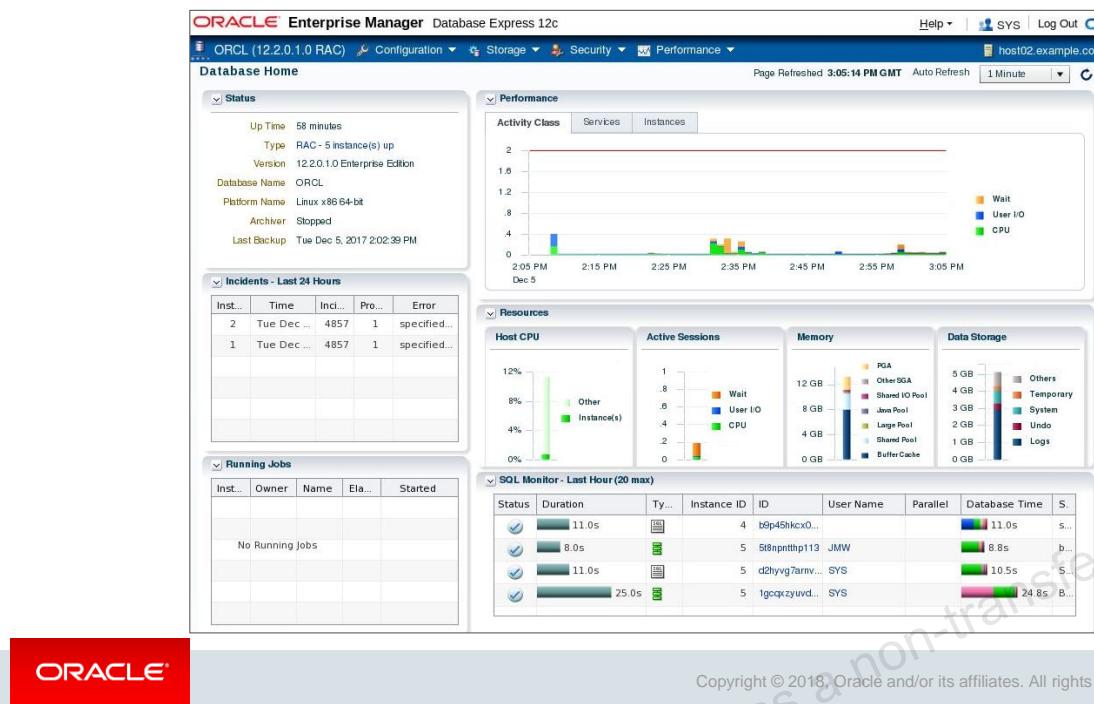
You can use Enterprise Manager to define blackouts for all managed targets of your RAC database to prevent alerts from being recorded. Blackouts are useful when performing scheduled or unscheduled maintenance or other tasks that might trigger extraneous or unwanted events. You can define blackouts for an entire cluster database or for specific cluster database instances.

To create a blackout event, select Control and then Create Blackouts from the Cluster Database pull-down menu. Click the Create button. The Create Blackout: Properties page appears. You must enter a name or tag in the Name field. If you want, you can also enter a descriptive comment in the Comments field. This is optional. Enter a reason for the blackout in the Reason field.

In the Targets area of the Properties page, you must choose a target Type from the drop-down list. Click the cluster database in the Available Targets list, and then click the Move button to move your choice to the Selected Targets list. Click the Next button to continue.

The Member Targets page appears next. Expand the Selected Composite Targets tree and ensure that all targets that must be included appear in the list. Continue and define your schedule in accordance to your needs.

Enterprise Manager Database Express



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

ORACLE®

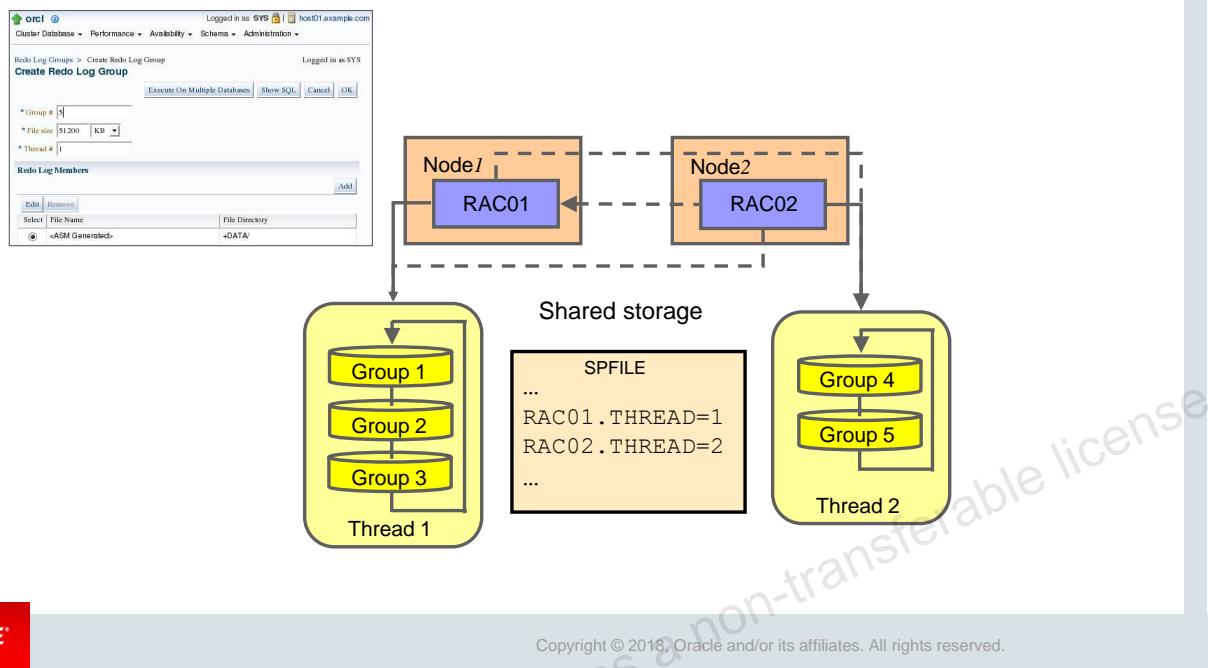
Enterprise Manager Database Express is designed to be lightweight and to incur minimal overhead on the database server. In order to achieve this goal, EM Express is built inside the Oracle Database and only uses internal infrastructure components such as XDB and SQL*Net. It does not require any separate middle-tier components. Because EM Express is built inside the database, the database has to be open in order to use EM Express, and EM Express cannot perform actions outside the database.

EM Express does not have background tasks or processes that periodically collect information. Instead, it uses data that is already collected by the database. Data is requested only when the user interacts with the UI and all UI processing is done in the browser, thus minimizing load on the database server.

EM Express is a database management tool that is automatically RAC-aware. When connected to a RAC system, the information displayed is for the entire database, that is, for all instances. For example, the average active session values is aggregated across all instances. In addition, an extra Instances tab is displayed in the Performance Region on the DB Home Page to show the distribution of average active sessions across instances.

EM Express provides support for CDB. Users can configure EM Express both at the root and the PDB containers, with each container using a different HTTPS/HTTP port.

Redo Log Files and RAC



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In an Oracle RAC database, each instance must have at least two groups of redo log files. You must allocate the redo log groups before enabling a new instance with the `ALTER DATABASE ENABLE INSTANCE instance_name` command. When you use DBCA to create the database, DBCA allocates redo log files to instances, as required, automatically. You can change the number of redo log groups and the size of the redo log files as required either during the initial database creation or as a post-creation step.

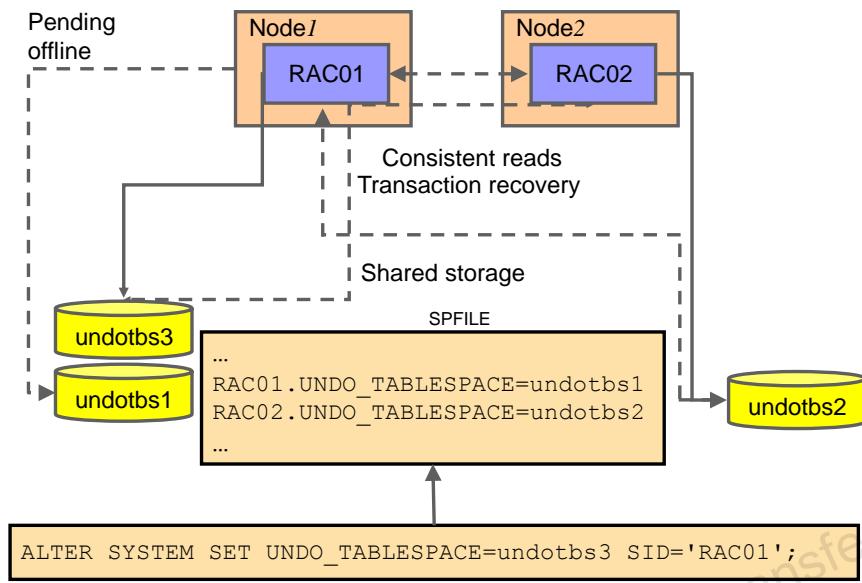
When the current group fills, an instance begins writing to the next log file group. If your database is in ARCHIVELOG mode, then each instance must save filled online log groups as archived redo log files that are tracked in the control file. During database recovery, all enabled instances are checked to see if recovery is needed. If you remove an instance from your Oracle RAC database, you should disable the instance's thread of redo so that Oracle does not have to check the thread during database recovery.

Redo log management must be considered when the number of instances for a particular production Oracle RAC database changes. For example, if you increase the cardinality of a server pool in a policy-managed database and a new server is allocated to the server pool, then Oracle Clusterware starts an instance on the new server if you have Oracle Managed Files (OMF) enabled. If the instance starts and there is no thread or redo log file available, then Oracle Clusterware automatically enables a thread of redo and allocates the associated redo log files and undo if the database uses Oracle ASM or any cluster file system.

For administrator-managed databases, each instance has its own online redo log groups. Create these redo log groups and establish group members. To add a redo log group to a specific instance, specify the `INSTANCE` clause in the `ALTER DATABASE ADD LOGFILE` statement. If you do not specify the instance when adding the redo log group, then the redo log group is added to the instance to which you are currently connected.

Each instance must have at least two groups of redo log files. You must allocate the redo log groups before enabling a new instance with the `ALTER DATABASE ENABLE INSTANCE instance_name` command. When the current group fills, an instance begins writing to the next log file group. If your database is in `ARCHIVELOG` mode, then each instance must save filled online log groups as archived redo log files that are tracked in the control file. During database recovery, all enabled instances are checked to see if recovery is needed. If you remove an instance from your Oracle RAC database, then you should disable the instance's thread of redo so that Oracle does not have to check the thread during database recovery. You can query `V$THREAD` to see the thread currently in use. Query `V$LOG` to list the redo log files with the threads they belong to.

Automatic Undo Management and RAC



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The Oracle database automatically manages undo segments within a specific undo tablespace that is assigned to an instance. Under normal circumstances, only the instance assigned to the undo tablespace can modify the contents of that tablespace. However, all instances can always read all undo blocks for consistent-read purposes. Also, any instance can update any undo tablespace during transaction recovery, as long as that undo tablespace is not currently used by another instance for undo generation or transaction recovery.

You assign undo tablespaces in your Oracle RAC administrator-managed database by specifying a different value for the `UNDO_TABLESPACE` parameter for each instance in your SPFILE or individual PFILEs. For policy-managed databases, Oracle automatically allocates the undo tablespace when the instance starts if you have Oracle Managed Files enabled. You cannot simultaneously use automatic undo management and manual undo management in an Oracle RAC database. In other words, all instances of an Oracle RAC database must operate in the same undo mode.

You can dynamically switch undo tablespace assignments by executing the `ALTER SYSTEM SET UNDO_TABLESPACE` statement. You can run this command from any instance. In the example above, the previously used undo tablespace assigned to the `RAC01` instance remains assigned to it until `RAC01`'s last active transaction commits. The pending offline tablespace may be unavailable for other instances until all transactions against that tablespace are committed. You cannot simultaneously use Automatic Undo Management (AUM) and manual undo management in a RAC database. It is highly recommended that you use AUM.

Local Temporary Tablespaces

- Local temporary tablespaces were introduced in Oracle Database 12.2.
- I/O is improved by writing spill-overs to the local disks on the reader nodes.
- Local temporary tablespaces improve temporary tablespace management in read-only instances by:
 - Storing temp files in reader node private storage to take advantage of the I/O benefits of local storage.
 - Avoiding expensive cross-instance temporary tablespace management.
 - Improving instance warm-up performance by eliminating on-disk space metadata management.
- You cannot use local temporary tablespaces to store database objects, such as tables or indexes.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To improve I/O operations, Oracle introduces a local temporary tablespace in Oracle Database 12c release 2 (12.2), so that spill-overs are written to the local disks on the reader nodes.

It is still possible for SQL operations, such as hash aggregation, sort, hash join, creation of cursor-duration temporary tables for the WITH clause, and star transformation to spill over to disk (specifically to the global temporary tablespace on shared disks). Management of the local temporary tablespace is similar to that of the existing temporary tablespace.

When many read-only instances access a single database, local temporary tablespaces improve temporary tablespace management in read-only instances by:

- Storing temp files in reader node private storage to take advantage of the I/O benefits of local storage.
- Avoiding expensive cross-instance temporary tablespace management.
- Increased addressability of temporary tablespace.
- Improving instance warm-up performance by eliminating on-disk space metadata management. It stores temp file metadata common to all instances in the control file, and instance-specific metadata (for example, the bitmaps for allocation, current temp file sizes, and file status) in the SGA.

You cannot use local temporary tablespaces to store database objects, such as tables or indexes. This same restriction is also true for space for Oracle global temporary tables.

Local Temporary Tablespace Organization

- Create local temporary tablespace using the FOR LEAF | FOR ALL extension.
- You can create local temporary tablespaces for both read-only and read-write instances.
 - LOCAL FOR LEAF creates a local temporary tablespace whose files reside on local disks for read-only instances.

```
SQL> CREATE LOCAL TEMPORARY TABLESPACE FOR LEAF local_temp_ts_leaf \
    TEMPFILE 'temp_file_leaf.dbf' SIZE 1M AUTOEXTEND ON;
```

- LOCAL FOR ALL creates a local temporary tablespace whose files reside on local disks for read-write and read-only instances.

```
SQL> CREATE LOCAL TEMPORARY TABLESPACE FOR ALL temp_ts_all \
    TEMPFILE 'temp_file_all.dbf' SIZE 5M AUTOEXTEND ON;
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Temporary Tablespace Hierarchy

The allocation of temporary space for spilling to a local temporary tablespace differs between read-only and read-write instances.

For **read-only** instances:

1. Allocate from a user's local temporary tablespace.
2. Allocate from the database default local temporary tablespace.
3. Allocate from a user's temporary tablespace.
4. Allocate from the database default temporary tablespace.

For **read-write** instances:

1. Allocate from a user's shared temporary tablespace.
2. Allocate from a user's local temporary tablespace.
3. Allocate from the database default shared temporary tablespace.
4. Allocate from the database default local temporary tablespace.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When you define local temporary tablespace and shared (existing) temporary tablespace, there is a hierarchy in which they are used. To understand the hierarchy, remember that there can be multiple shared temporary tablespaces in a database, such as the default shared temporary tablespace for the database and multiple temporary tablespaces assigned to individual users. If a user has a shared temporary tablespace assigned, then that tablespace is used first, otherwise the database default temporary tablespace is used.

Once a tablespace has been selected for spilling during query processing, there is no switching to another tablespace. For example, if a user has a shared temporary tablespace assigned and during spilling it runs out of space, then there is no switching to an alternative tablespace. The spilling, in that case, will result in an error. Additionally, remember that shared temporary tablespaces are shared among instances.

Note: In previous releases, the term *temporary tablespace* referred to what is now called a *shared temporary tablespace*.

Local Temporary Tablespace Considerations

- Instances cannot share local temporary tablespace and one instance cannot use local temporary tablespace from another.
- If an instance runs out of temporary tablespace during spilling, then the statement results in an error.
- Only one BIGFILE per local temporary tablespace is allowed.
- To address contention issues, multiple local temporary tablespaces can be assigned to different users.

```
SQL> ALTER USER MAYNARD LOCAL TEMPORARY TABLESPACE temp_ts_for_leaf;
```

- A user can have two default temporary tablespaces:
 - One local temporary when the user is connected to the read-only instance running on reader nodes.
 - One shared temporary tablespace to be used when the user is connected on the read-write instances running on a Hub Node.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Instances cannot share local temporary tablespace, hence one instance cannot take local temporary tablespace from another. If an instance runs out of temporary tablespace during spilling, then the statement results in an error.

- Local temporary tablespace support only one BIGFILE per tablespace, but the BIGFILE keyword is not required in the creation statement.
- To address contention issues arising from having only one BIGFILE-based local temporary tablespace, multiple local temporary tablespaces can be assigned to different users, as default.
- A database administrator can specify the default temporary tablespace for a user using ALTER USER syntax. For example:
`ALTER USER MAYNARD LOCAL TEMPORARY TABLESPACE temp_ts_for_leaf;`
- A user can be configured with two default temporary tablespaces:
 - One local temporary (created with the FOR LEAF option) when the user is connected to the read-only instance running on reader nodes.
 - One shared temporary tablespace to be used when the same user is connected on the read-write instances running on a Hub Node.

Managing Local Temporary Tablespaces

- Local temporary tablespaces and files are managed with:
 - ALTER TABLESPACE
 - ALTER DATABASE

- To resize a local temporary file:

```
SQL> ALTER TABLESPACE temp_ts_for_leaf RESIZE 10g;
```

- To change auto-extension attributes of a local temporary file:

```
SQL> ALTER TABLESPACE temp_ts_for_leaf AUTOEXTEND ON NEXT 20G;
```

- To decrease the size of a local temporary tablespace:

```
SQL> ALTER TABLESPACE temp_ts_for_leaf SHRINK SPACE KEEP 20M;
```

- To take a local temporary tablespace offline:

```
SQL> ALTER DATABASE TEMPFILE 'temp_file_leaf' OFFLINE;
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Local Temporary Tablespace Dictionary Views

- `USER_TABLESPACES` and `DBA_TABLESPACES` have a new column `SHARED`, indicating that the temp file is local or shared.
- `DBA_TEMP_FILES` is extended by two columns: `SHARED` and `INST_ID`.
- `DBA_TEMP_FREE_SPACE` is extended by two columns: `SHARED` and `INST_ID`.

```
SQL> select tablespace_name, shared, inst_id from DBA_TEMP_FILES;
```

TABLESPACE_NAME	SHARED	INST_ID
TEMP	SHARED	
TEMP_ALL	LOCAL_ON_ALL	1
TEMP_ALL	LOCAL_ON_ALL	2
TEMP_ALL	LOCAL_ON_ALL	3
TEMP_ALL	LOCAL_ON_ALL	4
TEMP_LEAF	LOCAL_ON_LEAF	1
TEMP_LEAF	LOCAL_ON_LEAF	2



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

All the diagnosability information related to temporary tablespaces and temporary files exposed through AWR, SQL monitor, and other utilities, is also available for local temporary tablespaces and local temporary files. This information is available with the existing dictionary views for temporary tablespaces and temporary files: `DBA_TEMP_FILES`, `DBA_TEMP_FREE_SPACE`.

The `USER_TABLESPACES` and `DBA_TABLESPACES` dictionary view are extended by a column, called `SHARED`, that indicates whether the temporary file is local (FOR LEAF or FOR ALL) or shared.

The `DBA_TEMP_FILES` dictionary view is extended by two columns: `SHARED` and `INST_ID`. The `SHARED` column indicates whether the temp file is local (FOR LEAF or FOR ALL) or shared. The `INST_ID` column contains the instance number. For shared temporary files, there is a single row per file, and the `INST_ID` is null. For local temporary files, this column contains information about temporary files per instance, such as the size of the file in bytes (`BYTES` column).

The `DBA_TEMP_FREE_SPACE` dictionary view is extended by two columns: `SHARED` and `INST_ID`. The `SHARED` column indicates whether the temporary file is local (FOR LEAF or FOR ALL) or shared. The `INST_ID` column contains the instance number. For shared temporary files, there is a single row per file, and the `INST_ID` is null. For local temporary files, this column contains information about temporary files per instance, such as total free space available (`FREE_SPACE` column).

Starting and Stopping RAC Instances

- Multiple instances can open the same database simultaneously.
- SHUTDOWN TRANSACTIONAL LOCAL does not wait for other instances' transactions to finish.
- RAC instances can be started and stopped by using:
 - Enterprise Manager
 - The Server Control (`srvctl`) utility
 - SQL*Plus
- Shutting down a RAC database means shutting down all instances accessing the database.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In a RAC environment, multiple instances can have the same RAC database open at the same time. The procedures for starting up and shutting down RAC instances are identical to the procedures used in single-instance Oracle, with the following exception:

The SHUTDOWN TRANSACTIONAL command with the LOCAL option is useful to shut down an instance after all active transactions on the instance have either committed or rolled back. Transactions on other instances do not block this operation. If you omit the LOCAL option, this operation waits until transactions on all other instances that started before the shutdown are issued either a COMMIT or a ROLLBACK.

You can start up and shut down instances by using Enterprise Manager, SQL*Plus, or Server Control (`srvctl`). Both Enterprise Manager and `srvctl` provide options to start up and shut down all the instances of a RAC database with a single step. Other Oracle utilities that can stop and start instances include RMAN and DGMGRL.

Shutting down a RAC database mounted or opened by multiple instances means that you need to shut down every instance accessing that RAC database. However, having only one instance opening the RAC database is enough to declare the RAC database open.

Starting and Stopping RAC Instances with `srvctl`

- start/stop syntax:

```
$ srvctl start|stop instance -db db_unique_name -node node_name  
-instance instance_name_list[-startoption|stopoption  
open|mount|nomount|normal|transactional|immediate|abort]
```

```
srvctl start|stop database -db <db_unique_name> -eval  
[-startoption|stopoption  
open|mount|nomount|normal|transactional|immediate|abort]
```

- Examples:

```
$ srvctl start instance -db orcl -instance orcl1,orcl2
```

```
$ srvctl stop instance -db orcl -instance orcl1,orcl2
```

```
$ srvctl start database -db orcl -startoption mount
```

```
$ srvctl start instance -db orcl -node host01  
*** This command will start a Policy-Managed database***
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The `srvctl start database` command starts a cluster database, its enabled instances, and services. The `srvctl stop database` command stops a database, its instances, and its services.

The `srvctl start instance` command starts instances of a cluster database. This command also starts all enabled and nonrunning services that have the listed instances either as preferred or as available instances.

The `srvctl stop instance` command stops instances, and all enabled and running services that have these instances as either preferred or available instances.

You must disable an object that you intend to keep stopped after you issue a `srvctl stop` command; otherwise, Oracle Clusterware can restart it as a result of another planned operation. `srvctl` does not support concurrent executions of commands on the same object. Therefore, run only one `srvctl` command at a time for each database, service, or other object. To use the `START` or `STOP` options of the `SRVCTL` command, your service must be an Oracle Clusterware–enabled, nonrunning service.

When shutting down an instance, using the `SHUTDOWN TRANSACTIONAL` command with the `LOCAL` option is useful to shut down a particular Oracle RAC database instance. Transactions on other instances do not block this operation. If you omit the `LOCAL` option, this operation waits until transactions on all other instances that started before you ran the `SHUTDOWN` command either commit or rollback, which is a valid approach, if you intend to shut down all instances of an Oracle RAC database.

Starting and Stopping RAC Instances with SQL*Plus

```
[host01] $ echo $ORACLE_SID  
orcl1  
sqlplus / as sysdba  
SQL> startup  
SQL> shutdown immediate  
  
[host02] $ echo $ORACLE_SID  
orcl2  
sqlplus / as sysdba  
SQL> startup  
SQL> shutdown immediate
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

If you want to start or stop just one instance and you are connected to your local node, you should first ensure that your current environment includes the SID for the local instance. Note that any subsequent commands in your session, whether inside or outside a SQL*Plus session, are associated with that same SID.

To start up or shut down your local instance, initiate a SQL*Plus session connected as `SYSDBA` or `SYSOPER`, and then issue the required command (for example, `STARTUP`).

You can start multiple instances from a single SQL*Plus session on one node by way of Oracle Net Services. To achieve this, you must connect to each instance by using a Net Services connection string, typically an instance-specific alias from your `tnsnames.ora` file. For example, you can use a SQL*Plus session on a local node to shut down two instances on remote nodes by connecting to each using the instance's individual alias name.

It is not possible to start up or shut down more than one instance at a time in SQL*Plus, so you cannot start or stop all the instances for a cluster database with a single SQL*Plus command. To verify that instances are running, on any node, look at `V$ACTIVE_INSTANCES`.

Note: SQL*Plus is integrated with Oracle Clusterware to make sure that corresponding resources are correctly handled when starting up and shutting down instances via SQL*Plus.

Starting and Stopping Pluggable Databases in Oracle RAC

- Manage RAC PDBs by managing services, regardless whether they are policy or administrator managed.
- Assign one database service to each PDB to coordinate start, stop, and placement of PDBs across instances.
 - Assume a CDB called `raccont` with a policy-managed PDB called `spark` in a server pool called `prod`:

```
$ srvctl add service -db raccont -pdb spark -service  
plug -serverpool prod
```

- To start and stop the PDB:

```
$ srvctl start service -db raccont -service plug
```

```
$ srvctl stop service -db raccont -service plug
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Administering an Oracle RAC-based multitenant container database (CDB) is somewhat similar to administering a non-CDB. The differences are only that some administrative tasks apply to the entire CDB, some apply only to the root, and some apply to specific pluggable databases (PDBs). Administering a PDB involves a small subset of the tasks required to administer a non-CDB. In this subset of tasks, most are the same for a PDB and a non-CDB. There are some differences, however, such as when you modify the open mode of a PDB. Also, a PDB administrator is limited to managing a single PDB and is not affected by other PDBs in the CDB.

You manage PDBs in an Oracle RAC-based CDB by managing services, regardless of whether the PDBs are policy managed or administrator managed. Assign one dynamic database service to each PDB to coordinate start, stop, and placement of PDBs across instances in a clustered container database.

For example, if you have a CDB called `raccont` with a policy-managed PDB called `spark`, which is in a server pool called `prod`, then assign a service called `plug` to this database using the following command:

```
srvctl add service -db raccont -pdb spark -service plug -serverpool prod
```

The service `plug` will be uniformly managed across all nodes in the server pool. If you want to have this service running as a singleton service in the same server pool, use the `-cardinality singleton` parameter with the preceding command.

To start the PDB `spark`, you must start the respective service, `plug`, as follows:

```
srvctl start service -db raccont -service plug
```

To stop the PDB `spark`, you must stop the respective service, `plug`, as follows:

```
srvctl stop service -db raccont -service plug
```

You can check the status of the database using the `srvctl status service` command.

Because PDBs are managed using dynamic database services, typical Oracle RAC-based management practices apply. So, if the service plug is in the `ONLINE` state when Oracle Clusterware is shut down on a server hosting this service, then the service will be restored to its original state after the restart of Oracle Clusterware on this server. This way, starting PDBs is automated as with any other Oracle RAC database.

Note: The service name associated with a PDB when it is created may not be used to stop and start the PDB.

Switch Between Automatic and Manual Policies

```
$ srvctl config database -db orcl -a
Database unique name: orcl
Database name: orcl
Oracle home: /u01/app/oracle/product/12.2.0/dbhome_1
Oracle user: oracle
Spfile: +DATA/orcl/spfileorcl.ora
Password file: +DATA/orcl/orapworcl
Domain: cluster01.example.com
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools: orcldb
Database instances:
Disk Groups:
Mount point paths:
Services:
Type: RAC
...
srvctl modify database -db orcl -policy MANUAL;
```

ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

By default, Oracle Clusterware controls database restarts in Oracle RAC environments. In some cases, you may need to minimize the level of control that Oracle Clusterware has over your Oracle RAC database, for example, during database upgrades.

To prevent Clusterware from restarting your RAC database when you restart your system, or to avoid restarting failed instances more than once, configure a management policy to define the degree of control. There are two management policies: AUTOMATIC, which is the default, and MANUAL. If the management policy is set to AUTOMATIC, the database is automatically restored to its previous running condition (started or stopped) upon restart of the database host computer. If MANUAL, the database is never automatically restarted upon restart of the database host computer. A MANUAL setting does not prevent Oracle Restart from monitoring the database while it is running and restarting it if a failure occurs.

Use the following SRVCTL command syntax to change the current management policy to either AUTOMATIC, MANUAL, or NORESTART:

```
srvctl modify database -db db_unique_name -policy [AUTOMATIC | MANUAL | NORESTART]
```

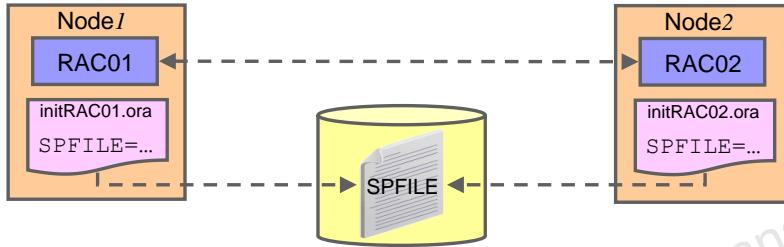
When you add a new database using the `srvctl add database` command, you can use the `-policy` parameter to specify the management policy:

```
srvctl add database -db db_unique_name -policy [AUTOMATIC | MANUAL | NORESTART] -oraclehome $ORACLE_HOME -dbname DATA
```

This command syntax places the new database under the control of Oracle Clusterware. If you do not provide a management policy option, then Oracle Database uses the default value of automatic. After you change the management policy, the Oracle Clusterware resource records the new value for the affected database.

RAC Initialization Parameter Files

- An **SPFILE** is created if you use the DBCA.
- The **SPFILE** must be created in an ASM disk group or a cluster file system file.
- All instances use the same **SPFILE**.
- If the database is created manually, create an **SPFILE** from a **PFILE**.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When you create the database, Oracle Database creates an **SPFILE** in the file location that you specify. This location can be either an Oracle ASM disk group or a cluster file system. If you manually create your database, then Oracle recommends that you create an **SPFILE** from an initialization parameter file (**PFILE**).

All instances in the cluster database use the same **SPFILE** at startup. Because the **SPFILE** is a binary file, do not directly edit the **SPFILE** with an editor. Instead, change **SPFILE** parameter settings using Oracle Enterprise Manager or **ALTER SYSTEM SQL** statements.

When creating an **SPFILE**, if you include the **FROM MEMORY** clause (for example, **CREATE PFILE FROM MEMORY** or **CREATE SPFILE FROM MEMORY**), then the **CREATE** statement creates a **PFILE** or **SPFILE** using the current system-wide parameter settings. In an Oracle RAC environment, the created file contains the parameter settings from each instance. Because the **FROM MEMORY** clause requires all other instances to send their parameter settings to the instance that is trying to create the parameter file, the total execution time depends on the number of instances, the number of parameter settings on each instance, and the amount of data for these settings.

Note: Oracle RAC uses a traditional **PFILE** only if an **SPFILE** does not exist or if you specify **PFILE** in your **STARTUP** command. Oracle recommends that you use an **SPFILE** to simplify administration, to maintain parameter setting consistency, and to guarantee parameter setting persistence across database shutdown and startup events.

SPFILE Parameter Values and RAC

- You can change parameter settings using the `ALTER SYSTEM SET` command from any instance:

```
ALTER SYSTEM SET <dpname> SCOPE=MEMORY sid='<sid|*>';
```

- SPFILE entries such as:

- `*.<pname>` apply to all instances
- `<sid>.<pname>` apply only to `<sid>`
- `<sid>.<pname>` takes precedence over `*.<pname>`

- Use current or future `*.<dpname>` settings for `<sid>`:

```
ALTER SYSTEM RESET <dpname> SCOPE=MEMORY sid='<sid>';
```

- Remove an entry from your SPFILE:

```
ALTER SYSTEM RESET <dpname> SCOPE=SPFILE sid='<sid|*>';
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can modify the value of your initialization parameters by using the `ALTER SYSTEM SET` command. This is the same as with a single-instance database except that you have the possibility to specify the `SID` clause in addition to the `SCOPE` clause.

By using the `SID` clause, you can specify the `SID` of the instance where the value takes effect. Specify `SID='*' if you want to change the value of the parameter for all instances. Specify SID='sid' if you want to change the value of the parameter only for the instance sid. This setting takes precedence over previous and subsequent ALTER SYSTEM SET statements that specify SID='*'. If the instances are started up with an SPFILE, then SID='*' is the default if you do not specify the SID clause.`

If you specify an instance other than the current instance, then a message is sent to that instance to change the parameter value in its memory if you are not using the `SPFILE` scope. The combination of `SCOPE=MEMORY` and `SID='sid'` of the `ALTER SYSTEM RESET` command allows you to override the precedence of a currently used `<sid>.<dpname>` entry. This allows for the current `*.<dpname>` entry to be used, or for the next created `*.<dpname>` entry to be taken into account on that particular SID. Using the last example, you can remove a line from your SPFILE.

In a multitenant environment, pluggable database (PDB) parameter values are inherited from the root database (CDB). If you want to change a parameter of a PDB, check the `GV$SYSTEM_PARAMETER` view.

If the `ISPDB_MODIFIABLE` value of the parameter is `TRUE`, you can change it by using the `ALTER SYSTEM SET` command.

```
SQL> ALTER SYSTEM SET CONTAINER=<PDB_NAME>
```

```
SQL> ALTER SYSTEM SET <DP_NAME>=VALUE SCOPE=MEMORY sid='<sid|*>';
```

EM and SPFILE Parameter Values

The parameter values listed here are currently used by the running instance(s). You can change static parameters in SPFILE mode.

Name	Basic	Modified	Dynamic	Category							
open	All	All	All	All							
<input type="text"/> Filter on a name or partial name:											
<input type="checkbox"/> Apply changes in current running instance(s) mode to SPFILE. For static parameters, you must restart the database.											
Select	Instance	Name	Help	Revisions	Value	Comments	Type	Basic	Modified	Dynamic	Category
<input checked="" type="radio"/>	*	open_cursors	D		300		Integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Cursors and Library Cache
<input type="radio"/>	*	open_links	D		4		Integer				Distributed, Replication and Snapshot
<input type="radio"/>	*	open_links_per_instance	D		4		Integer				Distributed, Replication and Snapshot
<input type="radio"/>	*	real_only_open_delayed	D		FALSE		Boolean				Memory
<input type="radio"/>	*	session_max_open_files	D		10		Integer				Objects and LOBs

[Execute On Multiple Databases](#) | [Show SQL](#) | [Revert](#) | [Apply](#)

ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can access the Initialization Parameters page on the Cluster Database page by clicking the Administration tab and selecting Initialization Parameters from the pull-down menu.

The Current tabbed page shows you the values currently used by the initialization parameters of all the instances accessing the RAC database. You can filter the Initialization Parameters page to show only those parameters that meet the criteria of the filter that you entered in the Name field.

The Instance column shows the instances for which the parameter has the value listed in the table. An asterisk (*) indicates that the parameter has the same value for all remaining instances of the cluster database.

Choose a parameter from the Select column and perform one of the following steps:

- Click Add to add the selected parameter to a different instance. Enter a new instance name and value in the newly created row in the table.
- Click Reset to reset the value of the selected parameter. Note that you can reset only those parameters that do not have an asterisk in the Instance column. The value of the selected column is reset to the value of the remaining instances.

Note: For both Add and Reset buttons, the `ALTER SYSTEM` command uses `SCOPE=MEMORY`.

RAC Initialization Parameters

The parameter values listed here are currently used by the running instance(s). You can change static parameters in SPFILE mode.

Name	Basic	Modified	Dynamic	Category
cluster	All	All	All	All

Apply changes in current running instance(s) mode to SPFILE. For static parameters, you must restart the database.

Add	Reset	Select	Instance	Name	Help	Revisions	Value	Comments	Type	Basic	Modified	Dynamic	Category
<input checked="" type="radio"/>		*		cluster_database	?		TRUE		Boolean	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		Cluster Database
<input type="radio"/>		*		cluster_database_instances	?		3		Integer		<input checked="" type="checkbox"/>		Cluster Database
<input type="radio"/>		*		cluster_interconnects	?				String			<input checked="" type="checkbox"/>	Cluster Database

[Execute On Multiple Databases](#) [Show SQL](#) [Revert](#) [Apply](#)

Database Identification:

<input checked="" type="radio"/>	*	db_name	?	orcl	String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Database Identification
----------------------------------	---	---------	-------------------	------	--------	-------------------------------------	-------------------------------------	-------------------------

Shared Server:

<input checked="" type="radio"/>	*	dispatchers	?	(PROTOCOL=TCP) (SERV)	String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Shared Server
----------------------------------	---	-------------	-------------------	-----------------------	--------	-------------------------------------	-------------------------------------	---------------

Miscellaneous:

<input checked="" type="radio"/>	orc_1	instance_name	?	+DATA/orcl/spfileorc1ora	String	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Instance Identification
<input type="radio"/>	orc_3	instance_name	?	orc_3	String		<input checked="" type="checkbox"/>	Instance Identification



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

CLUSTER_DATABASE: Enables a database to be started in cluster mode. Set this to TRUE.

CLUSTER_DATABASE_INSTANCES: Sets the number of instances in your RAC environment. A proper setting for this parameter can improve memory use.

CLUSTER_INTERCONNECTS: Specifies the cluster interconnect when there is more than one interconnect. Refer to your Oracle platform-specific documentation for the use of this parameter, its syntax, and its behavior. You typically do not need to set the CLUSTER_INTERCONNECTS parameter. For example, do not set this parameter for the following common configurations:

- If you have only one cluster interconnect
- If the default cluster interconnect meets the bandwidth requirements of your RAC database, which is typically the case
- If NIC bonding is being used for the interconnect
- When OIFCFG's global configuration can specify the right cluster interconnects. It only needs to be specified as an override for OIFCFG.

DB_NAME: If you set a value for DB_NAME in instance-specific parameter files, the setting must be identical for all instances.

DISPATCHERS: Set this parameter to enable a shared-server configuration, that is, a server that is configured to allow many user processes to share very few server processes.

With shared-server configurations, many user processes connect to a dispatcher. The DISPATCHERS parameter may contain many attributes. Oracle recommends that you configure at least the PROTOCOL and LISTENER attributes.

PROTOCOL specifies the network protocol for which the dispatcher process generates a listening end point. **LISTENER** specifies an alias name for the Oracle Net Services listeners. Set the alias to a name that is resolved through a naming method, such as a `tnsnames.ora` file. Other parameters that can affect RAC database configurations include:

- **ACTIVE_INSTANCE_COUNT**: This initialization parameter was deprecated in Oracle RAC 11.2. Instead, use a service with one preferred and one available instance.
- **GCS_SERVER_PROCESSES**: This static parameter specifies the initial number of server processes for an Oracle RAC instance's Global Cache Service (GCS). The GCS processes manage the routing of interinstance traffic among Oracle RAC instances. The default number of GCS server processes is calculated based on system resources with a minimum setting of 2. For systems with one CPU, there is one GCS server process. For systems with two to eight CPUs, there are two GCS server processes. For systems with more than eight CPUs, the number of GCS server processes equals the number of CPUs divided by 4, dropping any fractions. For example, if you have 10 CPUs, then 10 divided by 4 means that your system has 2 GCS processes. You can set this parameter to different values on different instances.
- **INSTANCE_NAME**: The instance's SID. The SID identifies the instance's shared memory on a host. Any alphanumeric characters can be used. The value for this parameter is automatically set to the database unique name followed by an incrementing number during the creation of the database when using DBCA.
- **RESULT_CACHE_MAX_SIZE**: In a clustered database, you can either set it to 0 on every instance to disable the result cache, or use a nonzero value on every instance to enable the result cache.
- **SERVICE_NAMES**: When you use services, Oracle recommends that you do not set a value for the **SERVICE_NAMES** parameter but instead you should create cluster managed services through the Cluster Managed Services page in EM Cloud Control. This is because Oracle Clusterware controls the setting for this parameter for the services that you create and for the default database service.
- **SPFILE**: When you use an SPFILE, all Oracle RAC database instances must use the SPFILE and the file must be on shared storage.
- **THREAD**: Specifies the number of the redo threads to be used by an instance. You can specify any available redo thread number if that thread number is enabled and is not used. If specified, this parameter must have unique values on all instances. The best practice is to use the **INSTANCE_NAME** parameter to specify redo log groups.

Parameters That Require Identical Settings

- COMPATIBLE
- CLUSTER_DATABASE
- CONTROL_FILES
- DB_BLOCK_SIZE
- DB_DOMAIN
- DB_FILES
- DB_NAME
- DB_RECOVERY_FILE_DEST
- DB_RECOVERY_FILE_DEST_SIZE
- DB_UNIQUE_NAME
- INSTANCE_TYPE (RDBMS or ASM)
- PARALLEL_EXECUTION_MESSAGE_SIZE
- REMOTE_LOGIN_PASSWORDFILE
- UNDO_MANAGEMENT



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Certain initialization parameters that are critical at database creation or that affect certain database operations must have the same value for every instance in an Oracle RAC database. Specify these parameter values in the SPFILE or in the individual PFILEs for each instance. The list in the slide above contains the parameters that must be identical on every instance.

Note: The setting for DML_LOCKS and RESULT_CACHE_MAX_SIZE must be identical on every instance only if set to zero. Disabling the result cache on some instances may lead to incorrect results.

Parameters That Require Unique Settings

Instance settings:

- INSTANCE_NAME
- INSTANCE_NUMBER
- UNDO_TABLESPACE
- CLUSTER_INTERCONNECTS
- ROLLBACK_SEGMENTS



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When it is necessary to set parameters with unique settings on a policy-managed database, you can ensure that instances always use the same name on particular nodes by running the `srvctl modify instance -n node -i instance_name` command for each server that can be assigned to the database's server pool. Then a unique value of the parameter can be specified for `instance_name` used whenever the database runs on `node_name`.

Specify the `ORACLE_SID` environment variable, which consists of the database name and the number of the `INSTANCE_NAME` assigned to the instance. Use the `CLUSTER_INTERCONNECTS` parameter to specify an alternative interconnect to the one Oracle Clusterware is using for the private network. Each instance of the RAC database gets a unique value when setting the `CLUSTER_INTERCONNECTS` initialization parameter.

Oracle Database uses the `INSTANCE_NUMBER` parameter to distinguish among instances at startup and the `INSTANCE_NAME` parameter to assign redo log groups to specific instances. The instance name can take the form `db_unique_name_instance_number` and when it has this form of name and number separated by an underscore, the number after the underscore is used as the `INSTANCE_NUMBER`. When `UNDO_TABLESPACE` is specified with automatic undo management enabled, then set this to a unique undo tablespace name for each instance.

If you use the `ROLLBACK_SEGMENTS` parameters, then Oracle recommends setting unique values for it by using the SID identifier in the SPFILE. However, you must set a unique value for `INSTANCE_NUMBER` for each instance and you cannot use a default value.

Quiescing RAC Databases

- Use the `ALTER SYSTEM QUIESCE RESTRICTED` statement from a single instance:

```
SQL> ALTER SYSTEM QUIESCE RESTRICTED;
```
- You must have the Database Resource Manager feature activated to issue the preceding statement.
- The database cannot be opened by other instances after the `ALTER SYSTEM QUIESCE...` statement starts.
- The `ALTER SYSTEM QUIESCE RESTRICTED` and `ALTER SYSTEM UNQUIESCE` statements affect all instances in a RAC environment.
- Cold backups cannot be taken when the database is in a quiesced state.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To quiesce a RAC database, use the `ALTER SYSTEM QUIESCE RESTRICTED` statement from one instance. It is not possible to open the database from any instance while the database is in the process of being quiesced from another instance. After all the non-DBA sessions become inactive, the `ALTER SYSTEM QUIESCE RESTRICTED` statement executes and the database is considered to be quiesced. In RAC, this statement affects all instances.

To issue the `ALTER SYSTEM QUIESCE RESTRICTED` statement in a RAC environment, you must have the Database Resource Manager feature activated, and it must have been activated since instance startup for all instances in the cluster database. It is through the Database Resource Manager that non-DBA sessions are prevented from becoming active. The following conditions apply to RAC:

- If you had issued the `ALTER SYSTEM QUIESCE RESTRICTED` statement, but the Oracle server has not finished processing it, then you cannot open the database.
- You cannot open the database if it is already in a quiesced state.
- The `ALTER SYSTEM QUIESCE RESTRICTED` and `ALTER SYSTEM UNQUIESCE` statements affect all instances in a RAC environment, not just the instance that issues the command.

Cold backups cannot be taken when the database is in a quiesced state because the Oracle background processes may still perform updates for internal purposes even when the database is in a quiesced state. Also, the file headers of online data files continue to appear as if they are being accessed. They do not look the same as they do when a clean shutdown is done.

Terminating Sessions on a Specific Instance

```
SQL> SELECT SID, SERIAL#, INST_ID
  2  FROM GV$SESSION WHERE USERNAME='JMW';
      SID      SERIAL#      INST_ID
-----  -----
    140        3340          2
SQL> ALTER SYSTEM KILL SESSION '140,3340,@2';
System altered.
SQL>
```

```
ALTER SYSTEM KILL SESSION '140,3340,@2'
*
ERROR at line 1:
ORA-00031: session marked for kill
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can use the `ALTER SYSTEM KILL SESSION` statement to terminate a session on a specific instance. The slide illustrates this by terminating a session started on a different instance than the one used to terminate the problematic session.

If the session is performing some activity that must be completed, such as waiting for a reply from a remote database or rolling back a transaction, then Oracle Database waits for this activity to complete, marks the session as terminated, and then returns control to you. If the waiting lasts a minute, Oracle Database marks the session to be terminated and returns control to you with a message that the session is marked to be terminated. The PMON background process then marks the session as terminated when the activity is complete.

The example above assumes that application continuity has not been configured. If application continuity has been configured and you want to terminate the session without being replayed, use the `-noreplay` option:

```
alter system kill|disconnect session 'sid, serial#, @inst' noreplay;
```

If you are terminating sessions in a multitenant environment, determine the container ID (`CON_ID`) by using `show con_id` and include it in the `GV$SESSION` query.

```
SQL> show con_id
```

```
CON_ID
```

```
-----
```

```
5
```

How SQL*Plus Commands Affect Instances

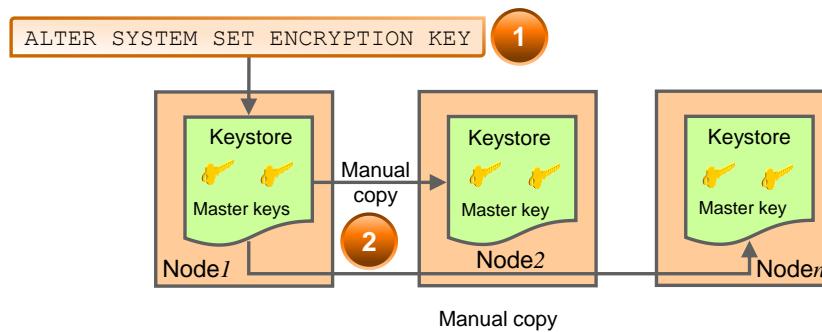
SQL*Plus Command	Associated Instance
ARCHIVE LOG	Generally affects the current instance
CONNECT	Affects the default instance if no instance is specified in the CONNECT command
RECOVER	Does not affect any particular instance, but rather the database
SHOW PARAMETER and SHOW SGA	Show the current instance parameter and SGA information
STARTUP and SHUTDOWN	Affect the current instance
SHOW INSTANCE	Displays information about the current instance



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Transparent Data Encryption and Keystores in RAC

- One wallet shared by all instances on shared storage:
 - No additional administration is required.
- One copy of the keystore on each local storage:
 - Local copies need to be synchronized each time master key is changed.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle Database enables RAC nodes to share the keystore (wallet). This eliminates the need to manually copy and synchronize the keystore across all nodes. Oracle recommends that you create the keystore on a shared file system. This allows all instances to access the same shared keystore. Oracle RAC uses keystores in the following ways:

1. Any keystore operation, such as opening or closing the keystore, performed on any one Oracle RAC instance is applicable for all other Oracle RAC instances. This means that when you open and close the keystore for one instance, then it opens and closes the keystore for all Oracle RAC instances.
2. When using a shared file system, ensure that the `ENCRYPTION_WALLET_LOCATION` parameter for all Oracle RAC instances points to the same shared keystore location. The security administrator must also ensure security of the shared keystore by assigning appropriate directory permissions.
3. A master key rekey performed on one instance is applicable for all instances. When a new Oracle RAC node comes up, it is aware of the current keystore open or close status.
4. Do not issue any keystore `ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN` or `CLOSE` SQL statements while setting up or changing the master key.

Deployments where shared storage does not exist for the keystore require that each Oracle RAC node maintain a local keystore. After you create and provision a keystore on a single node, you must copy the keystore and make it available to all of the other nodes, as follows:

- For systems using Transparent Data Encryption with encrypted keystores, you can use any standard file transport protocol, though Oracle recommends using a secured file transport.
- For systems using Transparent Data Encryption with auto-login keystores, file transport through a secured channel is recommended.

To specify the directory in which the keystore must reside, set the `ENCRYPTION_WALLET_LOCATION` parameter in the `sqlnet.ora` file. The local copies of the keystore need not be synchronized for the duration of Transparent Data Encryption usage until the server key is re-keyed through the `ADMINISTER KEY MANAGEMENT SET KEY SQL` statement. Each time you issue the `ADMINISTER KEY MANAGEMENT SET KEY` statement on a database instance, you must again copy the keystore residing on that node and make it available to all of the other nodes. Then, you must close and reopen the keystore on each of the nodes. To avoid unnecessary administrative overhead, reserve re-keying for exceptional cases where you believe that the server master key may have been compromised and that not re-keying it could cause a serious security problem.

Note: If Oracle Automatic Storage Management Cluster File System (Oracle ACFS) is available for your operating system, then Oracle recommends that you store the keystore in Oracle ACFS.

Quiz



If an instance starts in a policy-managed RAC environment and no thread or redo log file is available, then Oracle Clusterware automatically enables a thread of redo and allocates the redo log files and undo if the database uses Oracle ASM or any cluster file system and OMF is enabled.

- a. True
- b. False



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Quiz



Local temporary tablespaces and files are managed with the ALTER TABLESPACE or ALTER DATABASE SQL commands.

- a. True
- b. False



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.



Quiz

Which of the following statements is not true?

- a. Multiple instances can open the same database simultaneously.
- b. Shutting down one instance does not interfere with other running instances.
- c. SHUTDOWN TRANSACTIONAL LOCAL will wait for other instances' transactions to finish.
- d. Shutting down a RAC database means shutting down all instances accessing the database.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Use Enterprise Manager Cluster Database pages
- Define redo log files in a RAC environment
- Define undo tablespaces in a RAC environment
- Explain the benefits of Local Temporary Tablespaces
- Start and stop RAC databases and instances
- Modify initialization parameters in a RAC environment



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Practice 4: Overview

This practice covers the following topics:

- Using operating system and password file authenticated connections
- Using Oracle Database authenticated connections
- Stopping a complete ORACLE_HOME component stack
- Configuring Leaf Nodes to run Read-Only Instances
- Working with Temporary Tablespaces



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Upgrading and Patching Oracle RAC



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the types of patches available
- Plan for rolling patches and rolling upgrades
- Install a patch set with the Oracle Universal Installer (OUI) utility
- Install a patch with the `opatch` utility



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Patch and Patch Set: Overview

- Oracle issues product fixes for its software called patches.
- Patches are associated with particular releases and versions of Oracle products.
- The patching cycle involves downloading patches, applying patches, and verifying the applied patch.
- Patching involves migrating from one version of the software product to another, within a particular release.
- When a patch is applied to an Oracle software installation, it updates the executable files, libraries, and object files in the software home directory.
 - The patch application can also update configuration files and Oracle-supplied SQL schemas.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle issues product fixes for its software called patches. They are associated with particular releases and versions of Oracle products. The patching cycle involves downloading patches, applying patches, and verifying the applied patch to ensure that the bug fixes present in the patch reflect appropriately.

Patching involves migrating from one version of the software product to another, within a particular release, unlike upgrading which involves moving from one release of a product to another newer release of the software product. When you apply the patch to your Oracle software installation, it updates the executable files, libraries, and object files in the software home directory. The patch application can also update configuration files and Oracle-supplied SQL schemas.

Types of Patches

Patch Type	Description
Interim Patches	Released to fix a bug, or a collection of bugs. Previously called patch set exceptions (PSE), one-off patches, or hot fixes.
Interim Patches (for Security bug fixes)	Released to provide customer-specific security fixes. Previously referred to as a test patch, fix verification binary, or e-fix.
Diagnostic Patches	Mainly help diagnose and verify a fix, or a collection of bugfixes
Bundle Patch Updates	Cumulative collection of fixes for a specific product or component. Previously referred to as a maintenance pack, service pack, cumulative patch, update release, or MLR.
Patch Set Updates (PSU)	Cumulative patch bundles that contain well-tested and proven bug fixes for critical issues. PSUs have limited new content, and do not include any changes that require re-certification.
Security Patch Updates	A cumulative collection of security bug fixes. Previously known as Critical Patch Updates, or CPUs.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Interim patches are bug fixes available to customers in response to specific bugs. They require a particular base release or patch set to be installed before you can apply them. These patches are not versioned and are generally available in a future patch set as well as the next product release. Interim patches are applied by using Enterprise Manager Cloud Control or OPatch, which is included with your Oracle Database installation.

Patch sets updates (PSUs) and patch bundles are mechanisms for delivering fully tested and integrated product fixes. All the fixes in a patch set have been tested and are certified to work with each other. Because a patch set includes only low impact patches, it does not require you to certify applications or tools against the updated Oracle Database software. When you apply a patch set, many different files and utilities are modified. This results in a release number change for your Oracle software. You use OPatch to apply PSUs and Oracle Universal Installer (OUI) to install patch sets. PSUs are rolling installable but patch sets are not.

Obtaining Oracle RAC Patch Sets

The screenshot shows the Oracle My Oracle Support interface. The top navigation bar includes links for Dashboard, Knowledge, Service Requests, Patches & Updates (which is currently selected), Community, Certifications, Managed Cloud, CRM On Demand, and Sy. Below the navigation is a search bar with dropdowns for Patching Quick Links and Patch Search. The main content area is titled 'Patches and Updates' and features a 'Quick Links' sidebar with sections for Software and Patch Search Sites, Oracle E-Business Suite, and Oracle Server and Tools. Under Oracle Server and Tools, a link to 'Latest Patchsets' is highlighted with a red box and an arrow pointing to the 'Latest Oracle Server/Tools Patches' section on the right. This section contains two tables: 'Patch Sets for Product Bundles' and 'Patch Bundles for Individual Products or Components'. In the 'Patch Sets for Product Bundles' table, 'Oracle Database' is highlighted with a red box. The copyright notice at the bottom right reads: 'Copyright © 2018, Oracle and/or its affiliates. All rights reserved.'

The latest patch sets and recommended BPs can be downloaded from the My Oracle Support website at the following URL:

<http://support.oracle.com/>

After signing in to the website, click the “Patches & Updates” tab. For the latest patch sets and patch bundles, click the “Latest Patchsets” link under Oracle Servers and Tools. You can choose from patch sets for product bundles or patch bundles for individual products.

If you know the patch set number, you can click the Number/Name Sun CR ID link. You can enter a single patch set number or a comma-separated list. Select your platform and click the Search button.

To locate the Patch Set notes on My Oracle Support:

1. Log in to My Oracle Support.
2. Select the Patches & Updates tab.
3. Select Quick Links to the Latest Patchsets, Mini Packs, and Maintenance Packs.
4. Under the heading Latest Oracle Server/Tools Patchsets, select Oracle Database.
A list of operating systems appears.
5. Place your cursor over the entry that matches your operating system, or use the triangular arrows to search for your operating system.
When you place the cursor over the entry for your operating system, for example, Linux x86, a list of database versions appears.
6. Select 12.2.0
The Advanced Search page appears.
7. Scroll to the bottom of this page to see the list of available patch sets.
8. Select the number in the Patch column for the patch set you want to view or download.
The Patchset description and download page appears.
9. Click View Readme to see the patch set notes.
On this page you can also click Download to download the patch to your computer.
10. If you choose to download the patch, then follow the instructions in the ReadMe file of the patch set to apply the patch set to your software.

Obtaining Oracle RAC Patches

The screenshot shows the Oracle Patch Search interface. A red box highlights the 'Search' button, which has a red arrow pointing down to the search results page. The search filters are set to Product: Oracle Real Application Clusters, Release: 12.2.0.1.0, and Platform: Linux x86-64. The results table displays two patches:

Patch Name	Description	Release	Platform (Language)	Recommended
26737266	GRID INFRASTRUCTURE RELEASE UPDATE 12.2.0.1.171017 (System Patch)	12.2.0.1.0	Linux x86-64 (American English)	
26878187	GRID INFRASTRUCTURE RELEASE UPDATE REVISION 12.2.0.1.171017 (System Patch)	12.2.0.1.0	Linux x86-64 (American English)	



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You obtain patches and patch sets from My Oracle Support, which is the Oracle Support Services website. The Oracle Support Services website is located at: <https://support.oracle.com>

To locate patches on the My Oracle Support website:

1. Log in to your account on My Oracle Support.
2. Select the **Patches & Updates** tab.
3. If you know the patch number, then you can enter it into the Patch Name or Number field, then click **Search**.

If you want to search for all available patches for your system, then select **Product or Family (Advanced Search)**, which is located above the Patch Name or Number field. Supply the following information:

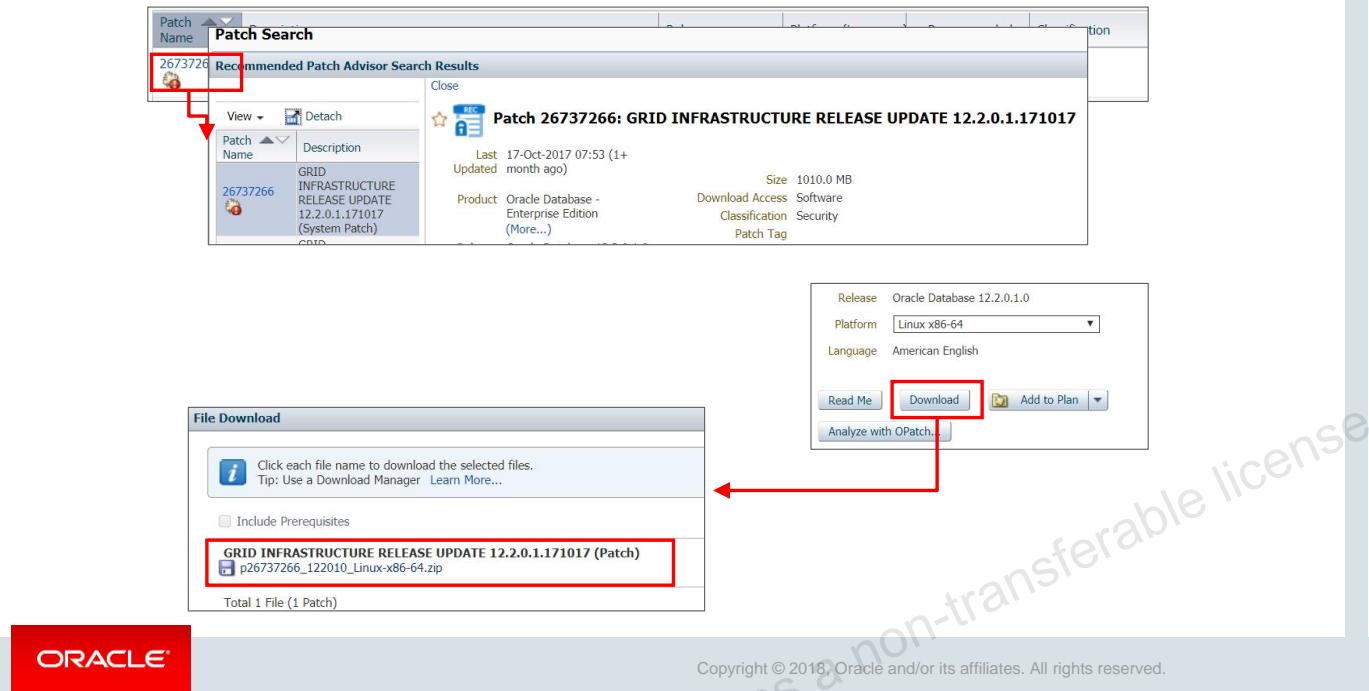
- Choose the products you want to patch (for example, Oracle Clusterware, Oracle Database, or an individual product such as Universal Installer)
- Specify the software release for the products you selected, for example, Oracle 12.2.0.1.0.
- Specify the platform on which the software is installed.

Click **Search** to look for available patches. The Patch Search Results page is displayed.

4. On the Patch Search Results page, select the number of the patch you want to download. A details page for that patch appears on your screen.
5. Click the **ReadMe** button to view the ReadMe file for the patch, or click **Download** to download the patch to your local computer.

Note: To locate recommended patches, start from the “Patches & Updates” tab and click the Recommended Patch Advisor link. Select Oracle Clusterware from the Product pull-down menu, and then select the release number and platform. Click the Search button for a list of recommended patches

Downloading Patches



Once you have located the patch you need, you can download it. Click the patch link on the search results page. Locate and click the Download link on the patch summary page, and then click the patch link in the File Download dialog box. Click the Save File button, and then click OK. Specify the directory location for the file, and then click Save.

RAC Patching methods

- OUI Installs *patch sets* as out-of-place upgrades, reducing the down time required for patching.
- OPatch supports 3 patch methods on a RAC environment
 - *All-Node Patch*: Patching RAC as a single instance
 - *Minimum Downtime Patch*: Patching RAC using a minimum down-time strategy
 - *Rolling Patch*: Patching RAC using a rolling strategy. Rolling patching strategy incur no downtime, however, some rolling patches may incur downtime due to post-installation steps. Please refer to patch readme to find out whether post-installation steps requires downtime or not.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Patch sets are now installed as out-of-place upgrades to the Grid Infrastructure software (Oracle Clusterware and Automatic Storage Management) and Oracle Database. This reduces the down time required for planned outages for patching.

OPatch supports 3 different patch methods on a RAC environment:

- Patching RAC as a single instance (*All-Node Patch*): In this mode, OPatch applies the patch to the local node first, then propagates the patch to all the other nodes, and finally updates the inventory. All instances must be down during the whole patching process.
- Patching RAC using a minimum down-time strategy (*Min. Downtime Patch*): In this mode, OPatch patches the local node, asks users for a sub-set of nodes, which will be the first subset of nodes to be patched. After the initial subset of nodes are patched, Opatch propagates the patch to the other nodes and finally updates the inventory. The downtime would happen between the shutdown of the second subset of nodes and the startup of the initial subset of nodes patched.
- Patching RAC using a rolling strategy - No down time (*Rolling Patch*): With this method, there is no downtime. Each node would be patched and brought up while all the other nodes are up and running, resulting in no disruption of the system. Rolling patching strategy incur no downtime, however, some rolling patches may incur downtime due to post-installation steps, i.e. running sql scripts to patch the actual database. Please refer to patch readme to find out whether post-installation steps requires downtime or not.

Out-of-Place Upgrades with OUI

- You must install the software for the new Oracle Database release before you can perform the upgrade.
- The software is installed to a new Oracle Home by using OUI.
- The Database Upgrade Assistant is used to finish the upgrade process.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Out-of-Place Database Upgrade with OUI

You must install the software for the new Oracle Database release before you can perform the upgrade of Oracle Database. The installation procedure installs the Oracle software into a new Oracle home. This is referred to as an out-of-place upgrade and is different from patch set releases for earlier releases of Oracle Database, where the patch set was always installed in place. The new Oracle Database software is installed by using the OUI. The upgrade process is completed by the Oracle Database Upgrade Assistant.

Out-of-Place Upgrades with OUI

1. Upgrade Grid Infrastructure first, if necessary.
2. Follow the instructions in your Oracle OS-specific documentation to prepare for installation of Oracle Database software.
3. Use OUI to install the Oracle Database software.
4. Execute the Pre-Upgrade Information Tool and correct any reported deficiencies.

```
$ cd /u01/app/oracle/product/12.2.0/dbhome_1/rdbms/admin
$ $<Earlier_Release_ORACLE_HOME>/jdk/bin/java -jar
preupgrade.jar [FILE|TERMINAL] [TEXT|XML] [DIR output_dir]
```

5. Run the `root.sh` script as directed by OUI.
6. Finish the upgrade process with DBUA.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

If you are upgrading an Oracle RAC database, then you must perform the following steps:

1. Upgrade Grid Infrastructure on your cluster, if necessary.
2. Follow the instructions in your Oracle operating system–specific documentation to prepare for installation of Oracle Database software.
3. Start the OUI. Select “Upgrade an existing database” on the Select Installation Option page.
4. It is recommended that you run the Pre-Upgrade Information Tool before you upgrade using DBUA, so that you can preview the types of items DBUA checks. The OUI will launch the DBUA when the root scripts have been executed, but you can elect to run DBUA at a later time. Oracle Database 12c release 2 introduces the `preupgrade.jar` Pre-Upgrade Information Tool. You can run the tool from the operating system command line. In previous Oracle Database releases, the Pre-Upgrade Information Tool was run within SQL*Plus as a SQL file.

If you do not specify an output directory with `DIR`, but you defined `ORACLE_BASE`, the generated scripts and log files are created in `ORACLE_BASE/cfgtoollogs/`.

If you do not specify an output directory and `ORACLE_BASE` is not defined, the generated scripts and log files are created in `ORACLE_HOME/cfgtoollogs/`.

Example: Non-CDB In the Source Oracle Home Example

- Set your user environment variables to point to the earlier release Oracle home.
\$ export ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1
\$ export ORACLE_BASE=/u01/app/oracle
\$ export ORACLE_SID=sales01
\$ export PATH=.::\$ORACLE_HOME/bin:\$PATH
- Run the new release Oracle Database Pre-Upgrade Information Tool on the earlier release Oracle Database server (12.2), using the environment settings you have set to the earlier release Oracle home.
\$ORACLE_HOME/jdk/bin/java -jar /u01/app/oracle/product/12.2.0/rdbms/admin/preupgrade.jar TERMINAL TEXT

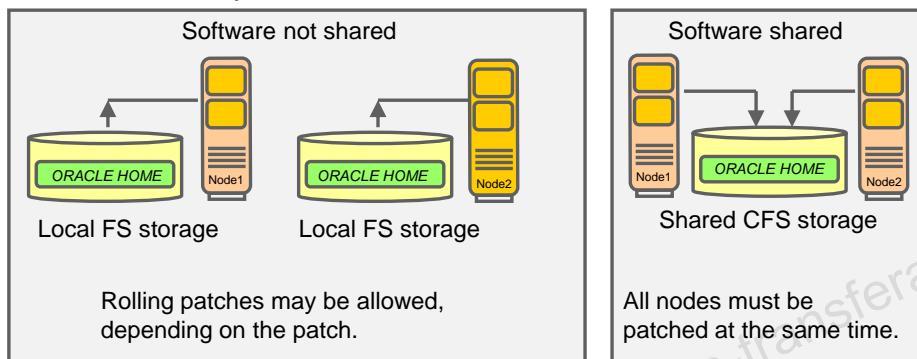
Example: CDB in a Source Oracle Home

- Open all the pluggable databases
 - Set your user environment variables to point to the earlier release Oracle home.
\$ export ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1
\$ export ORACLE_BASE=/u01/app/oracle
\$ export ORACLE_SID=sales01
\$ export PATH=.::\$ORACLE_HOME/bin:\$PATH
 - Run the Pre-Upgrade Information Tool with an inclusion list, using the -c option. In this example, the inclusion list is PDB1 and PDB2, and the command is run on a Linux or UNIX system. The output of the command is displayed to the terminal, and the output is displayed as text.
\$ORACLE_HOME/jdk/bin/java -jar /u01/app/oracle/product/12.2.0/rdbms/admin/preupgrade.jar TERMINAL TEXT -c 'pdb1 pdb2'
5. Run the root.sh script as directed by the OUI.
 6. If you are ready, use DBUA to complete the upgrade process.

Rolling Patches

A rolling patch allows one node at a time to be patched, while other nodes continue to provide service. It:

- Requires distinct software homes for each node
- Allows different versions to coexist temporarily
- May not be available for all patches



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

A rolling patch allows one node to be patched to the latest version, while other nodes continue to use the older version and provide business service. This methodology is best suited when you are applying one-off patches that support rolling methodology, maintaining high availability of your targets, so when one node is being patched, the other nodes are available for service.

Rolling patches are enabled by using locally accessible, nonshared file system to store the software files. Rolling patches cannot be done when the Oracle software files are stored in a shared cluster file system in which a single copy of the software is shared among all nodes. A single copy requires much less disk space and a single copy to patch or upgrade. However, to patch or upgrade, the software must be stopped. Stopping the Oracle Clusterware software also requires all databases, applications, and services that depend on Oracle Clusterware to be stopped. This technique requires a complete outage with down time to patch or upgrade.

Out-of-place rolling patching involves installing the patch in a new home, modifying the Oracle home of the database, and then restarting instances in a rolling fashion.

Note: A patch set that can be rolled for the clusterware may not be able to be rolled for the RDBMS.

OPatch: Overview

- OPatch is a utility that assists you with the process of applying interim patches to Oracle software.
- OPatch is a Java-based utility that allows the application and rolling back of interim patches.
- OPatch is included with the Oracle Clusterware 12c installation.
- For large IT environments, EM Cloud Control's patch automation capability can simplify the patching process.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

OPatch is an Oracle-supplied utility that assists you with the process of applying interim patches to Oracle's software. Opatch is a Java-based utility that can run on either OUI-based Oracle homes or standalone homes. It works on all operating systems for which Oracle releases software. OPatch is included with the Oracle Clusterware 12c installation.

For large-scale IT environments, patching individual GI/RAC/DB homes may not be practical because patching large numbers of targets manually is both monotonous and error prone. To maintain and deploy Oracle patches across many targets across your organization, you can use Enterprise Manager Cloud Control's patch automation capability.

OPatch: General Usage

- To define the `ORACLE_HOME` or `-oh` option on all commands:

```
$ export ORACLE_HOME=/u01/app/oracle/product/12.2.0/dbhome_1  
$ opatch command [options]
```

Or

```
$ opatch command -oh /u01/app/oracle/product/12.2.0/dbhome_1 [options]
```

- To obtain help with the OPatch syntax:

```
$ opatch command -help
```

- To check whether a patch supports a rolling application (Run from the `patch` directory):

```
$ opatch query -is_rolling_patch [unzipped patch location]
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The `opatch` utility requires that the `ORACLE_HOME` environment variable be defined or that the value of `ORACLE_HOME` be passed as an argument on the command line with the `-oh` option. For the case of Oracle RAC, `ORACLE_HOME` refers to the installation directory for Oracle RAC, not the location of other Oracle products that may be installed. In general, `ORACLE_HOME` refers to the home for the product to be patched.

The OPatch documentation can be found in the `$ORACLE_HOME/OPatch/docs` directory. The utility contains help for its syntax by using the `-help` option as follows:

```
opatch -help  
opatch apply -help  
opatch lsinventory -help  
opatch rollback -help  
opatch prereq -help  
opatch util -help
```

In general, RAC patches can be applied in a rolling fashion—that is, one node at a time. However, it is still important to check each patch for exceptions to this rule. To verify that a patch supports rolling applications, unzip the downloaded patch into a directory of your choosing and, from that directory, issue the following command:

```
$ORACLE_HOME/OPatch/opatch query -is_rolling_patch <patch_location>
```

Before Patching with OPatch

- Check the current setting of the `ORACLE_HOME` variable.
- Back up the directory being patched with an OS utility or Oracle Secure backup.
- Stage the patch to each node.
- Update the `PATH` environment variable for the `OPatch` directory.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The Oracle Patching utility, OPatch, verifies that the `ORACLE_HOME` environment variable names an actual directory. You should verify that the `ORACLE_HOME` variable is set to the Oracle home of the product you are trying to patch.

It is best practice to back up the software directory you are patching before performing any patch operation. This applies to Oracle RAC, ASM, or Oracle Clusterware software installation directories. The backup should include the Oracle Inventory directory as well.

If you manually download the patch and use OPatch to install the patch, you must stage the patch on each node. If you use Enterprise Manager to download the patch and you selected all the nodes in your cluster as targets for the patch, then the patch is automatically staged on those nodes.

The `opatch` binary file is located in the `$ORACLE_HOME/OPatch` directory. You can either specify this path when executing OPatch or update the `PATH` environment variable to include the `OPatch` directory. To change the `PATH` variable on Linux, use:

```
$ export PATH=$PATH:$ORACLE_HOME/OPatch
```

Installing a Rolling Patch with OPatch

1. Verify that Oracle Inventory is properly configured.

```
[oracle]$ opatch lsinventory
```

2. Determine if any installed interim patches conflict with a patch to be installed.

```
[oracle]$ opatch prereq CheckConflictAgainstOHWithDetail -ph ./
```

3. Shutdown the instance running on the local node.

```
[oracle]$ srvctl stop instance -db orcl -instance orcl_3
```

4. Start the patch installation in rolling fashion.

```
[oracle]$ opatch apply
```

5. Verify patch installation

```
[oracle]$ opatch lsinventory
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The slide shows an example of installing a rolling patch with OPatch.

1. Assume that you have set ORACLE_HOME and PATH for OPatch. Verify that Oracle Inventory is properly configured.

```
[oracle@host01 ~]$ opatch lsinventory
```

2. Determine whether any currently installed interim patches conflict with patch to be installed.

```
[oracle@host01 <patch directory>]$ opatch prereq  
CheckConflictAgainstOHWithDetail -ph ./
```

3. Shutdown the instance running on the local node (host01)

```
[oracle@host01 ~]$ srvctl status database -db orcl  
Instance orcl_1 is running on node host02  
Instance orcl_2 is running on node host03  
Instance orcl_3 is running on node host01
```

```
[oracle@host01 ~]$ srvctl stop instance -db orcl -instance orcl_3
```

4. Start the patch installation. When OPatch asks “Is the local system ready for patching?”, answer yes by typing Y and pressing Enter.

```
[oracle@host01 <patch directory>]$ opatch apply
```

When patching is finished on host01, the OPatch dialog will inform you that the instance can be restarted on host01 and will prompt you for the name of the next node to patch.

```
[oracle@host01 <patch directory>]$ opatch apply  
...  
The local system has been patched. You can restart Oracle instances on it.  
Patching in rolling mode.
```

Remaining nodes to be patched:
'host02' 'host03'

```
[oracle@host01 ~]$ srvctl start instance -db orcl -instance orcl_3  
[oracle@host01 ~]$ srvctl stop instance -db orcl -instance orcl_1
```

```
[oracle@host01 <patch directory>]$ opatch apply
```

What is the next node to be patched?
host02

...
Is the node ready for patching? [y|n]

y

...
The node 'host02' has been patched. You can restart Oracle instances on it.
...
Please shutdown Oracle instances running out of this ORACLE_HOME on 'host03'.
(Oracle Home = '/u01/app/oracle/product/12.2.0/dbhome_1')

```
[oracle@host01 ~]$ srvctl start instance -db orcl -instance orcl_1  
[oracle@host01 ~]$ srvctl stop instance -db orcl -instance orcl_2  
[oracle@host01 <patch directory>]$ opatch apply
```

Is the node ready for patching? [y|n]
y

...
The node 'host03' has been patched. You can restart Oracle instances on it.
...

OPatch succeeded.

```
[oracle@host01 ~]$ srvctl start instance -db orcl -instance orcl_2
```

Ensure the Oracle homes on all three nodes were patched successfully.

```
[oracle@host01 ~]$ opatch lsinventory
```

OPatch Automation

- OPatch has automated patch application for the Oracle Grid Infrastructure and Oracle RAC database homes.
- Existing configurations are queried and the steps required for patching each Oracle RAC database home of the same version and the Grid home are automated.
- The utility must be executed by an operating system user with root privileges.
- OPatch must be executed on each node in the cluster if the Grid home or RAC home is in non-shared storage.
- One invocation of OPatch can patch the Grid home, one or more RAC homes, or both Grid and Oracle RAC database homes of the same Oracle release version.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

OPatch Automation: Examples

- To patch Grid home and all Oracle RAC database homes of the same version:

```
# opatchauto apply <UNZIPPED_PATCH_LOCATION> <Grid_home>
-ocmrfs <ocm_response_file>
```

- To patch only the GI home:

```
# opatchauto apply <UNZIPPED_PATCH_LOCATION> -oh
<Grid_home> -ocmrfs <ocm_response_file>
```

- To patch one or more Oracle RAC database homes:

```
# opatchauto apply <UNZIPPED_PATCH_LOCATION> -database
db1, db2 -ocmrfs <ocm_response_file>
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

If you have not installed Oracle Configuration Manager (OCM), the OPatch utility will prompt for your OCM response file when it is run. You should enter a complete path of OCM response file if you already have created this in your environment. If you do not have the OCM response file (`ocm.rsp`), you should run the `emocmrsp` command to create it. As the software home owner, execute:

```
$ <ORACLE_HOME>/OPatch/ocm/bin/emocmrsp
```

Before executing OPatch, add the directory containing OPatch to the your path:

```
# export PATH=$PATH:<GI_HOME>/Opatch
```

To patch GI home and all Oracle RAC database homes of the same version:

```
# <Grid_home>/OPatch/opatchauto apply <Grid_home> -ocmrfs <ocm_response_file>
```

To patch only the GI home:

```
# <Grid_home>/OPatch/opatchauto apply <SYSTEM_PATCH_TOP_DIR> -oh <Grid_home> -ocmrfs
<ocm_response_file>
```

To patch one or more Oracle RAC databases and associated GI/RAC homes:

```
# opatchauto apply <UNZIPPED_PATCH_LOCATION> -database db1, db2 -ocmrfs
<ocm_response_file>
```

To roll back the patch from the GI home and each Oracle RAC database home:

```
# opatchauto rollback <UNZIPPED_PATCH_LOCATION> -ocmrfs <ocm_response_file>
```

To roll back the patch from the GI home:

```
# opatchauto rollback <UNZIPPED_PATCH_LOCATION> -oh <Grid_home> -ocmrfs
<ocm_response_file>
```

To roll back the patch from one or more Oracle RAC database homes:

```
# opatchauto rollback <UNZIPPED_PATCH_LOCATION> -database db1, db2 -ocmrfs
<ocm_response_file>
```

OPatch Log and Trace Files

- OPatch maintains logs for apply, rollback, and lsinventory operations.
- OPatch Log files are located in `ORACLE_HOME/cfgtoollogs/opatch`
- Each log file is tagged with the time stamp of the operation.
- Each time you run OPatch, a new log file is created.
- OPatch maintains an index of processed commands and log files in the `opatch_history.txt` file.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Logging and tracing is a common aid in debugging. OPatch maintains logs for apply, rollback, and lsinventory operations. Log files are located in `Oracle_home/cfgtoollogs/opatch`. Each log file is tagged with the time stamp of the operation. Log files are named as `opatch_mm-dd-yyyy_hh-mm-ss.log`, where `mm-dd-yyyy` is the current date and `hh-mm-ss` is the current time. Each time you run OPatch, a new log file is created.

For example, if a log file is created on May 17, 2016 at 11:55 PM, then it is named as follows:

`opatch_05-17-2016_23-55-00.log`

OPatch also maintains an index of the commands processed by OPatch and the log files associated with it in the `opatch_history.txt` file located in the `Oracle_home/cfgtoollogs/opatch` directory. A sample of the `opatch_history.txt` file is as follows:

```
Date & Time : Thu Jun 09 22:07:45 MDT 2016 Oracle Home :  
/u01/app/oracle/product/12.2.0/dbhome_1 OPatch Ver. : 12.2.0.1.0 Current Dir :  
/u01/app/oracle/product/12.2.0/dbhome_1 Command : lsinventory -xml  
/u01/app/oracle/product/12.2.0/dbhome_1 ... Log File :  
/u01/app/oracle/product/12.2.0/dbhome_1/cfgtoollogs/opatch/opatch2016-06-09_22-  
07-57PM_1.log
```

Queryable Patch Inventory

- The `DBMS_QOPATCH` package provides a PL/SQL or SQL interface to view the database patches that are installed.
- The interface provides all the patch information available as part of the OPatch `lsinventory -xml` command.
- The package accesses the OUI patch inventory in real time to provide patch and patch meta information.
- The `DBMS_QOPATCH` package allows users to:
 - Query what patches are installed from SQL*Plus
 - Write wrapper programs to create reports and do validation checks across multiple environments
 - Check patches installed on cluster nodes from a single location



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Using `DBMS_QOPATCH`, Oracle Database 12c provides a PL/SQL or SQL interface to view the database patches that are installed. The interface provides all the patch information available as part of the OPatch `lsinventory -xml` command. The package accesses the OUI patch inventory in real time to provide patch and patch meta information. Using this feature, users can:

- Query what patches are installed from SQL*Plus
- Write wrapper programs to create reports and do validation checks across multiple environments
- Check patches installed on cluster nodes from a single location instead of having to log onto each one in turn

Alternative Methods of Patching

- Using Oracle Enterprise Manager Cloud Control for Patching Operations
 - Using Cloud Control with its Provisioning & Patching functionality, you can automate the patching of your Oracle Grid Infrastructure and Oracle RAC software.
- Rapid Home Provisioning, Patching, and Upgrading
 - Rapid Home Provisioning is a method of deploying software homes to any number of nodes in a data center from a single cluster, and also facilitates patching and upgrading software.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

OPatch is a commonly used method for patching Oracle software homes, but this is not the only method of patching Oracle software.

Here are two possible alternative methods of patching. You might find these methods more appropriate for your organization than using OPatch.

1. Oracle Enterprise Manager Cloud Control for Patching Operations
 - Using Cloud Control with its Provisioning & Patching functionality, you can automate the patching of your Oracle Grid Infrastructure and Oracle RAC software.
 - Details on how to patch your Oracle Grid Infrastructure and Oracle RAC software using Cloud Control are available from the following PDF file: <http://www.oracle.com/technetwork/oem/pdf/512066.pdf>
2. Rapid Home Provisioning, Patching, and Upgrading
 - Rapid Home Provisioning is a method of deploying software homes to any number of nodes in a data center from a single cluster, and also facilitates patching and upgrading software.
 - See Also: [Oracle Clusterware Administration and Deployment Guide](#)



Quiz

Which tools can be used to install a patch set?

- a. Oracle Universal Installer
- b. OPatch
- c. Enterprise Manager Database Console
- d. Database Configuration Assistant
- e. Rapid Home Provisioning



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Describe the types of patches available
- Plan for rolling patches and rolling upgrades
- Install a patch set with the Oracle Universal Installer (OUI) utility
- Install a patch with the `opatch` utility



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Managing Backup and Recovery for RAC

ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to configure the following:

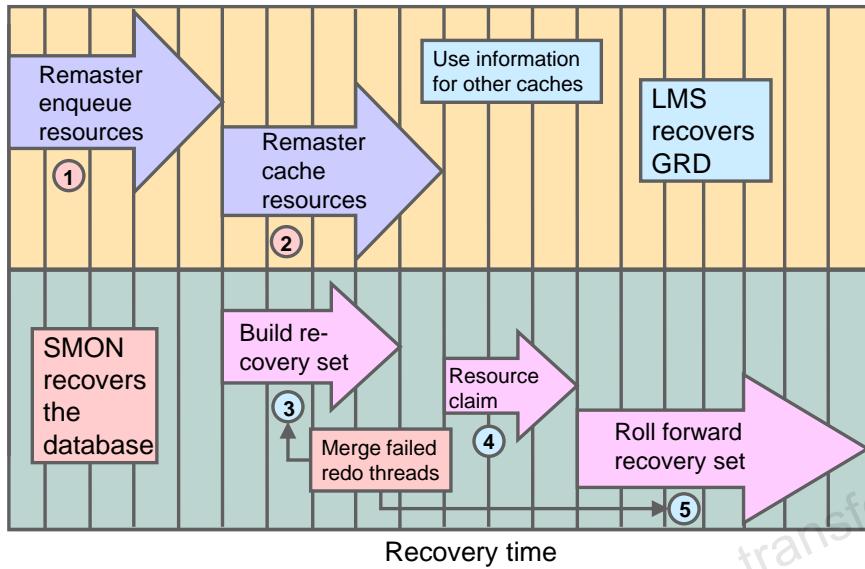
- The RAC database to use ARCHIVELOG mode and the fast recovery area
- RMAN for the RAC environment



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Instance Recovery and RAC



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When an instance fails and the failure is detected by another instance, the second instance performs the following recovery steps:

1. During the first phase of recovery, Global Enqueue Services remasters the enqueues.
2. The Global Cache Services (GCS) remasters its resources. The GCS processes remaster only those resources that lose their masters. During this time, all GCS resource requests and write requests are temporarily suspended. However, transactions can continue to modify data blocks as long as these transactions have already acquired the necessary resources.
3. After enqueues are reconfigured, one of the surviving instances can grab the Instance Recovery enqueue. Therefore, at the same time as GCS resources are remastered, SMON determines the set of blocks that need recovery. This set is called the recovery set. Because, with Cache Fusion, an instance ships the contents of its blocks to the requesting instance without writing the blocks to the disk, the on-disk version of the blocks may not contain the changes that are made by either instance. This implies that SMON needs to merge the content of all the online redo logs of each failed instance to determine the recovery set. This is because one failed thread might contain a hole in the redo that needs to be applied to a particular block. So, redo threads of failed instances cannot be applied serially. Also, redo threads of surviving instances are not needed for recovery because SMON could use past or current images of their corresponding buffer caches.
4. Buffer space for recovery is allocated and the resources that were identified in the previous reading of the redo logs are claimed as recovery resources. This is done to avoid other instances to access those resources.

5. All resources required for subsequent processing have been acquired and the Global Resource Directory (GRD) is now unfrozen. Any data blocks that are not in recovery can now be accessed. Note that the system is already partially available.

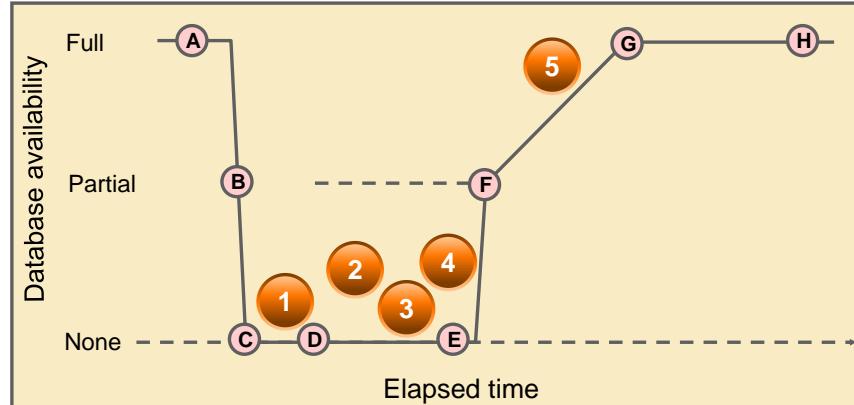
Then, assuming that there are past images or current images of blocks to be recovered in other caches in the cluster database, the most recent image is the starting point of recovery for these particular blocks. If neither the past image buffers nor the current buffer for a data block is in any of the surviving instances' caches, then SMON performs a log merge of the failed instances. SMON recovers and writes each block identified in step 3, releasing the recovery resources immediately after block recovery so that more blocks become available as recovery proceeds.

After all the blocks have been recovered and the recovery resources have been released, the system is again fully available.

In summary, the recovered database or the 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.

Note: The performance overhead of a log merge is proportional to the number of failed instances and to the size of the amount of redo written in the redo logs for each instance.

Instance Recovery and Database Availability



ORACLE®

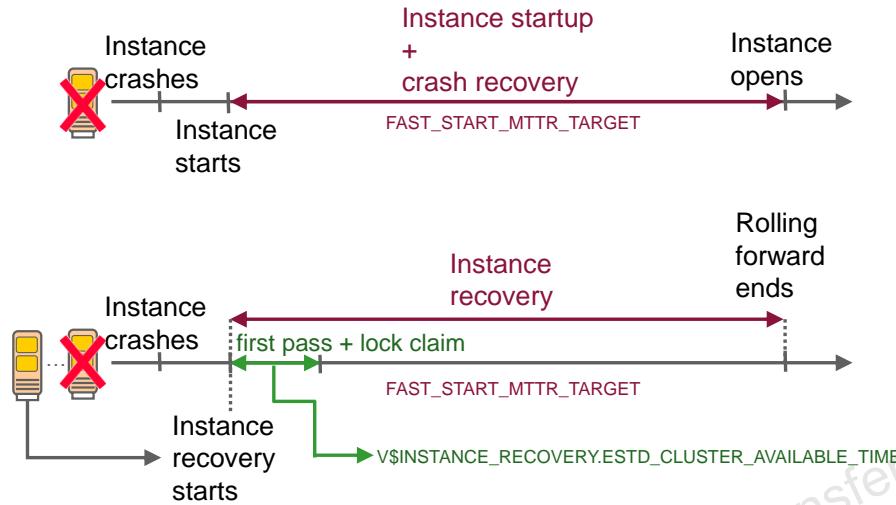
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The graphic illustrates the degree of database availability during each step of Oracle instance recovery:

- A. Real Application Clusters is running on multiple nodes.
- B. Node failure is detected.
- C. The enqueue part of the GRD is reconfigured; resource management is redistributed to the surviving nodes. This operation occurs relatively quickly.
- D. The cache part of the GRD is reconfigured and SMON reads the redo log of the failed instance to identify the database blocks that it needs to recover.
- E. SMON issues the GRD requests to obtain all the database blocks it needs for recovery. After the requests are complete, all other blocks are accessible.
- F. The Oracle server performs roll forward recovery. Redo logs of the failed threads are applied to the database, and blocks are available right after their recovery is completed.
- G. The Oracle server performs rollback recovery. Undo blocks are applied to the database for all uncommitted transactions.
- H. Instance recovery is complete and all data is accessible.

Note: The dashed line represents the blocks identified in step 2 of the previous slide. Also, the dotted steps represent the ones identified in the previous slide.

Instance Recovery and RAC



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In a single-instance environment, the instance startup combined with the crash recovery time is controlled by the setting of the `FAST_START_MTTR_TARGET` initialization parameter. You can set its value if you want incremental checkpointing to be more aggressive than the autotuned checkpointing. However, this is at the expense of a much higher I/O overhead.

In a RAC environment, including the startup time of the instance in this calculation is useless because one of the surviving instances is doing the recovery.

In a RAC environment, it is possible to monitor the estimated target, in seconds, for the duration from the start of instance recovery to the time when GRD is open for lock requests for blocks not needed for recovery. This estimation is published in the `V$INSTANCE_RECOVERY` view through the `ESTD_CLUSTER_AVAILABLE_TIME` column. Basically, you can monitor the time your cluster is frozen during instance-recovery situations.

In a RAC environment, the `FAST_START_MTTR_TARGET` initialization parameter is used to bound the entire instance-recovery time, assuming it is instance recovery for single-instance death.

If you really want to have short instance recovery times by setting `FAST_START_MTTR_TARGET`, you can safely ignore the alert log messages advising you to raise its value. Starting with Oracle RAC 12c Release 2, the Recovery Buddy capability can further reduce the instance recovery time.

When an instance fails, blocks that were recently modified by that instance may be in an inconsistent state on disk and must be recovered. A surviving instance must scan the redo log to recover these blocks before new transactions can modify the data. The Recovery Buddy capability creates a Buddy Instance to track modified data blocks on “buddy” nodes. If a node fails, the Buddy Instance can quickly identify which blocks require recovery, allowing new transactions on unaffected blocks to proceed rapidly and without having to wait for recovery determination.

This feature is especially significant reducing recovery time in clusters where a lot of update activity is taking place.

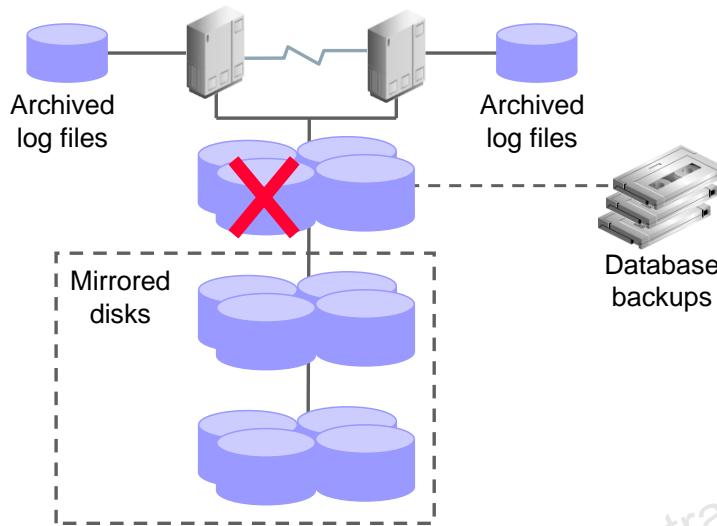
Instance Recovery and RAC

- Use parallel instance recovery.
- Set `PARALLEL_MIN_SERVERS`.
- Use asynchronous input/output (I/O).
- Increase the size of the default buffer cache.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Protecting Against Media Failure



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Although RAC provides you with methods to avoid or to reduce down time due to a failure of one or more (but not all) of your instances, you must still protect the database itself, which is shared by all the instances. This means that you need to consider disk backup and recovery strategies for your cluster database just as you would for a nonclustered database.

To minimize the potential loss of data due to disk failures, you may want to use disk mirroring technology (available from your server or disk vendor). As in nonclustered databases, you can have more than one mirror if your vendor allows it, to help reduce the potential for data loss and to provide you with alternative backup strategies. For example, with your database in ARCHIVELOG mode and with three copies of your disks, you can remove one mirror copy and perform your backup from it while the two remaining mirror copies continue to protect ongoing disk activity. To do this correctly, you must first put the tablespaces into backup mode and then, if required by your cluster or disk vendor, temporarily halt disk operations by issuing the `ALTER SYSTEM SUSPEND` command. After the statement completes, you can break the mirror and then resume normal operations by executing the `ALTER SYSTEM RESUME` command and taking the tablespaces out of backup mode.

Media Recovery in Oracle RAC

- Media recovery must be user-initiated through a client application.
- In these situations, use RMAN to restore backups of the data files and then recover the database.
- RMAN media recovery procedures for RAC do not differ substantially from those for single-instance environments.
- The node that performs the recovery must be able to restore all of the required data files.
 - That node must also be able to either read all required archived redo logs on disk or restore them from backups.
- When recovering a database with encrypted tablespaces, the wallet must be opened after database mount and before you open the database.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Media recovery must be user-initiated through a client application, whereas instance recovery is automatically performed by the database. In these situations, use RMAN to restore backups of the data files and then recover the database. The procedures for RMAN media recovery in Oracle RAC environments do not differ substantially from the media recovery procedures for single-instance environments.

The node that performs the recovery must be able to restore all of the required data files. That node must also be able to either read all the required archived redo logs on disk or be able to restore them from backups. Each instance generates its own archive logs that are copies of its dedicated redo log group threads. It is recommended that Automatic Storage Management (ASM) or a cluster file system be used to consolidate these files.

When recovering a database with encrypted tablespaces (for example, after a SHUTDOWN ABORT or a catastrophic error that brings down the database instance), you must open the Oracle Wallet after database mount and before you open the database, so the recovery process can decrypt data blocks and redo.

Parallel Recovery in RAC

- Oracle Database automatically selects the optimum degree of parallelism for:
 - Instance recovery
 - Crash recovery
- Archived redo logs are applied using an optimal number of parallel processes based on the availability of CPUs.
- With RMAN's RESTORE and RECOVER commands, the following three stages of recovery can use parallelism:
 - Restoring data files
 - Applying incremental backups
 - Applying archived redo logs
- To disable parallel instance and crash recovery, set the RECOVERY_PARALLELISM parameter to 0.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

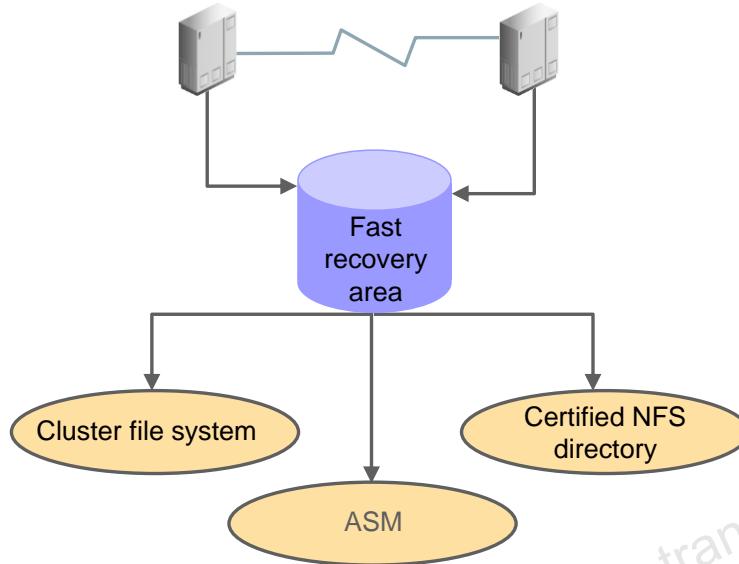
Oracle Database automatically selects the optimum degree of parallelism for instance and crash recovery. Oracle Database applies archived redo logs using an optimal number of parallel processes based on the availability of CPUs. With RMAN's RESTORE and RECOVER commands, Oracle Database automatically uses parallelism for the following three stages of recovery:

- **Restoring Data Files:** When restoring data files, the number of channels you allocate in the RMAN recover script effectively sets the parallelism that RMAN uses. For example, if you allocate five channels, you can have up to five parallel streams restoring data files.
- **Applying Incremental Backups:** Similarly, when you are applying incremental backups, the number of channels you allocate determines the potential parallelism.
- **Applying Archived Redo Logs with RMAN:** Oracle Database automatically selects the optimum degree of parallelism based on available CPU resources.

To disable parallel instance and crash recovery on a system with multiple CPUs, set the RECOVERY_PARALLELISM parameter to 0 or 1.

Use the NOPARALLEL clause of the RMAN RECOVER command or ALTER DATABASE RECOVER statement to force the RAC database to use nonparallel media recovery.

RAC and the Fast Recovery Area



ORACLE®

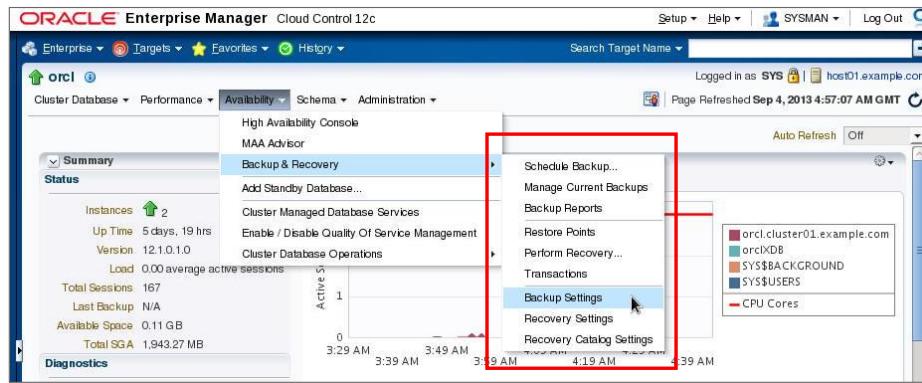
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To use a fast recovery area in RAC, you must place it on an ASM disk group, a cluster file system, or on a shared directory that is configured through certified NFS for each RAC instance. That is, the fast recovery area must be shared among all the instances of a RAC database. In addition, set the DB_RECOVERY_FILE_DEST parameter to the same value on all instances.

Oracle Enterprise Manager enables you to set up a fast recovery area. To use this feature:

1. From the Cluster Database Home page, click the Availability pull-down menu.
2. Select Backup and Recovery > Recovery Settings.
3. Specify your requirements in the Flash Recovery Area section of the page.

RAC Backup and Recovery Using EM



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can access the Cluster Database backup and recovery-related tasks by clicking the Availability tab on the Cluster Database Home page. On the Availability tabbed page, you can perform a range of backup and recovery operations using RMAN, such as scheduling backups, performing recovery when necessary, and configuring backup and recovery settings. Also, there are links related to Oracle Secure Backup and Service management.

Configuring RAC Recovery Settings with EM

The screenshot shows two tabs in the Recovery Settings section:

- Media Recovery:** Shows the database is in NOARCHIVELOG mode. A red box highlights the "ARCHIVELOG Mode" checkbox. Another red box highlights the "Add Another Row" button under Log Archive Filename Format. A third red box highlights the "Tip" message: "Tip It is recommended that archived redo log files be stored in a separate directory from the data files. You can specify up to 10 archived redo log files per instance." Below this, there's a note about possible data loss if the database is in NOARCHIVELOG mode.
- Fast Recovery Area:** Displays a pie chart titled "Fast Recovery Area Usage" showing the distribution of space. The data is as follows:

Category	Size (GB)	Percentage
Redo Log	~0.3 GB	5.7%
Control File	~0.02 GB	0.4%
Archived Redo Log	~0 GB	0%
Backup Piece	~0 GB	0%
Image Copy	~0 GB	0%
Flashback Log	~0 GB	0%
Auxiliary Datafile Copy	~0 GB	0%
Usable	~4.96 GB	94%

 Below the chart, it says "Fast Recovery Area Size must be set when the location is set." and lists Non-reclaimable, Reclaimable, and Free Fast Recovery Area sizes in MB. It also includes a "Flashback Retention Time" dropdown set to 24 hours and other flash recovery-related information.

ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can use Enterprise Manager to configure important recovery settings for your cluster database. On the Database Home page, click the Availability tab, and then click the Recovery Settings link. From here, you can ensure that your database is in ARCHIVELOG mode and configure flash recovery settings.

With a RAC database, if the Archive Log Destination setting is not the same for all instances, the field appears blank, with a message indicating that instances have different settings for this field. In this case, entering a location in this field sets the archive log location for all instances of the database. You can assign instance-specific values for an archive log destination by using the Initialization Parameters page.

Note: You can run the `ALTER DATABASE` SQL statement to change the archiving mode in RAC as long as the database is mounted by the local instance but not open in any instances. You do not need to modify parameter settings to run this statement. Set the initialization parameters `DB_RECOVERY_FILE_DEST` and `DB_RECOVERY_FILE_DEST_SIZE` to the same values on all instances to configure a fast recovery area in a RAC environment.

Archived Redo File Conventions in RAC

Variable	Description	Example
<code>%t</code>	Thread number, not padded	log_1
<code>%T</code>	Thread number, left-zero-padded	log_0001
<code>%s</code>	Log sequence number, not padded	log_251
<code>%S</code>	Log sequence number, left-zero-padded	log_0000000251
<code>%r</code>	Resetlogs identifier	log_23452345
<code>%R</code>	Padded resetlogs identifier	log_0023452345
<code>%t %s %r</code>	Using multiple variables	log_1_251_23452345



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

For any archived redo log configuration, uniquely identify the archived redo logs with the `LOG_ARCHIVE_FORMAT` parameter. The format of this parameter is operating system specific and it can include text strings, one or more variables, and a file name extension.

All of the thread parameters, in either uppercase or lowercase, are mandatory for RAC. This enables the Oracle database to create unique names for archive logs across the incarnation. This requirement is in effect when the `COMPATIBLE` parameter is set to 10.0 or greater. Use the `%R` or `%r` parameter to include the resetlogs identifier to avoid overwriting the logs from a previous incarnation. If you do not specify a log format, the default is operating system specific and includes `%t`, `%s`, and `%r`.

As an example, if the instance associated with redo thread number 1 sets `LOG_ARCHIVE_FORMAT` to `log_%t_%s_%r.arc`, then its archived redo log files are named as:

```
log_1_1000_23435343.arc  
log_1_1001_23452345.arc  
log_1_1002_23452345.arc  
...
```

Note: The `LOG_ARCHIVE_FORMAT` parameter will have no effect if Oracle Managed Files (OMF) has been implemented for the Oracle RAC database.

Configuring RAC Backup Settings with EM

The screenshot shows the 'Backup Settings' page in Oracle Enterprise Manager. It includes sections for Disk Settings, Tape Settings, Oracle Secure Backup Domain, Media Management Settings, and Host Credentials. The Disk Settings section allows configuring disk backup location and parallelism. The Tape Settings section allows specifying tape drives and backup type. The Oracle Secure Backup Domain section shows domain details and storage selector configuration. The Media Management Settings section provides vendor-specific parameters. The Host Credentials section manages operating system login credentials.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Persistent backup settings can be configured using Enterprise Manager. On the Cluster Database Home page, click the Availability link, select Backup and Recovery, and then click the Backup Settings link. You can configure disk settings, such as the directory location of your disk backups and level of parallelism. You can also choose the default backup type:

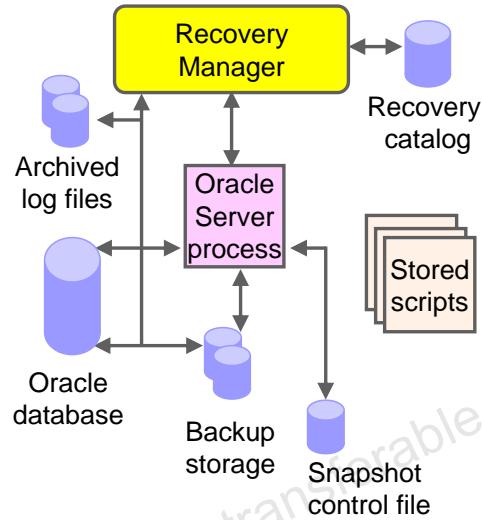
- Backup set
- Compressed backup set
- Image copy

You can also specify important tape-related settings, such as the number of available tape drives, vendor-specific media management parameters and Oracle Secure Backup domain.

Oracle Recovery Manager

RMAN provides the following benefits for Real Application Clusters:

- Can read cluster files or ASM files with no configuration changes
- Can access multiple archive log destinations



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle Database provides RMAN for backing up and restoring the database. RMAN enables you to back up, restore, and recover data files, control files, SPFILEs, and archived redo logs. You can run RMAN from the command line or you can use it from the Backup Manager in Enterprise Manager. In addition, RMAN is the recommended backup and recovery tool if you are using ASM. RMAN can use stored scripts, interactive scripts, or an interactive GUI front end. When using RMAN with your RAC database, use stored scripts to initiate the backup and recovery processes from the most appropriate node.

Create a snapshot control file in a location that exists on all your nodes. You can specify a cluster file system or a raw device destination for the location of your snapshot control file. This file is shared across all nodes in the cluster and must be accessible by all nodes in the cluster.

For recovery, you must ensure that each recovery node can access the archive log files from all instances by using one of the archive schemes discussed earlier, or make the archived logs available to the recovering instance by copying them from another location.

Configuring RMAN Snapshot Control File Location

- The snapshot control file path **must** be valid on every node from which you might initiate an RMAN backup.
- Configure the snapshot control file location in RMAN.
 - Determine the current location:

```
RMAN> SHOW SNAPSHOT CONTROLFILE NAME;
' /u01/app/oracle/product/12.2.0/dbhome_1/dbs/snapcf_orcl_3.f'
```

- You can use ASM or a shared file system location:

```
RMAN> CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'+FRA/SNAP/snap_prod.cf';
RMAN> CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'/ocfs2/oradata/dbs/scf/snap_prod.cf';
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The snapshot control file is a copy of a database control file created in an operating system–specific location by RMAN. RMAN creates the snapshot control file so that it has a consistent version of a control file to use when either resynchronizing the recovery catalog or backing up the control file. You can also create a snapshot control file by entering the following at the RMAN prompt: `DUPLICATE FROM ACTIVE`. You can specify a cluster file system or ASM disk group destination for the location of your snapshot control file. This file is shared across all nodes in the cluster and must be accessible by all nodes in the cluster.

You can change the configured location of the snapshot control file. For example, on Linux and UNIX systems you can change the snapshot control file location by using the `CONFIGURE SNAPSHOT CONTROLFILE NAME` RMAN command. This command sets the configuration for the location of the snapshot control file for every instance of your cluster database. Therefore, ensure that the location specified exists on all nodes that perform backups. The `CONFIGURE` command creates persistent settings across RMAN sessions. Therefore, you do not need to run this command again unless you want to change the location of the snapshot control file. To delete a snapshot control file you must first change the snapshot control file location, then delete the file at the older location, as follows:

```
CONFIGURE SNAPSHOT CONTROLFILE NAME TO 'new_name';
DELETE COPY OF CONTROLFILE;
```

Configuring Control File and SPFILE Autobackup

- RMAN automatically creates a control file and SPFILE backup after the BACKUP or COPY command:

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

- Change default location:

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE  
TYPE DISK TO '+DATA';
```

- Location **must** be available to all nodes in your RAC database.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

If you set CONFIGURE CONTROLFILE AUTOBACKUP to ON, RMAN automatically creates a control file and an SPFILE backup after you run the BACKUP or COPY command. RMAN can also automatically restore an SPFILE if this is required to start an instance to perform recovery. This means that the default location for the SPFILE must be available to all nodes in your RAC database.

These features are important in disaster recovery because RMAN can restore the control file even without a recovery catalog. RMAN can restore an autobackup of the control file even after the loss of both the recovery catalog and the current control file.

You can change the default location that RMAN gives to this file with the CONFIGURE CONTROLFILE AUTOBACKUP FORMAT command. If you specify an absolute path name in this command, this path must exist identically on all nodes that participate in backups.

RMAN performs the control file autobackup on the first allocated channel. Therefore, when you allocate multiple channels with different parameters, especially when you allocate a channel with the CONNECT command, determine which channel will perform the control file autobackup. Always allocate the channel for this node first. Besides using the RMAN control file, you can also use Oracle Enterprise Manager to use the RMAN features.

Crosschecking on Multiple RAC Clusters Nodes

When crosschecking on multiple nodes, make sure that all backups can be accessed by every node in the cluster.

- This allows you to allocate channels at any node in the cluster during restore or crosscheck operations.
- Otherwise, you must allocate channels on multiple nodes by providing the CONNECT option to the CONFIGURE CHANNEL command.
- If backups are not accessible because no channel was configured on the node that can access those backups, then those backups are marked EXPIRED.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When crosschecking on multiple RAC nodes, configure the cluster so that all backups can be accessed by every node, regardless of which node created the backup. When the cluster is configured this way, you can allocate channels at any node in the cluster during restore or crosscheck operations.

If you cannot configure the cluster so that each node can access all backups, then during restore and crosscheck operations, you must allocate channels on multiple nodes by providing the CONNECT option to the CONFIGURE CHANNEL command, so that every backup can be accessed by at least one node. If some backups are not accessible during crosscheck because no channel was configured on the node that can access those backups, then those backups are marked EXPIRED in the RMAN repository after the crosscheck.

For example, you can use CONFIGURE CHANNEL . . . CONNECT in an Oracle RAC configuration in which tape backups are created on various nodes in the cluster and each backup is accessible only on the node on which it is created.

Channel Connections to Cluster Instances

- When backing up, each allocated channel can connect to a different instance in the cluster.
- Instances to which the channels connect must be either all mounted or all open.
- When choosing a channel to use, RMAN gives preference to the nodes with faster access to the data files that you want to back up.

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT='sys/rac@orcl_1';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT='sys/rac@orcl_2';
CONFIGURE CHANNEL 3 DEVICE TYPE sbt CONNECT='sys/rac@orcl_3';
```

OR

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE CHANNEL DEVICE TYPE sbt CONNECT='sys/rac@bkp_serv';
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When making backups in parallel, RMAN channels can connect to a different instance in the cluster. The examples in the slide illustrate two possible configurations:

- If you want to dedicate channels to specific instances, you can control at which instance the channels are allocated by using separate connect strings for each channel configuration as shown by the first example.
- If you define a special service for your backup and recovery jobs, you can use the second example shown in the slide. If you configure this service with load balancing turned on, then the channels are allocated at a node as decided by the load balancing algorithm.

During a backup, the instances to which the channels connect must be either all mounted or all open. For example, if the `orcl_1` instance has the database mounted whereas the `orcl_2` and `orcl_3` instances have the database open, then the backup fails.

In some RAC database configurations, some cluster nodes have faster access to certain data files than to other data files. RMAN automatically detects this, which is known as node affinity awareness. When deciding which channel to use to back up a particular data file, RMAN gives preference to the nodes with faster access to the data files that you want to back up.

RMAN Channel Support for the Grid

- RAC allows the use of nondeterministic connect strings.
- RMAN can use connect strings that are not bound to a specific instance in the Grid environment.
- It simplifies the use of parallelism with RMAN in a RAC environment.
- It uses the load-balancing characteristics of the Grid environment.
 - Channels connect to RAC instances that are the least loaded.

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

RAC allows the use of nondeterministic connect strings that can connect to different instances based on RAC features, such as load balancing. Therefore, to support RAC, the RMAN polling mechanism no longer depends on deterministic connect strings, and makes it possible to use RMAN with connect strings that are not bound to a specific instance in the Grid environment. Previously, if you wanted to use RMAN parallelism and spread a job between many instances, you had to manually allocate an RMAN channel for each instance. To use dynamic channel allocation, you do not need separate CONFIGURE CHANNEL CONNECT statements anymore. You only need to define your degree of parallelism by using a command such as CONFIGURE DEVICE TYPE disk PARALLELISM, and then run backup or restore commands. RMAN then automatically connects to different instances and does the job in parallel. The Grid environment selects the instances that RMAN connects to, based on load balancing. As a result of this, configuring RMAN parallelism in a RAC environment becomes as simple as setting it up in a non-RAC environment. By configuring parallelism when backing up or recovering a RAC database, RMAN channels are dynamically allocated across all RAC instances.

Note: RMAN has no control over the selection of the instances. If you require a guaranteed connection to an instance, you should provide a connect string that can connect only to the required instance.

RMAN Default Autolocation

- Recovery Manager autlocates the following files:
 - Backup pieces
 - Archived redo logs during backup
 - Data file or control file copies
- If local archiving is used, a node can read only those archived logs that were generated on that node.
- When restoring, a channel connected to a specific node restores only those files that were backed up to the node.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Recovery Manager automatically discovers which nodes of a RAC configuration can access the files that you want to back up or restore. Recovery Manager autlocates the following files:

- Backup pieces during backup or restore
- Archived redo logs during backup
- Data file or control file copies during backup or restore

If you use a noncluster file system local archiving scheme, a node can read only those archived redo logs that were generated by an instance on that node. RMAN never attempts to back up archived redo logs on a channel that it cannot read.

During a restore operation, RMAN automatically performs the autolocation of backups. A channel connected to a specific node attempts to restore only those files that were backed up to the node. For example, assume that log sequence 1001 is backed up to the drive attached to node 1, whereas log 1002 is backed up to the drive attached to node 2. If you then allocate channels that connect to each node, the channel connected to node 1 can restore log 1001 (but not 1002), and the channel connected to node 2 can restore log 1002 (but not 1001).

Distribution of Backups

Several possible backup configurations for RAC:

- A dedicated backup server performs and manages backups for the cluster and the cluster database.
- One node has access to a local backup appliance and performs and manages backups for the cluster database.
- Each node has access to a local backup appliance and can write to its own local backup media.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When configuring the backup options for RAC, you have several possible configurations:

- **Network backup server:** A dedicated backup server performs and manages backups for the cluster and the cluster database. None of the nodes have local backup appliances.
- **One local drive:** One node has access to a local backup appliance and performs and manages backups for the cluster database. All nodes of the cluster should be on a cluster file system to be able to read all data files, archived redo logs, and SPFILEs. It is recommended that you do not use the noncluster file system archiving scheme if you have backup media on only one local drive.
- **Multiple drives:** Each node has access to a local backup appliance and can write to its own local backup media.

In the cluster file system scheme, any node can access all the data files, archived redo logs, and SPFILEs. In the noncluster file system scheme, you must write the backup script so that the backup is distributed to the correct drive and path for each node. For example, node 1 can back up the archived redo logs whose path names begin with /arc_dest_1, node 2 can back up the archived redo logs whose path names begin with /arc_dest_2, and node 3 can back up the archived redo logs whose path names begin with /arc_dest_3.

Managing Archived Redo Logs Using RMAN

- When an archived redo log is generated, Oracle records the name of the log in the control file of the target database.
- If a recovery catalog is used, RMAN records the archived redo log names in the catalog when a resynch occurs.
- The backup and recovery strategy that you choose depends on how the archive destinations are configured.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When a node generates an archived redo log, Oracle Database always records the file name of the log in the control file of the target database. If you are using a recovery catalog, then RMAN also records the archived redo log file names in the recovery catalog when a resynchronization occurs.

The archived redo log naming scheme that you use is important because when a node writes to a log with a specific file name on its file system, the file must be readable by any node that must access this archived redo log. For example, if node1 archives a log to `/oracle/arc_dest/log_1_100_23452345.arc`, then node2 can back up this archived redo log only if it can read `/oracle/arc_dest/log_1_100_23452345.arc` on its own file system.

The backup and recovery strategy that you choose depends on how you configure the archiving destinations for each node. Whether only one node or all nodes perform archived redo log backups, you must ensure that all archived redo logs are backed up. If you use RMAN parallelism during recovery, then the node that performs recovery must have read access to all archived redo logs in your cluster.

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 that are needed for the recovery operation. By default, the database determines the optimum number of parallel threads to use during the recovery operation. You can use the `PARALLEL` clause in the `RECOVER` command to change the number of parallel threads.

Noncluster File System Local Archiving Scheme

- When archiving locally to a noncluster file system, each node archives to a uniquely named local directory.
- If recovery is required, configure the recovery node so that it can access directories on the other nodes remotely.
 - Use NFS on Linux and UNIX computers.
 - Use mapped drives on Windows systems.
- Each node writes to a local destination, but can read archived redo log files in remote directories on other nodes.
- If noncluster local archiving is used for media recovery, configure the node performing recovery for remote access.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When archiving locally to a noncluster file system, each node archives to a uniquely named local directory. If recovery is required, then you can configure the recovery node so that it can access directories on the other nodes remotely. For example, use NFS on Linux and UNIX computers, or mapped drives on Windows systems. Therefore, each node writes only to a local destination, but each node can also read archived redo log files in remote directories on the other nodes.

If you use noncluster file system local archiving for media recovery, then you must configure the node that is performing recovery for remote access to the other nodes so that it can read the archived redo log files in the archive directories on the other nodes. In addition, if you are performing recovery and you do not have all of the available archive logs, then you must perform an incomplete recovery up to the first missing archived redo log sequence number. You do not have to use a specific configuration for this scheme. However, to distribute the backup processing onto multiple nodes, the easiest method is to configure channels.

Note: Because different file systems are used in a noncluster case, the archive log directories must be unique on each node. For example, /arc_dest_1 is only available on node1, /arc_dest_2 is only directly mounted on node2, and so on. Then node1 mounts /arc_dest_2 from node2 and /arc_dest_3 from node3 through NFS.

Configuring Non-Cluster, Local Archiving

- Set the `SID.LOG_ARCHIVE_DEST` parameter for each instance using the SID designator:

```
sid1.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest_1"
sid2.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest_2"
sid3.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest_3"
```

- For policy-managed databases, create a node and instance binding to ensure that sid1 always runs on the same node:

```
$ srvctl modify instance -d mydb -n node1 -i sid1
$ srvctl modify instance -d mydb -n node2 -i sid2
$ srvctl modify instance -d mydb -n node3 -i sid3
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can set the archiving destination values as follows in the initialization parameter file for either policy-managed or administrator-managed databases. Set the `SID.LOG_ARCHIVE_DEST` parameter for each instance using the SID designator, as shown in the following example:

```
sid1.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest_1"
sid2.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest_2"
sid3.LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest_3"
```

For policy-managed databases, manually create a node and instance binding to ensure that sid1 always runs on the same node, as follows:

```
$ srvctl modify instance -d mydb -n node1 -i sid1
$ srvctl modify instance -d mydb -n node2 -i sid2
$ srvctl modify instance -d mydb -n node3 -i sid3
```

The following list shows the possible archived redo log entries in the database control file. Note that any node can read archived redo logs from any of the threads, which must happen in order for the database to recover after a failure.

```
/arc_dest_1/log_1_1000_23435343.arc
/arc_dest_2/log_1_1001_23452345.arc <- thread 1 archived in node2
/arc_dest_2/log_3_1563_23452345.arc <- thread 3 archived in node2
/arc_dest_1/log_2_753_23452345.arc <- thread 2 archived in node1
/arc_dest_2/log_2_754_23452345.arc
/arc_dest_3/log_3_1564_23452345.arc
```

ASM and Cluster File System Archiving Scheme

- The preferred configuration for RAC is to use ASM disk group for a recovery area for the recovery set that is different from the disk group used for data files.
 - When you use ASM, it uses an OMF naming format.
- Alternatively, a cluster file system archiving scheme can be used.
 - Each node writes to a single location on the cluster file system when archiving the redo log files.
 - Each node can read the archived redo log files of the other nodes.
- The advantage of this scheme is that none of the nodes uses the network to archive logs.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The preferred configuration for Oracle RAC is to use ASM for a recovery area using a disk group for your recovery set that is different from the disk group used for your data files. When you use Oracle ASM, it uses an Oracle Managed Files naming format.

Alternatively, you can use a cluster file system archiving scheme. If you use a cluster file system, then each node writes to a single location on the cluster file system when archiving the redo log files. Each node can read the archived redo log files of the other nodes. For example, if Node 1 archives a redo log file to /arc_dest/log_1_100_23452345.arc on the cluster file system, then any other node in the cluster can also read this file.

If you do not use a cluster file system, then the archived redo log files cannot be on raw devices. This is because raw devices do not enable sequential writing of consecutive archive log files. The advantage of this scheme is that none of the nodes uses the network to archive logs. Because the file name written by a node can be read by any node in the cluster, RMAN can back up all logs from any node in the cluster. Backup and restore scripts are simplified because each node has access to all archived redo logs.

Configuring the CFS Archiving Scheme

- In the CFS scheme, each node archives to a directory with the same name on all instances within the database.
- To configure this directory, set values for the `LOG_ARCHIVE_DEST_1` parameter:

```
* .LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest"
```
- The below list shows archived redo log entry examples that would appear in the RMAN catalog or in the control file:

```
/arc_dest/log_1_999_23452345.arc
/arc_dest/log_1_1000_23435343.arc
/arc_dest/log_1_1001_23452345.arc <- thread 1 archived in node3
/arc_dest/log_3_1563_23452345.arc <- thread 3 archived in node2
/arc_dest/log_2_753_23452345.arc <- thread 2 archived in node1
/arc_dest/log_2_754_23452345.arc
```
- Each node can read the logs written by itself and any other node.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In the cluster file system scheme, each node archives to a directory that is identified with the same name on all instances within the cluster database (`/arc_dest`, in the following example). To configure this directory, set values for the `LOG_ARCHIVE_DEST_1` parameter, as shown in the following example:

```
* .LOG_ARCHIVE_DEST_1="LOCATION=/arc_dest"
```

The following list shows archived redo log entry examples that would appear in the RMAN catalog or in the control file based on the previous example. Note that any node can archive logs using any of the threads:

```
/arc_dest/log_1_999_23452345.arc
/arc_dest/log_1_1000_23435343.arc
/arc_dest/log_1_1001_23452345.arc <- thread 1 archived in node3
/arc_dest/log_3_1563_23452345.arc <- thread 3 archived in node2
/arc_dest/log_2_753_23452345.arc <- thread 2 archived in node1
/arc_dest/log_2_754_23452345.arc
/arc_dest/log_3_1564_23452345.arc
```

Because the file system is shared and because each node is writing its archived redo logs to the `/arc_dest` directory in the cluster file system, each node can read the logs written by itself and any other node.

Restoring and Recovering

- Media recovery may require one or more archived log files from each thread.
- The RMAN RECOVER command automatically restores and applies the required archived logs.
- Archive logs may be restored to any node performing the restore and recover operation.
- Logs must be readable from the node performing the restore and recovery activity.
- Recovery processes request additional threads enabled during the recovery period.
- Recovery processes notify you of threads no longer needed because they were disabled.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Media recovery of a database that is accessed by RAC may require at least one archived log file for each thread. However, if a thread's online redo log contains enough recovery information, restoring archived log files for any thread is unnecessary.

If you use RMAN for media recovery and you share archive log directories, you can change the destination of the automatic restoration of archive logs with the SET clause to restore the files to a local directory of the node where you begin recovery. If you backed up the archive logs from each node without using a central media management system, you must first restore all the log files from the remote nodes and move them to the host from which you will start recovery with RMAN. However, if you backed up each node's log files using a central media management system, you can use RMAN's AUTOLOCATE feature. This enables you to recover a database by using the local tape drive on the remote node.

If recovery reaches a time when an additional thread was enabled, the recovery process requests the archived log file for that thread. If you are using a backup control file, when all archive log files are exhausted, you may need to redirect the recovery process to the online redo log files to complete recovery. If recovery reaches a time when a thread was disabled, the process informs you that the log file for that thread is no longer needed.



Quiz

Which of the following statements regarding media recovery in RAC is not true?

- a. Media recovery must be user-initiated through a client application.
- b. RMAN media recovery procedures for RAC are quite different from those for single-instance environments.
- c. The node that performs the recovery must be able to restore all the required data files.
- d. The recovering node must be able to either read all required archived redo logs on disk or restore them from backups.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.



Quiz

To use a fast recovery area in RAC, you must place it on an ASM disk group, a cluster file system, or on a shared directory that is configured through certified NFS for each RAC instance.

- a. True
- b. False



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to configure the following:

- The RAC database to use ARCHIVELOG mode and the fast recovery area
- RMAN for the RAC environment



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Practice 6: Overview

This practice covers the following topics:

- Configuring the archive log mode
- Configuring RMAN and performing parallel backups



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Global Resource Management Concepts



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to describe:

- The need for global concurrency control
- Global Resource Directory
- How global resources are managed
- RAC global resource access coordination
 - Global enqueue and instance lock management
 - Global buffer cache management



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Need for Global Concurrency Control

- Oracle requires concurrency control because it is a multi-user system.
- Single-instance Oracle provides concurrency control:
 - Latches or mutexes for memory structures
 - Enqueues for resource control
 - Buffer cache pins for cache management
- In RAC, structures and resources may be accessed by or modified by a session running on any database instance.
- RAC, therefore, requires additional global concurrency controls to mediate access across instances.
 - Global locks control library and row cache access.
 - Global enqueues control resource access.
 - Cache fusion controls buffer cache access.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Global Resource Directory (GRD)

- An object under global concurrency control is called a resource.
- Resource metadata is held in the Global Resource Directory (GRD).
 - Global enqueue resources are used for enqueues and locks.
 - Global cache resources are used for buffer cache control.
- The GRD is distributed among all active instances of each database or ASM environment.
- Each currently managed GRD resource has:
 - A master metadata structure
 - One or more shadow metadata structures
- The GRD uses memory from the shared pool.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Global Resource Management

- After first access of a globally managed entity by any instance, a global resource is allocated.
- An internal algorithm is used to decide which instance should contain the master metadata structure for that entity.
 - This instance is known as the resource master.
 - The resource mastering instance may be any active instance of the database or ASM environment.
- Subsequent access to an entity from another instance for which resource master metadata exists causes resource shadow metadata to be allocated in the requesting instance and updates to be done to the master metadata.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The resource mastering instance is the instance containing the master metadata used to manage concurrency control for a specific entity.

Each instance will be the resource master for some of the database entities.

The resource shadowing instance is any instance containing shadow metadata used to manage concurrency control for a specific entity. Each instance will contain shadow resources for entities it has accessed and for which it is not the resource master.

Global Resource Remastering

- Remastering is the process of allocating control of the master metadata for a specific entity to another instance.
- Instance-level or lazy remastering occurs when:
 - A new instance of the same database or ASM starts
 - A current instance is shut down gracefully
- File affinity remastering occurs when:
 - Requests to access blocks in a data file occur frequently from an instance, and the resource masters for the blocks are often held by other instances
- Object-affinity remastering occurs when:
 - Requests to access blocks in a data object occur frequently from an instance, and the resource masters for the blocks are often held by other instances
- Resource Mastering is further optimized using Service Oriented Buffer Cache Access



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When a new instance starts, remastering is not done immediately. Instead it is done gradually, based on which instances are accessing which resources (hence the term “lazy”).

When an instance shuts down gracefully—meaning normal, immediate, or transactional—then resources mastered by the terminating instance are handed off to the surviving instances by using an optimized internal algorithm designed to minimize the remastering and subsequent concurrency control overheads.

The decision to perform file-affinity or object-affinity remastering is made automatically when an internal threshold is reached.

Starting with Oracle RAC 12c Release 2, Cache Fusion maintains an in-memory hash that tracks the blocks in the buffer cache and the service name used by the sessions to connect. Cache Fusion uses this hash for resource mastering optimization.

Resource mastering of a resource cached in the buffer is only considered on the node where the service that the session used to access the resource is running. This results in improved performance as it eliminates the need for sending additional messages on the private network for resource change operations.

Global Resource Recovery

- When one or more but not all instances fail:
 - The failing instance(s) resource masters are lost
 - Any resource master that had a shadow in a surviving instance must be recovered
- The surviving instances can rebuild resource master metadata for a specific resource, by aggregating details from surviving shadow metadata for the same resource.
- Global locks and enqueue metadata are done first, followed by global cache metadata.
- The rebuilding results in each surviving instance mastering some of the recovered resource master metadata.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Enqueues are done first, because Oracle must know who has access to which resource in order for recovery to proceed. A look into the RAC database alert log shows global resource activity during instance recovery:

```
...
Reconfiguration started (old inc 0, new inc 6)
List of instances:
 1 2 3 (myinst: 3)
Global Resource Directory frozen
* allocate domain 0, invalid = TRUE
Communication channels reestablished
* domain 0 valid = 0 according to instance 1
Master broadcasted resource hash value bitmaps
Non-local Process blocks cleaned out
LMS 0: 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 enqueueues granted
Submitted all GCS remote-cache requests
Fix write in gcs resources
Reconfiguration complete (total time 0.5 secs)
...
```

Global Resource Background Processes

- **ACMS:** Atomic Control file to Memory Service
- **LMHB:** Monitors LMON, LMD, and LMS n processes
- **LMD0:** Requests global enqueues and instance locks
- **LMON:** Issues heartbeats and performs recovery
- **LMS n :** Processes global cache fusion requests
- **LCK0:** Is involved in library and row cache locking
- **RCBG:** Processes Global Result Cache invalidations



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Note: There are other RAC backgrounds, but this lesson concentrates only on Global Concurrency Control.

- **Atomic Control File to Memory Service (ACMS):** In a RAC environment, the ACMS per-instance process is an agent that contributes to ensuring that a distributed SGA memory update is either globally committed on success or globally aborted if a failure occurs.
- **Global Enqueue Service Monitor (LMON):** The LMON process monitors global enqueues and resources across the cluster and performs global enqueue recovery operations.
- **Global Enqueue Service Daemon (LMD):** The LMD process manages incoming remote resource requests within each instance.
- **Global Cache Service Process (LMS):** The LMS process maintains records of the data file statuses and each cached block by recording information in the GRD. The LMS process also controls the flow of messages to remote instances and manages global data block access and transmits block images between the buffer caches of different instances. This processing is part of the cache fusion feature.
- **Instance Enqueue Process (LCK0):** The LCK0 process manages noncache fusion resource requests such as library and row cache requests.
- **Global Cache/Enqueue Service Heartbeat Monitor (LMHB):** LMHB monitors LMON, LMD, and LMS n processes to ensure that they are running normally without blocking or spinning.
- **Result Cache Background Process (RCBG):** This process is used for handling invalidation and other messages generated by server processes attached to other instances in Oracle RAC.

Global Resource Access Coordination

- There are two types of global resource coordination.
- Global enqueue management, which is used for:
 - Global enqueues
 - Global instance locks
- Global buffer cache, which:
 - Is also known as cache fusion or global cache
 - Is a logical buffer cache spanning all instances
 - Coordinates access to block images in the global cache
 - Supports Parallel Query across the global cache



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Global enqueues are used to control access to resources, where the owner(s), waiter(s) if any, or converter(s) if any, or both, may be sessions in the same or different instances. Some global enqueues serve the same purpose they would serve in a single instance. For example, table manipulation (`TM`) enqueues, transaction enqueues (`TX`), control file enqueues (`CF`), high watermark enqueues (`HW`), sequence cache replenishment (`SQ`), and redo thread enqueues (`RT`) all serve the same purpose as they would in a single instance. However, there are master and shadow metadata structures as described earlier in this lesson in the `GRD`, and the mastering instance will keep track of the waiters and converters.

Instance locks are enqueues that represent resources in the row cache or library cache protected within each instance by pins, mutexes, or latches. For cross-instance concurrency control, an enqueue is used, the owner(s) of which is or are the instance(s) that is or are currently the “source of truth” with regard to the current state of that resource. The `LCK0` process acts as the owner, waiter, or converter of the enqueue as a “proxy” process representing the instance. These enqueues are known as instance locks.

Global Enqueues

Processing starts in the requesting instance as follows:

1. A global enqueue request is made by a session.
2. The request is passed to `LMD0` in the requesting instance.
3. The foreground waits for the request on event.
4. `LMD0` determines the mastering instance.
5. `LMD0` forwards the request to the mastering instance if required.
6. The mastering instance adds a new master resource if required.
 - Process is made an owner, waiter, or converter as appropriate.
 - Once the resource can be granted to the requestor, `LMD0` in the mastering instance notifies `LMD0` in the requesting instance.
7. When the resource is available, the foreground is posted by `LMD0` in the requesting instance.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

If requesting and mastering instances are the same, then `LMD0` need not forward the request over the interconnect. `LMD0` in the mastering instance notifies `LMD0` in the requesting instance whether the resource is available to the requestor immediately.

If a dequeue request is passed to the mastering instance, then `LMD0` notifies the `LMD0` processes for any waiters or converters that need resuming and they are posted by the `LMD0` in their own instances.

Instance Locks

- Instance locks are used to represent which instance(s) has (have) control over an instance-wide structure:
 - Row cache entries
 - Library cache entries
 - Result cache entries
- The owner, waiter, or converter on an instance lock is the LCK0 process.
 - As long as the local LCK0 process in an instance owns the lock on a specific resource, any session in that instance can use the cached metadata, because it is considered current.
 - If the local instance does not own the lock, then a request must be made for the lock and the foreground waits on DFS Lock Handle wait event.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Instance locks use enqueue structures but the scope is different. LCK0 acts as the owner or waiter for such situations. The owner of an instance lock represents the instance having permission to access the related entity in the row cache or library cache. Assuming that an instance owns the instance lock, then the usual latches or pins or mutexes provide concurrency control within the instance as usual.

Global Cache Management: Overview

Global cache management provides:

- A concurrency mechanism for multiple buffer caches
- An optimization of block access for reads
- An optimization of writes for dirty buffers
- A mechanism to optimize parallel queries



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Global Cache Management Components

- The LMS_n processes
- Buffers
- Buffer headers
- Global Cache Master Resources
- Global Cache Shadow Resources



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Global Cache Buffer States

- Buffer states are visible in V\$BH.STATUS.
- Important buffer states are:
 - Shared Current: SCUR
 - Exclusive Current: XCUR
 - Consistent Read: CR
 - Built in the Instance
 - Sent by cache fusion
 - Converted from SCUR or PI
 - Past Image: PI
 - Converted from XCUR
 - Not normally written
 - Converted to CR after later XCUR image is written
 - Multiple PI images may exist for same block in different buffer caches.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Important buffer states for cache fusion in V\$BH.STATUS are:

- **Shared Current:** The buffer contains a block image that matches the one on disk. One or more instances may have images for the same block in the SCUR state. After an instance has one in this state, cache fusion is used if another instance reads the same block for read purposes.
- **Exclusive Current:** The buffer contains a block image that is about to be updated, or has been updated. It may or may not have been written by the database writer. Only one instance may have an XCUR image for a block.
- **Consistent Read:** The buffer contains a block image that is consistent with an earlier point in time. This image may have been created in the same way as in single-instance databases, but copying a block into an available buffer and using undo to roll back the changes to create the older image. It may also get created by converting a block image from SCUR or PI.
- **Past Image:** The buffer contains a block image that was XCUR but then shipped to another instance using cache fusion. A later image of this block now exists in another buffer cache. Once DBWn writes the later image to disk from the other instance, the PI image becomes a CR image.

Global Cache Management Scenarios for Single Block Reads

There are several scenarios for single block reads:

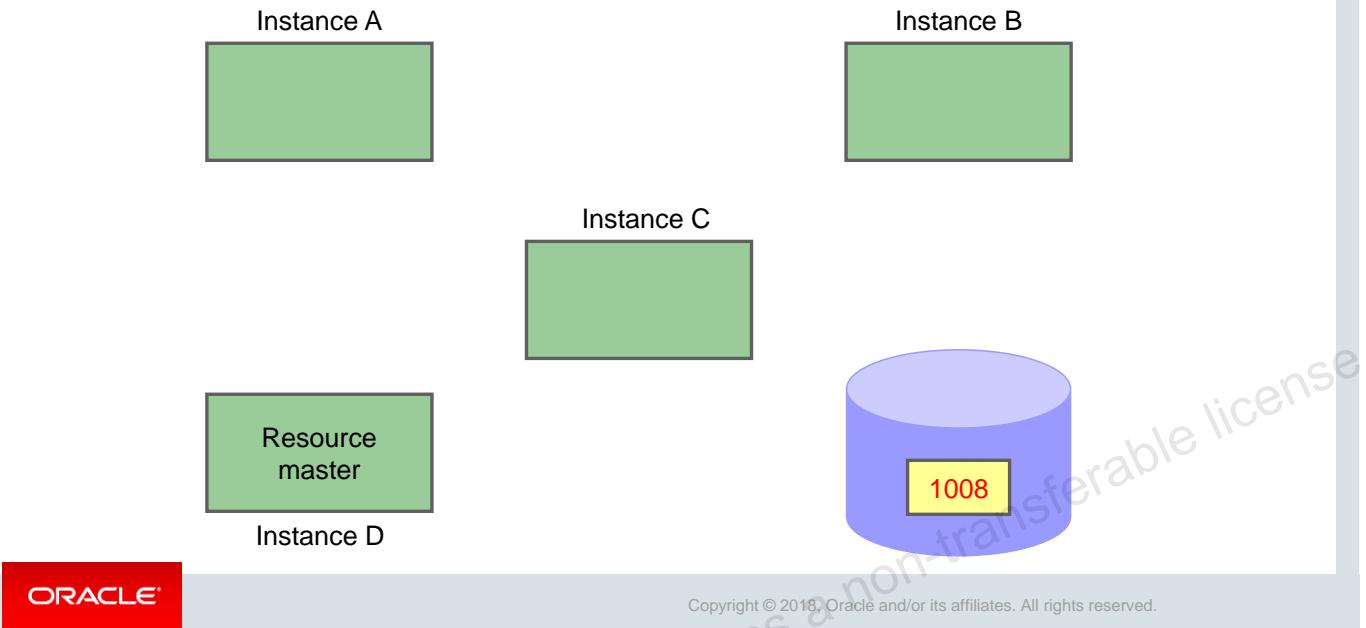
- Read from Disk
- Read – Read
- Read – Write
- Write – Write
- Write – Read
- Write to Disk



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

- **Read from Disk:** Occurs when an I/O request occurs for a block that has no image in any buffer cache
- **Read – Read:** Occurs when a block image exists in at least one buffer cache in shared current state (SCUR), and another instance wishes to access the block for read
- **Read – Write:** Occurs when a block image exists in at least one buffer cache in shared current state (SCUR), and another instance wishes to access the block for update (XCUR)
- **Write – Write:** Occurs when a block image exists in one buffer cache in exclusive current state (XCUR), and another instance wishes to access the same block for write in exclusive current state (XCUR)
- **Write – Read:** Occurs when a block image exists in one buffer cache in exclusive current state (XCUR), and another instance wishes to access the block for read. The instance doing the read may get it in CR or in SCUR as will be described later.
- **Write to Disk:** Occurs when DBWn writes a dirty buffer to disk. If the block was modified in multiple instances, then only the latest image will be written. This image will be (XCUR). All the older dirty images for the same block will be past images (PI).

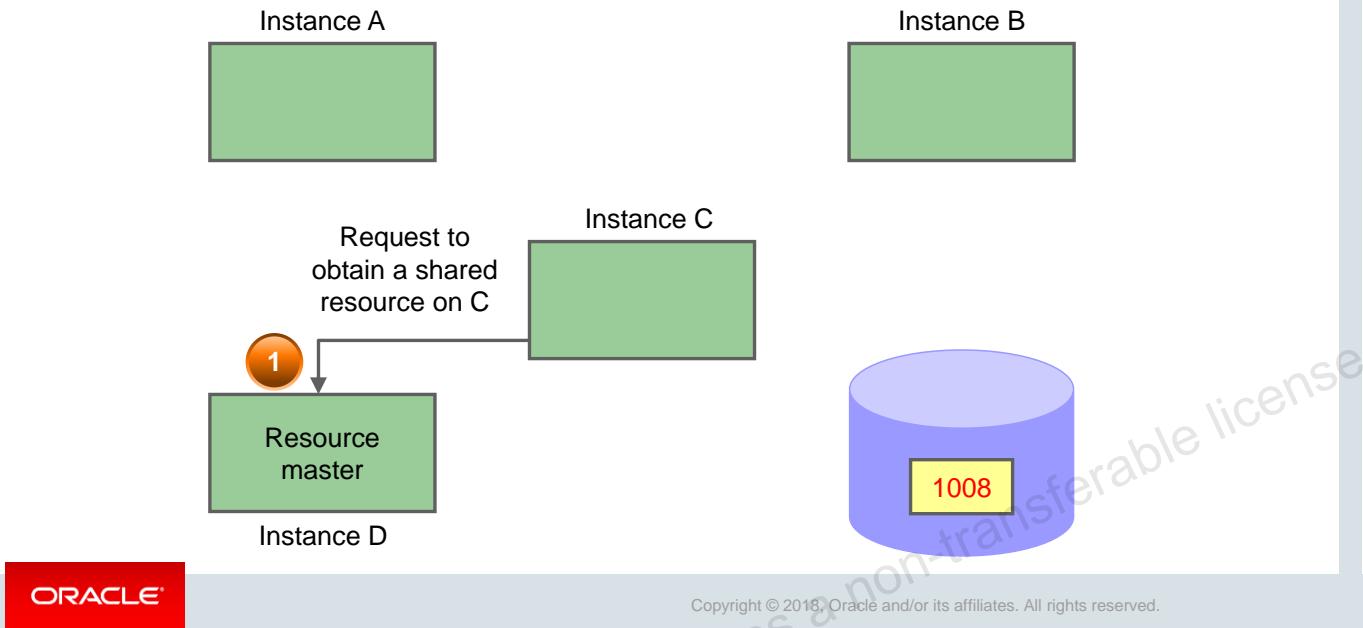
Global Cache Scenarios: Overview



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

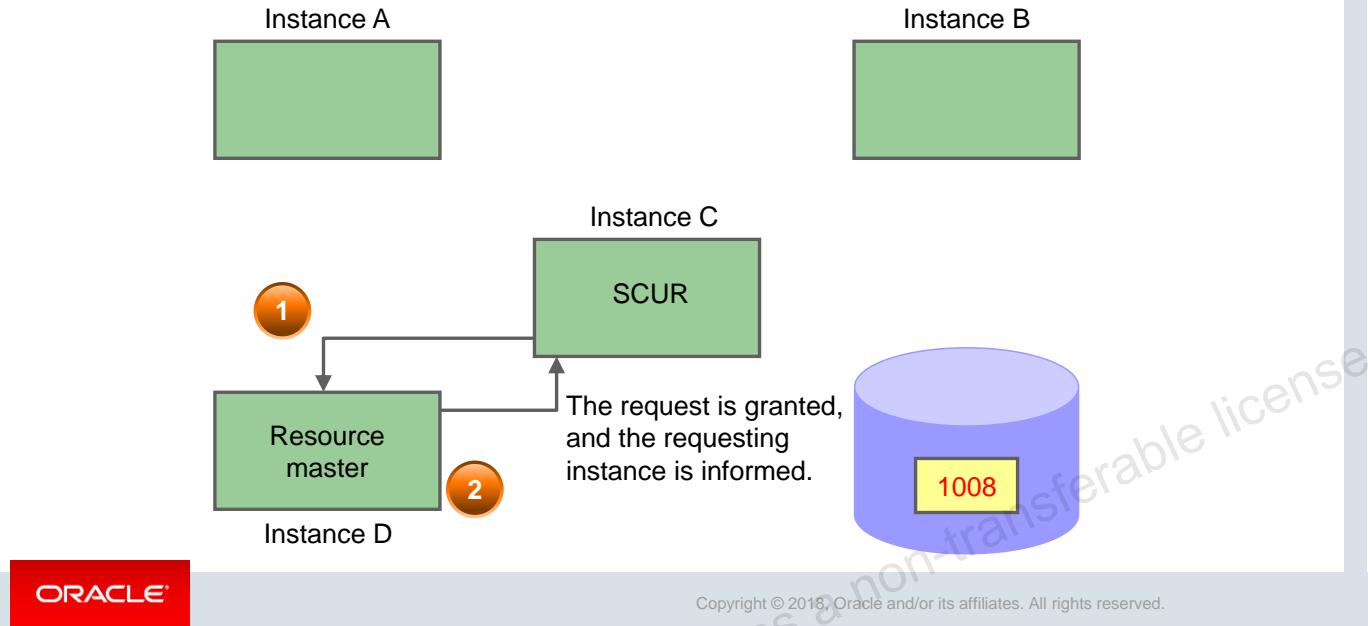
Scenario 1: Read from Disk



Instance C wishes to read a block in shared mode. The foreground process passes the request to a local LMS process, which passes it over the interconnect to an LMS process in the mastering instance for this block.

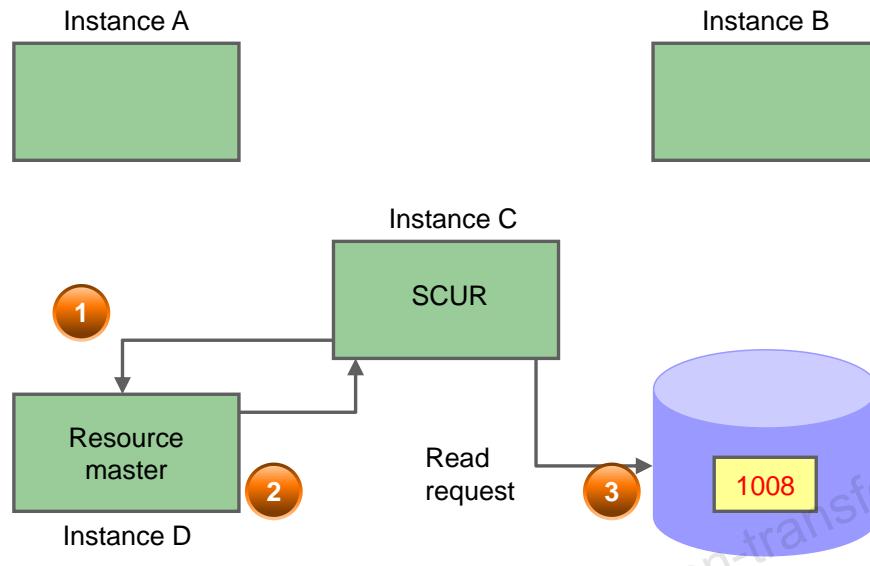
At the moment, there is no buffer in the buffer cache on instance C, nor in any other instance, containing a block image for the block.

Scenario 1: Read from Disk



The LMS process in the mastering instance updates master metadata in instance D and issues a “grant” to LMS on the requesting instance C, which creates a shadow metadata resource and notifies the foreground. The buffer header status column shows SCUR for shared current.

Scenario 1: Read from Disk

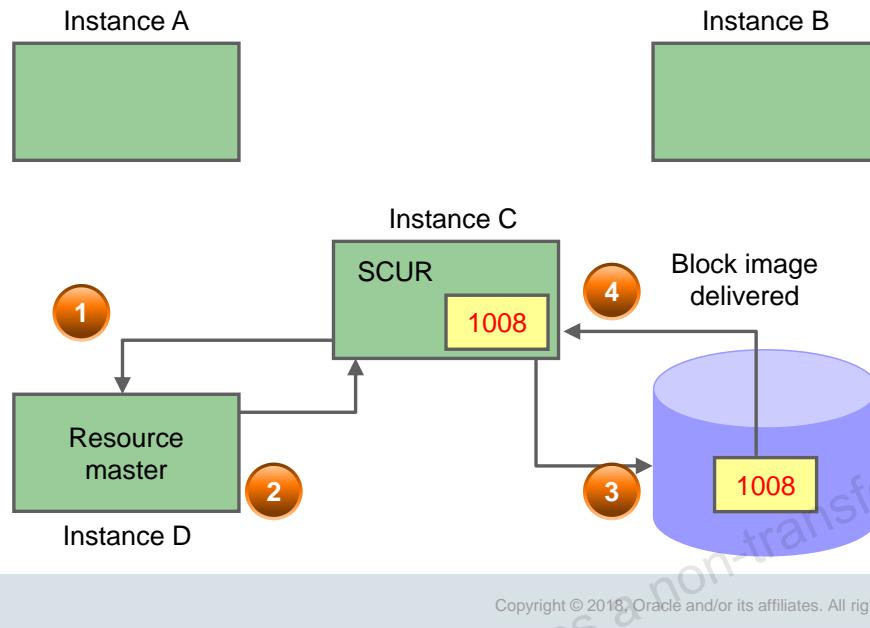


ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The foreground process on instance C then obtains an available buffer, and issues the I/O request.

Scenario 1: Read from Disk

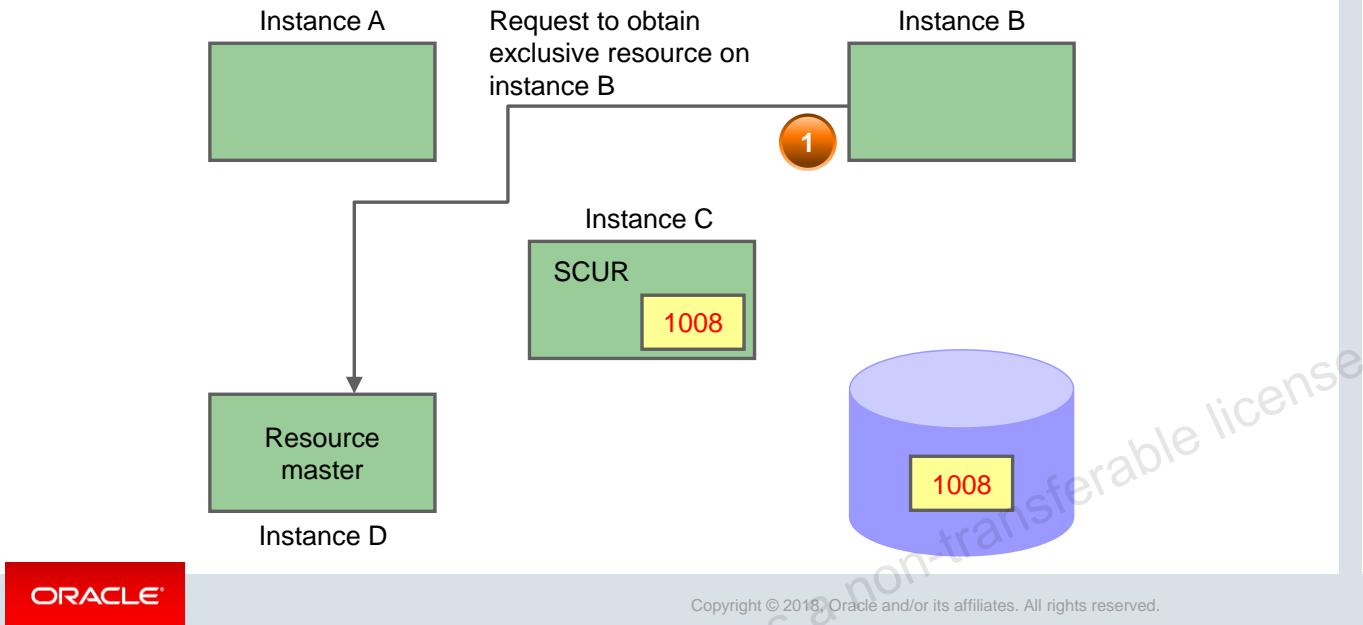


The status of this block image after I/O completes will be SCUR even though, currently, instance C is the only instance with an image of this block.

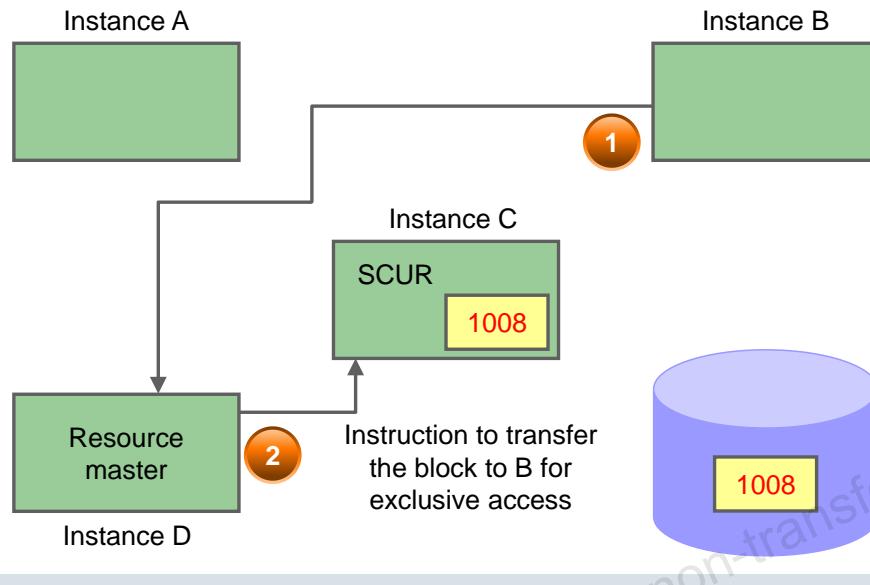
If other instances want to read the same block for read access, then the block image may be sent to them via cache fusion, and two or more instances may then have images of the same block in their buffer caches. If this occurs, then they will all be SCUR.

Note that the SCN for the block image in instance C matches the SCN of the block on disk.

Scenario 2: Read-Write Cache Fusion



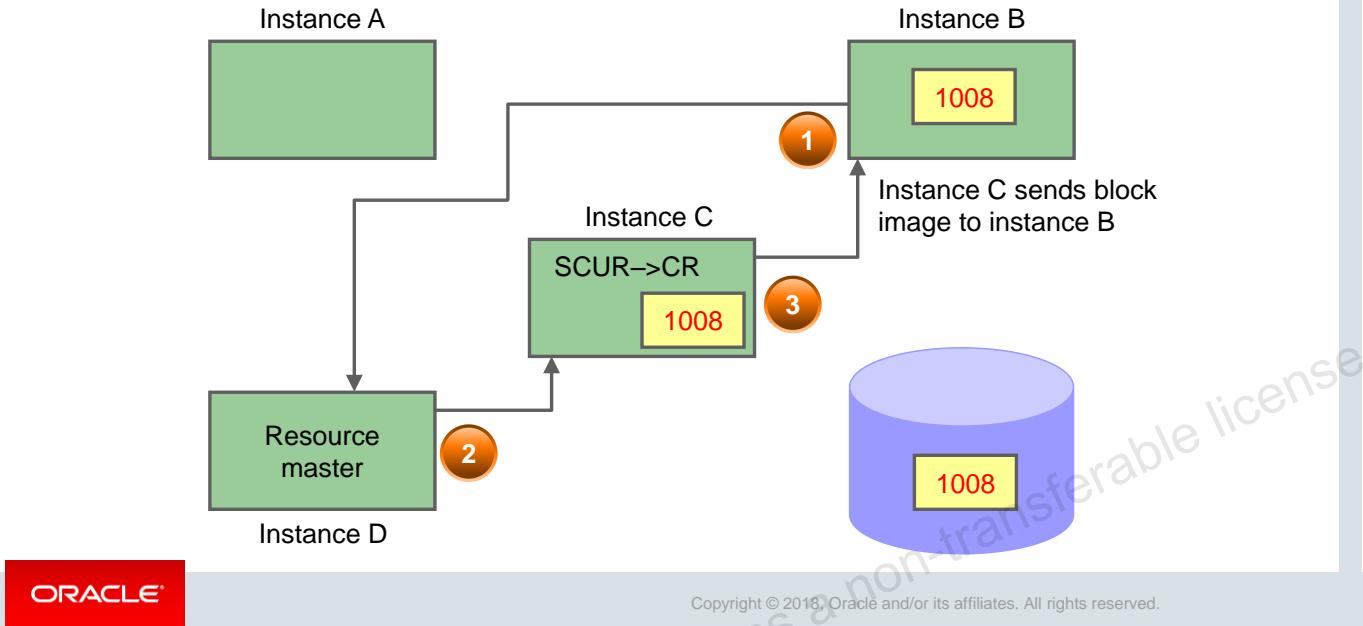
Scenario 2: Read-Write Cache Fusion



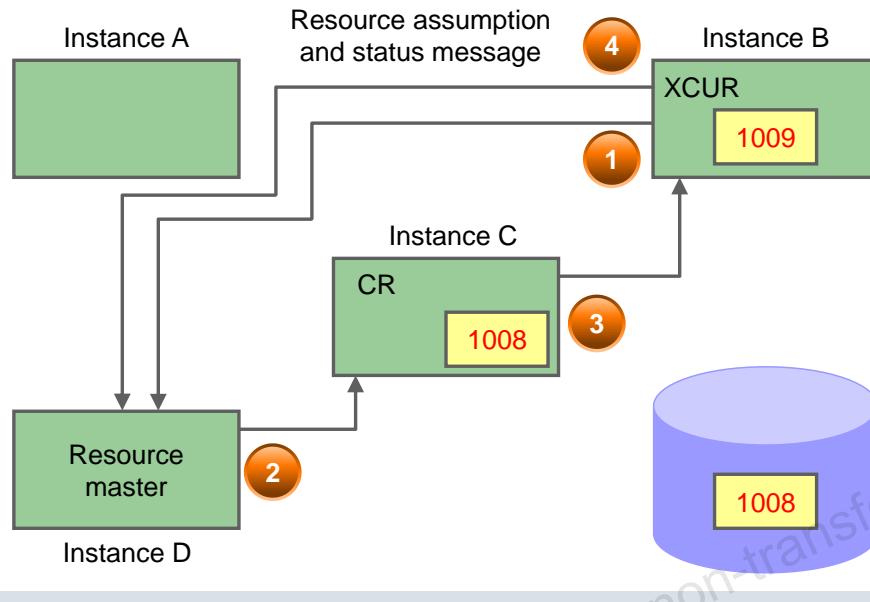
ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Scenario 2: Read-Write Cache Fusion



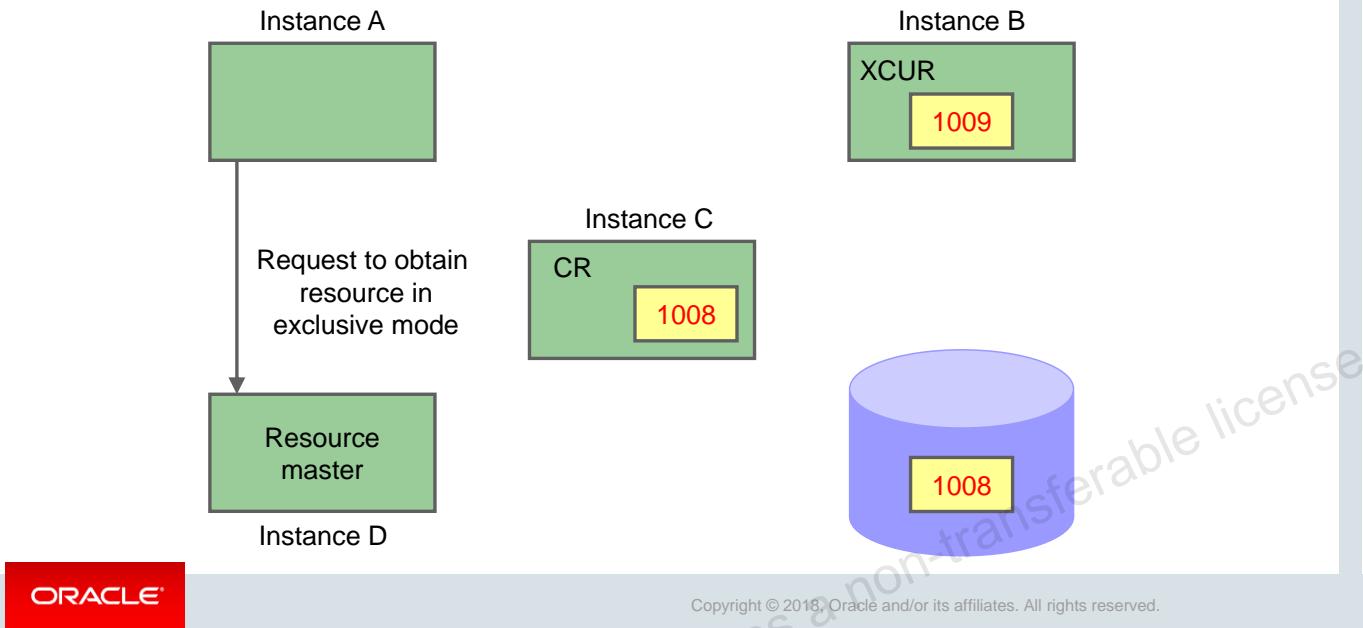
Scenario 2: Read-Write Cache Fusion



ORACLE®

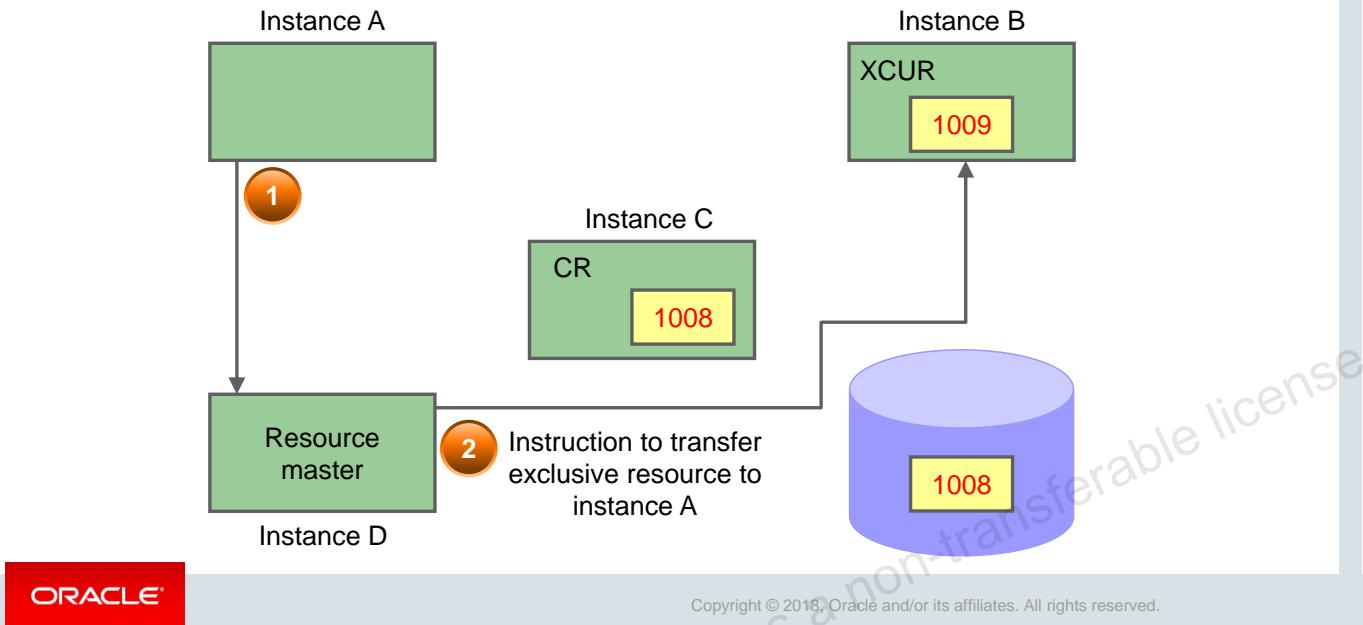
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Scenario 3: Write-Write Cache Fusion



Now a foreground process on instance A requests the local LMS on instance A to access the block for write. LMS on instance A sends a request to LMS on mastering instance D for access to the block in exclusive mode.

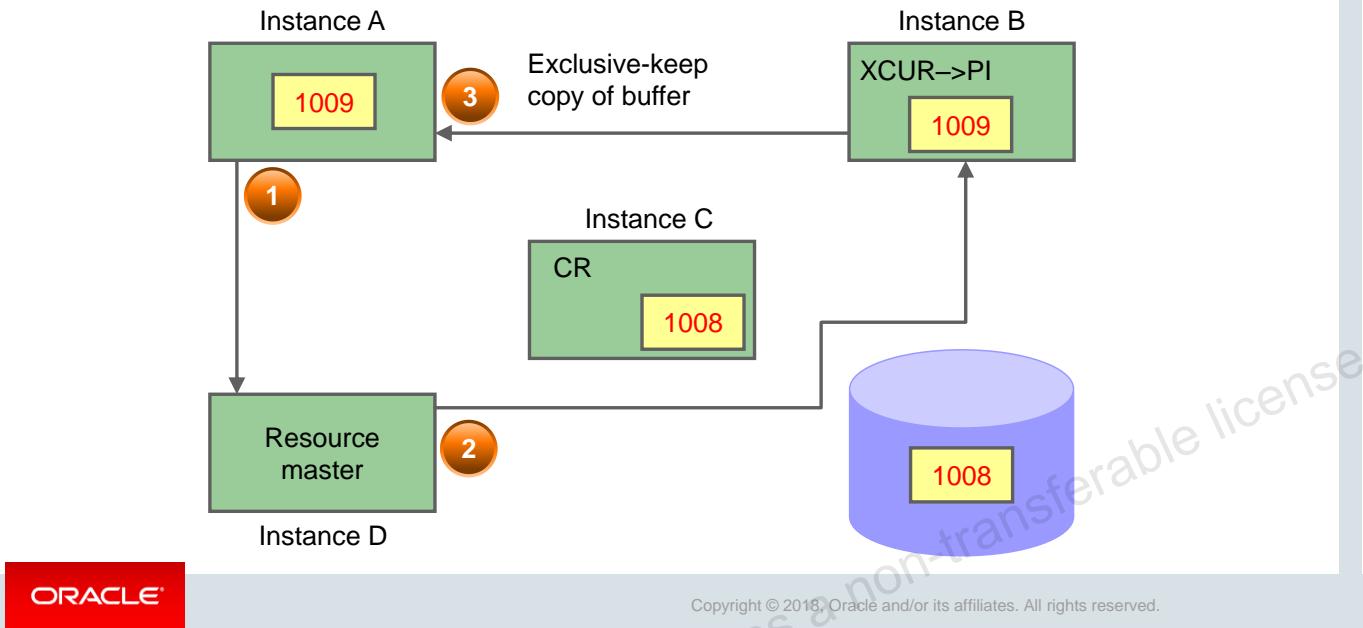
Scenario 3: Write-Write Cache Fusion



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Scenario 3: Write-Write Cache Fusion

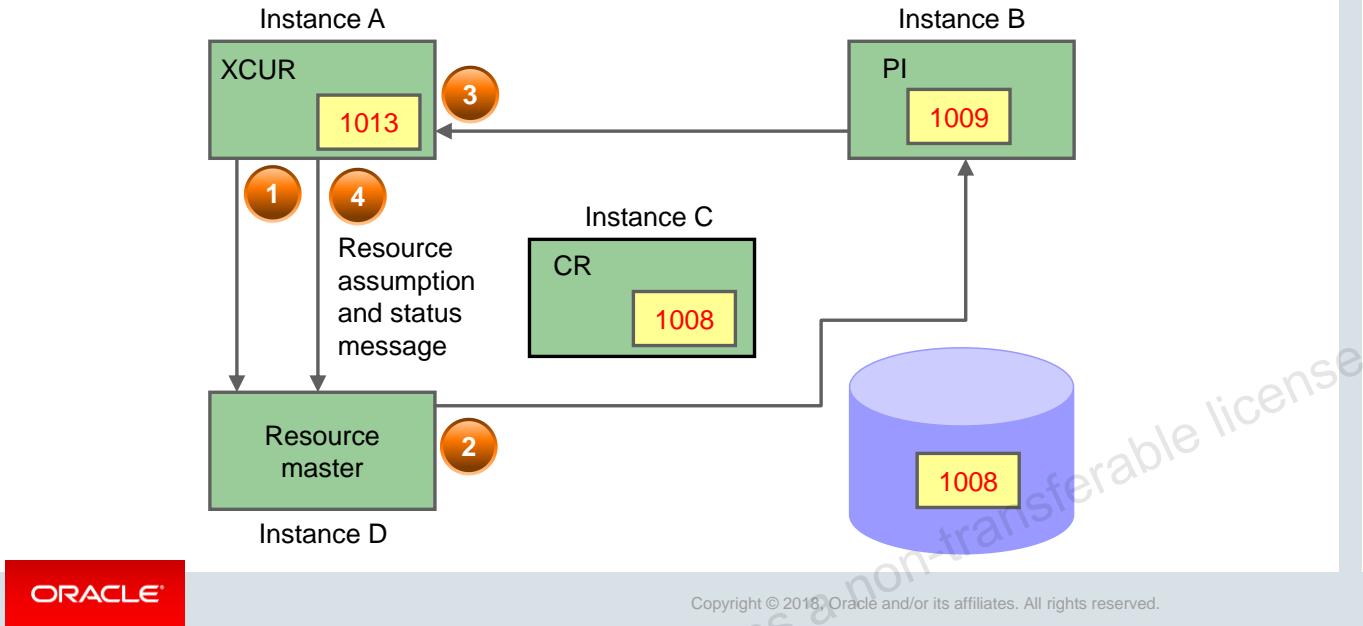


LMS on instance B then sends the block image to LMS on instance A, but retains the PI image in its own buffer cache. The past image of the block is held until one of the following occurs:

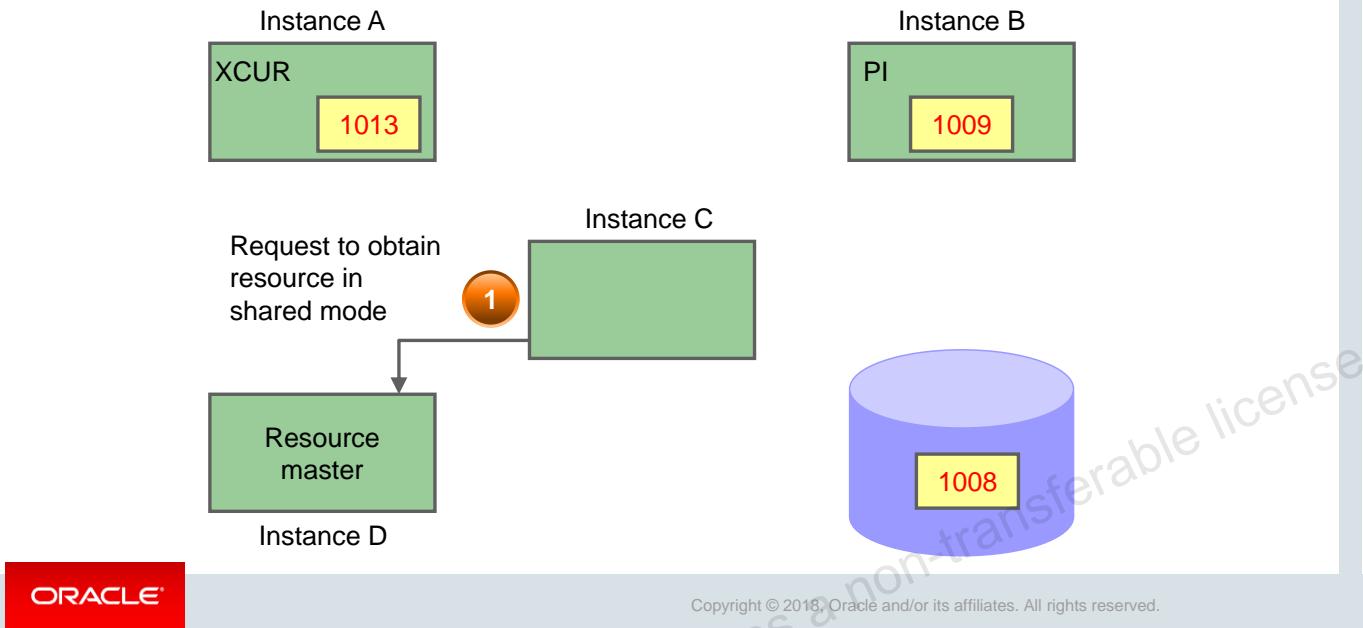
- A later XCUR version of the block is written by DBWn from an instance, at which time the PI image becomes a CR image.
- LMS instructs DBWn on instance B to write the PI image, due to instance recovery. After recovery is finished, the PI image becomes a CR image.

Note that at the moment, the SCN for the block images in instances A and B are both 1009, but this will change when the transaction in instance A updates the block image.

Scenario 3: Write-Write Cache Fusion



Scenario 4: Write-Read Cache Fusion

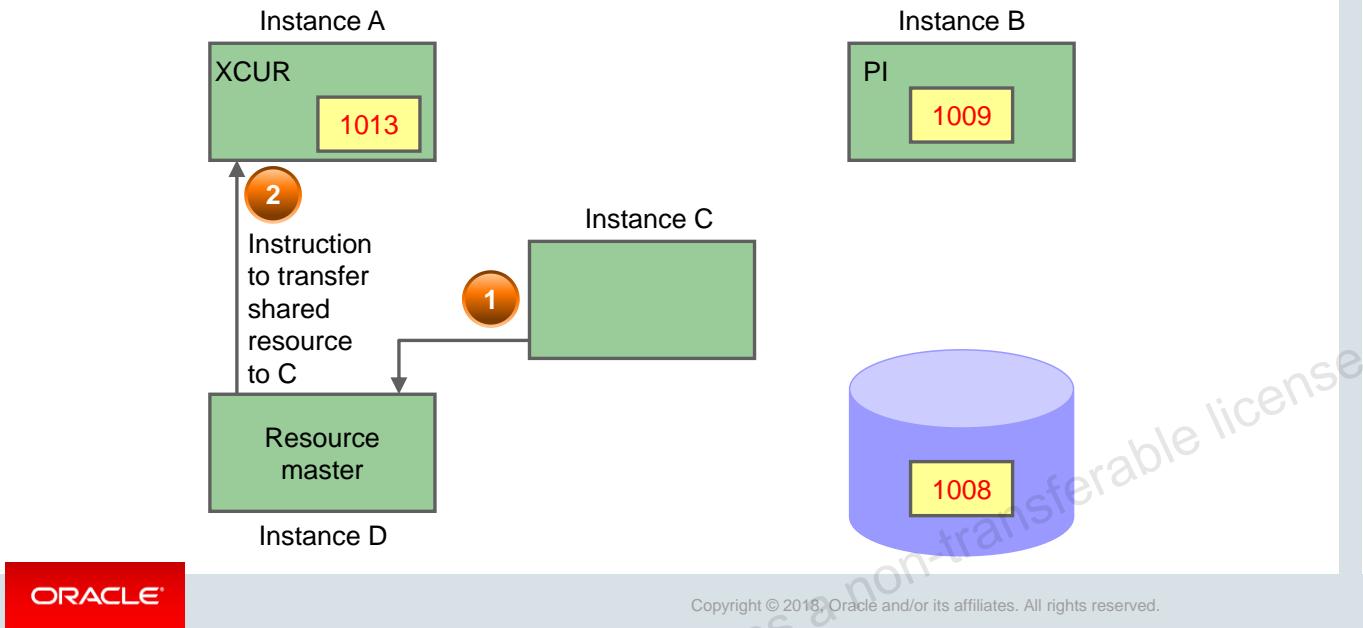


Now a foreground process on instance C requests local `LMS` on instance C to access the block for read. `LMS` on instance C sends a request to `LMS` on mastering instance D for access to the block in shared mode.

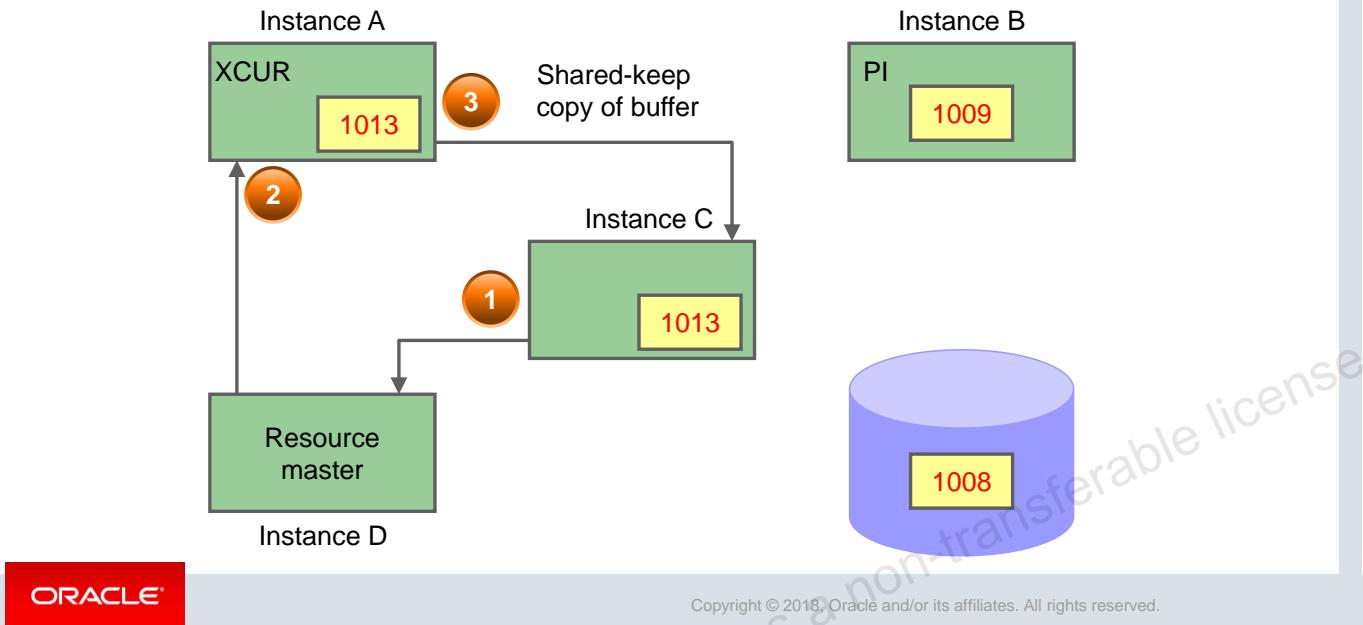
Note that the `PI` image in instance B may or may not have aged out at this point. This depends on whether `DBWn` in instance A has written the `XCUR` image to disk. If the `XCUR` image for the block in instance A, or a later `XCUR` image in any instance for the same block is written by `DBWn`, then the `PI` block image becomes a `CR` block image, and it might age out of the cache due to pressure on the replacement list for available buffers. This transition is controlled by the `LMSn` processes in the affected instances communicating with the `LMSn` process on the resource mastering instance.

Note: To simplify the slide, the `CR` image in instance C has aged out of the buffer cache and, therefore, is not present.

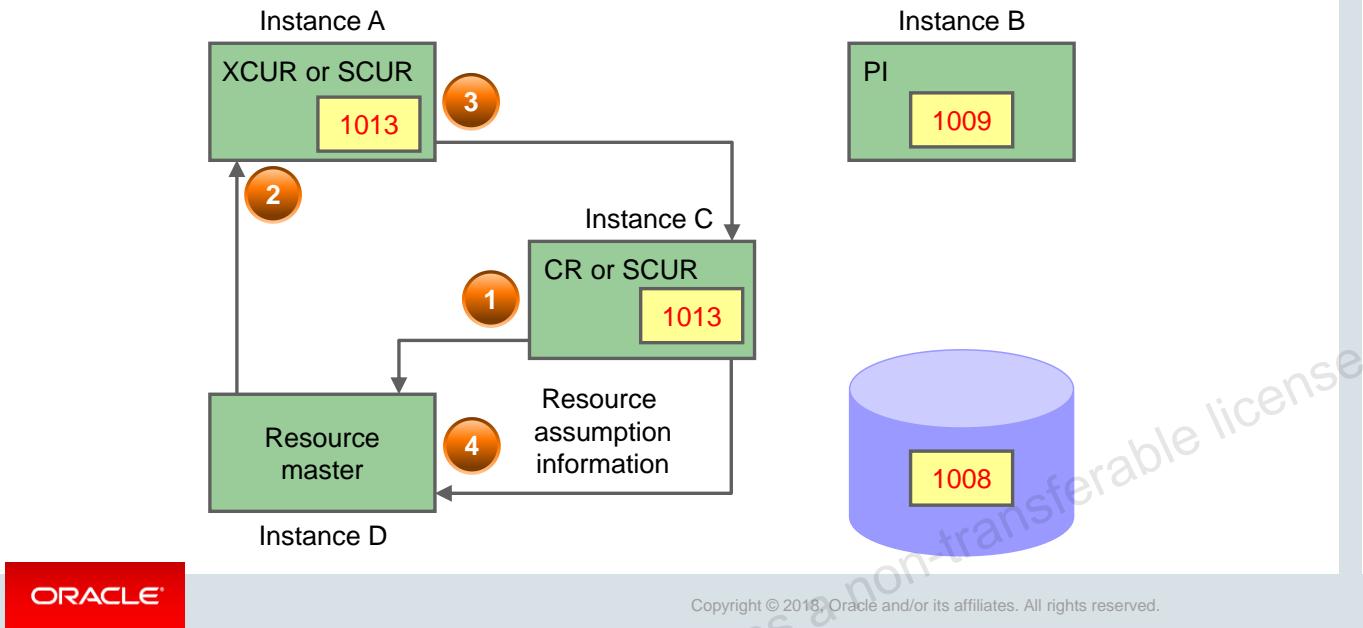
Scenario 4: Write-Read Cache Fusion



Scenario 4: Write-Read Cache Fusion



Scenario 4: Write-Read Cache Fusion



LMS on instance C now sends a status message to mastering instance D over the interconnect, which updates the mastering metadata.

LMS on instance C updates the shadowing metadata and posts the foreground process.

If no downgrade occurs, then the status is set to **CR** in the buffer header in instance C and remains **XCUR** in instance A.

Note that **CR** images may age out of buffer caches, and if the block remains **XCUR** in instance A, repeated cache fusion requests may occur, resulting in repeated construction and shipping of **CR** images to other instances for the same block. When enough such requests occur, LMS on the mastering instance D would request that instance A downgrade the block ownership to **SCUR** and ship the image to the other instance(s). The block would also be **SCUR** in the other instances.

Global Cache Management Scenarios for Multi-Block Reads

- When multi-block read requests occur:
 - The instance doing the I/O must acquire resources for each block in the correct state
 - LMSn coordination from the requesting instance to the LMSn on the mastering instance(s) happens
 - Different blocks in the same multi-block read may have different mastering instances
 - Dynamic remastering, described earlier, may help reduce the performance overhead
- There are several scenarios for multi-block reads:
 - No resource masters exist for any block.
 - Resource masters for some block(s), all are SCUR
 - Resource masters for some block(s), some are XCUR



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

No resource masters for any block in a particular multi-block read request: In this case, a request is made to the specific mastering instance for each block in the multi-block read and after being granted permission by LMSn, the server process does the multi-block read from disk.

Resource masters exist for at least one block in a particular multi-block read request, but it or they are Shared Current (SCUR): This means that the block has not been modified. In this case, a request is made to the specific mastering instance for each block in the multi-block read and, after being granted, the processing reads from disk.

Resource Masters exist for at least one block in a particular multi-block read request, but at least one is Exclusive Current (XCUR) and, therefore, a newer version may exist in a buffer cache than on disk. In this case, a request is made to the specific mastering instance for each block in the multi-block read and, after being granted, the XCUR images are transferred by cache fusion, as described earlier, and the remaining images are read from disk in smaller multi-block reads.

Useful Global Resource Management Views

- GV\$SESSION_WAIT
- GV\$SYSSTAT
- GV\$GES_STATISTICS
- V\$RESOURCE_LIMIT
- V\$BH
- GV\$LOCK
- V\$CR_BLOCK_SERVER
- V\$CURRENT_BLOCK_SERVER
- V\$INSTANCE_CACHE_TRANSFER
- V\$DYNAMIC_REMASTER_STATS
- GV\$RESULT_CACHE_STATS
- V\$GCSPFMASTER_INFO



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Quiz



Which statement about the Global Resource Directory is *not* true?

- a. Resource metadata is held in the Global Resource Directory (GRD).
- b. An object under global concurrency control is called an asset.
- c. Global enqueue resources are used for enqueues and locks.
- d. Global cache resources are used for buffer cache control.
- e. The GRD is distributed among all active instances of each database or ASM environment.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to describe:

- The need for global concurrency control
- Global Resource Directory
- How global resources are managed
- RAC global resource access coordination
 - Global enqueue and instance lock management
 - Global buffer cache management



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

RAC Database Monitoring and Tuning



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

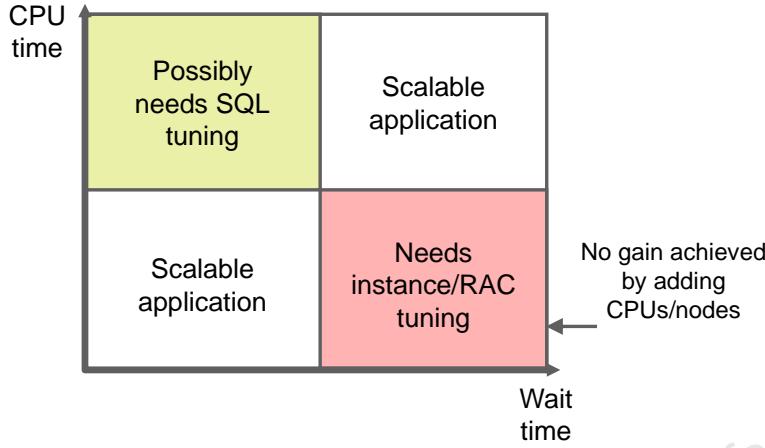
- Determine RAC-specific tuning components
- Determine RAC-specific wait events, global enqueues, and system statistics
- Implement the most common RAC tuning tips
- Use the Cluster Database Performance pages
- Use the Automatic Workload Repository (AWR) in RAC
- Use Automatic Database Diagnostic Monitor (ADDM) in RAC



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

CPU and Wait Time Tuning Dimensions



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

RAC-Specific Tuning

- Tune for a single instance first.
- Tune for RAC:
 - Instance recovery
 - Interconnect traffic
 - Point of serialization can be exacerbated.
- RAC-reactive tuning tools:
 - Specific wait events
 - System and enqueue statistics
 - Enterprise Manager performance pages
 - Statspack and AWR reports
- RAC-proactive tuning tools:
 - AWR snapshots
 - ADDM reports

} Certain combinations
are characteristic of
well-known tuning cases.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Analyzing Cache Fusion Impact in RAC

- The cost of block access and cache coherency is represented by:
 - Global Cache Services statistics
 - Global Cache Services wait events
- The response time for cache fusion transfers is determined by:
 - Overhead of the physical interconnect components
 - IPC protocol
 - GCS protocol
- The response time is not generally affected by disk I/O factors.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The effect of accessing blocks in the global cache and maintaining cache coherency is represented by:

- The Global Cache Services statistics for current and cr blocks—for example, gc current blocks received, gc cr blocks received, and so on
- The Global Cache Services wait events for gc current block 3-way, gc cr grant 2-way, and so on

The response time for cache fusion transfers is determined by the messaging time and processing time imposed by the physical interconnect components, the IPC protocol, and the GCS protocol. It is not affected by disk input/output (I/O) factors other than occasional log writes. The cache fusion protocol does not require I/O to data files to guarantee cache coherency, and RAC inherently does not cause any more I/O to disk than a nonclustered instance.

Typical Latencies for RAC Operations

AWR Report Latency Name	Lower Bound	Typical	Upper Bound
Average time to process cr block request	0.1	1	10
Avg global cache cr block receive time (ms)	0.3	4	12
Average time to process current block request	0.1	3	23
Avg global cache current block receive time(ms)	0.3	8	30

Global Cache and Enqueue Workload Characteristics											
#	CR Blocks						CU Blocks				
	GE Get Time (ms)	Receive Time (ms)	Build Time (ms)	Send Time (ms)	Flush Time (ms)	Log Flush CR Srvd %	Receive Time (ms)	Pin Time (ms)	Send Time (ms)	Flush Time (ms)	Log Flush CU Srvd %
3	2.03	7.46	0.00	0.00	30.72	2.68	6.38	0.11	0.00	22.30	5.72



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In a RAC AWR report, there is a table in the RAC Statistics section containing average times (latencies) for some Global Cache Services and Global Enqueue Services operations. This table is shown in the slide and is called “Global Cache and Enqueue Services: Workload Characteristics.” Those latencies should be monitored over time, and significant increases in their values should be investigated. The table presents some typical values, based on empirical observations. Factors that may cause variations to those latencies include:

- Utilization of the IPC protocol. User-mode IPC protocols are faster, but only Tru64’s RDG is recommended for use.
- Scheduling delays, when the system is under high CPU utilization
- Log flushes for current blocks served

Other RAC latencies in AWR reports are mostly derived from `V$GES_STATISTICS` and may be useful for debugging purposes, but do not require frequent monitoring.

Note: The time to process consistent read (CR) block request in the cache corresponds to (build time + flush time + send time), and the time to process current block request in the cache corresponds to (pin time + flush time + send time).

Wait Events for RAC

- Wait events help to analyze what sessions are waiting for.
- Wait times are attributed to events that reflect the outcome of a request:
 - Placeholders while waiting
 - Precise events after waiting
- Global cache waits are summarized in a broader category called Cluster Wait Class.
- These wait events are used in ADDM to enable cache fusion diagnostics.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Analyzing what sessions are waiting for is an important method to determine where time is spent. In RAC, the wait time is attributed to an event that reflects the exact outcome of a request. For example, when a session on an instance is looking for a block in the global cache, it does not know whether it will receive the data cached by another instance or whether it will receive a message to read from disk. The wait events for the global cache convey precise information and wait for global cache blocks or messages. They are mainly categorized by the following:

- Summarized in a broader category called Cluster Wait Class
- Temporarily represented by a placeholder event that is active while waiting for a block
- Attributed to precise events when the outcome of the request is known

The wait events for RAC convey information valuable for performance analysis. They are used in ADDM to enable precise diagnostics of the impact of cache fusion.

Wait Event Views

Total waits for an event	GV\$SYSTEM_EVENT
Waits for a wait event class by a session	GV\$SESSION_WAIT_CLASS
Waits for an event by a session	GV\$SESSION_EVENT
Activity of recent active sessions	GV\$ACTIVE_SESSION_HISTORY
Last 10 wait events for each active session	GV\$SESSION_WAIT_HISTORY
Events for which active sessions are waiting	GV\$SESSION_WAIT
Identify SQL statements impacted by interconnect latencies	GV\$SQLSTATS



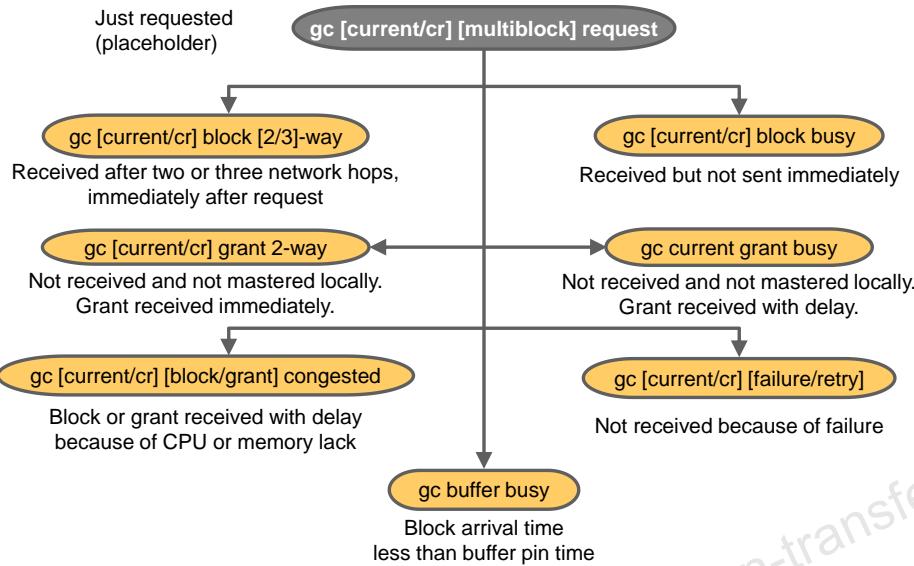
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When it takes some time to acquire resources because of the total path length and latency for requests, processes sleep to avoid spinning for indeterminate periods of time. When the process decides to wait, it wakes up either after a specified timer value expires (timeout) or when the event it is waiting for occurs and the process is posted. The wait events are recorded and aggregated in the views shown in the slide. The first three are aggregations of wait times, timeouts, and the number of times waited for a particular event, whereas the rest enable the monitoring of waiting sessions in real time, including a history of recent events waited for.

The individual events distinguish themselves by their names and the parameters that they assume. For most of the global cache wait events, the parameters include file number, block number, the block class, and access mode dispositions, such as mode held and requested. The wait times for events presented and aggregated in these views are very useful when debugging response time performance issues. Note that the time waited is cumulative, and that the event with the highest score is not necessarily a problem. However, if the available CPU power cannot be maximized, or response times for an application are too high, the top wait events provide valuable performance diagnostics.

Note: Use the CLUSTER_WAIT_TIME column in GV\$SQLSTATS to identify SQL statements impacted by interconnect latencies, or run an ADDM report on the corresponding AWR snapshot.

Global Cache Wait Events: Overview



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The main global cache wait events are described briefly in the slide:

- **gc current/cr request:** These wait events are relevant only while a gc request for a cr block or current buffer is in progress. They act as placeholders until the request completes.
- **gc [current/cr] block [2/3]-way:** A current or cr block is requested and received after two or three network hops. The request is processed immediately; the block is not busy or congested.
- **gc [current/cr] block busy:** A current or cr block is requested and received, but is not sent immediately by LMS because some special condition that delayed the sending was found.
- **gc [current/cr] grant 2-way:** A current or cr block is requested and a grant message received. The grant is given without any significant delays. If the block is not in its local cache, a current or cr grant is followed by a disk read on the requesting instance.
- **gc current grant busy:** A current block is requested and a grant message received. The busy hint implies that the request is blocked because others are ahead of it or it cannot be handled immediately.

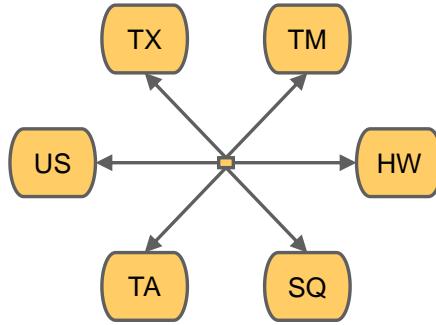
Note: For dynamic remastering, two events are of most importance: `gc remaster` and `gc quiesce`. They can be symptoms of the impact of remastering on the running processes.

- **gc [current/cr] [block/grant] congested:** A current or cr block is requested and a block or grant message is received. The congested hint implies that the request spent more than 1 ms in internal queues.
- **gc [current/cr] [failure/retry]:** A block is requested and a failure status received or some other exceptional event has occurred.
- **gc buffer busy:** If the time between buffer accesses becomes less than the time the buffer is pinned in memory, the buffer containing a block is said to become busy and as a result interested users may have to wait for it to be unpinned.

Note: For more information, refer to *Oracle Database Reference*.

Global Enqueue Waits

- Enqueues are synchronous.
- Enqueues are global resources in RAC.
- The most frequent waits are for:



- The waits may constitute serious serialization points.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

An enqueue wait is not RAC specific, but involves a global lock operation when RAC is enabled. Most of the global requests for enqueues are synchronous, and foreground processes wait for them. Therefore, contention on enqueues in RAC is more visible than in single-instance environments. Most waits for enqueues occur for enqueues of the following types:

- **TX:** Transaction enqueue; used for transaction demarcation and tracking
- **TM:** Table or partition enqueue; used to protect table definitions during DML operations
- **HW:** High-water mark enqueue; acquired to synchronize a new block operation
- **SQ:** Sequence enqueue; used to serialize incrementing of an Oracle sequence number
- **US:** Undo segment enqueue; mainly used by the Automatic Undo Management (AUM) feature
- **TA:** Enqueue used mainly for transaction recovery as part of instance recovery

In all of the preceding cases, the waits are synchronous and may constitute serious serialization points that can be exacerbated in a RAC environment.

Note: The enqueue wait events specify the resource name and a reason for the wait—for example, “TX Enqueue index block split.” This makes diagnostics of enqueue waits easier.

Session and System Statistics

- Use GV\$SYSSTAT to characterize the workload.
- Use GV\$SESSTAT to monitor important sessions.
- GV\$SEGMENT_STATISTICS includes RAC statistics.
- RAC-relevant statistic groups are:
 - Global Cache Service statistics
 - Global Enqueue Service statistics
 - Statistics for messages sent
- GV\$ENQUEUE_STATISTICS determines the enqueue with the highest impact.
- GV\$INSTANCE_CACHE_TRANSFER breaks down GCS statistics into block classes.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Using system statistics based on GV\$SYSSTAT enables characterization of the database activity based on averages. It is the basis for many metrics and ratios used in various tools and methods, such as AWR, Statspack, and Database Control.

To drill down to individual sessions or groups of sessions, GV\$SESSTAT is useful when the important session identifiers to monitor are known. Its usefulness is enhanced if an application fills in the MODULE and ACTION columns in GV\$SESSION.

GV\$SEGMENT_STATISTICS is useful for RAC because it also tracks the number of CR and current blocks received by the object.

The RAC-relevant statistics can be grouped into:

- **Global Cache Service statistics:** *gc cr blocks received, gc cr block receive time, and so on*
- **Global Enqueue Service statistics:** *global enqueue gets, and so on*
- **Statistics for messages sent:** *gcs messages sent and ges messages sent*

GV\$ENQUEUE_STATISTICS can be queried to determine which enqueue has the highest impact on database service times and, eventually, response times.

GV\$INSTANCE_CACHE_TRANSFER indicates how many current and CR blocks per block class are received from each instance, including how many transfers incurred a delay.

Note: For more information about statistics, refer to *Oracle Database Reference*.

Most Common RAC Tuning Tips

Application tuning is often the most beneficial!

- Reduce long full-table scans in OLTP systems.
- Use Automatic Segment Space Management (ASSM).
- Increase sequence caches.
- Use partitioning to reduce interinstance traffic.
- Avoid unnecessary parsing.
- Minimize locking usage.
- Remove unselective indexes.
- Configure interconnect properly.
- Employ In Memory-Parallel Query



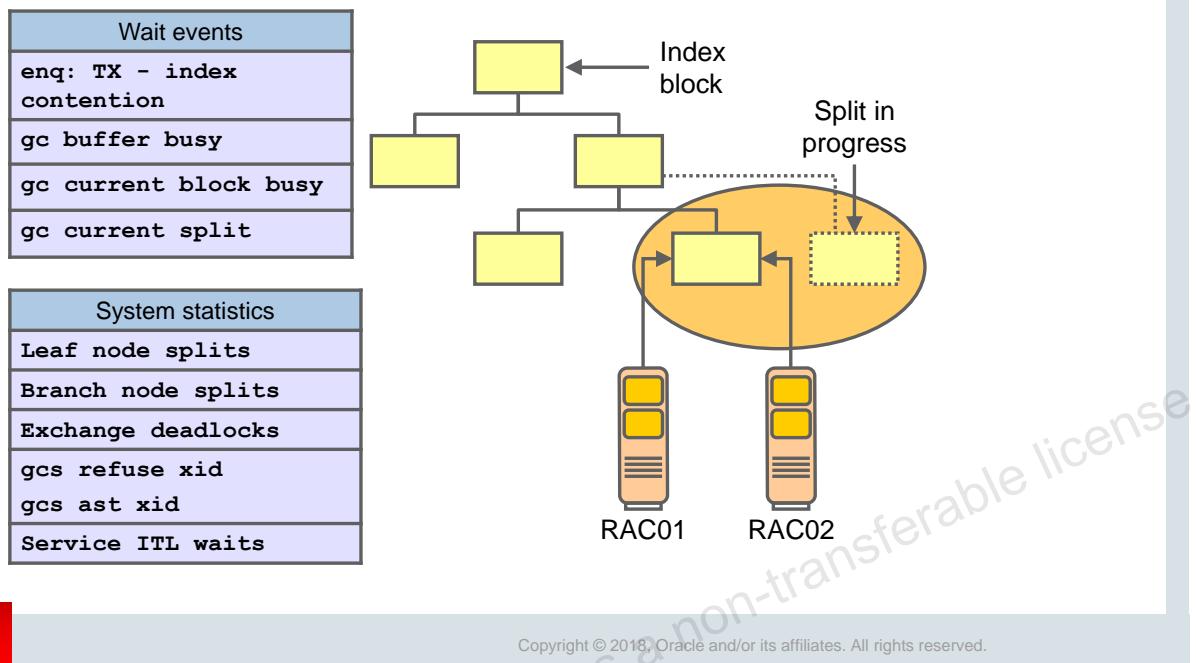
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In any database system, RAC or single-instance, the most significant performance gains are usually obtained from traditional application-tuning techniques. The benefits of those techniques are even more remarkable in a RAC database. In addition to traditional application tuning, some of the techniques that are particularly important for RAC include the following:

- Try to avoid long full-table scans to minimize GCS requests. The overhead caused by the global CR requests in this scenario is because when queries result in local cache misses, an attempt is first made to find the data in another cache, based on the assumption that the chance is high that another instance has cached the block.
- Automatic Segment Space Management can provide instance affinity to table blocks.
- Increasing sequence caches improves instance affinity to index keys deriving their values from sequences. That technique may result in significant performance gains for multi-instance insert-intensive applications.
- Range or list partitioning may be very effective in conjunction with data-dependent routing, if the workload can be directed to modify a particular range of values from a particular instance.
- Hash partitioning may help to reduce buffer busy contention by making buffer access distribution patterns sparser, enabling more buffers to be available for concurrent access.
- In RAC, library cache and row cache operations are globally coordinated. So, excessive parsing means additional interconnect traffic. Library cache locks are heavily used, in particular by applications that use PL/SQL or Advanced Queuing. Library cache locks are acquired in exclusive mode whenever a package or procedure has to be recompiled.
- Because transaction locks are globally coordinated, they also deserve special attention in RAC. For example, using tables instead of Oracle sequences to generate unique numbers is not recommended because it may cause severe contention even for a single-instance system.

- Indexes that are not selective do not improve query performance, but can degrade DML performance. In RAC, unselective index blocks may be subject to inter-instance contention, increasing the frequency of cache transfers for indexes belonging to insert-intensive tables.
- Always verify that you use a private network for your interconnect, and that your private network is configured properly. Ensure that a network link is operating in full duplex mode. Ensure that your network interface and Ethernet switches support MTU size of 9 KB. Note that a single-gigabit Ethernet interface can scale up to ten thousand 8 KB blocks per second before saturation.

Index Block Contention: Considerations



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

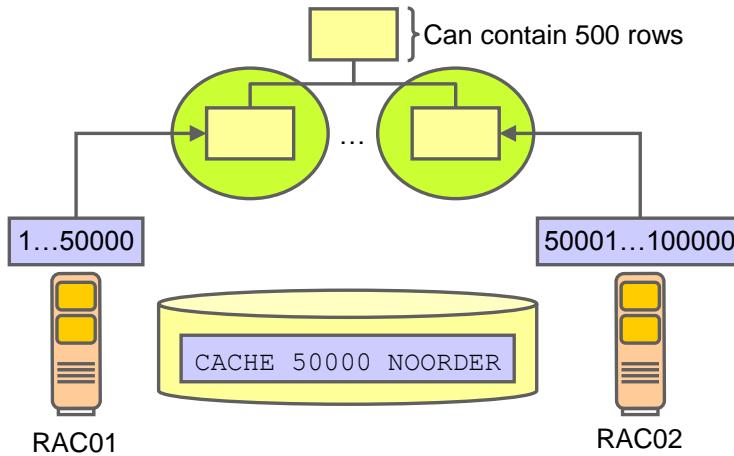
In application systems where the loading or batch processing of data is a dominant business function, there may be performance issues affecting response times because of the high volume of data inserted into indexes. Depending on the access frequency and the number of processes concurrently inserting data, indexes can become hot spots and contention can be exacerbated by:

- Ordered, monotonically increasing key values in the index (right-growing trees)
- Frequent leaf block splits
- Low tree depth: All leaf block access goes through the root block.

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 tables in the slide sum up the most common symptoms associated with the splitting of index blocks, listing wait events and statistics that are commonly elevated when index block splits are prevalent. As a general recommendation, to alleviate the performance impact of globally hot index blocks and leaf block splits, a more uniform, less skewed distribution of the concurrency in the index tree should be the primary objective. This can be achieved by:

- Global index hash partitioning
- Increasing the sequence cache, if the key value is derived from a sequence
- Using natural keys as opposed to surrogate keys
- Using reverse key indexes

Oracle Sequences and Index Contention



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Indexes with key values generated by sequences tend to be subject to leaf block contention when the insert rate is high. That is because the index leaf block holding the highest key value is changed for every row inserted, as the values are monotonically ascending. In RAC, this may lead to a high rate of current and CR blocks transferred between nodes.

One of the simplest techniques that can be used to limit this overhead is to increase the sequence cache, if you are using Oracle sequences. Because the difference between sequence values generated by different instances increases, successive index block splits tend to create instance affinity to index leaf blocks. For example, suppose that an index key value is generated by a `CACHE NOORDER` sequence and each index leaf block can hold 500 rows. If the sequence cache is set to 50000, while instance 1 inserts values 1, 2, 3, and so on, instance 2 concurrently inserts 50001, 50002, and so on. After some block splits, each instance writes to a different part of the index tree.

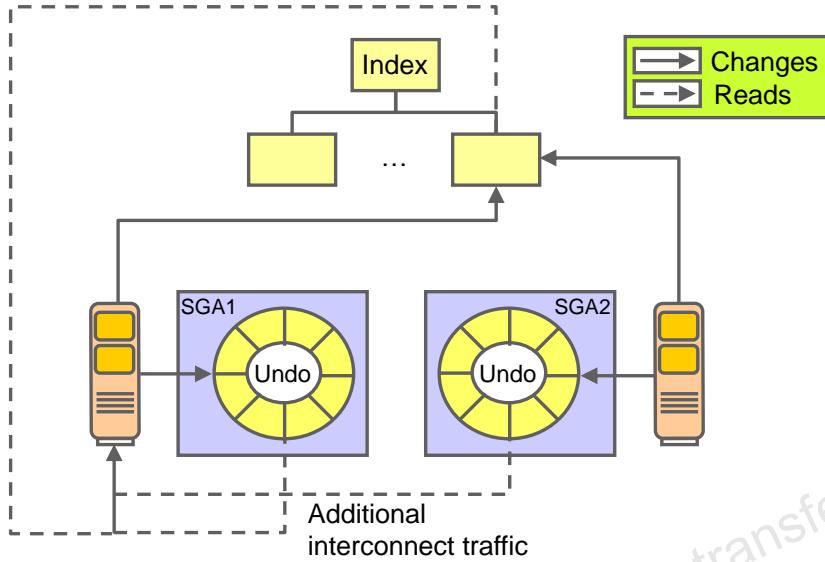
So, what is the ideal value for a sequence cache to avoid inter-instance leaf index block contention, yet minimizing possible gaps? One of the main variables to consider is the insert rate: the higher it is, the higher must be the sequence cache. However, creating a simulation to evaluate the gains for a specific configuration is recommended.

Note: By default, the cache value is 20. Typically, 20 is too small for the preceding example.

If sequences are not implemented properly, performance can be negatively impacted. Setting the sequence `CACHE_VALUE` too low or overusing non-cached sequences can degrade performance.

If you use sequence numbers, then always use `CACHE` with the `NOORDER` option for optimal performance in sequence number generation. With the `CACHE` option, however, you may have gaps in the sequence numbers. If your environment cannot tolerate sequence number gaps, then use the `NOCACHE` option or consider pre-generating the sequence numbers. If your application requires sequence number ordering but can tolerate gaps, then use `CACHE` and `ORDER` to cache and order sequence numbers in Oracle RAC. If an application requires ordered sequence numbers without gaps, then use `NOCACHE` and `ORDER`. The `NOCACHE` and `ORDER` combination has the most negative effect on performance compared to other caching and ordering combinations.

Undo Block Considerations



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Excessive undo block shipment and contention for undo buffers usually happens when index blocks containing active transactions from multiple instances are read frequently.

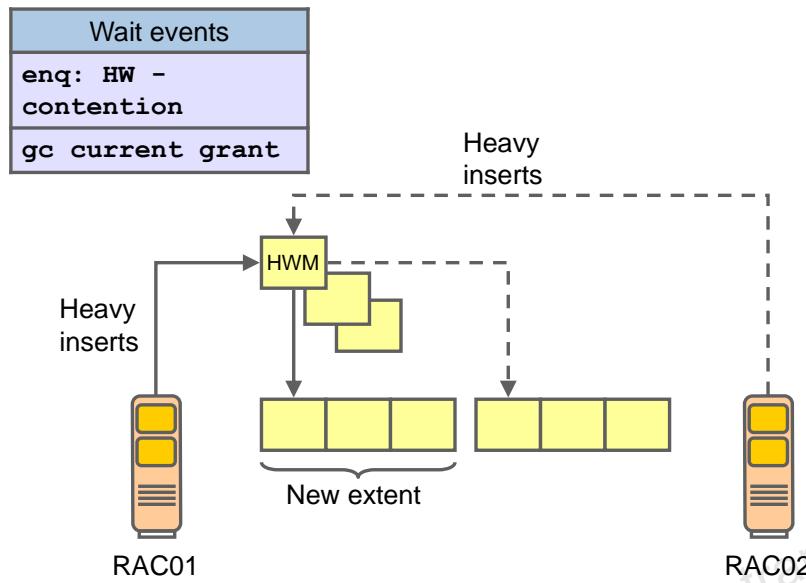
When a `SELECT` statement needs to read a block with active transactions, it has to undo the changes to create a CR version. If the active transactions in the block belong to more than one instance, there is a need to combine local and remote undo information for the consistent read. Depending on the amount of index blocks changed by multiple instances and the duration of the transactions, undo block shipment may become a bottleneck.

Usually this happens in applications that read recently inserted data very frequently, but commit infrequently. Techniques that can be used to reduce such situations include the following:

- Shorter transactions reduce the likelihood that any given index block in the cache contains uncommitted data, thereby reducing the need to access undo information for consistent read.
- As explained earlier, increasing sequence cache sizes can reduce inter-instance concurrent access to index leaf blocks. CR versions of index blocks modified by only one instance can be fabricated without the need of remote undo information.

Note: In RAC, the problem is exacerbated by the fact that a subset of the undo information has to be obtained from remote instances.

High-Water Mark Considerations



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

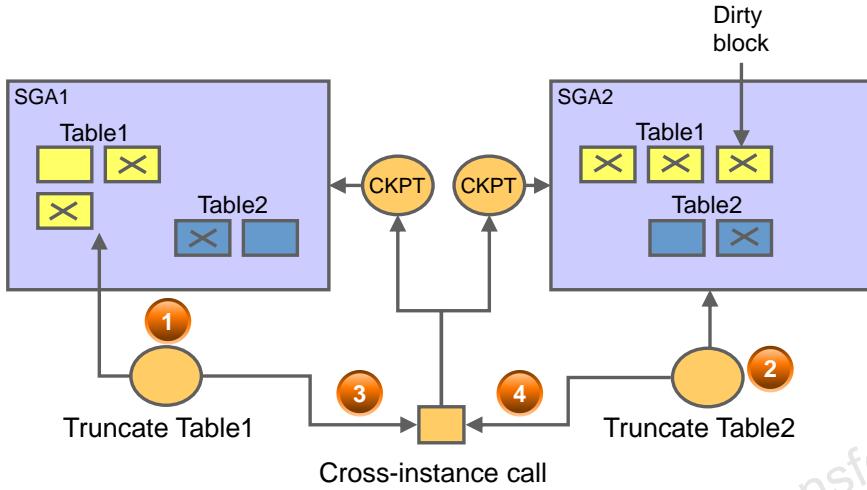
A certain combination of wait events and statistics presents itself in applications where the insertion of data is a dominant business function and new blocks have to be allocated frequently to a segment. If data is inserted at a high rate, new blocks may have to be made available after unfruitful searches for free space. This has to happen while holding the high-water mark (HWM) enqueue.

Therefore, the most common symptoms for this scenario include:

- A high percentage of wait time for enq: HW – contention
- A high percentage of wait time for gc current grant events

The former is a consequence of the serialization on the HWM enqueue, and the latter is because of the fact that current access to the new data blocks that need formatting is required for the new block operation. In a RAC environment, the length of this space management operation is proportional to the time it takes to acquire the HWM enqueue and the time it takes to acquire global locks for all the new blocks that need formatting. This time is small under normal circumstances because there is never any access conflict for the new blocks. Therefore, this scenario may be observed in applications with business functions requiring a lot of data loading, and the main recommendation to alleviate the symptoms is to define uniform and large extent sizes for the locally managed and automatic space-managed segments that are subject to high-volume inserts.

Concurrent Cross-Instance Calls: Considerations



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In data warehouse and data mart environments, it is not uncommon to see a lot of TRUNCATE operations. These essentially happen on tables containing temporary data.

In a RAC environment, truncating tables concurrently from different instances does not scale well, especially if, in conjunction, you are also using direct read operations such as parallel queries.

As shown in the slide, a truncate operation requires a cross-instance call to flush dirty blocks of the table that may be spread across instances. This constitutes a point of serialization. So, while the first TRUNCATE command is processing, the second has to wait until the first one completes.

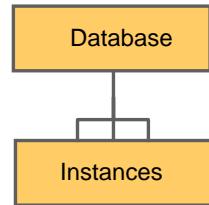
There are different types of cross-instance calls. However, all use the same serialization mechanism.

For example, the cache flush for a partitioned table with many partitions may add latency to a corresponding parallel query. This is because each cross-instance call is serialized at the cluster level, and one cross-instance call is needed for each partition at the start of the parallel query for direct read purposes.

Monitoring RAC Database and Cluster Performance

Directly from EM Cloud Control:

- View the status of each node in the cluster.
- View the aggregated alert messages across all the instances.
- Review the issues that are affecting the entire cluster or each instance.
- Monitor the cluster cache coherency statistics.
- Determine whether any of the services for the cluster database are having availability problems.
- Review any outstanding Clusterware interconnect alerts.

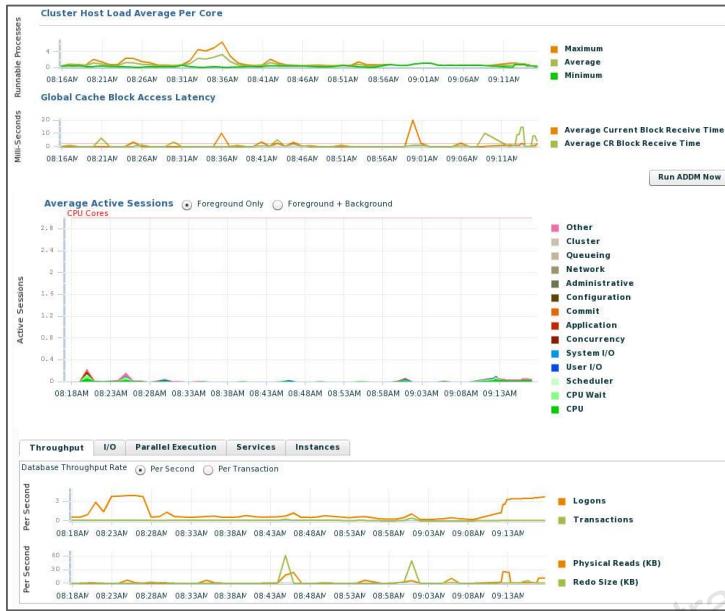


Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Both Oracle Enterprise Manager Database Control and Grid Control are cluster-aware and provide a central console to manage your cluster database. From the Cluster Database Home page, you can do all of the following:

- View the overall system status, such as the number of nodes in the cluster and their current status, so you do not have to access each individual database instance for details.
- View the alert messages aggregated across all the instances with lists for the source of each alert message.
- Review the issues that are affecting the entire cluster as well as those that are affecting individual instances.
- Monitor cluster cache coherency statistics to help you identify processing trends and optimize performance for your Oracle RAC environment. Cache coherency statistics measure how well the data in caches on multiple instances is synchronized.
- Determine whether any of the services for the cluster database are having availability problems. A service is deemed to be a problem service if it is not running on all preferred instances, if its response time thresholds are not met, and so on.
- Review any outstanding Clusterware interconnect alerts.

Cluster Database Performance Page



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

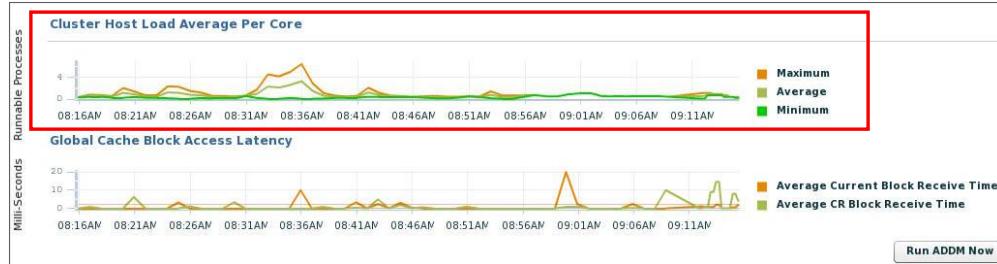
The Cluster Database Performance page provides a quick glimpse of the performance statistics for a database. Enterprise Manager accumulates data from each instance over specified periods of time, called collection-based data. Enterprise Manager also provides current data from each instance, known as real-time data.

Statistics are rolled up across all the instances in the cluster database. Using the links next to the charts, you can get more specific information and perform any of the following tasks:

- Identify the causes of performance issues.
- Decide whether resources need to be added or redistributed.
- Tune your SQL plan and schema for better optimization.
- Resolve performance issues.

The screenshot in the slide shows a partial view of the Cluster Database Performance page. You access this page by selecting Performance Home from the Performance pull-down menu from the Cluster Database Home page.

Determining Cluster Host Load Average



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

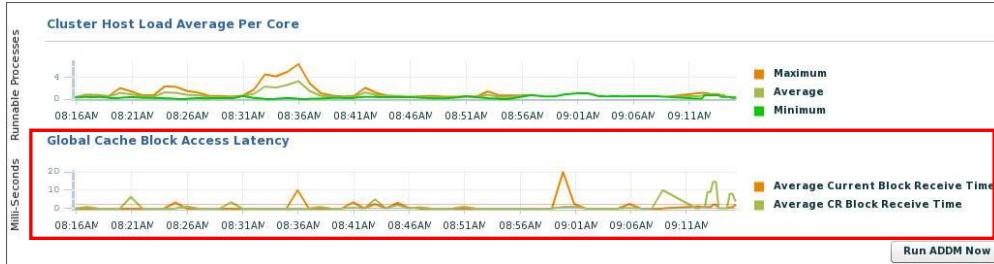
The Cluster Host Load Average chart in the Cluster Database Performance page shows potential problems that are outside the database. The chart shows maximum, average, and minimum load values for available nodes in the cluster for the previous hour.

If the load average is higher than the average of the total number of CPUs across all the hosts in the cluster, then too many processes are waiting for CPU resources. SQL statements that are not tuned often cause high CPU usage. Compare the load average values with the values displayed for CPU Used in the Average Active Sessions chart. If the sessions value is low and the load average value is high, this indicates that something else on the host, other than your database, is consuming the CPU.

You can click any of the load value labels for the Cluster Host Load Average chart to view more detailed information about that load value. For example, if you click the Average label, the Hosts: Average Load page appears, displaying charts that depict the average host load for up to four nodes in the cluster.

You can select whether the data is displayed in a summary chart, combining the data for each node in one display, or using tile charts, where the data for each node is displayed in its own chart. You can click Customize to change the number of tile charts displayed in each row or the method of ordering the tile charts.

Determining Global Cache Block Access Latency



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The Global Cache Block Access Latency chart shows the latency for each type of data block request: current and consistent-read (CR) blocks. That is the elapsed time it takes to locate and transfer consistent-read and current blocks between the buffer caches.

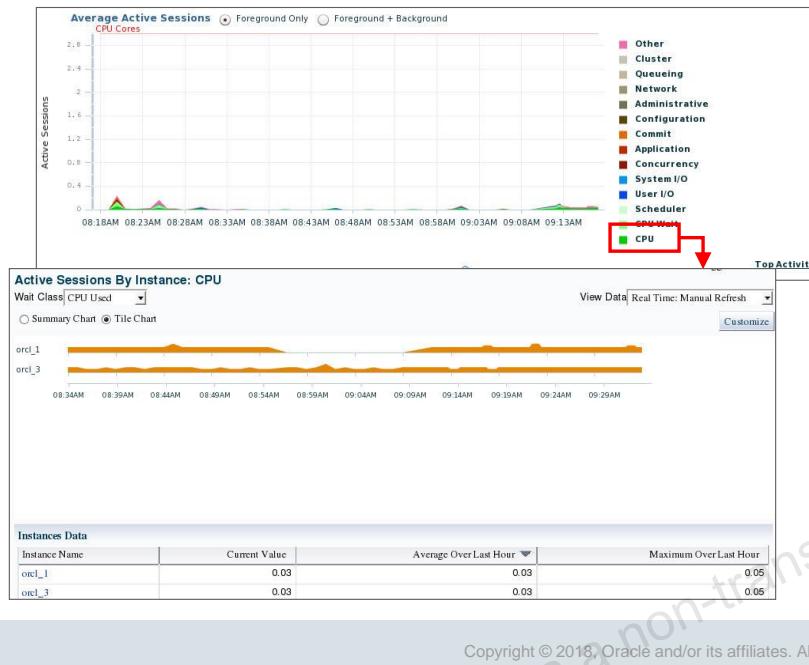
You can click either metric for the Global Cache Block Access Latency chart to view more detailed information about that type of cached block.

If the Global Cache Block Access Latency chart shows high latencies (high elapsed times), this can be caused by any of the following:

- A high number of requests caused by SQL statements that are not tuned
- A large number of processes in the queue waiting for the CPU, or scheduling delays
- Slow, busy, or faulty interconnects. In these cases, check your network connection for dropped packets, retransmittals, or cyclic redundancy check (CRC) errors.

Concurrent read and write activity on shared data in a cluster is a frequently occurring activity. Depending on the service requirements, this activity does not usually cause performance problems. However, when global cache requests cause a performance problem, optimizing SQL plans and the schema to improve the rate at which data blocks are located in the local buffer cache, and minimizing I/O is a successful strategy for performance tuning. If the latency for consistent-read and current block requests reaches 10 milliseconds, then see the Cluster Cache Coherency page for more detailed information.

Determining Average Active Sessions



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The Average Active Sessions chart on the Cluster Database Performance page shows potential problems inside the database. Categories, called wait classes, show how much of the database is using a resource, such as CPU or disk I/O. Comparing CPU time with wait time helps to determine how much of the response time is consumed with useful work rather than waiting for resources that are potentially held by other processes.

At the cluster database level, this chart shows the aggregate wait class statistics across all the instances. For a more detailed analysis, you can click the Clipboard icon at the bottom of the chart to view the ADDM analysis for the database for that time period.

If you click the wait class legends beside the Average Active Sessions chart, you can view instance-level information stored on the “Active Sessions by Instance” pages. You can use the Wait Class action list on the “Active Sessions by Instance” page to view the different wait classes. The “Active Sessions by Instance” pages show the service times for up to four instances. Using the Customize button, you can select the instances that are displayed. You can view the data for the instances separately by using tile charts, or you can combine the data into a single summary chart.

Determining Database Throughput



ORACLE®

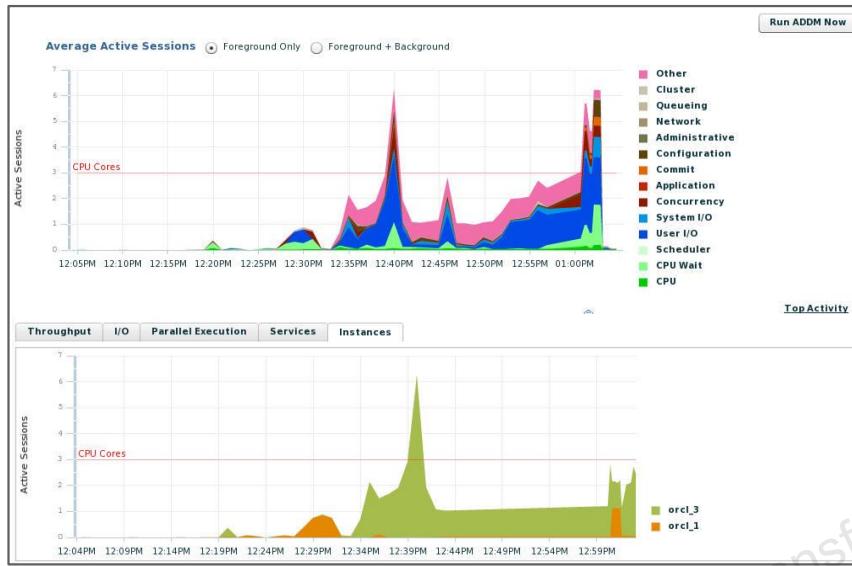
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The last chart on the Performance page monitors the usage of various database resources. Click the Throughput tab at the top of this chart to view the Database Throughput chart. Compare the peaks on the Average Active Sessions chart with those on the Database Throughput charts. If internal contention is high and throughput is low, consider tuning the database.

The Database Throughput charts summarize any resource contention that appears in the Average Active Sessions chart, and also show how much work the database is performing on behalf of the users or applications. The Per Second view shows the number of transactions compared to the number of logons, and (not shown here) the number of physical reads compared to the redo size per second. The Per Transaction view shows the number of physical reads compared to the redo size per transaction. Logons is the number of users that are logged on to the database.

To obtain information at the instance level, access the “Database Throughput by Instance” page by clicking one of the legends to the right of the charts. This page shows the breakdown of the aggregated Database Throughput chart for up to four instances. You can select the instances that are displayed. You can drill down further on the “Database Throughput by Instance” page to see the sessions of an instance consuming the greatest resources. Click an instance name legend under the chart to go to the Top Consumers page for that instance.

Determining Database Throughput



ORACLE®

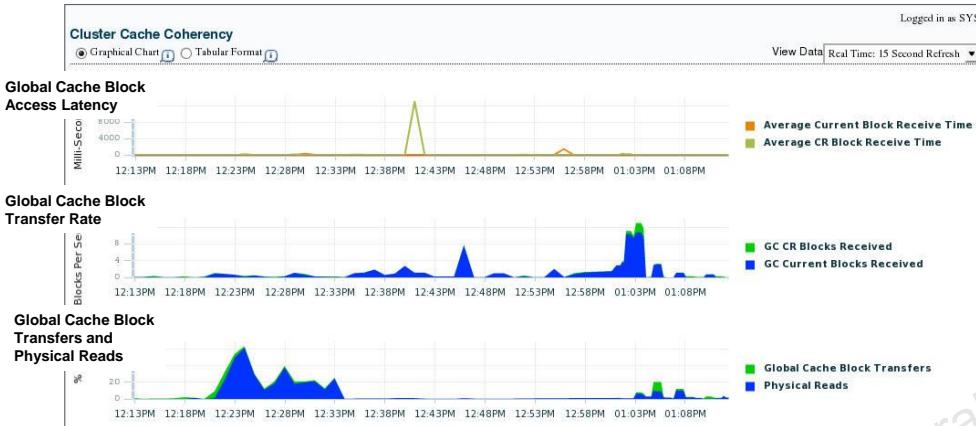
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The last chart on the Performance page monitors the usage of various database resources. By clicking the Instances tab at the top of this chart, you can view the “Active Sessions by Instance” chart.

The “Active Sessions by Instance” chart summarizes any resource contention that appears in the Average Active Sessions chart. Using this chart, you can quickly determine how much of the database work is being performed on each instance.

You can also obtain information at the instance level by clicking one of the legends to the right of the chart to access the Top Sessions page. On the Top Sessions page, you can view real-time data showing the sessions that consume the greatest system resources. In the graph in the slide, the orcl_3 instance after 12:30 PM is consistently showing more active sessions than the orcl_1 instance.

Accessing the Cluster Cache Coherency Page



ORACLE®

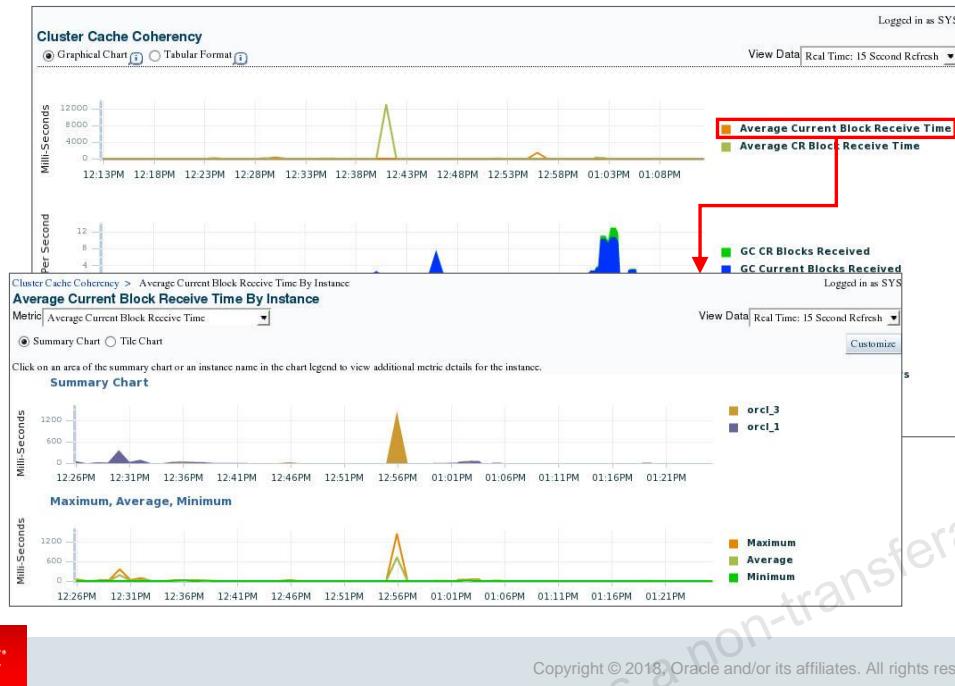
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To access the Cluster Cache Coherency page, click the Performance tab on the Cluster Database Home page, and click Cluster Cache Coherency in the Additional Monitoring Links section at the bottom of the page. Alternatively, click either of the legends to the right of the Global Cache Block Access Latency chart.

The Cluster Cache Coherency page contains summary charts for cache coherency metrics for the cluster:

- **Global Cache Block Access Latency:** Shows the total elapsed time, or latency, for a block request. Click one of the legends to the right of the chart to view the average time it takes to receive data blocks for each block type (current or CR) by instance. On the “Average Block Receive Time by Instance” page, you can click an instance legend under the chart to go to the “Block Transfer for Local Instance” page, where you can identify which block classes, such as undo blocks, data blocks, and so on, are subject to intense global cache activity. This page displays the block classes that are being transferred, and which instances are transferring most of the blocks. Cache transfer indicates how many current and CR blocks for each block class were received from remote instances, including how many transfers incurred a delay (busy) or an unexpected longer delay (congested).

Accessing the Cluster Cache Coherency Page



- **Global Cache Block Transfer Rate:** Shows the total aggregated number of blocks received by all instances in the cluster by way of an interconnect. Click one of the legends to the right of the chart to go to the “Global Cache Blocks Received by Instance” page for that type of block. From there, you can click an instance legend under the chart to go to the “Segment Statistics by Instance” page, where you can see which segments are causing cache contention.
- **Global Cache Block Transfers and Physical Reads:** Shows the percentage of logical read operations that retrieved data from the buffer cache of other instances by way of Direct Memory Access and from disk. It is essentially a profile of how much work is performed in the local buffer cache, rather than the portion of remote references and physical reads, which both have higher latencies. Click one of the legends to the right of the chart to go to the “Global Cache Block Transfers vs. Logical Reads by Instance” and “Physical Reads vs. Logical Reads by Instance” pages. From there, you can click an instance legend under the chart to go to the “Segment Statistics by Instance” page, where you can see which segments are causing cache contention.

Viewing the Database Locks Page

Cluster Database: RDBA > Logged in As SYS

Database Locks

Page Refreshed Aug 5, 2009 4:34:51 PM EDT [Refresh]

View All Database Locks ▾
 Blocking Locks
 User Locks
 Locks
[All Database Locks](#) [Kill Session](#) [Session Details](#) [View Object](#) [View SQL](#)

Expand All | Collapse All

Select Username	Sessions Blocked Name	Instance ID	Session ID	Serial Number	Process ID	SQL Hash Value	Lock Type	Mode Held	Mode Requested	Object Type	Object Owner	Object Name	ROWID	Time in current mode (seconds)
All Database Locks														
GEN0	O RDBA2	4	1	4100			XR	NULL	NONE					689986
GEN0	O RDBA1	4	1	6221			XR	NULL	NONE					689902
DBW0	O RDBA1	17	1	6249			RT	EXCLUSIVE	NONE					689994
DBW0	O RDBA1	17	1	6249			DM	SHARE	NONE					689993
DBW0	O RDBA2	18	1	4137			DM	SHARE	NONE					689974
DBW0	O RDBA2	18	1	4137			RT	EXCLUSIVE	NONE					689975
DBSNMP	O RDBA2	34	45336	24475	1tu4ybcx7vks0		PS	SHARE	NONE					0
DBSNMP	O RDBA1	41	3015	18368			AE	SHARE	NONE					428174
DBSNMP	O RDBA1	43	53999	19695	1tu4ybcx7vks0		PS	SHARE	NONE					1
SYS	O RDBA2	46	1	4323			AE	SHARE	NONE					689965
SYSMAN	O RDBA1	46	1839	18972			AE	SHARE	NONE					428133
SYSMAN	O RDBA2	48	35028	24448			TO	ROW EXCLUSIVE	NONE					426919
DBSNMP	O RDBA2	59	11707	30013	1tu4ybcx7vks0		PS	SHARE	NONE					0
DBSNMP	O RDBA2	59	11707	30013	1tu4ybcx7vks0		PS	SHARE	NONE					0
DBSNMP	O RDBA2	59	11707	30013	1tu4ybcx7vks0		PS	SHARE	NONE					0
DBSNMP	O RDBA2	59	11707	30013	1tu4ybcx7vks0		PS	SHARE	NONE					0
DBSNMP	O RDBA2	59	11707	30013	1tu4ybcx7vks0		AE	SHARE	NONE					39

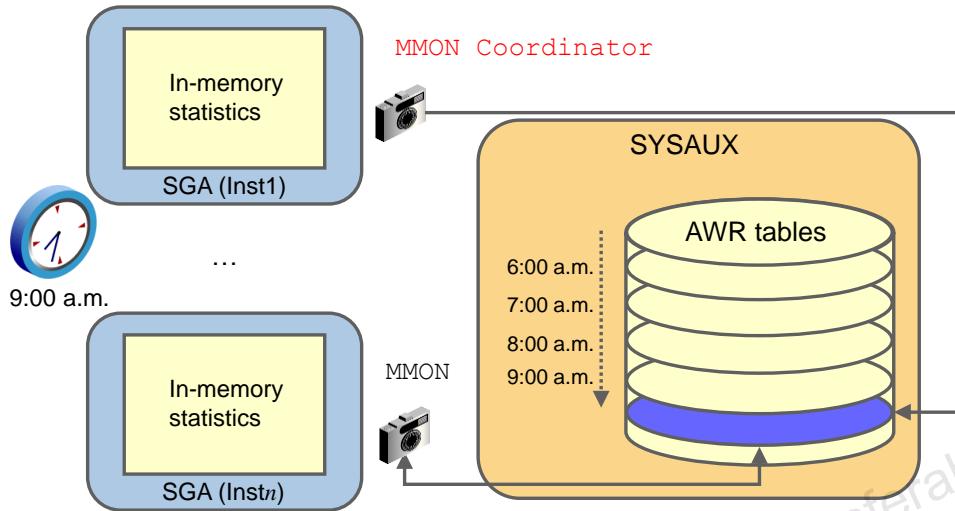


Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Use the Database Locks link found in the Additional Monitoring Links section to determine whether multiple instances are holding locks for the same object. The page shows user locks, all database locks, or locks that are blocking other users or applications. You can use this information to stop a session that is unnecessarily locking an object.

To access the Database Locks page, select Performance on the Cluster Database Home page, and click Database Locks in the Additional Monitoring Links section at the bottom of the Performance subpage.

AWR Snapshots in RAC



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

AWR automatically generates snapshots of the performance data once every hour and collects the statistics in the workload repository. In RAC environments, each AWR snapshot captures data from all active instances within the cluster. The data for each snapshot set that is captured for all active instances is from roughly the same point in time. In addition, the data for each instance is stored separately and is identified with an instance identifier. For example, the `buffer_busy_wait` statistic shows the number of buffer waits on each instance. The AWR does not store data that is aggregated from across the entire cluster. That is, the data is stored for each individual instance.

The statistics snapshots generated by the AWR can be evaluated by producing reports displaying summary data such as load and cluster profiles based on regular statistics and wait events gathered on each instance.

The AWR functions in a similar way as Statspack. The difference is that the AWR automatically collects and maintains performance statistics for problem detection and self-tuning purposes. Unlike in Statspack, in the AWR, there is only one `snapshot_id` per snapshot across instances.

AWR Reports and RAC: Overview

The diagram illustrates the structure of an AWR report for RAC. At the top right is the **Main Report** menu, which lists various categories of statistics. Below it is the **Report Summary** section, which displays top ADDM findings by average active sessions. To the left of the summary is the **WORKLOAD REPOSITORY REPORT (RAC)**, which provides a database summary and a list of database instances included in the report. The report summary also includes a table of findings and cache sizes. On the right side, there are three detailed statistics sections: **Global Cache Efficiency Percentages**, **Global Cache and Enqueue Statistics Summary**, and **Global Cache and Enqueue Workload Characteristics**. The **ORACLE** logo is located at the bottom left of the report area.

The RAC-related statistics in an AWR report are listed in the AWR report by category. A look at the Main Report menu lists the categories in the report. Available information includes:

- The number of instances open at the time of the begin snapshot and the end snapshot to indicate whether instances joined or left between the two snapshots
- The Global Activity Load Profile, which essentially lists the number of blocks and messages that are sent and received, as well as the number of fusion writes
- The Global Cache Efficiency Percentages, which indicate the percentage of buffer gets broken up into buffers received from the disk, local cache, and remote caches. Ideally, the percentage of disk buffer access should be close to zero.
- Global Cache and Enqueue Workload Characteristics, which gives you an overview of the more important numbers first. Because the global enqueue convert statistics have been consolidated with the global enqueue get statistics, the report prints only the average global enqueue get time. The round-trip times for CR and current block transfers follow, as well as the individual sender-side statistics for CR and current blocks. The average log flush times are computed by dividing the total log flush time by the number of actual log flushes. Also, the report prints the percentage of blocks served that actually incurred a log flush.
- Global Cache and Enqueue Messaging Statistics. The most important statistic here is the *queue time on ksxp*, which indicates how well the IPC works. Average numbers should be less than 1 ms.

The Segment Statistics section also includes the GC Buffer Busy Waits, CR Blocks Received, and CUR Blocks Received information for relevant segments.

Note: For more information about wait events and statistics, refer to *Oracle Database Reference*.

Active Session History Reports for RAC

- Active Session History (ASH) report statistics provide details about the RAC Database session activity.
- The database records information about active sessions for all active RAC instances.
- Two ASH report sections specific to Oracle RAC are **Top Cluster Events** and **Top Remote Instance**.

ASH Report For ORCL/orcl_3									
DB Name	DB Id	Instance	Inst num	Release	RAC	Host			
ORCL	1352492209	orcl_3		12.1.0.1.0	YES	host@example.com			
CPU	SGA Size	Buffer Cache	Shared Pool	ASH Buffer Size					
1	972M (100%)	524M (53.9%)	364M (39.9%)	2.0M (0.2%)					
Sample Time		Data Source							
Analysis Begin Time:	05-Sep-13 05:17:22	V\$ACTIVE_SESSION_HISTORY							
Analysis End Time:	05-Sep-13 05:32:28	V\$ACTIVE_SESSION_HISTORY							
Elapsed Time:	15.1 (mins)								
Sample Count:	225								
Average Active Sessions:	0.25								
Avg. Active Session per CPU:	0.25								
Report Target:	None specified								

ASH Report

- Top Events
- Load Profile
- Top SQL
- Top PL/SQL
- Top Java
- Top Call Types
- Top Wait Classes
- Top Object/Files/Latches
- Activity Over Time



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Active Session History (ASH) is an integral part of the Oracle Database self-management framework and is useful for diagnosing performance problems in Oracle RAC environments. ASH report statistics provide details about Oracle Database session activity. Oracle Database records information about active sessions for all active Oracle RAC instances and stores this data in the System Global Area (SGA). Any session that is connected to the database and using CPU is considered an active session. The exception to this is sessions that are waiting for an event that belongs to the idle wait class.

ASH reports present a manageable set of data by capturing only information about active sessions. The amount of the data is directly related to the work being performed, rather than the number of sessions allowed on the system. ASH statistics that are gathered over a specified duration can be put into ASH reports.

Each ASH report is divided into multiple sections to help you identify short-lived performance problems that do not appear in the ADDM analysis. Two ASH report sections that are specific to Oracle RAC are Top Cluster Events and Top Remote Instance.

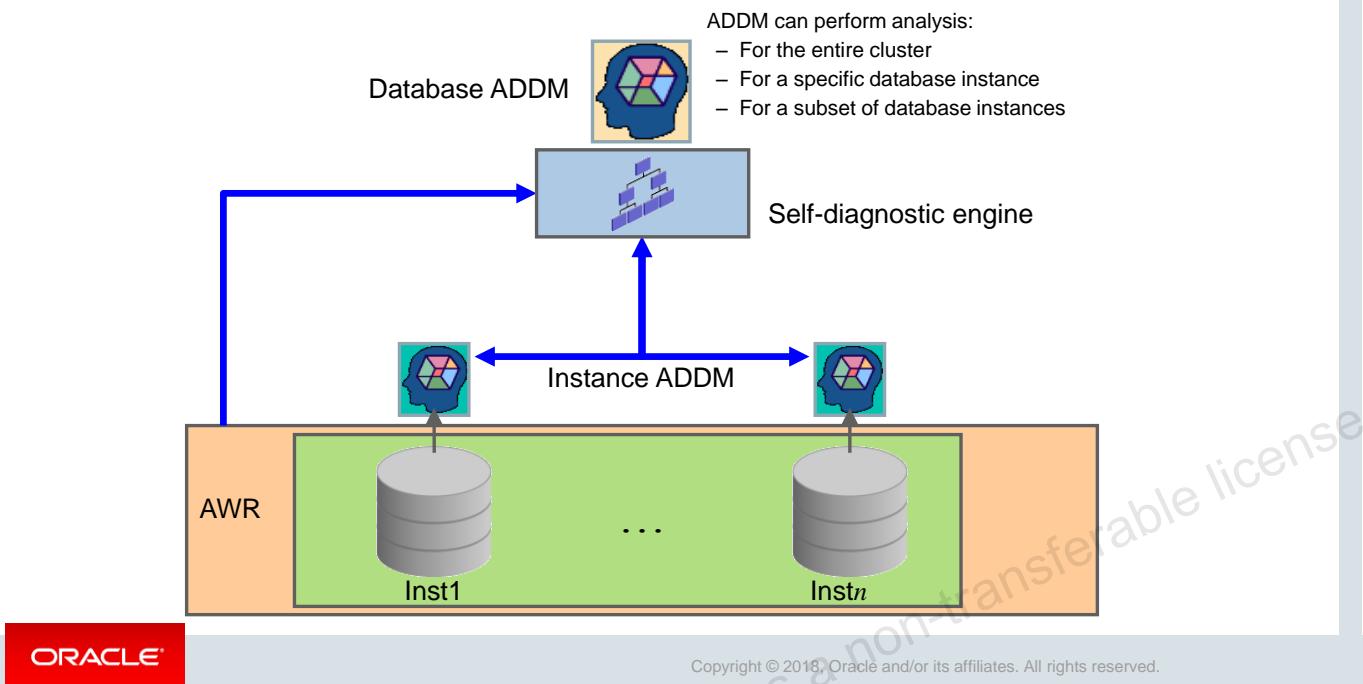
Top Cluster Events

The ASH report Top Cluster Events section is part of the Top Events report that is specific to Oracle RAC. The Top Cluster Events report lists events that account for the highest percentage of session activity in the cluster wait class event along with the instance number of the affected instances. You can use this information to identify which events and instances caused a high percentage of cluster wait events.

Top Remote Instance

The ASH report Top Remote Instance section is part of the Top Load Profile report that is specific to Oracle RAC. The Top Remote Instance report shows cluster wait events along with the instance numbers of the instances that accounted for the highest percentages of session activity. You can use this information to identify the instance that caused the extended cluster wait period.

Automatic Database Diagnostic Monitor for RAC



Using the Automatic Database Diagnostic Monitor (ADDM), you can analyze the information collected by AWR for possible performance problems with your Oracle database. ADDM presents performance data from a clusterwide perspective, thus enabling you to analyze performance on a global basis. In an Oracle RAC environment, ADDM can analyze performance using data collected from all instances and present it at different levels of granularity, including:

- Analysis for the entire cluster
- Analysis for a specific database instance
- Analysis for a subset of database instances

To perform these analyses, you can run the ADDM Advisor in Database ADDM for RAC mode to perform an analysis of the entire cluster, in Local ADDM mode to analyze the performance of an individual instance, or in Partial ADDM mode to analyze a subset of instances. Database ADDM for RAC is not just a report of reports but has independent analysis that is appropriate for RAC. You activate ADDM analysis by using the advisor framework through Advisor Central in Oracle Enterprise Manager, or through the `DBMS_ADVISOR` and `DBMS_ADDM` PL/SQL packages.

Automatic Database Diagnostic Monitor for RAC

- Identifies the most critical performance problems for the entire RAC cluster database
- Runs automatically when taking AWR snapshots
- Performs database-wide analysis of:
 - Global resources (for example I/O and global locks)
 - High-load SQL and hot blocks
 - Global cache interconnect traffic
 - Network latency issues
 - Skew in instance response times
- Is used by DBAs to analyze cluster performance
- Does not require investigation of n reports to spot common problems

ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

What Does ADDM Diagnose for RAC?

- Latency problems in interconnect
- Congestion (identifying top instances affecting the entire cluster)
- Contention (buffer busy, top objects, and so on)
- Top consumers of multiblock requests
- Lost blocks
- Reports information about interconnect devices; warns about using PUBLIC interfaces
- Reports throughput of devices, and how much of it is used by Oracle and for what purpose (GC, locks, PQ)



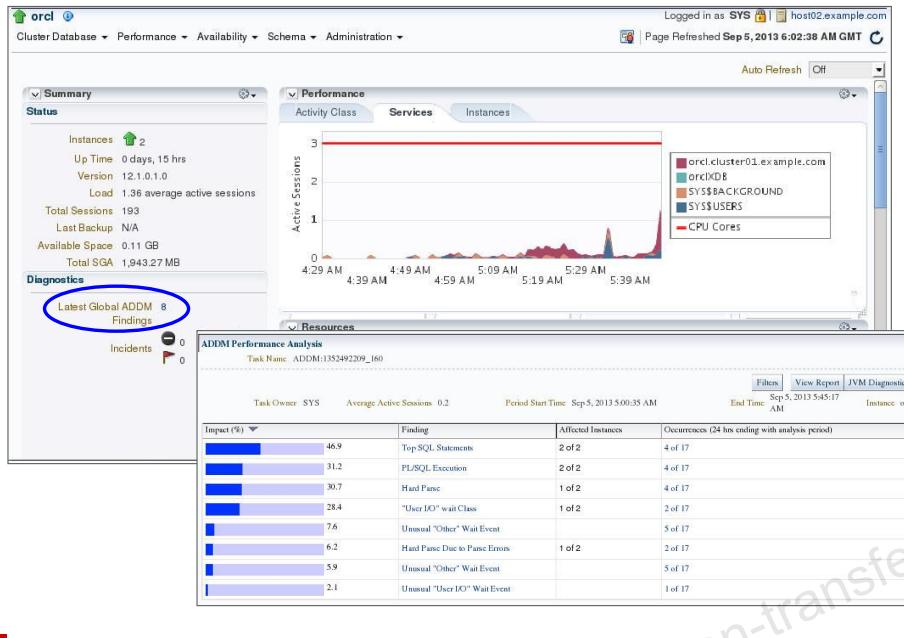
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Data sources are:

- Wait events (especially Cluster class and buffer busy)
- Active Session History (ASH) reports
- Instance cache transfer data
- Interconnect statistics (throughput, usage by component, pings)

ADDM analyzes the effects of RAC for both the entire database (DATABASE analysis mode) and for each instance (INSTANCE analysis mode).

EM Support for ADDM for RAC



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

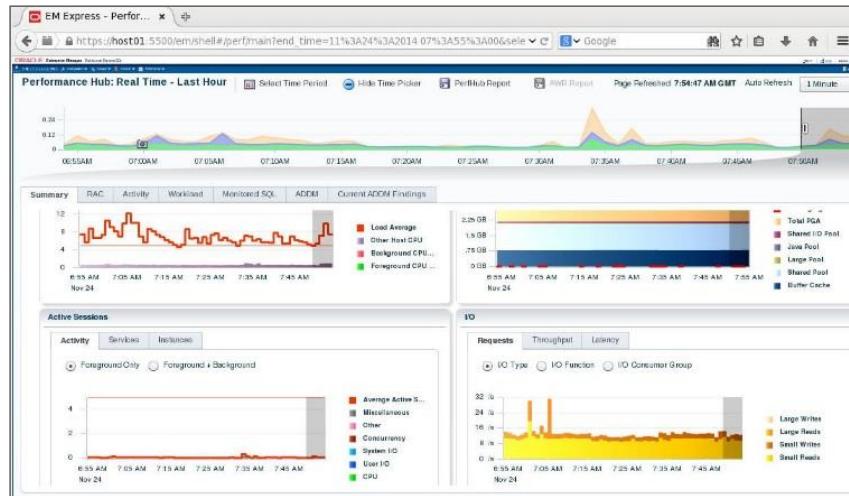
Enterprise Manager displays the ADDM analysis on the Cluster Database Home page.

On the Automatic Database Diagnostic Monitor (ADDM) page, the Database Activity chart (not shown here) plots the database activity during the ADDM analysis period. Database activity types are defined in the legend based on its corresponding color in the chart. Each icon below the chart represents a different ADDM task, which in turn corresponds to a pair of individual Oracle Database snapshots saved in the Workload Repository.

In the ADDM Performance Analysis section, the ADDM findings are listed in descending order, from highest impact to least impact. For each finding, the Affected Instances column displays the number (*m* of *n*) of instances affected. Drilling down further on the findings takes you to the Performance Findings Detail page. The Informational Findings section lists the areas that do not have a performance impact and are for informational purpose only.

The DB Time Breakdown chart shows how much each instance is impacted by these findings. The display indicates the percentage impact for each instance.

EM Database Express Performance Hub



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The Performance Hub provides a consolidated view of all performance data for a given time range. The Performance Hub can be used to view both historical and real-time data. In real-time mode, performance data is retrieved from in-memory views. The Performance Hub organizes performance data by dividing it into different tabs.

The Summary tab is available in both real-time and historical modes. In real-time mode, this tab shows metrics data that gives an overview of system performance in terms of Host Resource Consumption (CPU, I/O and Memory), and Average Active Sessions. In historical mode, the tab displays system performance in terms of resource consumption, average active sessions, and load profile information.

The Activity tab displays ASH Analytics, and is available in both real-time and historical modes.

The RAC tab displays RAC-specific metrics such as the number of global cache blocks received, and the average block latency.

The Monitored SQL tab displays Monitored Executions of SQL, PL/SQL, and Database Operations, and is available in both real-time and historical modes.

The ADDM tab is available in both real-time and historical modes. It displays ADDM and Automatic Real Time ADDM reports.

The Current ADDM Findings tab is available only in real-time mode, and displays a real-time analysis of system performance for the past 5 minutes.

Monitoring RAC With Cluster Health Advisor (CHA)

- CHA is deployed on each node by default when Grid Infrastructure is installed for RAC or RAC One Node database.
- CHA does not require any additional configuration.
- Run the following command to monitor a database:

```
$ chactl monitor database -db racdb
```

- Run the following command to stop monitoring a database:

```
$ chactl unmonitor database -db racdb
```

- Run the following command to check monitoring status of all cluster nodes and databases:

```
$ chactl status  
Monitoring nodes host01, host02  
Monitoring databases racdb
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

CHA is automatically provisioned on each node by default when Grid Infrastructure is installed for RAC or RAC One Node database. CHA does not require any additional configuration. The credentials of OCHAD daemon user in the Grid Infrastructure Management Repository (GIMR), are securely and randomly generated and stored in the Oracle Grid Infrastructure Credential Store.

When Cluster Health Advisor detects an Oracle RAC or RAC One Node database instance as running, CHA autonomously starts monitoring the cluster nodes. You must use CHACTL, while logged in as the Grid user, to explicitly turn on monitoring of the database.

To monitor the Oracle RAC environment, run the following command:

```
$ chactl monitor database -db db_unique_name
```

Cluster Health Advisor monitors all instances of the RAC or RAC One Node database using the default model. CHA cannot monitor single-instance Oracle databases, even if the single-instance Oracle databases share the same cluster as Oracle RAC databases.

CHA preserves database monitoring status across cluster restarts as CHA stores the status information in the GIMR. Each database instance is monitored independently both across Oracle RAC database nodes and when more than one databases run on a single node.

Note: For more information about Cluster Health Advisor, refer to *Oracle Autonomous Health Framework User's Guide*.



Quiz

Although there are specific tuning areas for RAC, such as instance recovery and interconnect traffic, you get most benefits by tuning your system like a single-instance system.

- a. True
- b. False



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.



Quiz

Which of the following RAC tuning tips are correct?

- a. Application tuning is often the most beneficial.
- b. Reduce long full-table scans in OLTP systems.
- c. Eliminate sequence caches.
- d. Use partitioning to reduce inter-instance traffic.
- e. Configure the interconnects properly.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Determine RAC-specific tuning components
- Determine RAC-specific wait events, global enqueues, and system statistics
- Implement the most common RAC tuning tips
- Use the Cluster Database Performance pages
- Use the Automatic Workload Repository (AWR) in RAC
- Use Automatic Database Diagnostic Monitor (ADDM) in RAC



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Practice 8: Overview

This practice covers manually discovering performance issues by using the EM performance pages as well as ADDM.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Managing High Availability of Services



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Configure and manage services in a RAC environment
- Use services with client applications
- Use services with the Database Resource Manager
- Use services with the Scheduler
- Configure services aggregation and tracing



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle Services

- To manage workloads or a group of applications, you can define services for a particular application or a subset of an application's operations.
- You can also group work by type under services.
- For example, OLTP users can use one service while batch processing can use another to connect to the database.
- Users who share a service should have the same service-level requirements.
- Use `srvctl` or Enterprise Manager to manage services, **not DBMS_SERVICE**.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To manage workloads or a group of applications, you can define services that you assign to a particular application or to a subset of an application's operations. You can also group work by type under services. For example, online users can use one service, while batch processing can use another, and reporting can use yet another service to connect to the database.

It is recommended that all users who share a service have the same service-level requirements. You can define specific characteristics for services and each service can be a separate unit of work. There are many options that you can take advantage of when using services. Although you do not have to implement these options, using them helps optimize application performance. You can define services for both policy-managed and administrator-managed databases.

Do not use `DBMS_SERVICE` with cluster-managed services. When Oracle Clusterware starts a service, it updates the database with the attributes stored in the CRS resource. If you use `DBMS_SERVICE` to modify the service and do not update the CRS resource, the next time CRS resource is started, it will override the database attributes set by `DBMS_SERVICE`.

Service Usage in an Oracle RAC Database

- Services provide location transparency.
- A service name can identify multiple database instances and an instance can belong to multiple services.
- Resource profiles are automatically created when you define a service.
 - A resource profile describes how Oracle Clusterware should manage the service.
 - Resource profiles also define service dependencies for the instance and the database.
- Services are integrated with Resource Manager, enabling you to restrict the resources that users use to connect to an instance by using a service.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Services provide location transparency. A service name can identify multiple database instances, and an instance can belong to multiple services. Several database features use services for an Oracle RAC database.

Resource profiles are automatically created when you define a service. A resource profile describes how Oracle Clusterware should manage the service and which instance the service should failover to if the preferred instance stops. Resource profiles also define service dependencies for the instance and the database. Due to these dependencies, if you stop a database, then the instances and services are automatically stopped in the correct order.

Services are integrated with Oracle Resource Manager, which enables you to restrict the resources that users use to connect to an instance by using a service. Oracle Resource Manager enables you to map a consumer group to a service so that users who connect to an instance using that service are members of the specified consumer group. Oracle Resource Manager operates at an instance level.

The metric data generated by AWR is organized into various groups, such as event, event class, session, service, and tablespace metrics. Typically, you view the AWR data by using Oracle Enterprise Manager or AWR reports.

In a Global Data Services environment, the RAC service model is applied to sets of globally distributed databases. GDS works with single instance or RAC databases by using Data Guard, GoldenGate, or other replication technologies.

Parallel Operations and Services

- By default, in an RAC environment, a SQL statement run in parallel can run across all of the nodes in the cluster.
- To perform well, the interconnect must be sized correctly as inter-node parallel execution may result in a lot of interconnect traffic.
- You can control parallel execution in a RAC environment with the `PARALLEL_FORCE_LOCAL` initialization parameter.
 - The parallel execution servers can only execute on the same Oracle RAC node where the SQL statement was started.
- Services can be used to limit the number of instances that participate in a parallel SQL operation.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

By default, in an Oracle RAC environment, a SQL statement executed in parallel can run across all of the nodes in the cluster. For this cross-node or inter-node parallel execution to perform well, the interconnect in the Oracle RAC environment must be sized appropriately because inter-node parallel execution may result in a lot of interconnect traffic. To limit inter-node parallel execution, you can control parallel execution in an Oracle RAC environment by using the `PARALLEL_FORCE_LOCAL` initialization parameter. By setting this parameter to `TRUE`, the parallel execution servers can only execute on the same Oracle RAC node where the SQL statement was started.

Services are used to limit the number of instances that participate in a parallel SQL operation. When the default database service is used, the parallel SQL operation can run on all available instances. You can create any number of services, each consisting of one or more instances. When a parallel SQL operation is started, the parallel execution servers are only spawned on instances which offer the specified service used in the initial database connection.

`INSTANCE_GROUPS` is a RAC parameter that you can specify only in parallel mode. Used in conjunction with the `PARALLEL_INSTANCE_GROUP` parameter, it lets you restrict parallel query operations to a limited number of instances. To restrict parallel query operations to a limited number of instances, set the `PARALLEL_INSTANCE_GROUP` initialization parameter to the name of a service. This does not affect other parallel operations such as parallel recovery or the processing of `GV$` queries.

Service-Oriented Buffer Cache Access

- Pre-Warm the Buffer Cache:
 - During planned maintenance, when a singleton service is failed over, Service-oriented buffer cache access allows RAC to pre-warm the buffer of the instance to which the service is going to failover.
- Resource Mastering Optimization:
 - Service-oriented buffer cache access improves performance by managing data with the service to which the data belongs.
- Access of an object, over time, through a service is mapped and persisted to the database.
- Blocks that are accessed through the service:
 - Are cached in the instances where the services are running
 - Are not cached where the services are not running.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Cluster-managed services are used to allocate workloads across various Oracle RAC database instances running in a cluster. These services are used to access database objects cached in the buffer caches of the respective database instances.

Service-oriented buffer cache access optimization allows Oracle RAC to cache or pre-warm instances with data blocks for objects accessed through a service. This feature improves access time of Oracle RAC Database instances.

Service-oriented buffer cache access improves performance by managing data with the service to which the data belongs. Access of an object, over time, through a service is mapped and persisted to the database, and this information can be used to improve performance. Blocks that are accessed through the service are cached in the instances where the services are running and, more importantly, the information is not cached where the services are not running.

This information can also be used to pre-warm the cache prior to a singleton service starting. The service start-up can be triggered either by instance start-up or by service relocation. Service-oriented buffer cache access provides consistent performance to any user of that service because the blocks that the service user accesses are cached in the new relocated instance.

Service Characteristics

The characteristics of a service include:

- Service Name
- Service Edition
- Service Management Policy
- Database Role for a Service
- Instance Preference
- Server Pool Assignment
- Load Balancing Advisory Goal for Run-time Connection Load Balancing
- Connection Load Balancing Goal



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When you create new services for your database, you should define the automatic workload management characteristics for each service. The characteristics of a service include:

- **Service Name:** The service name is used by clients to connect to one or more instances. The service name must be unique throughout your system.
- **Service Edition:** Edition-based redefinition of database objects enables you to upgrade an application's objects while the objects are in use. When the service edition is set, connections that use this service use this edition as the initial session edition. If the service does not specify the edition name, then the initial session edition is the database default edition.
- **Service Management Policy:** When you use Clusterware to manage your database, you can configure startup options for each individual database service when you add the service by using the `srvctl add service` command with the `-policy` parameter. If you set the management policy for a service to `AUTOMATIC` (the default), then the service starts automatically.
- **Database Role for a Service:** If you configured Oracle Data Guard in your environment, then you can define a role for each service using SRVCTL with the `-1` parameter.
- **Instance Preference:** When you define a service for an administrator-managed database, you define which instances support that service using SRVCTL with the `-preferred` parameter. These are known as the preferred instances.
- **Server Pool Assignment:** When you define services for a policy-managed database, you assign the service to a server pool in which the database is hosted using SRVCTL with the `-serverpool` parameter. You can define the service as either `UNIFORM` (running on all instances in the server pool) or `SINGLETON` (running on only one instance in the server pool) using the `-cardinality` parameter. For singleton services, Oracle RAC chooses on which instance in the server pool the service is active. If that instance fails, then the service fails over to another instance in the server pool. A service can only run in one server pool and Oracle recommends that every server pool has at least one service.

- **Load Balancing Advisory Goal for Run-time CLB:** With run-time connection load balancing, applications can use load balancing advisory events to provide better service to users. Oracle JDBC, Oracle UCP for Java, OCI session pool, ODP.NET, and Oracle WebLogic Server Active GridLink for Oracle RAC clients are automatically integrated to take advantage of load balancing advisory events. To enable the load balancing advisory, use SRVCTL with the `-rlbgoal` parameter when creating or modifying the service. The load balancing advisory also recommends how much of the workload should be sent to that instance. The goal determines whether connections are made to the service based on best service quality (how efficiently a single transaction completes) or best throughput (how efficiently a complete job or long-running query completes).
- **Connection Load Balancing Goal:** Oracle Net Services provides connection load balancing to enable you to spread user connections across all of the instances that are supporting a service. For each service, you can use SRVCTL to define the method you want the listener to use for load balancing by setting the connection load balancing goal, specified with the `-clbgoal` parameter. Connections are classified as `LONG` (such as connection pools and SQL*FORMS), which tells the listener to use session count, or `SHORT`, which tells the listener to use response-time or throughput statistics. If the load balancing advisory is enabled, then its information is used to balance connections; otherwise, CPU utilization is used to balance connections.

Default Service Connections

- Your RAC database includes an Oracle database service identified by DB_UNIQUE_NAME, if set.
 - DB_NAME or PDB_NAME, if not.
- This default service is always available on all instances in an Oracle RAC environment.
- Additionally, the database supports two internal services:
 - SYS\$BACKGROUND is used by background processes only.
 - SYS\$USERS is the default service for user sessions that are not associated with any application service.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Your RAC database includes an Oracle database service identified by DB_UNIQUE_NAME, if set, or DB_NAME or PDB_NAME, if not. This default service is always available on all instances in an Oracle RAC environment, unless an instance is in restricted mode. You cannot alter this service or its properties. In a multi-tenant environment, every PDB has a default service created at PDB creation. Additionally, the database supports the following two internal services:

- SYS\$BACKGROUND is used by the background processes only.
- SYS\$USERS is the default service for user sessions that are not associated with any application service.

All of these services are used for internal management. You cannot stop or disable any of these internal services to do planned outages or to fail over to Oracle Data Guard. Do not use these services for client connections. You can explicitly manage only the services that you create. If a feature of the database creates an internal service, you cannot manage it using the methods presented here.

Restricted Service Registration

- Restricted Service Registration allows listener registration only from local IP addresses, by default.
 - Provides the ability to configure and update a set of addresses or subnets from which registration requests are allowed by the listener.
- Database Instance registration with a listener succeeds only when the request originates from a valid node.
- The network administrator can specify a list of valid nodes, excluded nodes, or disable valid node checking.
- The control of dynamic registration results in increased manageability and security of Oracle RAC deployments.
- Valid node checking for registration (VNCR) is enabled by default.

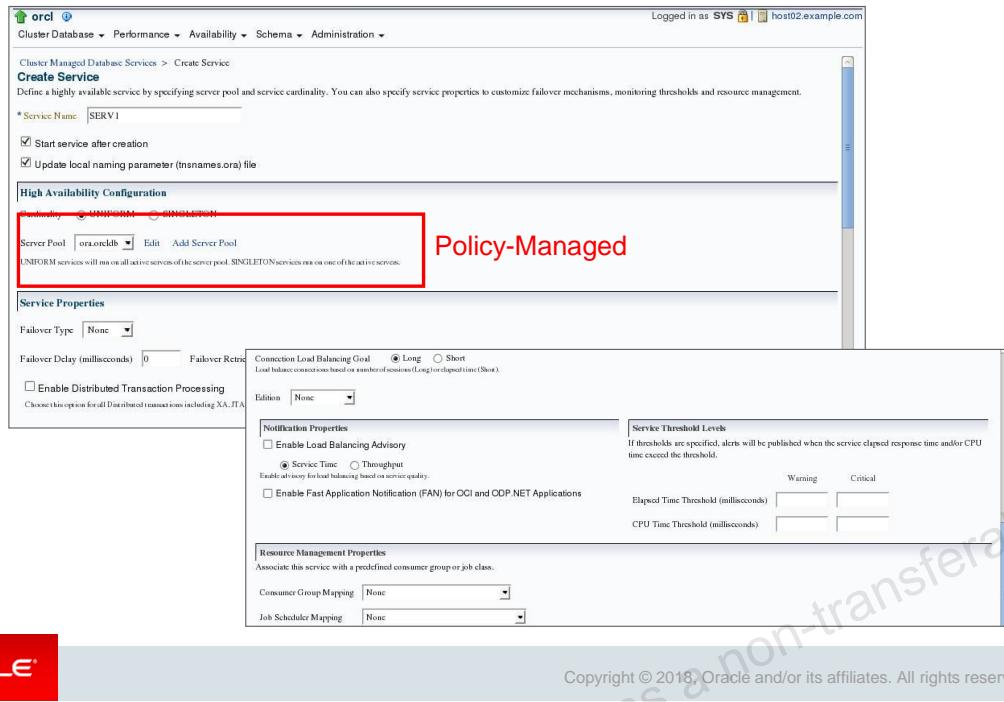


Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Security is a high priority to all enterprises, and network security and controlling access to the database is a critical component of overall security endeavors. This feature allows listener registration only from local IP addresses, by default, and provides the ability to configure and dynamically update a set of IP addresses or subnets from which registration requests are allowed by the listener. Database Instance registration with a listener succeeds only when the request originates from a valid node. The network administrator can specify a list of valid nodes, excluded nodes, or disable valid node checking. The list of valid nodes explicitly lists the nodes and subnets that can register with the database. The list of excluded nodes explicitly lists the nodes that cannot register with the database. The control of dynamic registration results in increased manageability and security of Oracle RAC deployments. By default, valid node checking for registration (VNCR) is enabled. In the default configuration, registration requests are only allowed from nodes within the cluster, because they are redirected to the private subnet, and only nodes within the cluster can access the private subnet. Non-SCAN listeners only accept registration from instances on the local node. You must manually include remote nodes or nodes outside the subnet of the SCAN listener on the list of valid nodes by using the `registration_invited_nodes_alias` parameter in the `listener.ora` file or by modifying the SCAN listener by using SRVCTL as follows:

```
$ srvctl modify scan_listener -invitednodes node_list -invitedsubnets subnet_list
```

Creating Service with Enterprise Manager



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

From your Cluster Database home page, click the Availability link, and then select Cluster Managed Database Services from the pull-down list. On the Cluster Managed Database Services page, click Create Service.

Use the Create Service page to configure a new service in which you do the following:

- Select the desired service policy for each instance configured for the cluster database.
- Select the desired service properties.

If your database is administration managed, the High Availability Configuration section allows you to configure preferred and available servers. If your database employs policy-managed administration, you can configure the service cardinality to be UNIFORM or SINGLETON and assign the service to a server pool.

You can also define the management policy for a service. You can choose either an automatic or a manual management policy.

- **Automatic:** The service always starts when the database starts.
- **Manual:** Requires that the service be started manually. Prior to Oracle RAC 11g Release 2, all services worked as though they were defined with a manual management policy.

Note: Enterprise Manager now generates the corresponding entries in your `tnsnames.ora` files for your services. Just click the “Update local naming parameter (`tnsnames.ora`) file” check box when creating the service.

Creating Services with SRVCTL

- To create a service called **GL** with preferred instance RAC02 and an available instance RAC01:

```
$ srvctl add service -db PROD1 -service GL -preferred RAC02  
-available RAC01
```

- To create a service called **AP** with preferred instance RAC01 and an available instance RAC02:

```
$ srvctl add service -db PROD1 -service AP -preferred RAC01  
-available RAC02
```

- Create a **SINGLETON** service called **BATCH** using server pool **SP1** and a **UNIFORM** service called **ERP** using pool **SP2**:

```
$ srvctl add service -db PROD2 -service BATCH -serverpool  
SP1 -cardinality singleton -policy manual
```

```
$ srvctl add service -db PROD2 -service ERP -serverpool SP2  
-cardinality UNIFORM -policy manual
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

For the example in the slide, assume a two-node, administration-managed database called PROD1 with an instance named RAC01 on one node and an instance called RAC02 on the other. Two services are created, AP and GL, to be managed by Oracle Clusterware. The AP service is defined with a preferred instance of RAC01 and an available instance of RAC02.

If RAC01 dies, the AP service member on RAC01 is restored automatically on RAC02. A similar scenario holds true for the GL service.

Next, assume a policy-managed cluster database called PROD2. Two services are created, a SINGLETON service called BATCH and a UNIFORM service called ERP. SINGLETON services run on one of the active servers and UNIFORM services run on all active servers of the server pool. The characteristics of the server pool determine how resources are allocated to the service.

Note: When services are created with `srvctl`, `tnsnames.ora` is not updated and the service is not started.

Managing Services with Enterprise Manager

Cluster Managed Database Services

The following shows the status of all cluster managed services defined for the current database. Select a service to manage the state of its instances.

Select	Service Name	Status	Running Servers	Server Pool	Response Time (ms)	% CPU Load	Service Alerts	Status Details
<input checked="" type="radio"/>	BATCH		host02	ora.orcldb	0.00 0.00 0.00	0.00		SINGLETON Service is running on one server in server pool.
<input type="radio"/>	ERP		host02, host01	ora.orcldb	0.00 0.00 0.00	0.00		UNIFORM Service is running on all servers in server pool.
<input type="radio"/>	oracl		host02, host01	ora.orcldb	0.0 0.0 0.0	0.0		Default Database Service is running.

TIP Response Time and % CPU Load data is average over the last 5 minutes

[Create Service](#) [Refresh](#) [Return](#)

Additional Links

[Manage Server Pools](#)



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can use Enterprise Manager to manage services within a GUI framework. The screenshot in the slide shows the main page for administering services within RAC. It shows you some basic status information about a defined service.

To access this page, select Cluster Managed Database Services from the Availability pull-down menu.

You can perform simple service management such as enabling, disabling, starting, stopping, and relocating services. All possible operations are shown in the slide.

If you choose to start a service on the Cluster Managed Database Services page, then EM attempts to start the service on every preferred instance. Stopping the service stops it on all instances that it is currently running.

To relocate a service, select the service that you want to administer, select the Manage option from the Actions drop-down list, and then click Go.

Note: On the Cluster Managed Database Services page, you can test the connection for a service.

Managing Services with EM

The screenshot shows the Oracle Enterprise Manager interface for managing services. At the top, it says 'Logged in as SYS host02.example.com'. Below that, it displays 'Cluster Managed Database Service: ERP'. It notes that the service has been configured to run on multiple instances. A message states: 'The service has been configured to run on the following instances. A service may have been stopped on an instance if the instance was down or the service was disabled. Starting a service on a down instance will first bring up the down instance.' The page is refreshed at 9/6/13 12:31 PM.

Key service details shown:

- Service Status: UNIFORM Service is running on all servers in server pool.
- Edition: None
- % CPU Load: 0.00
- Server Pool: ora.orcldb
- Cardinality: UNIFORM
- Failover Type: SESSION
- Failover Type: SESSION

Links for 'Top Consumers' and 'Service Properties' are also present.

Instances										
	Enable	Disable	Start	Stop	Relocate	Service Status for Node	Instance Status	Response Time (per user call) (milliseconds)	CPU Time (per user call) (milliseconds)	Status Details
Select	Node Name	Instance Name								
<input checked="" type="radio"/>	host02	orc1_1								
<input type="radio"/>	host01	orc1_3								



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To access the Cluster Managed Database Service page for an individual service, you must select a service from the Cluster Managed Database Services page, select the Manage option from the Actions drop-down list, and then click Go.

This is the Cluster Managed Database Service page for an individual service. It offers you the same functionality as the previous page, except that actions performed here apply to specific instances of a service.

This page also offers you the added functionality of relocating a service to an available instance. Relocating a service from one instance to another stops the service on the first instance and then starts it on the second.

Managing Services with `srvctl`

- Start a named service on all configured instances:

```
$ srvctl start service -db orcl -service AP
```

- Stop a service:

```
$ srvctl stop service -db orcl -service AP -instance orcl4
```

- Disable a service at a named instance:

```
$ srvctl disable service -db orcl -service AP -instance orcl4
```

- Set an available instance as a preferred instance:

```
$ srvctl modify service -db orcl -service AP -instance orcl5 -preferred
```

- Relocate a service from one instance to another:

```
$ srvctl relocate service -db orcl -service AP -oldinst orcl5 -newinst orcl4
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The slide demonstrates some management tasks with services by using SRVCTL.

Assume that an AP service has been created with four preferred instances: orcl1, orcl2, orcl3, and orcl4. An available instance, orcl5, has also been defined for AP.

In the first example, the AP service is started on all instances. If any of the preferred or available instances that support AP are not running but are enabled, then they are started.

The `stop` command stops the AP service on instance orcl4. The instance itself is not shut down, but remains running possibly supporting other services. The AP service continues to run on orcl1, orcl2, and orcl_3. The intention might have been to perform maintenance on orcl4, and so the AP service was disabled on that instance to prevent automatic restart of the service on that instance. The OCR records the fact that AP is disabled for orcl4. Thus, Oracle Clusterware will not run AP on orcl4 until the service is enabled.

The next command in the slide changes orcl5 from being an available instance to a preferred one. This is beneficial if the intent is to always have four instances run the service because orcl4 was previously disabled. The last example relocates the AP service from instance orcl5 to orcl4. Do not perform other service operations while the online service modification is in progress. The following command relocates the AP service from host01 to host03 using node syntax:

```
$ srvctl relocate service -db orcl -service AP -currentnode host01  
-targetnode node3
```

Using Services with Client Applications

```
ERP= (DESCRIPTION=
      ## Using SCAN ##
      (LOAD_BALANCE=on)
      (ADDRESS= (PROTOCOL=TCP) (HOST=cluster01-scan) (PORT=1521))
      (CONNECT_DATA= (SERVICE_NAME=ERP)) )
```

```
ERP= (DESCRIPTION=
      ## Using VIPs ##
      (LOAD_BALANCE=on)
      (ADDRESS= (PROTOCOL=TCP) (HOST=node1-vip) (PORT=1521))
      (ADDRESS= (PROTOCOL=TCP) (HOST=node2-vip) (PORT=1521))
      (ADDRESS= (PROTOCOL=TCP) (HOST=node3-vip) (PORT=1521))
      (CONNECT_DATA= (SERVICE_NAME=ERP)) )
```

```
url="jdbc:oracle:oci:@ERP"      ## Thick JDBC ##
```

```
url="jdbc:oracle:thin:@(DESCRIPTION=
      ## Thin JDBC ##
      (LOAD_BALANCE=on)
      (ADDRESS= (PROTOCOL=TCP) (HOST=cluster01-scan) (PORT=1521)))
      (CONNECT_DATA= (SERVICE_NAME=ERP)) )"
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The first example in the slide shows the TNS connect descriptor that can be used to access the ERP service. It uses the cluster's Single Client Access Name (SCAN). The SCAN provides a single name to the clients connecting to Oracle RAC that does not change throughout the life of the cluster, even if you add or remove nodes from the cluster. Clients connecting with SCAN can use a simple connection string, such as a thin JDBC URL or EZConnect, and still achieve the load balancing and client connection failover. The second example uses virtual IP addresses as in previous versions of the Oracle Database.

The third example shows the thick JDBC connection description using the previously defined TNS connect descriptor.

The third example shows the thin JDBC connection description using the same TNS connect descriptor as the first example.

Note: The `LOAD_BALANCE=ON` clause is used by Oracle Net to randomize its progress through the protocol addresses of the connect descriptor. This feature is called client connection load balancing.

Services and Connection Load Balancing

- The two load balancing methods that you can implement are:
 - **Client-side load balancing:** Balances the connection requests across the listeners
 - **Server-side load balancing:** The listener directs a connection request to the best instance currently providing the service by using the load balancing advisory (LBA).
- FAN, Fast Connection Failover, and LBA depend on a connection load balancing configuration that includes setting the connection load balancing goal for the service.
- The load balancing goal for the service can be either:
 - **LONG:** For applications having long-lived connections. This is typical for connection pools and SQL*Forms sessions.
 - **SHORT:** For applications that have short-lived connections

```
srvctl modify service -service service_name -clbgoal  
LONG|SHORT
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle Net Services provides the ability to balance client connections across the instances in an Oracle RAC configuration. You can implement two types of load balancing: client-side and server-side. Client-side load balancing balances the connection requests across the listeners. With server-side load balancing, the SCAN listener directs a connection request to the best instance currently providing the service, based on the `-clbgoal` and `-rlbgoal` settings for the service. In a RAC database, client connections should use both types of connection load balancing.

FAN, Fast Connection Failover, and the load balancing advisory depend on an accurate connection load balancing configuration that includes setting the connection load balancing goal for the service. You can use a goal of either `LONG` or `SHORT` for connection load balancing. These goals have the following characteristics:

- `LONG:` Use the `LONG` load balancing method for applications that have long-lived connections. This is typical for connection pools and SQL*Forms sessions. `LONG` is the default connection load balancing goal.
- `SHORT:` Use the `SHORT` connection load balancing method for applications that have short-lived connections. The following example modifies the `ORDER` service, using `srvctl` to set the goal to `SHORT`:
`srvctl modify service -s ORDER -j SHORT`

Services and Transparent Application Failover

- Services simplify the deployment of Transparent Application Failover (TAF).
- You can define a TAF policy for a service and all connections using this service will automatically have TAF enabled.
- The TAF setting on a service can be **NONE**, **BASIC**, or **PRECONNECT** and overrides any TAF setting in the client connection definition.
- To define a TAF policy for a service, the **srvctl** utility can be used as follows:

```
srvctl modify service -db crm -service GL  
-failovermethod BASIC -failovertype SELECT  
-failoverretry 10 -failoverdelay 30
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When Oracle Net Services establishes a connection to an instance, the connection remains open until the client closes the connection, the instance is shut down, or a failure occurs. If you configure TAF for the connection, then Oracle Database moves the session to a surviving instance when an outage occurs.

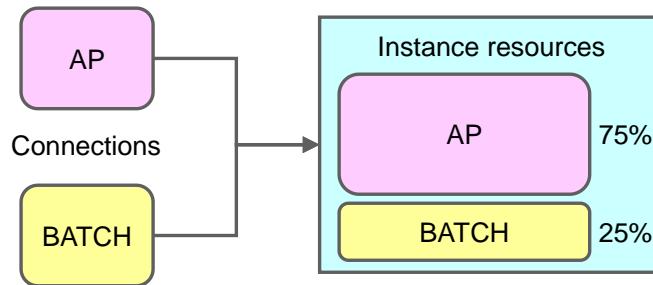
TAF can restart a query after failover has completed but for other types of transactions, such as **INSERT**, **UPDATE**, or **DELETE**, the application must roll back the failed transaction and resubmit the transaction. You must re-execute any session customizations, in other words, **ALTER SESSION** statements, after failover has occurred. However, with TAF, a connection is not moved during normal processing, even if the workload changes over time.

Services simplify the deployment of TAF. You can define a TAF policy for a service, and all connections using this service will automatically have TAF enabled. This does not require any client-side changes. The TAF setting on a service overrides any TAF setting in the client connection definition.

Note: TAF applies only to an admin-managed database and not to policy-managed databases.

Using Services with the Resource Manager

- Consumer groups are automatically assigned to sessions based on session services.
- Work is prioritized by service inside one instance.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The Database Resource Manager (also called Resource Manager) enables you to identify work by using services. It manages the relative priority of services within an instance by binding services directly to consumer groups. When a client connects by using a service, the consumer group is assigned transparently at connect time. This enables the Resource Manager to manage the work requests by service in the order of their importance.

For example, you define the `AP` and `BATCH` services to run on the same instance, and assign `AP` to a high-priority consumer group and `BATCH` to a low-priority consumer group. Sessions that connect to the database with the `AP` service specified in their TNS connect descriptor get priority over those that connect to the `BATCH` service.

This offers benefits in managing workloads because priority is given to business functions rather than the sessions that support those business functions.

Services and Resource Manager with EM

The screenshot illustrates the Oracle Enterprise Manager (EM) interface for managing consumer group mappings. The main window shows the 'Consumer Group Mappings' page with the 'General' tab active. It lists various consumer groups and allows users to define mapping rules by selecting a service and setting priorities. A specific rule is being configured on the right-hand panel, where a service is mapped to a consumer group. The 'Priorities' tab is also visible, providing another way to manage session assignments.

Enterprise Manager (EM) presents a GUI through the Consumer Group Mapping page to automatically map sessions to consumer groups. You can access this page by selecting Resource Manager from the Cluster Administration pull-down menu and then clicking the Consumer Group Mappings link.

Using the General tabbed page of the Consumer Group Mapping page, you can set up a mapping of sessions connecting with a service name to consumer groups as illustrated in the slide above.

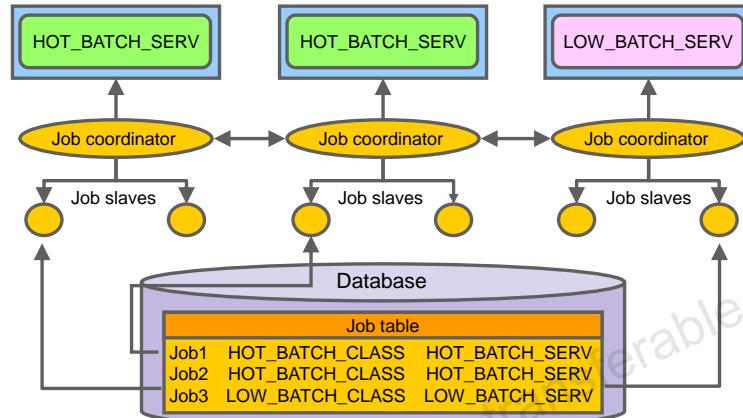
With the ability to map sessions to consumer groups by service, module, and action, you have greater flexibility when it comes to managing the performance of different application workloads.

Using the Priorities tabbed page of the Consumer Group Mapping page, you can change priorities for the mappings that you set up on the General tabbed page. The mapping options correspond to columns in V\$SESSION. When multiple mapping columns have values, the priorities you set determine the precedence for assigning sessions to consumer groups.

Note: You can also map a service to a consumer group directly from the Create Service page as shown in the bottom-left corner of the slide.

Using Services with the Scheduler

- Services are associated with Scheduler classes.
- Scheduler jobs have service affinity:
 - High availability
 - Load balancing



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Just as in other environments, the Scheduler in a RAC environment uses one job table for each database and one job coordinator (CJQ0 process) for each instance. The job coordinators communicate with each other to keep information current.

The Scheduler can use the services and the benefits they offer in a RAC environment. The service that a specific job class uses is defined when the job class is created. During execution, jobs are assigned to job classes and job classes run within services. Using services with job classes ensures that the work of the Scheduler is identified for workload management and performance tuning. For example, jobs inherit server-generated alerts and performance thresholds for the service they run under.

For high availability, the Scheduler offers service affinity instead of instance affinity. Jobs are not scheduled to run on any specific instance. They are scheduled to run under a service. So, if an instance dies, the job can still run on any other instance in the cluster that offers the service.

Note: By specifying the service where you want the jobs to run, the job coordinators balance the load on your system for better performance.

Services and the Scheduler with EM

The screenshot shows the Oracle Enterprise Manager (EM) interface for creating a job class. The main form includes fields for Name, Description, Logging Level (set to 'Log job runs only (RUNS)'), Log Retention Period (Days), Resource Consumer Group, and Service Name (highlighted with a red box). To the right, there are sections for Connection Load Balancing Goal (set to 'Long'), Edition (None), Notification Properties (checkboxes for Load Balancing Advisory and Fast Application), Resource Management Properties (checkbox for mapping to a consumer group), and Job Scheduler Mapping (dropdown set to 'None'). The background features a watermark that reads 'ANG LIU (angliu@uwaterloo.edu) has a non-transferrable license to use this material'.

ORACLE®

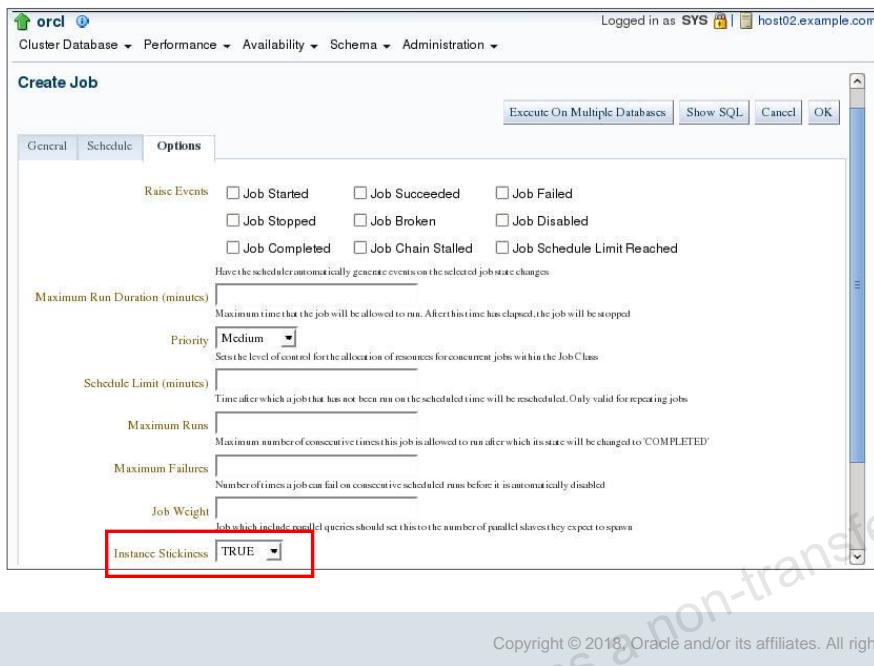
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To configure a job to run under a specific service, select Oracle Scheduler from the Administration pull-down menu and select the Job Classes link. This opens the Scheduler Job Classes page. On the Scheduler Job Classes page, you can see services assigned to job classes.

When you click the Create button on the Scheduler Job Classes page, the Create Job Class page is displayed. On this page, you can enter details of a new job class, including which service it must run under.

Note: Similarly, you can map a service to a job class on the Create Service page as shown at the bottom of the slide.

Services and the Scheduler with EM



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

After your job class is set up with the service that you want it to run under, you can create the job. To create the job, select Oracle Scheduler from the Administration pull-down menu, and select the Jobs link on the Server page. The Scheduler Jobs page appears, on which you can click the Create button to create a new job. When you click the Create button, the Create Job page is displayed. This page has different tabs: General, Schedule, and Options. Use the General tabbed page to assign your job to a job class.

Use the Options page (displayed in the slide) to set the Instance Stickiness attribute for your job. Basically, this attribute causes the job to be load balanced across the instances for which the service of the job is running. The job can run only on one instance. If the Instance Stickiness value is set to TRUE, which is the default value, the Scheduler runs the job on the instance where the service is offered with the lightest load. If Instance Stickiness is set to FALSE, then the job is run on the first available instance where the service is offered.

Note: It is possible to set job attributes, such as INSTANCE_STICKINESS, by using the SET_ATTRIBUTE procedure of the DBMS_SCHEDULER PL/SQL package.

Using Distributed Transactions with RAC

- An XA transaction can span RAC instances, allowing any application that uses XA to take full advantage of the Oracle RAC environment.
- Tightly coupled XA transactions no longer require the special type of singleton services (DTP).
- XA transactions are transparently supported on Oracle RAC databases with any type of services configuration.
- However, DTP services will improve performance for many distributed transaction scenarios.
- DTP services allow you to direct all branches of a distributed transaction to a single instance in the cluster.
- To load balance, it is better to have several groups of smaller application servers with each group directing its transactions to a single service.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

A global (XA) transaction can span RAC instances by default, allowing any application that uses the Oracle XA library to take full advantage of the Oracle RAC environment to enhance the availability and scalability of the application. GTXn background processes support XA transactions in an Oracle RAC environment. The GLOBAL_TXN_PROCESSES initialization parameter, which is set to 1 by default, specifies the initial number of GTXn background processes for each Oracle RAC instance. Use the default value for this parameter clusterwide to allow distributed transactions to span multiple Oracle RAC instances. Using the default value allows the units of work performed across these Oracle RAC instances to share resources and act as a single transaction (that is, the units of work are tightly coupled). It also allows 2 phase-commit requests to be sent to any node in the cluster. Tightly coupled XA transactions no longer require the special type of singleton services (that is, Oracle Distributed Transaction Processing [DTP] services) to be deployed on Oracle RAC database. XA transactions are transparently supported on Oracle RAC databases with any type of services configuration.

To provide improved application performance with distributed transaction processing (DTP) in RAC, you may want to take advantage of DTP services or XA affinity. Using DTP services, you can direct all branches of a distributed transaction to a single instance in the cluster. To load balance across the cluster, it is better to have several groups of smaller application servers with each group directing its transactions to a single service, or set of services, than to have one or two larger application servers. DTP or XA affinity is required, if suspending and resuming the same XA branch.

Distributed Transactions and Services

To create distributed transaction processing (DTP) services for distributed transaction processing, perform the following steps:

1. Create a singleton service using EM or SRVCTL. For an **administrator-managed database**, define only one instance as the preferred instance:

```
$ srvctl add service -db crm -service xa_01.example.com  
-preferred RAC01 -available RAC02,RAC03
```

For a **policy-managed database**, specify the server pool to use, and set the cardinality of the service to SINGLETON:

```
$ srvctl add service -db crm -service xa_01.example.com  
-serverpool dtp_pool -cardinality SINGLETON
```

2. Set the DTP parameter (-dtp) for the service to TRUE:

```
$ srvctl modify service -db crm -service xa_01.example.com  
-dtp TRUE
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To enhance the performance of distributed transactions, you can use services to manage DTP environments. By defining the DTP property of a service, the service is guaranteed to run on one instance at a time in an Oracle RAC database. All global distributed transactions performed through the DTP service are ensured to have their tightly coupled branches running on a single Oracle RAC instance. This has the following benefits:

- The changes are available locally within one Oracle RAC instance when tightly coupled branches need information about changes made by each other.
- Relocation and failover of services are fully supported for DTP.
- By using more DTP services than there are Oracle RAC instances, Oracle Database can balance the load by services across all the Oracle RAC database instances.

To leverage all the instances in a cluster, create one or more DTP services for each Oracle RAC instance that hosts distributed transactions. Choose one DTP service for one distributed transaction. Choose different DTP services for different distributed transactions to balance the workload among the Oracle RAC database instances.

Because all the branches of a distributed transaction are on one instance, you can leverage all the instances to balance the load of many DTP transactions through multiple singleton services, thereby maximizing application throughput.

An external transaction manager, such as OraMTS, coordinates DTP/XA transactions. However, an internal Oracle transaction manager coordinates distributed SQL transactions. Both DTP/XA and distributed SQL transactions must use the DTP service in Oracle RAC.

To create distributed transaction processing (DTP) services for distributed transaction processing, perform the following steps:

1. Create a singleton service using Oracle Enterprise Manager or SRVCTL. For an administrator-managed database, define only one instance as the preferred instance. You can have as many available instances as you want, for example:

```
$ srvctl add service -db crm -service xa_01.example.com -preferred RAC01  
-available RAC02,RAC03
```

For a policy-managed database, specify the server pool to use, and set the cardinality of the service to SINGLETON, for example:

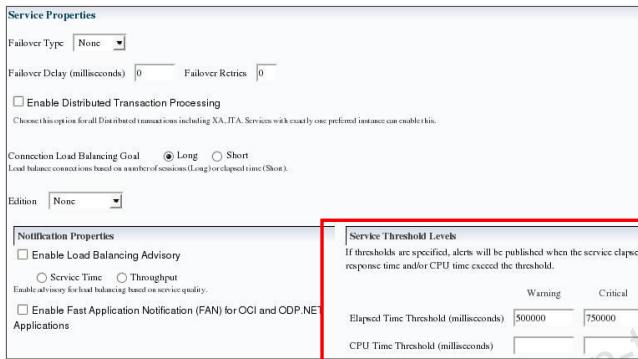
```
$ srvctl add service -db crm -service xa_01.example.com -serverpool  
dtp_pool -cardinality SINGLETON
```

2. Set the DTP parameter (-dtp) for the service to TRUE (the default value is FALSE). You can use Oracle Enterprise Manager or SRVCTL to modify the DTP property of the singleton service. The following example shows how to modify the xa_01.example.com service by using SRVCTL:

```
$ srvctl modify service -db crm -service xa_01.example.com -dtp TRUE
```

Service Thresholds and Alerts

- Service-level thresholds enable you to compare achieved service levels against accepted minimum required levels.
- You can explicitly specify two performance thresholds for each service:
 - **SERVICE_ELAPSED_TIME**: The response time for calls
 - **SERVICE_CPU_TIME**: The CPU time for calls



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

ORACLE®

Service-level thresholds enable you to compare achieved service levels against accepted minimum required levels. This provides accountability for the delivery or the failure to deliver an agreed service level. The end goal is a predictable system that achieves service levels. There is no requirement to perform as fast as possible with minimum resource consumption; the requirement is to meet the quality of service.

You can explicitly specify two performance thresholds for each service: the response time for calls, or **SERVICE_ELAPSED_TIME**, and the CPU time for calls, or **SERVICE_CPU_TIME**. The response time goal indicates that the elapsed time should not exceed a certain value, and the response time represents wall clock time. Response time is a fundamental measure that reflects all delays and faults that might be blocking the call from running on behalf of the user. Response time can also indicate differences in node power across the nodes of an Oracle RAC database.

The service time and CPU time are calculated as the moving average of the elapsed, server-side call time. The AWR monitors the service time and CPU time and publishes AWR alerts when the performance exceeds the thresholds. You can then respond to these alerts by changing the priority of a job, stopping overloaded processes, or by relocating, expanding, shrinking, starting, or stopping a service. This permits you to maintain service availability despite changes in demand.

Services and Thresholds Alerts: Example

```
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');
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To check the thresholds for the `servall` service, use the AWR report. You should record output from the report over several successive intervals during which time the system is running optimally. For example, assume that for an email server, the AWR report runs each Monday during the peak usage times of 10:00 AM to 2:00 PM. The AWR report would contain the response time, or DB time, and the CPU consumption time, or CPU time, for calls for each service. The AWR report would also provide a breakdown of the work done and the wait times that are contributing to the response times.

Using `DBMS_SERVER_ALERT`, set a warning threshold for the `payroll` service at 0.5 seconds and a critical threshold for the payroll service at 0.75 seconds. You must set these thresholds at all instances within an Oracle RAC database. You can schedule actions using Enterprise Manager jobs for alerts, or you can schedule actions to occur programmatically when the alert is received. In this example, thresholds are added for the `servall` service and set as shown in the slide.

Verify the threshold configuration by using the following `SELECT` statement:

```
SELECT metrics_name, instance_name, warning_value, critical_value,
observation_period FROM dba_thresholds;
```

Service Aggregation and Tracing

- Statistics are always aggregated by service to measure workloads for performance tuning.
- Statistics can be aggregated at finer levels:
 - MODULE
 - ACTION
 - Combination of SERVICE_NAME, MODULE, ACTION
- Tracing can be done at various levels:
 - SERVICE_NAME
 - MODULE
 - ACTION
 - Combination of SERVICE_NAME, MODULE, ACTION
- This is useful for tuning systems that use shared sessions.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

By default, important statistics and wait events are collected for the work attributed to every service. An application can further qualify a service by MODULE and ACTION names to identify the important transactions within the service. This enables you to locate exactly the poorly performing transactions for categorized workloads. This is especially important when monitoring performance in systems by using connection pools or transaction processing monitors. For these systems, the sessions are shared, which makes accountability difficult.

SERVICE_NAME, MODULE, and ACTION are actual columns in V\$SESSION. SERVICE_NAME is set automatically at login time for the user. MODULE and ACTION names are set by the application by using the DBMS_APPLICATION_INFO PL/SQL package or special OCI calls. MODULE should be set to a user-recognizable name for the program that is currently executing. Likewise, ACTION should be set to a specific action or task that a user is performing within a module (for example, entering a new customer).

Another aspect of this workload aggregation is tracing by service. The traditional method of tracing each session produces trace files with SQL commands that can span workloads. This results in a hit-or-miss approach to diagnose problematic SQL. With the criteria that you provide (SERVICE_NAME, MODULE, or ACTION), specific trace information is captured in a set of trace files and combined into a single output trace file. This enables you to produce trace files that contain SQL that is relevant to a specific workload being done.

Top Services Performance Page

The screenshot shows the Oracle Database 12c R2 Top Services Performance Page. At the top, there are tabs for Overview, Top Services, Top Modules, Top Actions, Top Clients, and Top Sessions. The Top Services tab is highlighted with a red box. Below the tabs, there are two pie charts: one for Top Services and one for Top Modules (by Service). A red arrow points from the Top Services pie chart down to a detailed table below it. The table lists services with their activity percentages: orcl.cluster01.example.com (50.9%), SYSSBACKGROUND (45.4%), and SYSSUSERS (3.7%). The table also includes columns for Instance, Activity (% for the last 5 minutes), Aggregation Enabled, SQL Trace Enabled, Delta Elapsed Time (seconds), Cumulative Elapsed Time (seconds), and Delta CPU Time (seconds).

Service	Instance	Activity (% for the last 5 minutes)	Aggregation Enabled	SQL Trace Enabled	Delta Elapsed Time (seconds)	Cumulative Elapsed Time (seconds)	Delta CPU Time (seconds)
orcl.cluster01.example.com		43.8	TRUE	FALSE	0.0	1612.0	0.0
SYSSBACKGROUND		50.3	TRUE	FALSE	0.0	5228.0	0.0
SYSSUSERS		5.7	TRUE	FALSE	0.0	6554.0	0.0

From the Performance page, you can access the Top Consumers page by clicking the Top Consumers link.

The Top Consumers page has several tabs for displaying your database as a single-system image. The Overview tabbed page contains four pie charts: Top Clients, Top Services, Top Modules, and Top Actions. Each chart provides a different perspective regarding the top resource consumers in your database.

The Top Services tabbed page displays performance-related information for the services that are defined in your database. Using this page, you can enable or disable tracing at the service level, as well as view the resulting SQL trace file.

Service Aggregation Configuration

- Automatic service aggregation level of statistics
- DBMS_MONITOR used for finer granularity of service aggregations:
 - SERV_MOD_ACT_STAT_ENABLE
 - SERV_MOD_ACT_STAT_DISABLE
- Possible additional aggregation levels:
 - SERVICE_NAME/MODULE
 - SERVICE_NAME/MODULE/ACTION
- Tracing services, modules, and actions:
 - SERV_MOD_ACT_TRACE_ENABLE
 - SERV_MOD_ACT_TRACE_DISABLE
- Database settings persist across instance restarts.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

On each instance, important statistics and wait events are automatically aggregated and collected by service. You do not have to do anything to set this up, except connect with different connect strings by using the services that you want to connect to. However, to achieve a finer level of granularity of statistics collection for services, you must use the SERV_MOD_ACT_STAT_ENABLE procedure in the DBMS_MONITOR package. This procedure enables statistics gathering for additional hierarchical combinations of SERVICE_NAME/MODULE and SERVICE_NAME/MODULE/ACTION. The SERV_MOD_ACT_STAT_DISABLE procedure stops the statistics gathering that was turned on.

The enabling and disabling of statistics aggregation within the service applies to every instance accessing the database. These settings are persistent across instance restarts.

The SERV_MOD_ACT_TRACE_ENABLE procedure enables tracing for services with three hierarchical possibilities: SERVICE_NAME, SERVICE_NAME/MODULE, and SERVICE_NAME/MODULE/ACTION. The default is to trace for all instances that access the database. A parameter is provided that restricts tracing to specified instances where poor performance is known to exist. This procedure also gives you the option of capturing relevant waits and bind variable values in the generated trace files. SERV_MOD_ACT_TRACE_DISABLE disables the tracing at all enabled instances for a given combination of service, module, and action. Like the statistics gathering mentioned previously, service tracing persists across instance restarts.

Service, Module, and Action Monitoring

- For the ERP service, enable monitoring for the exceptions pay action in the PAYROLL module.

```
EXEC DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(
    service_name => 'ERP', module_name=> 'PAYROLL',
    action_name => 'EXCEPTIONS PAY')
```

- For the ERP service, enable monitoring for all the actions in the PAYROLL module:

```
EXEC DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(service_name =>
    'ERP', module_name=> 'PAYROLL', action_name => NULL);
```

- For the HOT_BATCH service, enable monitoring for all actions in the POSTING module:

```
EXEC DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(service_name =>
    'HOT_BATCH', module_name =>'POSTING', action_name => NULL);
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can enable performance data tracing for important modules and actions within each service. The performance statistics are available in the V\$SERV_MOD_ACT_STATS view. Consider the following actions, as implemented in the slide:

- For the ERP service, enable monitoring for the exceptions pay action in the payroll module.
- Under the ERP service, enable monitoring for all the actions in the payroll module.
- Under the HOT_BATCH service, enable monitoring for all actions in the posting module.

Verify the enabled service, module, action configuration with the following SELECT statement:

```
COLUMN AGGREGATION_TYPE FORMAT A21 TRUNCATED HEADING 'AGGREGATION'
COLUMN PRIMARY_ID FORMAT A20 TRUNCATED HEADING 'SERVICE'
COLUMN QUALIFIER_ID1 FORMAT A20 TRUNCATED HEADING 'MODULE'
COLUMN QUALIFIER_ID2 FORMAT A20 TRUNCATED HEADING 'ACTION'
SELECT * FROM DBA_ENABLED_AGGREGATIONS ;
```

The output might appear as follows:

AGGREGATION	SERVICE	MODULE	ACTION
SERVICE_MODULE_ACTION	ERP	PAYROLL	EXCEPTIONS PAY
SERVICE_MODULE_ACTION	ERP	PAYROLL	
SERVICE_MODULE_ACTION	HOT_BATCH	POSTING	

The following sample SQL*Plus script provides service quality statistics for a five-second interval. You can use these service quality statistics to monitor the quality of a service, to direct work, and to balance services across Oracle RAC instances:

```
SET PAGESIZE 60 COLSEP ' | ' NUMWIDTH 8 LINESIZE 132 VERIFY OFF FEEDBACK
OFF
COLUMN service_name FORMAT A20 TRUNCATED HEADING 'Service'
COLUMN begin_time HEADING 'Begin Time' FORMAT A10
COLUMN end_time HEADING 'End Time' FORMAT A10
COLUMN instance_name HEADING 'Instance' FORMAT A10
COLUMN service_time HEADING 'Service Time|mSec/Call' FORMAT 999999999
COLUMN throughput HEADING 'Calls/sec' FORMAT 99.99
BREAK ON service_name SKIP 1
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 ;
```

Service	Begin Time	End Time	Service Time		
			Instance	Sec/Call	Calls/sec
BATCH.cluster01.exam	06:52:38	06:52:43	orcl_1	117	.90
ERP.cluster01.example	06:52:38	06:52:43	orcl_1	21	2.00
	07:16:39	07:16:44	orcl_3	6	.90
SYS\$BACKGROUND	06:52:38	06:52:43	orcl_1	655	.80
	07:16:39	07:16:44	orcl_3	782	.80
SYS\$USERS	06:52:38	06:52:43	orcl_1	420	2.20
	07:16:39	07:16:44	orcl_3	761	.60
orcl.cluster01.example	06:52:38	06:52:43	orcl_1	0	.00
	07:16:39	07:16:44	orcl_3	0	.00
orclXDB	06:52:38	06:52:43	orcl_1	0	.00
	07:16:39	07:16:44	orcl_3	0	.00

Service Performance Views

- Service, module, and action information in:
 - GV\$SESSION
 - GV\$ACTIVE_SESSION_HISTORY
- Service performance in:
 - GV\$SERVICE_STATS
 - GV\$SERVICE_EVENT
 - GV\$SERVICE_WAIT_CLASS
 - GV\$SERVICEMETRIC
 - GV\$SERVICEMETRIC_HISTORY
 - GV\$SERV_MOD_ACT_STATS
 - DBA_ENABLED_AGGREGATIONS
 - DBA_ENABLED_TRACES



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The service, module, and action information are visible in GV\$SESSION and GV\$ACTIVE_SESSION_HISTORY.

The call times and performance statistics are visible in GV\$SERVICE_STATS, GV\$SERVICE_EVENT, VG\$SERVICE_WAIT_CLASS, V\$SERVICEMETRIC, and GV\$SERVICEMETRIC_HISTORY.

When statistics collection for specific modules and actions is enabled, performance measures are visible at each instance in GV\$SERV_MOD_ACT_STATS.

More than 600 performance-related statistics are tracked and visible in GV\$SYSSTAT. Of these, 28 statistics are tracked for services. To see the statistics measured for services, run the following query: `SELECT DISTINCT stat_name FROM v$service_stats`

Of the 28 statistics, DB time and DB CPU are worth mentioning. DB time is a statistic that measures the average response time per call. It represents the actual wall clock time for a call to complete. DB CPU is an average of the actual CPU time spent per call. The difference between response time and CPU time is the wait time for the service. After the wait time is known, and if it consumes a large percentage of response time, then you can trace at the action level to identify the waits.

Note: DBA_ENABLED_AGGREGATIONS displays information about enabled on-demand statistic aggregation. DBA_ENABLED_TRACES displays information about enabled traces.

Quiz



Which of the following statements regarding Oracle Services is *not* correct?

- a. You can group work by type under services.
- b. Users who share a service should have the same service-level requirements.
- c. You use DBMS_SERVICE to manage services, not `srvctl` or Enterprise Manager.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.



Quiz

Is the following statement regarding performance thresholds true or false? The two performance thresholds that can be explicitly set for each service are:

- (a) SERVICE_ELAPSED_TIME: The response time for calls
- (b) SERVICE_CPU_TIME: The CPU time for calls

- a. True
- b. False



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Configure and manage services in a RAC environment
- Use services with client applications
- Use services with the Database Resource Manager
- Use services with the Scheduler
- Set performance-metric thresholds on services
- Configure services aggregation and tracing



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Practice 9: Overview

This practice covers the following topics:

- Working with Services
- Monitoring Services



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

High Availability for Connections and Applications

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Configure client-side connect-time load balancing
- Configure client-side connect-time failover
- Configure server-side connect-time load balancing
- Configure Transparent Application Failover (TAF)
- Describe the purpose of Transaction Guard and Application Continuity
- Describe the key concepts relating to Application Continuity



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Types of Workload Distribution

- Connection balancing is rendered possible by configuring multiple listeners on multiple nodes:
 - Client-side connect-time load balancing
 - Client-side connect-time failover
 - Server-side connect-time load balancing
- Runtime connection load balancing is rendered possible by using connection pools:
 - Work requests automatically balanced across the pool of connections
 - Native feature of Oracle Universal Connection Pool (UCP) for Java, and ODP.NET connection pool



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

With RAC, multiple listeners on multiple nodes can be configured to handle client connection requests for the same database service.

A multiple-listener configuration enables you to leverage the following failover and load-balancing features:

- Client-side connect-time load balancing
- Client-side connect-time failover
- Server-side connect-time load balancing

These features can be implemented either one by one, or in combination with each other.

Moreover, if you are using connection pools, you can benefit from readily available runtime connection load balancing to distribute the client work requests across the pool of connections established by the middle tier. This possibility is offered by the Oracle Universal Connection Pool (UCP) for Java feature as well as Oracle Data Provider for .NET (ODP.NET) connection pool.

Note: Starting with Oracle Database 11g Release 1 (11.1.0.7), Oracle has released the new Universal Connection Pool for JDBC. Consequently, Oracle is deprecating the JDBC connection pool (that is, Implicit Connection Cache) that was introduced in Oracle Database 10g Release 1.

Client-Side Connect-Time Load Balancing

- Client-side load balancing is defined in `tnsnames.ora` by setting the parameter `LOAD_BALANCE=ON`.
 - When set, the Oracle Database randomly selects an address in the address list, and connects to that node's listener.
 - Client connections are balanced across the available SCAN listeners in the cluster.

```
ERP =  
  (DESCRIPTION =  
    (ADDRESS_LIST =  
      (LOAD_BALANCE=ON)  
      (ADDRESS=(PROTOCOL=TCP) (HOST=node1vip) (PORT=1521))  
      (ADDRESS=(PROTOCOL=TCP) (HOST=node2vip) (PORT=1521))  
    )  
    (CONNECT_DATA=(SERVICE_NAME=ERP)))
```

- If SCAN is configured, then client-side load balancing is not relevant for clients supporting SCAN access.
- If SCAN is used, connections are balanced across the three IP addresses defined for SCAN, unless you are using EZConnect.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Client-side load balancing is defined in your client connection definition (`tnsnames.ora` file, for example) by setting the parameter `LOAD_BALANCE=ON`. When you set this parameter to `ON`, Oracle Database randomly selects an address in the address list, and connects to that node's listener. This balances client connections across the available SCAN listeners in the cluster.

If you configured SCAN for connection requests, then client-side load balancing is not relevant for those clients that support SCAN access. When clients connect using SCAN, Oracle Net automatically balances the load of client connection requests across the three IP addresses you defined for the SCAN, unless you are using EZConnect.

The SCAN listener redirects the connection request to the local listener of the instance that is least loaded (if `-clbgoal` is set to `SHORT`) and provides the requested service. When the listener receives the connection request, the listener connects the user to an instance that the listener knows provides the requested service. To see what services a listener supports, run the `lsnrctl services` command.

When clients connect using SCAN, Oracle Net automatically load balances client connection requests across the three IP addresses you defined for the SCAN, unless you are using EZConnect.

Fast Application Notification (FAN): Overview

- FAN is used by RAC to notify other processes about configuration and service-level information.
 - This includes service status changes like UP or DOWN events.
 - UP and DOWN events can apply to instances, services, and nodes.
- Oracle client drivers and Oracle connection pools respond to FAN events and take immediate action.
- Oracle connection pools use FAN to:
 - Quickly detect failures
 - Balance connections following failures
 - Balance connections after failed components are repaired
- Using FAN events eliminates:
 - Applications waiting on TCP timeouts
 - Time wasted processing the last result at the client after a failure has occurred



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

FAN is a high-availability notification mechanism that Oracle RAC uses to notify other processes about configuration and service-level information that includes service status changes, such as UP or DOWN events. The Oracle client drivers and Oracle connection pools respond to FAN events and take immediate action. FAN UP and DOWN events can apply to instances, services, and nodes.

Oracle connection pools, for example, use FAN to receive very fast detection of failures, to balance connections following failures, and to balance connections again after the failed components are repaired. So, when a service at an instance starts, the FAN event is used immediately to route work to that resource. When a service at an instance or node fails, the FAN event is used immediately to interrupt applications to recover.

Using FAN events eliminates applications waiting on TCP timeouts, time wasted processing the last result at the client after a failure has occurred, and time wasted executing work on slow, hung, or dead nodes. For cluster configuration changes, the Oracle RAC high availability framework publishes a FAN event immediately when a state change occurs in the cluster. Instead of waiting for the application to time out against the database and detect a problem, applications can receive FAN events and react immediately. With FAN, in-flight transactions are immediately terminated and the client notified when the instance fails.

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.

Fast Application Notification: Benefits

- No need for connections to rely on connection timeouts
- Used by Load Balancing Advisory to propagate load information
- Designed for enterprise application and management console integration
- Reliable distributed system that:
 - Detects high-availability event occurrences in a timely manner
 - Pushes notification directly to your applications
- Tightly integrated with:
 - Oracle JDBC applications using connection pools
 - ODP.NET connection pool, OCI session pool
 - Oracle WebLogic Server Active Gridlink for Oracle RAC



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

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 down times are extended and more noticeable.

With FAN, important high-availability events are pushed as soon as they are detected, which results in a more efficient use of existing computing resources, and a better integration with your enterprise applications, including mid-tier connection managers, or IT management consoles, including trouble ticket loggers and email/paging servers.

FAN is, in fact, a distributed system that is enabled on each participating node. This makes it very reliable and fault tolerant because the failure of one component is detected by another. Therefore, event notification can be detected and pushed by any of the participating nodes.

FAN events are tightly integrated with Oracle JDBC Universal Connection Pool, ODP.NET connection pool, OCI session pool, Oracle WebLogic Server Active Gridlink for Oracle RAC, and OCI and ODP.NET clients. This includes applications that use Application Continuity or Transaction Guard. For example, Oracle JDBC applications managing connection pools do not need custom code development. They are automatically integrated with the ONS if implicit connection cache and fast connection failover are enabled.

Implementing FAN Events

You can take advantage of FAN events in the following ways:

- Your application can use FAN without programmatic changes if you use an integrated Oracle client:
 - Oracle JDBC Universal Connection Pool (UCP)
 - ODP.NET connection pool
 - Oracle WebLogic Server Active Gridlink for RAC
 - OCI and ODP.NETclients
- Applications can use FAN programmatically:
 - By using the JDBC and RAC FAN API
 - By using callbacks with OCI and ODP.NET to subscribe to FAN events
- You can implement FAN with server-side callouts on your database tier.



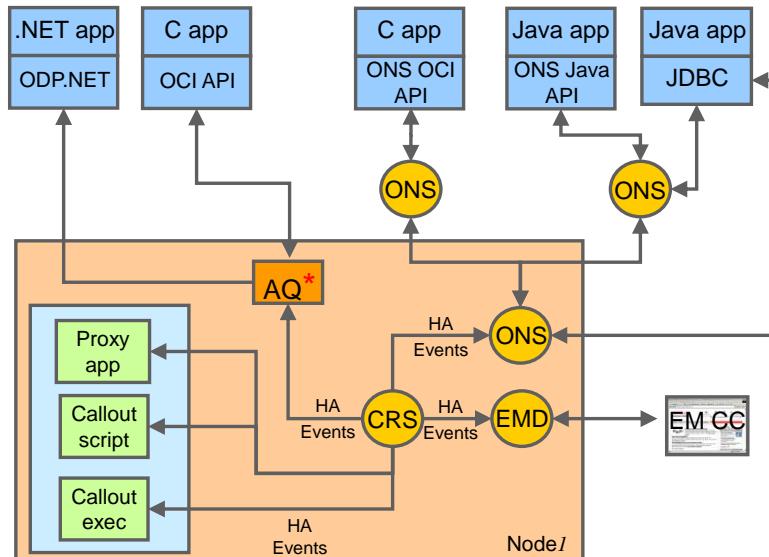
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can take advantage of FAN events in the following three ways:

1. Your application can use FAN without programmatic changes if you use an integrated Oracle client. The integrated clients for FAN events include Oracle JDBC Universal Connection Pool, ODP.NET connection pool, OCI session pool, Oracle WebLogic Server Active Gridlink for Oracle RAC, and OCI and ODP.NET clients. This includes applications that use Application Continuity or Transaction Guard. The integrated Oracle clients must be Oracle Database 10g release 2 or later to take advantage of the FAN high-availability events. The pooled clients can also take advantage of the load balancing advisory FAN events.
2. Applications can use FAN programmatically by using the JDBC and Oracle RAC FAN application programming interface (API) or by using callbacks with OCI and ODP.NET to subscribe to FAN events and to execute event handling actions upon the receipt of an event.
3. You can implement FAN with server-side callouts on your database tier. If you use one of the integrated clients listed in item 1 of the preceding list, then, for DOWN events, the disruption to the application is minimized because the FAN-aware client terminates the sessions to the failed instance or node before they are reused. Incomplete transactions are terminated and the application user is immediately notified. Application users who request connections are directed to available instances only.

For UP events when services and instances are started, new connections are created so the application can quickly take advantage of the extra hardware resources or additional capacity.

FAN and Oracle Integrated Clients



* Streams is continued for backward compatibility to previous Oracle Database releases.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Fast Application Notification (FAN) enables end-to-end, lights-out recovery of applications and load balancing based on real transaction performance in a RAC environment. Applications use the FAN high availability (HA) events to achieve very fast detection of failures, balancing of connection pools following failures, and distribution of connections again when the failed components are repaired.

The FAN events carrying load balancing advice help connection pools consistently deliver connections to available instances that provide the best service. FAN HA is integrated with:

- JDBC-thin
- OCI drivers
- JDBC Universal Connection Pool (and the deprecated Implicit Connection Cache)
- OCI session pools
- ODP.NET connection pool, and
- Oracle WebLogic Server Active GridLink for Oracle RAC.

Because of integration with FAN, integrated clients are more aware of the current status of a RAC cluster. This prevents client connections from waiting or trying to connect to instances or services that are no longer available. When instances start, Oracle RAC uses FAN to notify the connection pool so that the connection pool can create connections to the recently started instance and take advantage of the additional resources that this instance provides.

Oracle client drivers that are integrated with FAN can:

- Remove terminated connections immediately when a service is declared DOWN at an instance, and immediately when nodes are declared DOWN
- Report errors to clients immediately when Oracle Database detects the NOT RESTARTING state, instead of making the client wait while the service repeatedly attempts to restart

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Oracle connection pools that are integrated with FAN can:

- Balance connections across all of the Oracle RAC instances when a service starts; this is preferable to directing the sessions that are defined for the connection pool to the first Oracle RAC instance that supports the service
- Balance work requests at run time using load balancing advisory events

The use of client drivers or connection pools and FAN requires that you properly configure the Oracle Notification Service to deliver the FAN events to the clients. In addition, for load balancing, configure database connection load balancing across all of the instances that provide the services used by the connection pool. Oracle recommends that you configure both client-side and server-side load balancing with Oracle Net Services. If you use DBCA to create your database, then both client-side and server-side load balancing are configured by default.

Note: As indicated in the slide graphic, FAN events are published using ONS Service and Oracle Streams, the latter being continued for backward compatibility to previous Oracle Database releases.

FAN-Supported Event Types

Event type	Description
SERVICE	Primary application service
SRV_PRECONNECT	Shadow application service event (mid-tiers and TAF using primary and secondary instances)
SERVICEMEMBER	Application service on a specific instance
DATABASE	Oracle database
INSTANCE	Oracle instance
ASM	Oracle ASM instance
NODE	Oracle cluster node
SERVICEMETRICS	Load Balancing Advisory



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

FAN Event Reasons

Event Reason	Description
USER	User-initiated commands, such as srvctl and sqlplus
FAILURE	Managed resource polling checks detecting a failure
DEPENDENCY	Dependency of another managed resource that triggered a failure condition
UNKNOWN	Unknown or internal application state when an event is triggered
AUTOSTART	Initial cluster boot: Managed resource has profile attribute AUTO_START=1, and was offline before the last Oracle Clusterware shutdown.
BOOT	Initial cluster boot: Managed resource was running before the last Oracle Clusterware shutdown.
PUBLIC_NW_DOWN	The node is up, but a downed network prevents connectivity.
MEMBER_LEAVE	A node has failed and is no longer part of the cluster.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The event status for each managed resource is associated with an event reason. The reason further describes what triggered the event. The table in the slide gives you the list of possible reasons with a corresponding description.

FAN Event Status

Event status	Description
UP	Managed resource comes up.
DOWN	Managed resource goes down.
PRECONN_UP	Shadow application service comes up.
PRECONN_DOWN	Shadow application service goes down.
NODEDOWN	Managed node goes down.
NOT_RESTARTING	Managed resource cannot fail over to a remote node.
UNKNOWN	Status is unrecognized.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

This table describes the event status for each of the managed cluster resources seen previously.

FAN Event Format

```
<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>
```

```
SERVICE VERSION=1.0 service=ERP.example.com
database=ORCL status=up reason=user card=4 timestamp=16-
Jul-2013 13:21:11
```

```
NODE VERSION=1.0 host=host01
status=nodedown timestamp=16-Jul-2013 11:42:05
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In addition to its type, status, and reason, a FAN event has other payload fields to further describe the unique cluster resource whose status is being monitored and published:

- The event payload version
- The name of the primary or shadow application service. This name is excluded from NODE events.
- The name of the RAC database, which is also excluded from NODE events
- The name of the RAC instance, which is excluded from SERVICE, DATABASE, and NODE events
- The name of the cluster host machine, which is excluded from SERVICE and DATABASE events
- The service cardinality, which is excluded from all events except for SERVICE STATUS=UP events
- The server-side date and time when the event is detected

The general FAN event format is described in the slide along with possible FAN event examples. Note the differences in event payload for each FAN event type.

Load Balancing Advisory: FAN Event

Parameter	Description
VERSION	Version of the event payload
EVENT_TYPE	SERVICEMETRICS
SERVICE	Matches DBA_SERVICES
DATABASE	Unique DB name supporting the service
TIMESTAMP	Date and time stamp (local time zone)
INSTANCE	Instance name supporting the service
PERCENT	Percentage of work to send to this database and instance
FLAG	GOOD, VIOLATING, NO DATA, BLOCKED



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The Load Balancing Advisory FAN event is described in the slide. Basically, it contains a calculated percentage of work requests that should be sent to each instance. The flag indicates the behavior of the service on the corresponding instance relating to the thresholds set on that instance for the service. The easiest way to take advantage of these events is to use the run-time connection load balancing feature of an Oracle integrated client such as JDBC, Universal Connection Pool, ODP.NET Connection Pools, OCI session pools, or Oracle WebLogic Server Active GridLink for Oracle RAC. Other client applications can take advantage of FAN programmatically by using the Oracle RAC FAN API to subscribe to FAN events and execute event-handling actions upon receipt. Here is an example:

Notification Type: database/event/servicemetrics/prod

```
VERSION=1.0 database=PROD service=myServ { {instance=PROD2 percent=38 flag=GOOD aff=TRUE}{instance=PROD3 percent=62 flag=GOOD aff=TRUE} } timestamp=2013-07-30 08:47:06
```

Note: Applications using UCP and Oracle Database 11g or later can take advantage of the affinity feature. If the affinity flag is turned on in the Load Balancing Advisory event, then UCP creates an Affinity Context for the web session such that when that session does a get connection from the pool, the pool always tries to give it a connection to the instance it connected to the first time it acquired a session. The choice of instance for the first connection is based on the current load balancing advisory information. The affinity hint is automatic when load balancing advisory is turned on through setting the goal on the service.

Server-Side Callouts Implementation

- The callout directory:
 - *<Grid Home>/racg/usrco*
 - Can store more than one callout
 - Grants execution on callout directory and callouts only to the Oracle Clusterware user
- Callout execution order is nondeterministic.
- Writing callouts involves:
 1. Parsing callout arguments: The event payload
 2. Filtering incoming FAN events
 3. Executing event-handling programs



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Each database event detected by the RAC High Availability (HA) framework results in the execution of each executable script or program deployed in the standard Oracle Clusterware callout directory. On UNIX, it is *<Grid Home>/racg/usrco*. Unless your Oracle Clusterware home directory is shared across the network, you must deploy each new callout on each RAC node. The order in which these callouts are executed is nondeterministic. However, RAC guarantees that all callouts are invoked once for each recognized event in an asynchronous fashion. Thus, it is recommended to merge callouts whose executions need to be in a particular order.

You can install as many callout scripts or programs as your business requires, provided each callout does not incur expensive operations that delay the propagation of HA events. If many callouts are going to be written to perform different operations based on the event received, it might be more efficient to write a single callout program that merges each single callout.

Writing server-side callouts involves the steps shown in the slide. In order for your callout to identify an event, it must parse the event payload sent by the RAC HA framework to your callout. After the sent event is identified, your callout can filter it to avoid execution on each event notification. Then, your callout needs to implement a corresponding event handler that depends on the event itself and the recovery process required by your business.

Note: As a security measure, make sure that the callout directory and its contained callouts have write permissions only to the system user who installed Oracle Clusterware.

Server-Side Callout Parse: Example

```
#!/bin/sh
NOTIFY_EVENTTYPE=$1
for ARGS in $*; do
  PROPERTY=`echo $ARGS | $AWK -F"=" '{print $1}'` 
  VALUE=`echo $ARGS | $AWK -F"=" '{print $2}'` 
  case $PROPERTY in
    VERSION|version) NOTIFY_VERSION=$VALUE ;;
    SERVICE|service) NOTIFY_SERVICE=$VALUE ;;
    DATABASE|database) NOTIFY_DATABASE=$VALUE ;;
    INSTANCE|instance) NOTIFY_INSTANCE=$VALUE ;;
    HOST|host) NOTIFY_HOST=$VALUE ;;
    STATUS|status) NOTIFY_STATUS=$VALUE ;;
    REASON|reason) NOTIFY_REASON=$VALUE ;;
    CARD|card) NOTIFY_CARDINALITY=$VALUE ;;
    TIMESTAMP|timestamp) NOTIFY_LOGDATE=$VALUE ;;
    ?:?:?:?) NOTIFY_LOGTIME=$PROPERTY ;;
  esac
done
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Unless you want your callouts to be executed on each event notification, you must first identify the event parameters that are passed automatically to your callout during its execution. The example in the slide shows you how to parse these arguments by using a sample Bourne shell script.

The first argument that is passed to your callout is the type of event that is detected. Then, depending on the event type, a set of `PROPERTY=VALUE` strings are passed to identify exactly the event itself.

The script given in the slide identifies the event type and each pair of `PROPERTY=VALUE` strings. The data is then dispatched into a set of variables that can be used later in the callout for filtering purposes.

As mentioned in the previous slide, it might be better to have a single callout that parses the event payload, and then executes a function or another program on the basis of information in the event, as opposed to having to filter information in each callout. This becomes necessary only if many callouts are required.

Note: Make sure that executable permissions are set correctly on the callout script.

Server-Side Callout Filter: Example

```
if ((( [ $NOTIFY_EVENTTYPE = "SERVICE" ] ||  
      [ $NOTIFY_EVENTTYPE = "DATABASE" ] ||  
      [ $NOTIFY_EVENTTYPE = "NODE" ] )  
    ) &&  
    ( [ $NOTIFY_STATUS = "not_restarting" ] ||  
      [ $NOTIFY_STATUS = "restart_failed" ] )  
    ) &&  
    ( [ $NOTIFY_DATABASE = "HQPROD" ] ||  
      [ $NOTIFY_SERVICE = "ERP" ]  
    ))  
then  
  /usr/local/bin/logTicket $NOTIFY_LOGDATE \  
    $NOTIFY_LOGTIME \  
    $NOTIFY_SERVICE \  
    $NOTIFY_DBNAME \  
    $NOTIFY_HOST  
fi
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

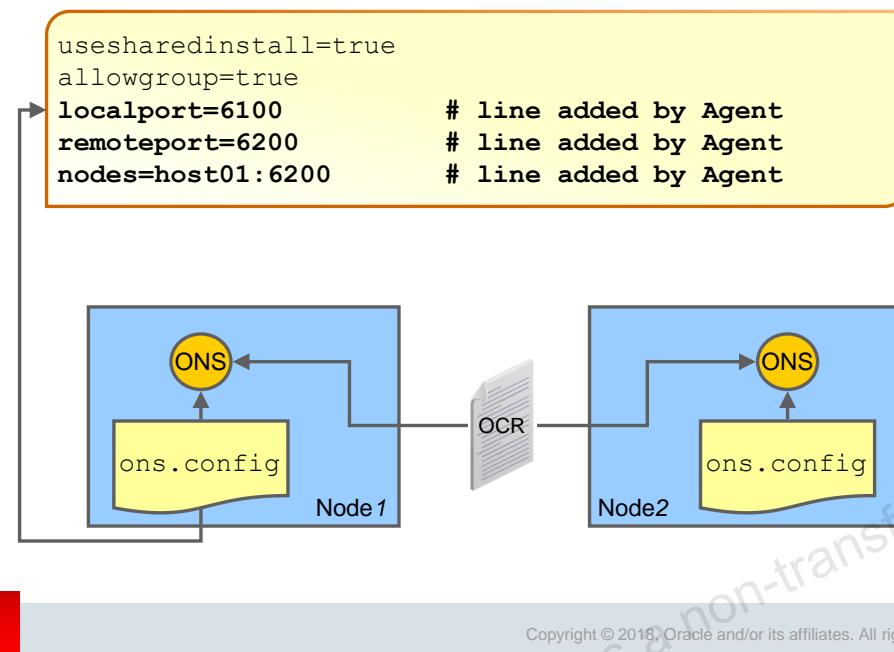
The example in the slide shows you a way to filter FAN events from a callout script. This example is based on the example in the previous slide.

Now that the event characteristics are identified, this script triggers the execution of the trouble-logging program `/usr/local/bin/logTicket` only when the RAC HA framework posts a SERVICE, DATABASE, or NODE event type, with a status set to either `not_restarting` or `restart_failed`, and only for the production HQPROD RAC database or the ERP service.

It is assumed that the `logTicket` program is already created and that it takes the arguments shown in the slide.

It is also assumed that a ticket is logged only for `not_restarting` or `restart_failed` events, because they are the ones that exceeded internally monitored timeouts and seriously need human intervention for full resolution.

Server-Side ONS



ORACLE®

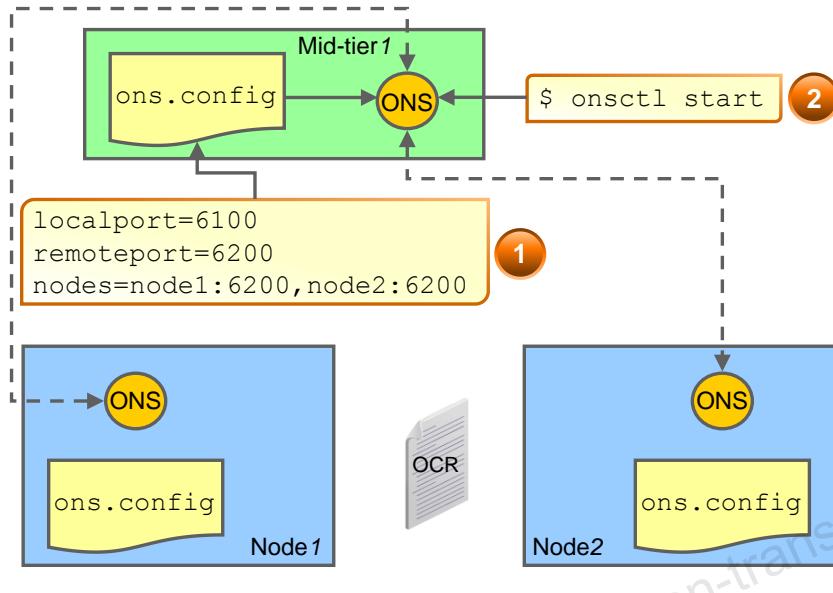
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The ONS configuration is controlled by the `<GRID_HOME>/opmn/conf/ons.config` configuration file. This file is automatically created during installation. Starting with Oracle Database 11g Release 2 (11.2) it is automatically maintained by the CRS ONS agent using information stored in the OCR. There are three important parameters that are always configured for each ONS:

- The first is `localport`, the port that ONS uses to talk to local clients.
- The second is `remoteport`, the port that ONS uses to talk to other ONS daemons.
- The third parameter is called `nodes`. It specifies the list of other ONS daemons to talk to. This list includes all RAC ONS daemons, and all mid-tier ONS daemons. Node values are given as either host names or IP addresses followed by their `remoteport`. This information is stored in Oracle Cluster Registry (OCR).

In the slide, it is assumed that ONS daemons are already started on each cluster node. This should be the default situation after a correct RAC installation.

Optionally Configuring the Client-Side ONS



ORACLE®

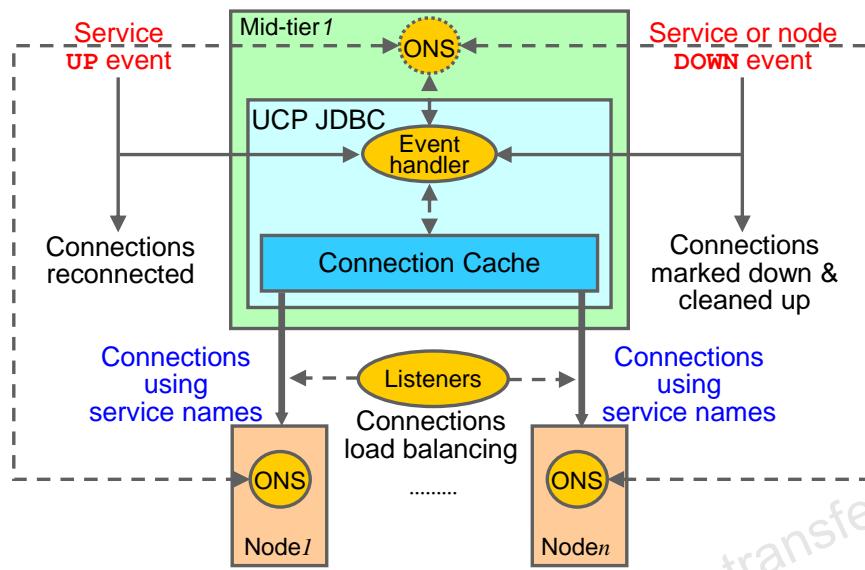
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle Database 11g Release 2 introduced a new set of APIs for Oracle RAC Fast Application Notification (FAN) events. These APIs provide an alternative for taking advantage of the high-availability (HA) features of Oracle Database, if you do not use Universal Connection Pool or Oracle JDBC connection caching. These APIs are not a part of Oracle JDBC APIs. For using Oracle RAC Fast Application Notification, the `ons.jar` file must be present in the `CLASSPATH` or an Oracle Notification Services (ONS) client must be installed and running in the client system. To use ONS on the client side, you must configure all the RAC nodes in the ONS configuration file. A sample configuration file might look like the one shown in the slide.

After configuring ONS, you start the ONS daemon with the `onsctl start` command. It is your responsibility to make sure that an ONS daemon is running at all times. You can check that the ONS daemon is active by executing the `onsctl ping` command.

Note: With Oracle Database 10g Release 2 and later, there is no requirement to use ONS daemons on the mid-tier when using the Oracle Universal Connection Pool. To configure this option, use either the `OracleDataSource` property or a setter API `setONSConfiguration (configStr)`. The input to this API are the contents of the `ons.config` file specified as a string. The `ons.jar` file must be on the client's `CLASSPATH`. There are no daemons to start or manage.

UCP JDBC Fast Connection Failover: Overview



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Oracle Universal Connection Pool (UCP) provides a tight integration with Oracle RAC database features like Fast Connection Failover (FCF). Basically, FCF is a FAN client implemented through the connection pool. FCF quickly and automatically recovers lost or damaged connections. This automatic connection management results from FAN events received by the local ONS daemon, or by a remote ONS if a local one is not used, and is handled by a special event handler thread. Both JDBC thin and JDBC OCI drivers are supported.

Therefore, if UCP and FCF are enabled, your Java program automatically becomes an ONS subscriber without having to manage FAN events directly.

Whenever a service or node down event is received by the mid-tier ONS, the event handler automatically marks the corresponding connections as down and cleans them up. This prevents applications that request connections from the cache from receiving invalid or bad connections.

Whenever a service up event is received by the mid-tier ONS, the event handler recycles some unused connections and reconnects them using the event service name. The number of recycled connections is automatically determined by the connection cache. Because the listeners perform connection load balancing, this automatically rebalances connections across the preferred instances of the service without waiting for connection requests or retries.

For more information see *Oracle Universal Connection Pool for JDBC Developer's Guide*.

JDBC/ODP.NET FCF Benefits

- Database connections are balanced across preferred instances according to LBA.
- Database work requests are balanced across preferred instances according to LBA.
- Database connections are anticipated.
- Database connection failures are immediately detected and stopped.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

By enabling FCF, your existing Java applications connecting through Oracle JDBC and application services, or your .NET applications using ODP.NET connection pools and application services benefit from the following:

- All database connections are balanced across all RAC instances that support the new service name, instead of having the first batch of sessions routed to the first RAC instance. This is done according to the Load Balancing Advisory algorithm you use (see the next slide). Connection pools are rebalanced upon service, instance, or node up events.
- The connection cache immediately starts placing connections to a particular RAC instance when a new service is started on that instance.
- The connection cache immediately shuts down stale connections to RAC instances where the service is stopped on that instance, or whose node goes down.
- Your application automatically becomes a FAN subscriber without having to manage FAN events directly by just setting up flags in your connection descriptors.
- An exception is immediately thrown as soon as the service status becomes not_restarting, which avoids wasteful service connection retries.

Note: For more information about how to subscribe to FAN events, refer to the *Oracle Database JDBC Developer's Guide* and *Oracle Data Provider for .NET Developer's Guide*.

Load Balancing Advisory

- The Load Balancing Advisory (LBA) is an advisory for sending work across RAC instances.
- The LBA advice is available to all applications that send work:
 - JDBC and ODP connection pools
 - Connection load balancing
- The LBA advice sends work to where services are executing well and resources are available:
 - Relies on service goodness
 - Adjusts distribution for different power nodes, different priority and shape workloads, and changing demand
 - Stops sending work to slow, hung, or failed nodes



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

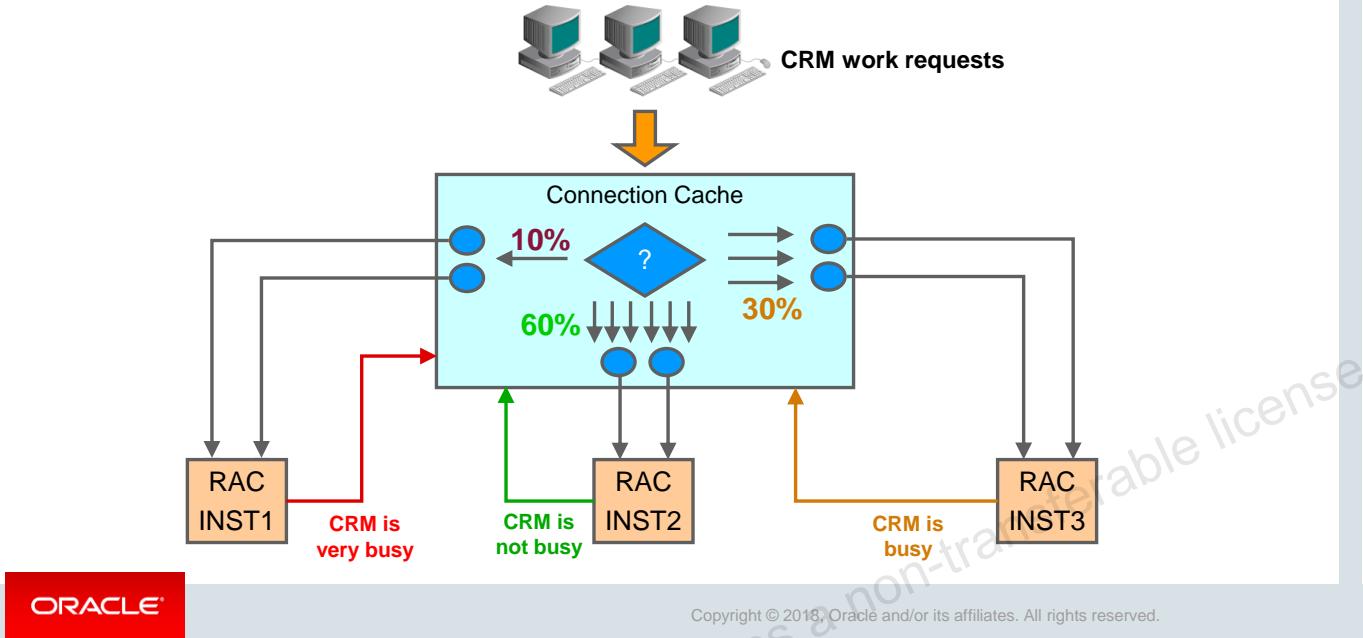
Load balancing distributes work across all of the available RAC database instances. Well-written applications use persistent connections that span the instances of RAC offering a service. Connections are created infrequently and exist for a long duration. Work comes into the system with high frequency, borrows these connections, and exists for a relatively short duration.

The Load Balancing Advisory has the task of advising the direction of incoming work to the RAC instances that provide optimal quality of service for that work. The LBA algorithm uses metrics sensitive to the current performance of services across the system.

The load balancing advisory is deployed with key Oracle clients, such as a listener, the JDBC universal connection pool, OCI session pool, Oracle WebLogic Server Active GridLink for Oracle RAC, and the ODP.NET Connection Pools. Third-party applications can also subscribe to load balancing advisory events by using JDBC and Oracle RAC FAN API or by using callbacks with OCI.

Using the Load Balancing Advisory for load balancing recognizes machine power differences, sessions that are blocked in wait, failures that block processing, as well as competing services of different importance. Using the Load Balancing Advisory prevents sending work to nodes that are overworked, hung, or failed.

UCP JDBC/ODP.NET Runtime Connection Load Balancing: Overview



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Without using the Load Balancing Advisory, work requests to RAC instances are assigned on a random basis, which is suitable when each instance is performing equally well. However, if one of the instances becomes more burdened than the others because of the amount of work resulting from each connection assignment, the random model does not perform optimally.

The Runtime Connection Load Balancing feature provides assignment of connections based on the Load Balancing Advisory information from the instances in the RAC cluster. The Connection Cache assigns connections to clients on the basis of a relative number indicating what percentage of work requests each instance should handle.

In the diagram in the slide, the feedback indicates that the CRM service on INST1 is so busy that it should service only 10% of the CRM work requests; INST2 is so lightly loaded that it should service 60%; and INST3 is somewhere in the middle, servicing 30% of requests. Note that these percentages apply to, and the decision is made on, a per-service basis. In this example, CRM is the service in question.

Note: Runtime Connection Load Balancing is a feature of Oracle connection pools.

Connection Load Balancing in RAC

- Connection load balancing allows scan listeners to distribute connection requests to the best instances.
- This is dependant on the `-clbgoal` setting for the service.
- For connection load balancing, you can use a goal of:
 - **LONG**: Use this connection load balancing method if run-time load balancing is not required. (Typical for batch operations)

```
$ srvctl modify service -db orcl -service batchconn  
-clbgoal LONG
```
 - **SHORT**: Use this method for applications that use run-time load balancing or when using connection pools that are integrated with LBA.

```
$ srvctl modify service -db orcl -service oltpapp  
-clbgoal SHORT
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle Net Services provides the ability to distribute client connections across the instances in a RAC configuration. With server-side load balancing, the SCAN listener directs a connection request to the best instance currently providing the service, based on the `-clbgoal` setting for the service.

When you create a RAC database with DBCA, it automatically configures and enables server-side load balancing. A sample client-side load balancing connection definition in the `tnsnames.ora` file on the server is also created. FAN, Fast Connection Failover, and the load balancing advisory depend on an accurate connection load balancing configuration that includes setting the connection load balancing goal for the service. the goal can be either `LONG` or `SHORT` for connection load balancing. These goals have the following characteristics:

- **SHORT**: Use the `SHORT` connection load balancing method for applications that use run-time load balancing. When using connection pools that are integrated with Load Balancing Advisory, set the goal to `SHORT`.
- **LONG**: Use the `LONG` method if run-time load balancing is not required. This is typical for batch operations. `LONG` is the default connection load balancing goal.

Setting the run-time connection load balancing goal to `NONE` disables load balancing for the service. You can see the goal settings for a service in the data dictionary by querying the `DBA_SERVICES`, `V$SERVICES`, and `V$ACTIVE_SERVICES` views. You can also review the load balancing settings for a service using Oracle Enterprise Manager.

Monitoring LBA FAN Events

```
SQL> SELECT TO_CHAR(enq_time, 'HH:MI:SS') Enq_time, user_data
  2  FROM sys.sys$service_metrics_tab
  3  ORDER BY 1 ;

ENQ_TIME USER_DATA
-----
...
04:19:46 SYS$RLBTYP('MYSERV', 'VERSION=1.0 database=orcl
  service=MYSERV { {instance=orcl_2 percent=50
  flag=UNKNOWN} {instance=orcl_1 percent=50 flag=UNKNOWN}
  } timestamp=2013-07-19 11:07:32')
04:20:16 SYS$RLBTYP('MYSERV', 'VERSION=1.0 database=orcl
  service=MYSERV { {instance=orcl_2 percent=80
  flag=UNKNOWN} {instance=orcl_1 percent=20 flag=UNKNOWN}
  } timestamp=2013-07-19 11:08:11')
SQL>
```

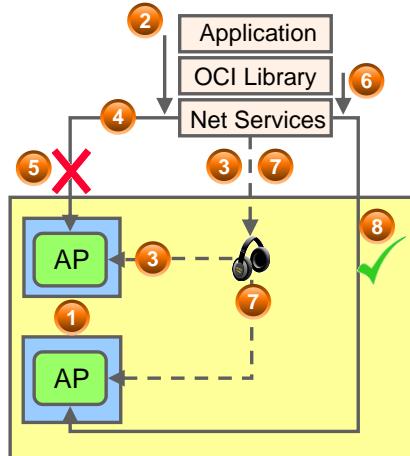


Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

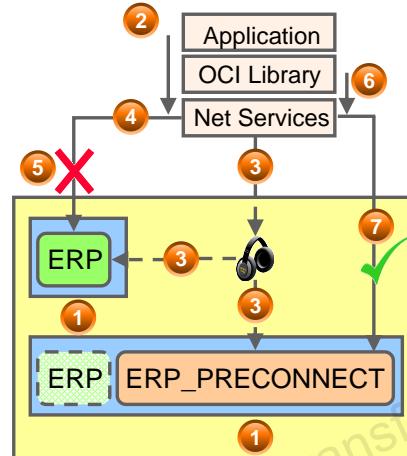
You can use the SQL query shown in the slide to monitor the Load Balancing Advisory FAN events for each of your services.

Transparent Application Failover: Overview

TAF Basic



① TAF Preconnect



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

TAF is a runtime feature of the OCI driver. It enables your application to automatically reconnect to the service if the initial connection fails. During the reconnection, although your active transactions are rolled back, TAF can optionally resume the execution of a `SELECT` statement that was in progress. TAF supports two failover methods:

- With the `BASIC` method, the reconnection is established at failover time. After the service has been started on the nodes (1), the initial connection (2) is made. The listener establishes the connection (3), and your application accesses the database (4) until the connection fails (5) for any reason. Your application then receives an error the next time it tries to access the database (6). Then, the OCI driver reconnects to the same service (7), and the next time your application tries to access the database, it transparently uses the newly created connection (8). TAF can be enabled to receive FAN events for faster down events detection and failover.
- The `PRECONNECT` method is similar to the `BASIC` method except that it is during the initial connection that a shadow connection is also created to anticipate the failover. TAF guarantees that the shadow connection is always created on the available instances of your service by using an automatically created and maintained shadow service.

Note: Optionally, you can register TAF callbacks with the OCI layer. These callback functions are automatically invoked at failover detection and allow you to have some control of the failover process. For more information, refer to the *Oracle Call Interface Programmer's Guide*.

TAF Basic Configuration on Server-Side: Example

```
$ srvctl add service -db RACDB -service APSVC
  -failovermethod BASIC -failovertype SELECT
  -failoverretry 10 -failoverdelay 30 -serverpool sp1
$ srvctl start service -db RACDB -service APSVC
```

```
apsvc =
(DESCRIPTION =
(ADDRESS = (PROTOCOL = TCP) (HOST = cluster01-scan)
(PORT = 1521))
(CONNECT_DATA =
(SERVICE_NAME = apsvc)))
```

```
$ sqlplus AP/AP@apsvc
SQL> SELECT inst_id, username, service_name, failover_type,
      failover_method
    FROM gv$session WHERE username='AP';

INST_ID USERNAME SERVICE_NAME FAILOVER_TYPE FAILOVER_M
----- ----- -----
1          AP        apsvc       SELECT      BASIC
```

ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Before using TAF, it is recommended that you create and start a service that is to be used when establishing connections. By doing so, you benefit from the integration of TAF and services. When you wish to use BASIC TAF with a service, you should use the `-failovermethod BASIC` option when creating the service (TAF failover method is used for backward compatibility only). You can define the TAF policy by setting the number of times that a failed session attempts to reconnect to the service and how long it should wait between reconnection attempts using the `-failoverretry` and `-failoverdelay` parameters, respectively. After the service is created, you simply start it on your database.

TAF can be configured at the client side in `tnsnames.ora` or at the server side using the `srvctl` utility as shown above. Configuring it at the server is preferred as it is convenient to put the configuration in a single place (the server).

Your application needs to connect to the service by using a connection descriptor similar to the one shown in the slide. In the example above, notice that the cluster SCAN is used in the descriptor. Once connected, the `GV$SESSION` view will reflect that the connection is TAF-enabled. The `FAILOVER_METHOD` and `FAILOVER_TYPE` column reflects this and confirms that the TAF configuration is correct.

TAF Basic Configuration on a Client-Side: Example

```
$ srvctl add service -db RACDB -service AP -serverpool sp1
```

```
$ srvctl start service -db RACDB -service AP
```

```
AP =
(DESCRIPTION =(FAILOVER=ON) (LOAD_BALANCE=ON)
 (ADDRESS=(PROTOCOL=TCP) (HOST=N1VIP) (PORT=1521))
 (ADDRESS=(PROTOCOL=TCP) (HOST=N2VIP) (PORT=1521))
 (CONNECT_DATA =
   (SERVICE_NAME = AP)
   (FAILOVER_MODE= (TYPE=select)
     (METHOD=basic)
     (RETRIES=20)
     (DELAY=15))))
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

TAF Preconnect Configuration: Example

```
$ srvctl add service -db RACDB -service ERP
  -preferred I1 -available I2
  -tafpolicy PRECONNECT

$ srvctl start service -db RACDB -service ERP
```

```
ERP =
(DESCRIPTION = (FAILOVER=ON) (LOAD_BALANCE=ON)
 (ADDRESS=(PROTOCOL=TCP) (HOST=N1VIP) (PORT=1521))
 (ADDRESS=(PROTOCOL=TCP) (HOST=N2VIP) (PORT=1521))
 (CONNECT_DATA = (SERVICE_NAME = ERP)
  (FAILOVER_MODE = (BACKUP=ERP_PRECONNECT)
   (TYPE=SESSION) (METHOD=PRECONNECT))))
```



```
ERP_PRECONNECT =
(DESCRIPTION = (FAILOVER=ON) (LOAD_BALANCE=ON)
 (ADDRESS=(PROTOCOL=TCP) (HOST=N1VIP) (PORT=1521))
 (ADDRESS=(PROTOCOL=TCP) (HOST=N2VIP) (PORT=1521))
 (CONNECT_DATA = (SERVICE_NAME = ERP_PRECONNECT)))
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In order to use PRECONNECT TAF, it is mandatory that you create a service with preferred and available instances. Also, in order for the shadow service to be created and managed automatically by Oracle Clusterware, you must define the service with the `-tafpolicy PRECONNECT` option. TAF policy specification is for administrator-managed databases only. The shadow service is always named using the format `<service_name>_PRECONNECT`.

When the TAF service settings are defined on the client side only, you need to configure a special connection descriptor in your `tnsnames.ora` file to use the PRECONNECT method. One such connection descriptor is shown in the slide.

The main differences with the previous example are that `METHOD` is set to `PRECONNECT` and an additional parameter is added. This parameter is called `BACKUP` and must be set to another entry in your `tnsnames.ora` file that points to the shadow service.

TAF Verification

```
SELECT machine, failover_method, failover_type,
       failed_over, service_name, COUNT(*)
  FROM v$session
 GROUP BY machine, failover_method, failover_type,
          failed_over, service_name;
```

1st node

MACHINE FAILOVER_M FAILOVER_T FAI SERVICE_N COUNT(*)					
-----	-----	-----	-----	-----	-----
node1	BASIC	SESSION	NO	AP	1
node1	PRECONNECT	SESSION	NO	ERP	1

2nd node

MACHINE FAILOVER_M FAILOVER_T FAI SERVICE_N COUNT(*)					
-----	-----	-----	-----	-----	-----
node2	NONE	NONE	NO	ERP_PRECO	1

2nd node after

MACHINE FAILOVER_M FAILOVER_T FAI SERVICE_N COUNT(*)					
-----	-----	-----	-----	-----	-----
node2	BASIC	SESSION	YES	AP	1
node2	PRECONNECT	SESSION	YES	ERP_PRECO	1



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To determine whether TAF is correctly configured and that connections are associated with a failover option, you can examine the `V$SESSION` view. To obtain information about the connected clients and their TAF status, examine the `FAILOVER_TYPE`, `FAILOVER_METHOD`, `FAILED_OVER`, and `SERVICE_NAME` columns. The example includes one query that you could execute to verify that you have correctly configured TAF. This example is based on the previously configured `AP` and `ERP` services, and their corresponding connection descriptors.

The first output in the slide is the result of the execution of the query on the first node after two SQL*Plus sessions from the first node have connected to the `AP` and `ERP` services, respectively. The output shows that the `AP` connection ended up on the first instance. Because of the load-balancing algorithm, it can end up on the second instance. Alternatively, the `ERP` connection must end up on the first instance because it is the only preferred one.

The second output is the result of the execution of the query on the second node before any connection failure. Note that there is currently one unused connection established under the `ERP_PRECONNECT` service that is automatically started on the `ERP` available instance.

The third output is the one corresponding to the execution of the query on the second node after the failure of the first instance. A second connection has been created automatically for the `AP` service connection, and the original `ERP` connection now uses the preconnected connection.

FAN Connection Pools and TAF Considerations

- Both techniques are integrated with services and provide service connection load balancing.
- Connection pools that use FAN are always preconnected.
- TAF may rely on operating system (OS) timeouts to detect failures.
- FAN never relies on OS timeouts to detect failures.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Because connection load balancing is a listener functionality, both FCF and TAF automatically benefit from connection load balancing for services.

When you use FCF, there is no need to use TAF.

For example, you do not need to preconnect if you use FAN in conjunction with connection pools. The connection pool is always preconnected.

With both techniques, you automatically benefit from VIPs at connection time. This means that your application does not rely on lengthy operating system connection timeouts at connect time, or when issuing a SQL statement. However, when in the SQL stack, and the application is blocked on a read/write call, the application needs to be integrated with FAN in order to receive an interrupt if a node goes down. In a similar case, TAF may rely on OS timeouts to detect the failure. This takes much more time to fail over the connection than when using FAN.

Introducing Transaction Guard and Application Continuity

Oracle Database 12c introduces two fundamental capabilities for ensuring continuity of applications after database outages:

- **Transaction Guard:** A new reliable protocol and API that returns the outcome of the last transaction after a recoverable error has occurred
- **Application Continuity:** A feature that attempts to mask database session outages by recovering the in-flight work for requests submitted to the database



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c introduces two fundamental capabilities for ensuring continuity of applications after database outages:

1. A foolproof way for applications to know the outcome of transactions
2. The ability to mask outages by reconnecting to the database and replaying the workload

These capabilities are provided by two new features: Transaction Guard and Application Continuity.

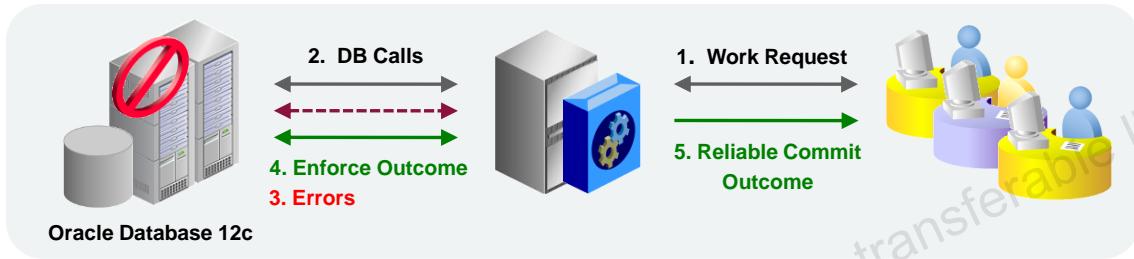
- Transaction Guard is an API that applications use in error handling. It is a new and reliable way to return the outcome of the last transaction after a recoverable error has occurred. By itself, Transaction Guard can dramatically improve the end-user experience by erasing the doubt over whether the last transaction was committed or not.
- Application Continuity is a feature that masks recoverable outages from end users and applications. Application Continuity attempts to replay the transactional and nontransactional work that constitutes a database request. When replay is successful, the outage appears to the end user as if the execution was slightly delayed. With Application Continuity, the end user experience is improved because users may never sense that an outage has occurred. Furthermore, Application Continuity can simplify application development by removing the burden of dealing with recoverable outages.

What Is Transaction Guard?

Transaction Guard:

- Is a tool that provides a reliable commit outcome for the last transaction after errors
- Is an API available for JDBC Thin, C/C++ (OCI/OCCI), and ODP.NET
- Is used by Application Continuity for at-most-once execution
- Can be used independently of Application Continuity

Without Transaction Guard, retry can cause logical corruption.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Benefits of Transaction Guard

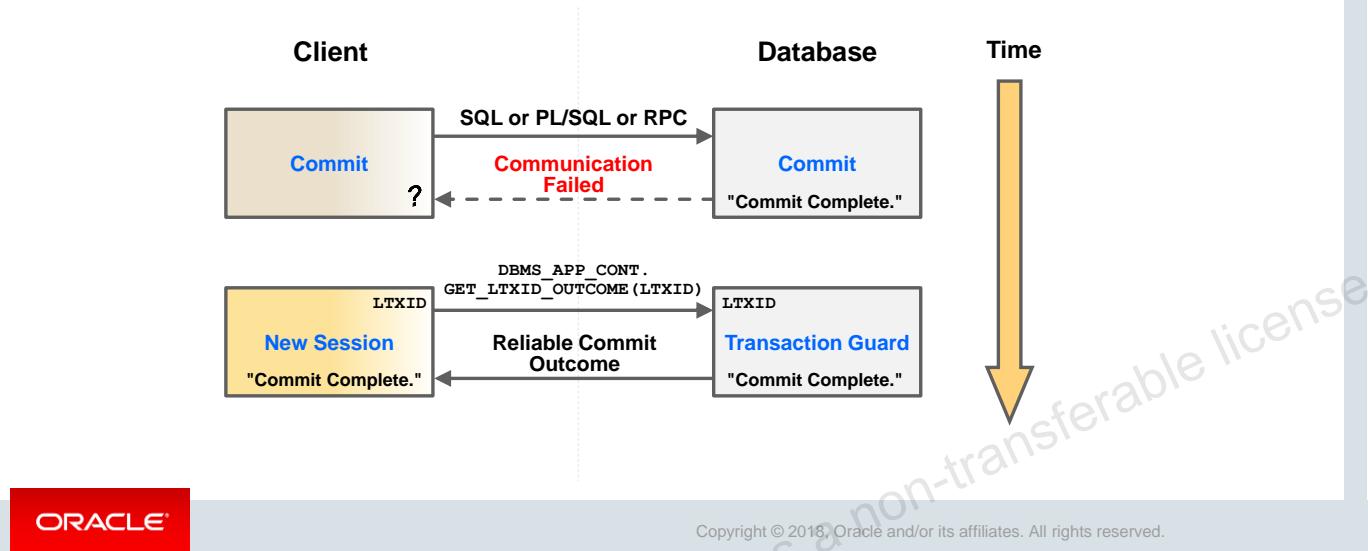
- After outages, users know what happened to their in-flight transactions, such as fund transfers, flight bookings, and bill payments.
- Transaction Guard provides better performance and reliability than home-built code for idempotence.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

How Transaction Guard Works

Transaction Guard is a reliable protocol and API that enables applications to know the outcome of the last transaction.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In the standard commit case, the database commits a transaction and returns a success message to the client. In the illustration shown in the slide, the client submits a commit statement and receives a message stating that communication failed. This type of failure can occur due to several reasons, including a database instance failure or network outage. In this scenario, without Transaction Guard, the client does not know the outcome of the transaction.

Oracle Database solves the problem by using a globally unique identifier called a logical transaction ID (LTXID). When the application is running, both the database and client hold the logical transaction ID. The database supplies the client with a logical transaction ID at authentication and at each round trip from the client driver that executes one or more commit operations.

When a recoverable outage occurs, the logical transaction ID uniquely identifies the last database transaction submitted on the session that failed. A new PL/SQL interface (`DBMS_APP_CONT.GET_LTXID_OUTCOME`) interface returns the reliable commit outcome. Further detail on using Transaction Guard APIs is provided later in the lesson.

Using Transaction Guard

- Supported transaction types:
 - Local commit
 - Auto-commit and Commit on Success
 - Commit embedded in PL/SQL
 - DDL, DCL, and Parallel DDL
 - Remote, Distributed commit
 - XA transactions using One Phase Optimizations
- Not supported in release 12.2:
 - Two Phase XA transactions
 - Read-write database links from Active Data Guard
- Server configuration:
 - Set the `COMMIT_OUTCOME=TRUE` service attribute
 - Optionally, set the `RETENTION_TIMEOUT` service attribute
- Supported clients:
 - JDBC Thin, OCI, OCCI, and ODP.NET



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You may use Transaction Guard on each database in your system, including restarting on and failing over between single instance database, Real Application Clusters, Data Guard and Active Data Guard.

Transaction Guard is supported with the following Oracle Database 12c configurations:

- Single Instance Oracle RDBMS
- Real Application Clusters
- Data Guard
- Active Data Guard
- Multitenant including unplug/plug and for 12.2 relocates across the PDB/CDB, but excludes "with clone" option
- Global Data Services for the above database configurations

Transaction Guard supports all the transaction types listed in the slide. The primary exclusions in Oracle Database12c release 12.2 are:

- Two Phase XA transactions
- Active Data Guard with read/write database links to another database

To enable Transaction Guard, set the service attribute `COMMIT_OUTCOME=TRUE`. Optionally, change the `RETENTION_TIMEOUT` service attribute to specify the amount of time that the commit outcome is retained. The retention timeout value is specified in seconds; the default is 86400 (24 hours), and the maximum is 2592000 (30 days).

Transaction Guard supports the following client drivers:

- 12c JDBC type 4 driver
- 12c OCI and OCCI client drivers
- 12c Oracle Data Provider for .NET (ODP.NET), Unmanaged Driver
- 12c ODP.NET, Managed Driver in ODAC 12c Release 4 or higher

For more information about Transaction Guard, refer to *Oracle Database Development Guide, Release 2 (12.2)*

Creating Services for Transaction Guard

- To create a service using Transaction Guard but not Application continuity:

```
$ srvctl add service -db racdb -service app3
  -serverpool Srvpool1
  -commit_outcome TRUE          Mandatory Settings for Transaction Guard
  -retention 86400 -failoverretry 30 -failoverdelay 10
  -notification TRUE -rlbgoal SERVICE_TIME -clbgoal SHORT
```

- To modify an existing service to enable Transaction Guard:

```
$ srvctl modify service -db racdb -service app4
  -commit_outcome TRUE -retention 86400
  -notification TRUE
```

- To use Transaction Guard, a DBA must grant permission:

```
SQL> GRANT execute ON DBMS_APP_CONT;
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To enable Transaction Guard, but not Application Continuity, create the service using SRVCTL and set only `-commit_outcome` to TRUE.

You can use SRVCTL to modify an existing service to enable Transaction Guard, similar to the following command, where `racdb` is the name of your Oracle RAC database, and `app2` is the name of the service you are modifying:

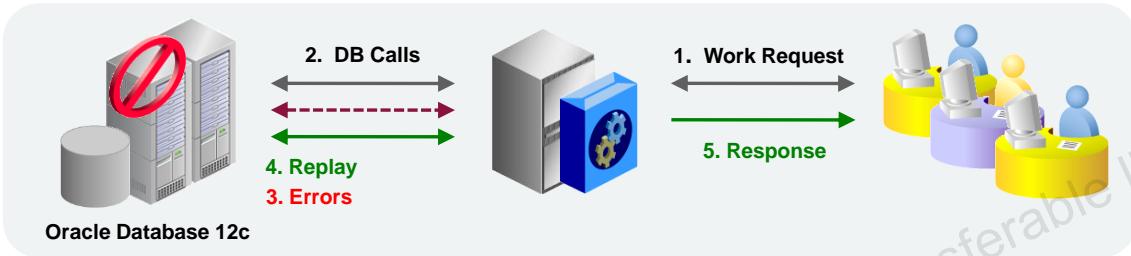
```
$ srvctl modify service -db racdb -service app2 -commit_outcome TRUE
  -retention 86400 -notification TRUE
```

In the preceding example, the `-retention` parameter specifies how long, in seconds, to maintain the history. Additionally the `-notification` parameter is set to TRUE, enabling FAN events. To use Transaction Guard, a DBA must grant permission, as follows:

```
GRANT EXECUTE ON DBMS_APP_CONT;
```

What Is Application Continuity?

- Replays in-flight work on recoverable errors
- Masks many hardware, software, network, storage errors, and outages, when successful
- Improves the end-user experience



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Without Application Continuity, network outages, instance failures, hardware failures, repairs, configuration changes, patches, and so on can result in the failure of a user session followed by an error message of some sort.

Application Continuity masks many recoverable database outages from applications and users. It achieves the masking by restoring the database session, the full session (including session state, cursors, and variables), and the last in-flight transaction (if there is one).

If the database session becomes unavailable due to a recoverable error, Application Continuity attempts to rebuild the session and any open transactions to the correct states. If the last transaction was successful and does not need to be reexecuted, the successful status is returned to the application. Otherwise, Application Continuity will replay the last in-flight transaction.

To be successful, the replay must return to the client exactly the same data that the client received previously in the original request. This ensures that any decisions based on previously queried data are honored during the replay. If the replay is successful, the application continues safely without duplication.

If the replay is not successful, the database rejects the replay and the application receives the original error. This ensures that the replay does not proceed if circumstances change between the original request and the replay.

Benefits of Application Continuity

- Uninterrupted user service, when replay is successful
- Can help relocate database sessions to remaining servers for planned outages
- Improves developer productivity by masking outages that can be masked
- Few or no application changes
- Simple configuration

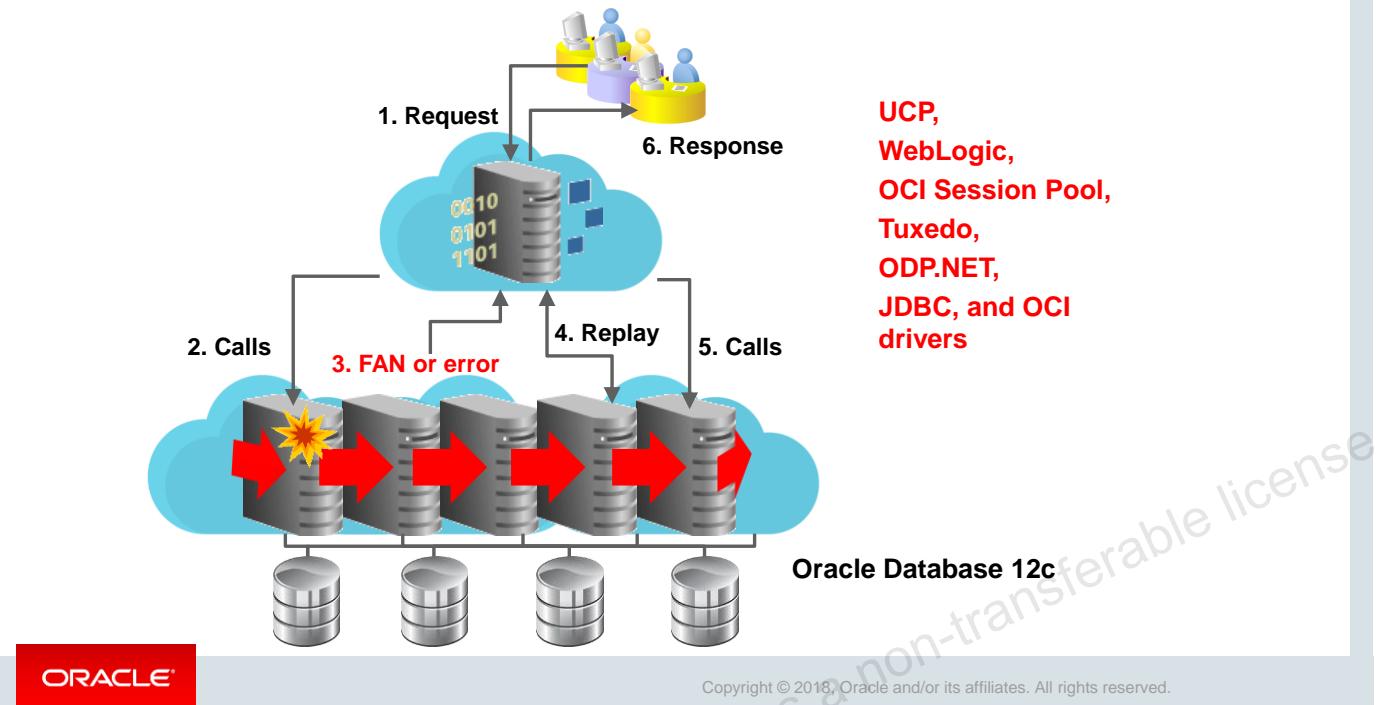


Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Here are some benefits of Application Continuity:

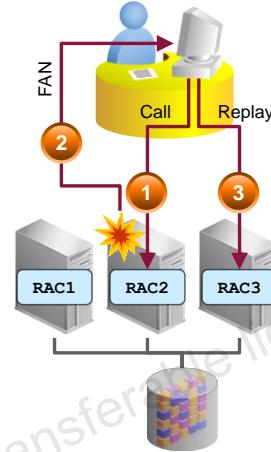
- User service is uninterrupted when the request replay is successful.
- Application Continuity can be used to migrate the database sessions to the remaining servers without the users perceiving an outage.
- Masking outages that can be masked improves developer productivity. Error handling code will be invoked less often and can potentially be simplified.
- Replay often requires few or no application changes.
- Application Continuity is simple to configure.

How Does Application Continuity Work?



RAC and Application Continuity

- Application Continuity transparently replays database requests after a failed session.
 - Users are shielded from many types of problems.
- Using Application Continuity with RAC provides:
 - Protection against a wider variety of failure scenarios
 - Faster reconnect and replay
 - Request replay on another RAC instance



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Application Continuity is a feature that rapidly and transparently replays a request against the database after a recoverable error that makes the database session unavailable. The request can contain transactional and nontransactional work. With Application Continuity, the end user experience is improved by masking many system, communication, hardware, and storage problems from the end user.

When Application Continuity is used in conjunction with Oracle RAC, failed sessions can be quickly restarted and replayed on another RAC database instance. Using Application Continuity in conjunction with Oracle RAC provides protection against a wider variety of possible failures compared with using Application Continuity against a single instance database. Also, using Application Continuity in conjunction with Oracle RAC enables quicker replay compared with using Application Continuity in conjunction with Data Guard because reconnecting to another already running database instance can be completed in a few seconds while a Data Guard failover operation may take a few minutes.

Using Application Continuity

- Supported database operations:
 - SQL, PL/SQL, and JDBC RPC: SELECT, ALTER SESSION, DML, DDL, COMMIT, ROLLBACK, SAVEPOINT, and JDBC RPCs
 - Transaction models: Local, Parallel, Remote, Distributed, and Embedded PL/SQL
 - Mutable functions
 - Transaction Guard
- Works in conjunction with:
 - Oracle RAC and RAC One
 - Oracle Active Data Guard
- Hardware acceleration on current Intel and SPARC chips
- Supported clients:
 - JDBC Thin driver, OCI drivers, Universal Connection Pool, WebLogic Server, ODP.NET, OCI Session Pool, and Tuxedo



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The slide lists key points relating to the use of Application Continuity. The following notes elaborate further:

- Application Continuity recovers the database request, including any in-flight transaction and the database session states. The requests may include most SQL and PL/SQL, RPCs, and local JDBC calls. Note that for remote and distributed transactions, all databases involved must be release 12.1 or later.
- Application Continuity offers the ability to keep the original values for some Oracle functions, such as SEQUENCE.NEXTVAL, that change their values each time that they are called. This improves the likelihood that the replay will succeed.
- Application Continuity uses Transaction Guard. Transaction Guard tags each database session with a logical transaction ID (LTXID), so that the database recognizes whether a request committed the transaction before the outage.
- Application Continuity works in conjunction with Oracle Real Application Clusters (Oracle RAC), RAC One, and Oracle Active Data Guard.
- On the database server, the validation performed by Application Continuity is accelerated using processor extensions built into current SPARC and Intel chips.
- Application Continuity provides client support for thin JDBC, Universal Connection Pool, and WebLogic Server, OCI drivers, ODP.NET, OCI Session Pool, and Tuxedo

Configuration Guidelines for the supported clients:

- Use Oracle Database 12c Release 1 (12.1.0.1), or later, for Java. Use Oracle Database 12c Release 2 (12.2), or later, for OCI-based applications..
- For .NET applications, use ODP.NET, Unmanaged Driver 12.2, or later, connecting to an Oracle Database 12c Release 2 (12.2) or later. By default, Application Continuity is enabled for ODP.NET applications in this configuration.

- For Java-based applications, use Universal Connection Pool 12.1 (or later) or WebLogic Server 12.1.2 (or later) configured with the JDBC Replay data source; or for third party applications, including third party JDBC pools, use JDBC replay driver. For IBM WebSphere, Apache Tomcat, RedHat Spring, and custom Java solutions, the most effective solution is to use UCP as the pooled data source.

For more information about application continuity, refer to *Oracle Real Application Clusters Administration and Deployment Guide*.

Creating Services for Application Continuity

- To create a service for Application Continuity for a policy-managed RAC database:

```
$ srvctl add service -db racedb -service app2
  -serverpool Srvpool1
  -failovertype TRANSACTION
  -commit_outcome TRUE
  -replay_init_time 1800 -failoverretry 30 -failoverdelay 10
  -retention 86400
  -notification TRUE -rlbgoal SERVICE_TIME -clbgoal SHORT
```

Mandatory Settings
for Application Continuity

Optional Settings for Application Continuity

- To modify an existing service for Application Continuity:

```
$ srvctl modify service -db racedb -service app1 -clbgoal SHORT
  -rlbgoal SERVICE_TIME -failoverretry 30 -failoverdelay 10
  -failovertype TRANSACTION -commit_outcome TRUE
  -replay_init_time 1800 -retention 86400 -notification TRUE
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To use Application Continuity, set the following mandatory service attributes:

- FAILOVERTYPE=TRANSACTION to enable Application Continuity
- COMMIT_OUTCOME=TRUE to enable Transaction Guard

Additionally, you can set values for these other service parameters for Application Continuity and load balancing:

- REPLAY_INIT_TIME:** This setting specifies the number of seconds within which replay must start. If replay does not start within the specified time then it is abandoned. Oracle recommends that you choose a value based on how long you will allow replay to be initiated. The default value is 300 seconds.
- RETENTION:** This setting specifies the number of seconds that the commit outcome is stored in the database. The default is 86400 (24 hours), and the maximum is 2592000 (30 days).

After a COMMIT has executed, if the session state was changed in that transaction, then it is not possible to replay the transaction to reestablish that state if the session is lost. When configuring Application Continuity, the applications are categorized depending on whether the session state after the initial setup is dynamic or static, and then whether it is correct to continue past a COMMIT operation within a request.

- DYNAMIC:** (default) A session has a dynamic state if the session state changes are not fully encapsulated by the initialization, and cannot be fully captured in a callback at failover. Once the first transaction in a request commits, failover is internally disabled until the next request begins. This is the default mode that almost all applications should use for requests.
- STATIC:** (special—on request) A session has a static state if all session state changes, such as NLS settings and PL/SQL package state, can be repeated in an initialization callback. This setting is used only for database diagnostic applications that do not change session state. Do not specify STATIC if there are any non-transactional state changes in the request that cannot be reestablished by a callback. If you are unsure what state to specify, use DYNAMIC.

- **-FAILOVERRETRY:** This setting specifies the number of connection retries for each replay attempt. If replay does not start within the specified number of retries then it is abandoned. Oracle recommends a value of 30.
- **-FAILOVERDELAY:** This setting specifies the delay in seconds between connection retries. Oracle recommends a value of 10.
- **-NOTIFICATION:** FAN is highly recommended—set this value to `TRUE` to enable FAN for OCI and ODP.Net clients.
- **-CLBGOAL:** For connection load balancing, use `SHORT` when using run-time load balancing.
- **-RLBGOAL:** For run-time load balancing, set to `SERVICE_TIME`.

Quiz



Which of the following are benefits of implementing Fast Application Notification?

- a. No need for connections to rely on connection timeouts
- b. Used by Load Balancing Advisory to efficiently propagate load information
- c. Database connection failures are immediately detected and stopped.
- d. Designed for enterprise application and management console integration



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.



Quiz

Application Continuity attempts to mask recoverable database outages from applications and users by restoring database sessions and replaying database calls.

- a. True
- b. False



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Configure client-side connect-time load balancing
- Configure client-side connect-time failover
- Configure server-side connect-time load balancing
- Configure Transparent Application Failover (TAF)
- Describe the purpose of Transaction Guard and Application Continuity
- Describe the key concepts relating to Application Continuity



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Practice 10 Overview: Using Application Continuity

This practice covers using Application Continuity against a RAC database to demonstrate how Application Continuity helps an application to seamlessly recover after the failure of a RAC instance.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle RAC One Node



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Perform an online database relocation
- Add an Oracle RAC One Node Database to an existing Cluster
- Convert an Oracle RAC One Node database to a RAC database
- Use DBCA to convert a single instance database to a RAC One Node database



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle RAC One Node

- Oracle RAC One Node is a single instance of a RAC-enabled database running on one node in the cluster only.
- This adds to the flexibility for database consolidation while reducing management overhead by providing a standard deployment for Oracle databases in the enterprise.
- Oracle RAC One Node requires Grid Infrastructure and, requires the same hardware setup as a RAC database.
- If applications require more resources than a single node can supply, you can upgrade online to Oracle RAC.
- If the node running Oracle RAC One Node becomes overloaded, you can relocate the instance to another node.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle RAC One Node is a single instance of an Oracle RAC-enabled database running on one node in the cluster, only, under normal operations. This option adds to the flexibility that Oracle offers for database consolidation while reducing management overhead by providing a standard deployment for Oracle databases in the enterprise. Oracle RAC One Node database requires Oracle Grid Infrastructure and, therefore, requires the same hardware setup as an Oracle RAC database.

Oracle supports Oracle RAC One Node on all platforms on which Oracle RAC is certified. Similar to Oracle RAC, Oracle RAC One Node is certified on Oracle Virtual Machine (Oracle VM). Using Oracle RAC or Oracle RAC One Node with Oracle VM increases the benefits of Oracle VM with the high availability and scalability of Oracle RAC.

With Oracle RAC One Node, there is no limit to server scalability and, if applications grow to require more resources than a single node can supply, then you can upgrade your applications online to Oracle RAC. If the node that is running Oracle RAC One Node becomes overloaded, then you can relocate the instance to another node in the cluster. With Oracle RAC One Node you can use the Online Database Relocation feature to relocate the database instance with no downtime for application users. Alternatively, you can limit the CPU consumption of individual database instances per server within the cluster using Resource Manager Instance Caging and dynamically change this limit, if necessary, depending on the demand scenario.

Creating an Oracle RAC One Node Database

- You can create Oracle RAC One Node databases by using DBCA.
- Single instance or RAC databases can be converted to Oracle RAC One Node.
- For Oracle RAC One Node databases, you must configure at least one dynamic database service.
- With an administrator-managed RAC One Node database, service registration is performed as with any other RAC database.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can create Oracle RAC One Node databases by using the Database Configuration Assistant (DBCA), as with any other Oracle database (manually created scripts are also a valid alternative). Oracle RAC One Node databases may also be the result of a conversion from either a single-instance Oracle database (using rconfig, for example) or an Oracle RAC database. Typically, Oracle-provided tools register the Oracle RAC One Node database with Oracle Clusterware. Depending on your configuration, automatic registration of an Oracle RAC One Node database with Oracle Clusterware may not have happened. If this is the case, then follow the steps in this section to register the Oracle RAC One Node database with Oracle Clusterware.

If your Oracle RAC One Node database did not register automatically with Clusterware, then use the `srvctl add database` command to add an Oracle RAC One Node database to your cluster. For example:

```
srvctl add database -dbtype RACONENODE [-server server_list] [-instance instance_name] [-timeout timeout]
```

Use the `-server` option and the `-instance` option when adding an administrator-managed Oracle RAC One Node database.

Note: Oracle recommends that you manage Oracle RAC One Node databases with SRVCTL. You can perform only certain operations (such as Online Database Relocation) using SRVCTL.

For Oracle RAC One Node databases, you must configure at least one dynamic database service (in addition to and opposed to the default database service). When using an administrator-managed Oracle RAC One Node database, service registration is performed as with any other Oracle RAC database. When you add services to a policy-managed Oracle RAC One Node database, SRVCTL does not accept any placement information, but instead configures those services using the value of the `SERVER_POOLS` attribute.

Verifying an Existing RAC One Node Database

```
[oracle@host01 ~]$ 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/spfileorcl.ora
Password file: +DATA/orcl/orapworcl
Domain: cluster01.example.com
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools: orcldb
Database instances:
Disk Groups: DATA
Mount point paths:
Services: SVC1
Type: RACOneNode
Online relocation timeout: 30
Instance name prefix: orcl
Candidate servers: host01,host02
...
...
```

*Information specific to
RAC One Node*



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Executing the `srvctl config database` command displays Oracle RAC One Node database configuration data. The data in the output specific to Oracle RAC One Node includes Type, Online relocation timeout, and candidate servers. As you can see in the example in the slide, the RAC One Node database `orcl` can run on `host01` or `host02` and the online relocation timeout value is 30 minutes.

Executing the `srvctl config database` command without the `-db` option returns a list of all databases that are registered with Oracle Clusterware.

Oracle RAC One Node Online Relocation

- Oracle RAC One Node allows the online relocation of the database from one server to another.
- The relocation period can be customized up to 12 hours.
- Use the `srvctl relocate database` command to initiate relocation of an Oracle RAC One Node database:

```
srvctl relocate database -db db_unique_name {[-node target]
[-timeout timeout_value] | -abort [-revert]} [-verbose]

-db <db_unique_name> Unique name of database to relocate
-node <target> Target node to which to relocate database
-timeout <timeout> Online relocation timeout in minutes
-abort Abort failed online relocation
-revert Remove target node of failed online relocation request
from the candidate server list of administrator-managed RAC One
Node database
-verbose Verbose output
-help Print usage
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle RAC One Node allows the online relocation of an Oracle RAC One Node database from one server to another, which provides increased availability for applications based on an Oracle Database. The relocation period can be customized up to 12 hours. You can now move a database for workload balancing as well as for performing planned maintenance on the server, on the operating system, or when applying patches to the Oracle software in a rolling fashion.

Only during a planned online database relocation is a second instance of an Oracle RAC One Node database created, so that any database sessions can continue while the database is relocated to a new node.

If your Oracle RAC One Node database is administrator managed, then the target node to which you want to relocate the database must be in the candidate list or in the Free server pool. If the target node is in the Free server pool, then the node is added to the candidate list.

When you relocate a database instance to a target node that is not currently in the candidate server list for the database, you must copy the password file, if configured, to the target node.

Oracle Corporation recommends using OS authentication, instead, or using Oracle Clusterware to start and stop the database, and defining users in the data dictionary for other management.

Online Relocation Considerations

- Use either Application Continuity and Oracle Fast Application Notification or Transparent Application Failover to minimize the impact of a relocation on the client.
- If FAN or TAF is not used, any in-flight transactions will be allowed to complete within the timeout value constraint.
 - If the timeout is exceeded, clients will receive an ORA-3113 error when the session is terminated due to the shutdown of the instance.
- If the shutdown of the original instance takes longer than the timeout value, the instance is aborted.
 - The new instance will then perform recovery to clean up any transactions that were aborted due to the shutdown.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Using the Single Client Access Name (SCAN) to connect to the database, clients can locate the service independently of the node on which it is running. Relocating an Oracle RAC One Node instance is therefore mostly transparent to the client, depending on the client connection. Oracle recommends to use either Application Continuity and Oracle Fast Application Notification or Transparent Application Failover to minimize the impact of a relocation on the client..

If you do not use Application Continuity, FAN or TAF, any in-flight transactions will be allowed to complete as long as they complete within the timeout period entered, after which the clients will receive an ORA-3113 error when their session is terminated due to the shutdown of the Oracle RAC One Node instance (ORA-3113 End of Line on communication channel). Because the new instance will be running, the client can immediately log in again.

If the shutdown of the original instance takes longer than the set timeout, this database instance is aborted. The new instance will then perform recovery to clean up any transactions that were aborted due to the shutdown.

Performing an Online Relocation

```
[oracle@host01 ~]$ srvctl relocate database -db orcl \
-node host02 -timeout 15 -verbose

Configuration updated to two instances
Instance orcl_2 started
Services relocated
Waiting for 15 minutes for instance orcl_1 to stop.....
Instance orcl_1 stopped
Configuration updated to one instance
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Execute the `srvctl relocate database` command specifying the database to be migrated, the host to which it will be migrated to, and optionally a timeout value used to allow connections with active transactions to finish. If no value is specified, the default is 30 minutes. If you retrieve the status of the database before the relocation starts, you should see something like this:

```
[oracle@host01 ~]$ srvctl status database -db orcl
Instance orcl_2 is running on node host01
Online relocation: INACTIVE
```

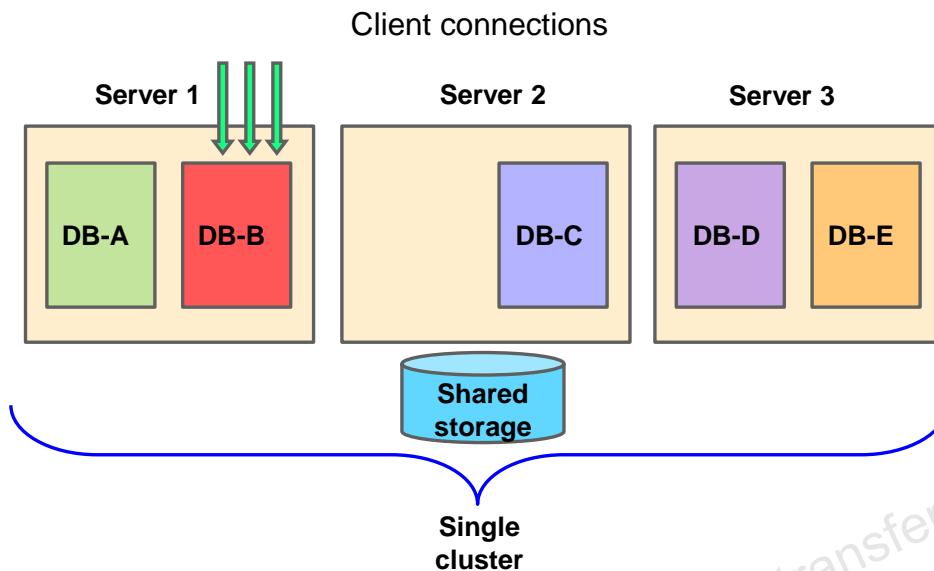
During the relocation:

```
[oracle@host01 ~]$ srvctl status database -db orcl
Instance orcl_1 is running on node host01
Online relocation: ACTIVE
Source instance: orcl_1 on host01
Destination instance: orcl_2 on host02
```

After the relocation is complete:

```
[oracle@host01 ~]$ srvctl status database -db orcl
Instance orcl_2 is running on node host02
Online relocation: INACTIVE
```

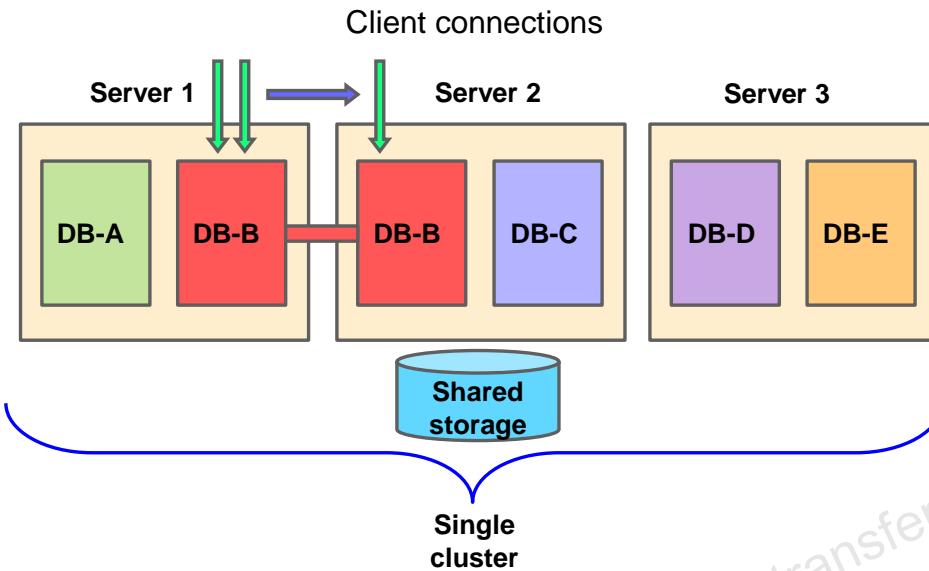
Online Relocation Illustration



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

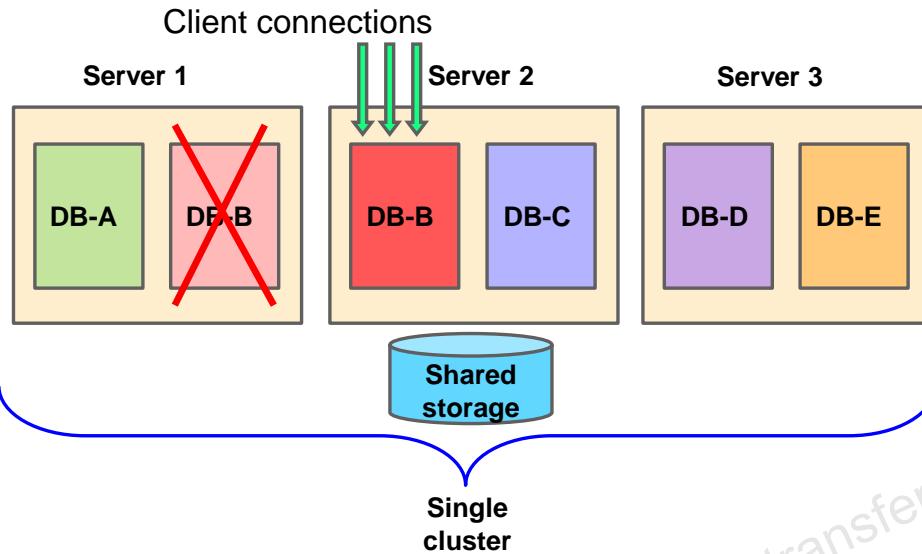
Online Relocation Illustration



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

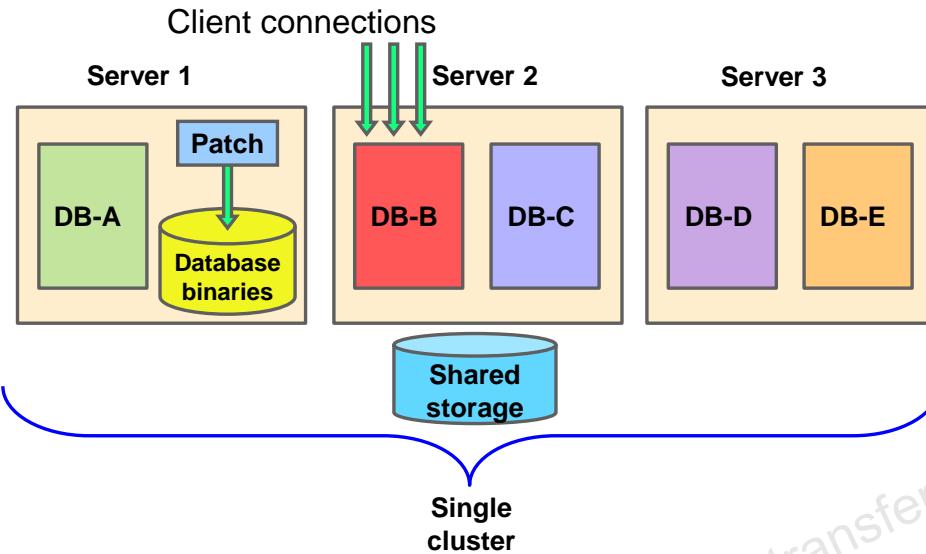
Online Relocation Illustration



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Online Maintenance: Rolling Patches



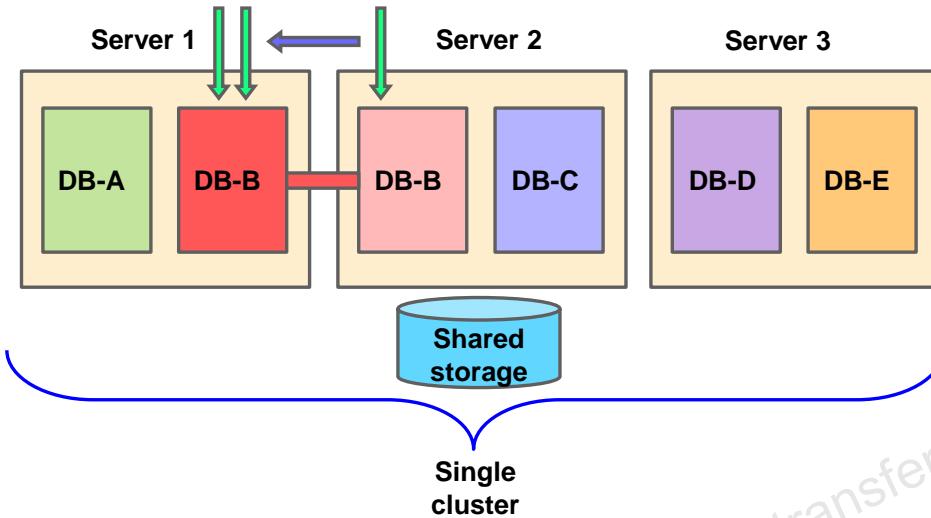
ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

RAC One Node administrators can online migrate the databases off the server to a spare server, and then perform all types of maintenance on the idle server, including hardware maintenance, OS upgrades, OS patches, and database patches. With online database relocation, a new database instance is created on a new server (running in a different operating system), and work is online migrated to the new instance. Thus, the old operating system and database home remain on the former host server, and can be upgraded (OS) or patched (DB).

You continue with your deployment example from the previous slide. The illustration in the slide depicts a RAC One Node deployment after using online database relocation to move database B from server 1 to server 2. After the relocation, the database binaries that had hosted the instance formerly running on server 1 remain available for patching.

Online Maintenance: Rolling Patches



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Once the database binaries on server 1 have been patched, database B can be migrated via online database relocation back to server 1. Because online database relocation supports relocation between instances at different patch levels, the operation is completely online and requires no disruption to end users. In this case, the online database relocation migrates the connections back to server 1, then the instance on server 2 is shut down, completing the relocation. Similarly, online database relocation can be used to move all databases off a node in preparation for an online operating system upgrade.

Adding an Oracle RAC One Node Database to an Existing Cluster

- Use the `srvctl add database` command to add an Oracle RAC One Node database to an existing cluster.

```
srvctl add database -dbtype RACONENODE [-server server_list]
[-instance instance_name] [-timeout timeout_value]
```

- When adding an administrator-managed Oracle RAC One Node database, you can optionally supply an instance prefix with the `-instance instance_name` option of the `srvctl add database` command.
- Each service is configured by using the same value for the `SERVER_POOLS` attribute as the underlying database.
 - When you add services to an Oracle RAC One Node database, `srvctl` configures those services using the value of the `SERVER_POOLS` attribute.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Converting Oracle RAC One Node to Oracle RAC

Use the `srvctl add database` command to add an Oracle RAC One Node database to an existing cluster. For example:

```
srvctl add database -dbtype RACONENODE [-server server_list] [-instance
instance_name] [-timeout timeout_value]
```

Use the `-server` option and the `-instance` option when adding an administrator-managed Oracle RAC One Node database.

Each service on an Oracle RAC One Node database is configured by using the same value for the `SERVER_POOLS` attribute as the underlying database. When you add services to an Oracle RAC One Node database, `srvctl` does not accept any placement information, but instead configures those services using the value of the `SERVER_POOLS` attribute.

When adding an administrator-managed Oracle RAC One Node database, you can optionally supply an instance prefix with the `-instance instance_name` option of the `srvctl add database` command. The name of the instance will then be `instance_name_1`. If you do not specify an instance prefix, then the first 12 characters of the unique name of the database becomes the prefix. The instance name changes to `instance_name_2` during an online database relocation and reverts to `instance_name_1` during a subsequent online database relocation. The same instance name is used on failover.

Converting a RAC One Node Database to RAC

To convert a RAC One Node database to RAC:

1. Execute the `srvctl convert database` command.

```
srvctl convert database -db <db_unique_name> -dbtype RAC  
[-node <node_1>]
```

2. Create server pools for each service that the database has, in addition to the database server pool.
3. Add the instances on the remaining nodes with the `srvctl add instance` command.

```
srvctl add instance -db <db_unique_name> -instance  
instance_name -node <node_2>  
srvctl add instance -db <db_unique_name> -instance  
instance_name -node <node_n>
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can convert an Oracle RAC One Node database to an Oracle RAC database by logging in as the Oracle RAC One Node database owner and executing the `srvctl convert database` command.

After you run the command, you must create server pools for each service that the database has, in addition to the database server pool. The values for `SERVER_NAMES` of the service server pools must be set to the node that you converted from an Oracle RAC One Node to an Oracle RAC node.

Converting an administrator-managed Oracle RAC One Node database to an Oracle RAC database sets all database services so that the single instance is preferred. After you convert the database, you can add instances by running the `srvctl add instance` command.

Converting a policy-managed Oracle RAC One Node database to an Oracle RAC database sets all database services to UNIFORM cardinality. It also results in reusing the server pool in which the database currently runs. The conversion reconfigures the database to run on all of the nodes in the server pool. The command does not start any additional instances but running the `srvctl start database` command starts the database on all of the nodes in the server pool.

Converting a RAC One Node Database to RAC

```
$ srvctl convert database -db orcl -dbtype RAC -node host01
$ srvctl add instance -db orcl -instance orcl_2 -node host02
$ srvctl start instance -db orcl -instance orcl_2
$ srvctl config database -d 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/spfileorcl.ora
Domain:
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools: orcl
Database instances: orcl_1,orcl_2
Disk Groups: DATA,FRA
Services: SERV1
Type: RAC
Database is administrator managed
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, the RAC One Node database ORCL is converted to an Oracle RAC database using the `srvctl convert database` command. The cluster consists of two nodes, `host01` and `host02`. After the `srvctl convert database` command has finished executing, the second instance, `orcl_2` is added to `host02` using the `srvctl add instance` command as illustrated in the example in the slide.

Once the instance has been added to the second node, it can be started using the `srvctl start instance` command. Use the `srvctl config database` command to verify that the database conversion and instance addition was successful.

Converting a Single Instance Database to RAC One Node

- Use DBCA to convert from single-instance Oracle databases to Oracle RAC One Node.
- Before you use DBCA to convert a single-instance database to an Oracle RAC One Node database, ensure that your system meets the following conditions:
 - It is a supported hardware and operating system software configuration.
 - The nodes have access to shared storage.
 - Your applications have no design characteristics that preclude their use in a clustered environment.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can use DBCA to convert from single-instance Oracle databases to Oracle RAC One Node. DBCA automates the configuration of the control file attributes, creates the undo tablespaces and the redo logs, and makes the initialization parameter file entries for cluster-enabled environments. DBCA also configures Oracle Net Services, Oracle Clusterware resources, and the configuration for Oracle database management for use by Oracle Enterprise Manager or the SRVCTL utility.

Before you use DBCA to convert a single-instance database to an Oracle RAC One Node database, ensure that your system meets the following conditions:

- It is a supported hardware and operating system software configuration.
- The nodes have access to shared storage; for example, either Oracle Cluster File System or Oracle ASM is available and accessible from all nodes. On Linux on POWER systems, ensure that GPFS is available and accessible from all nodes..
- Your applications have no design characteristics that preclude their use in a clustered environment.

Oracle strongly recommends that you use the Oracle Universal Installer to perform an Oracle Database 12c installation that sets up the Oracle home and inventory in an identical location on each of the selected nodes in your cluster.

Converting a RAC Database to RAC One Node

- When converting a RAC database to RAC One Node, first ensure that the RAC database has only one instance.
- If the RAC database is admin-managed, change the configuration of all services to set the preferred instance to the one you want to keep as an RAC One Node database.
 - If a service had a PRECONNECT TAF policy, then the policy must be updated to BASIC or NONE before conversion.
- If the RAC database is policy managed, then change the configuration of all services so they use the same server pool before you convert the RAC database.
- Use the `srvctl convert database` command to convert a RAC database to RAC One Node:

```
srvctl convert database -db db_unique_name -dbtype  
RACONENODE [-instance instance_name -timeout timeout]
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can convert an Oracle RAC database with one instance to an Oracle RAC One Node database using the `srvctl convert database` command, as follows:

```
srvctl convert database -db db_unique_name -dbtype RACONENODE [-instance  
instance_name -timeout timeout]
```

Prior to converting an Oracle RAC database to an Oracle RAC One Node database, you must first ensure that the Oracle RAC database has only one instance.

If the Oracle RAC database is administrator managed, then you must change the configuration of all services to set the preferred instance to the instance that you want to keep as an Oracle RAC One Node database after conversion. If any service had a PRECONNECT TAF policy, then its TAF policy must be updated to BASIC or NONE before starting the conversion process. These services must no longer have any available instance.

If the Oracle RAC database is policy managed, then you must change the configuration of all services so that they all use the same server pool before you convert the Oracle RAC database to an Oracle RAC One Node database.



Quiz

The `srvctl add database` command is used to add an Oracle RAC One Node database to an existing cluster.

- a. True
- b. False



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Quiz



Which of the following conditions must be met before a single instance database can be converted to a RAC One Node Database? (Choose three)

- a. Your environment is a supported hardware and operating system software configuration.
- b. It has shared storage: either Oracle Cluster File System or Oracle ASM is available and accessible from all nodes.
- c. You must disable Oracle Clusterware on all nodes.
- d. Your applications have no design characteristics that preclude their use in a clustered environment.



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Perform an online database relocation
- Add an Oracle RAC One Node Database to an existing cluster
- Convert an Oracle RAC One Node database to a RAC database
- Use DBCA to convert a single instance database to a RAC One Node database.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Practice 11: Overview

The practices for this lesson cover the following topics:

- RAC One Node Database creation by converting the existing RAC database
- RAC One Node online relocation
- Converting a RAC One Node database to a RAC Database



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle Database In-Memory in RAC



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

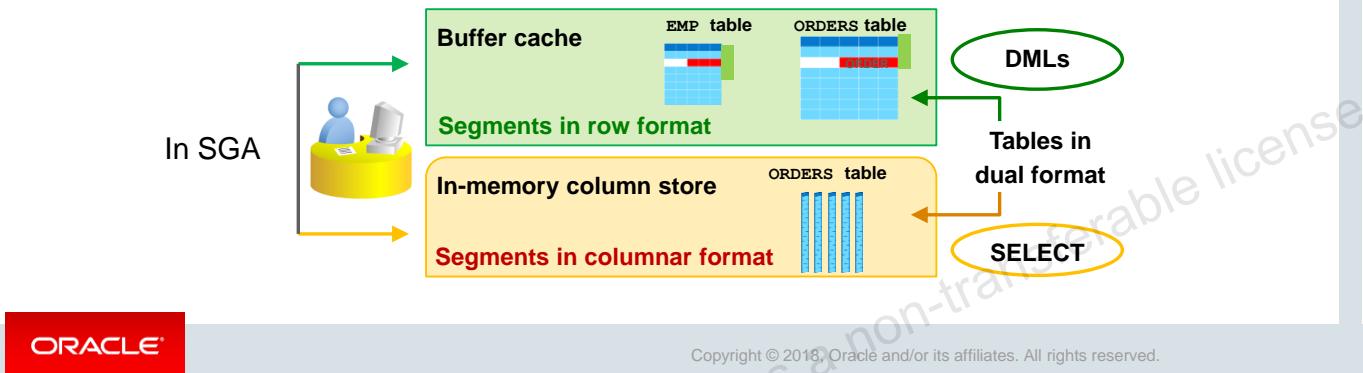
- Explain the goals, benefits, and architecture of the In-Memory Column Store
- Implementing the In-Memory Column Store in a database
- Explain and Implement In-Memory FastStart



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In-Memory Column Store

- In-Memory (IM) column store is a new pool in SGA introduced in Oracle Database Release 12.1, Patchset 1.
 - Segments populated into the IM column store are converted into a columnar format
 - In-Memory segments are transactionally consistent with the buffer cache
- Only one segment on disk and in row format



ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The In-Memory Column Store enables objects (tables, partitions, and other types) to be stored in memory in a new format known as the columnar format. This format enables scans, joins, and aggregates to perform much faster than the traditional on-disk format, providing fast reporting and DML performance for both OLTP and DW environments.

The in-memory columnar format does not replace the on-disk or buffer cache format. This means that when a segment such as a table or a partition is populated into the IM column store, the on-disk format segment is automatically converted into a columnar format and optionally compressed. The columnar format is a pure in-memory format. There is no columnar format storage on disk. It never causes additional writes to disk and therefore does not require any logging or undo space.

All data is stored on disk in the traditional row format. Moreover, the columnar format of a segment is a transaction-consistent copy of the segment either on disk or in the buffer cache. Transaction consistency between the two pools is maintained.

If sufficient space is allocated to the IM column store in SGA, a query accessing objects that are populated into the IM column store performs much faster. The improved performance allows more ad-hoc analytic queries to be executed directly on the real-time transaction data without impacting the existing workload. A lack of IM column store space does not prevent statements to execute against tables that could have been populated into IM column store.

The DBA must decide, according to the types of queries and DMLs executed against the segments, which segments should be defined as non in-memory segments and those as in-memory segments . The DBA can also define more precisely which columns are good candidates for IM column store:

- **In row format exclusively:** The segments being frequently accessed by OLTP style queries, operating on few rows returning many columns are good candidates for the buffer cache. These segments should not necessarily be defined as in-memory segments and will be sent to the buffer cache only.

- **In dual format simultaneously:** The segments being frequently accessed by analytical style queries, operating on many rows returning few columns are good candidates for IM column store. If a segment is defined as an in-memory segment but has some columns defined as non in-memory columns, the queries that select any non in-memory columns are sent to the buffer cache and those selecting in-memory columns only are sent to the IM column store. Any fetch-by-rowid is performed on the segment through the buffer cache.

Any DML performed on these objects is executed via the buffer cache.

Advantages of In-Memory Column Store

- Queries run a lot faster.
 - Data is populated in memory in a compressed columnar format.
 - No index is required and used.
- DMLs are faster.
 - Analytics indexes can be eliminated..
 - Replaced by scans of the IM column store representation of the table.
- Ad-hoc queries run with good performance.
 - The table behaves as if all columns are indexed.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

There are three distinct advantage of In-Memory Column Store:

- Queries run a lot faster: All data can be populated in memory in a compressed columnar format. No index is required and used. Queries run at least 100 times faster than when fetching data from buffer cache thanks to the columnar compressed format.
- DMLs are faster: Analytics indexes can be eliminated being replaced by scans of the IM column store representation of the table.
- Arbitrary ad-hoc queries run with good performance, since the table behaves as if all columns are indexed.

In-Memory Column Store Pools

- The In-Memory area is sub-divided into two pools:
 - A 1MB pool used to store the actual column formatted data populated into memory
 - A 64K pool used to store metadata about the objects that are populated into the IM column store

```
select pool, alloc_bytes, used_bytes, population_status from V$INMEMORY_AREA;
```

POOL	ALLOC_BYTES	USED_BYTES	POLULATE STATUS
1MB POOL	1710227456	16777216	DONE
64 KB POOL	419430400	1900544	DONE

- The size of the In-Memory area within the SGA, is controlled by the initialization parameter `INMEMORY_SIZE`.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The In-Memory area is sub-divided into two pools: a 1MB pool used to store the actual column formatted data populated into memory, and a 64K pool used to store metadata about the objects that are populated into the IM column store. The amount of available memory in each pool is visible in the `V$INMEMORY_AREA` view. The relative size of the two pools is determined by internal heuristics; the majority of the In-Memory area memory is allocated to the 1MB pool.

The size of the In-Memory area, within the SGA, is controlled by the initialization parameter `INMEMORY_SIZE` (default 0). The In-Memory area must have a minimum size of 100MB. The current size of the In-Memory area is visible in `V$SGA`. Starting in 12.2, it is possible to increase the size of the In-Memory area on the fly, by increasing the `INMEMORY_SIZE` parameter via an `ALTER SYSTEM` command, assuming there is spare memory within the SGA. The `INMEMORY_SIZE` parameter must be increased by 128MB or more in order for this change to take effect. It is not possible to shrink the size of the In-Memory area on the fly. A reduction in the size of the `INMEMORY_SIZE` parameter will not take effect until the database instance is restarted. It is important to note that the In-Memory area is not impacted or controlled by Oracle Automatic Memory Management (AMM).

Implementing In-Memory Column Store

1. Log in to the database with administrative privileges.

2. Set `INMEMORY_SIZE` to a non-zero value:

```
SQL> ALTER SYSTEM SET INMEMORY_SIZE = 1G SCOPE=SPFILE;
```

3. Shut down the database, and then reopen it.

4. Check the of memory allocated for the IM column store:

```
SQL> SHOW PARAMETER INMEMORY_SIZE
```

NAME	TYPE	VALUE
inmemory_size	big integer	1G

5. To dynamically increase the `INMEMORY_SIZE` parameter:

```
SQL> ALTER SYSTEM SET INMEMORY_SIZE = 2G SCOPE=BOTH;
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Before tables or materialized views can be populated into the IM column store, you must enable the IM column store for the database. Before this can be accomplished, you must ensure that the database is open and The `COMPATIBLE` initialization parameter is set to 12.1.0 or higher. To enable the In-Memory column store:

- In SQL*Plus or SQL Developer, log in to the database with administrative privileges.
- Set the `INMEMORY_SIZE` initialization parameter to a non-zero value. The minimum setting is 100M. When you set this initialization parameter in a server parameter file (SPFILE) using the `ALTER SYSTEM` statement, you must specify `SCOPE=SPFILE`.
- For example, the following statement sets the In-Memory Area size to 10 GB:
`ALTER SYSTEM SET INMEMORY_SIZE = 10g SCOPE=SPFILE;`
 Shut down the database, and then reopen it to initialize the IM column store in the SGA.
- Check the amount of memory currently allocated for the IM column store:
`SHOW PARAMETER INMEMORY_SIZE.`

If the compatibility level is 12.2 or greater, you can dynamically increase the IM column store using the SQL `ALTER SYSTEM SET INMEMORY_SIZE` command. Note that the size of the IM column store cannot be decreased dynamically. If you wish to do this, you must issue an `ALTER SYSTEM SET INMEMORY_SIZE...SCOPE=SPFILE` command then restart the database.

In-Memory Column Store Population

- Only objects with the `INMEMORY` attribute are eligible for population.
- The `INMEMORY` attribute can be specified on a tablespace, table, partition, or materialized view.

```
SQL> ALTER TABLE sales INMEMORY
```

- All columns in an object with the `INMEMORY` attribute will be populated into the IM column store.
- To exclude specific columns:

```
SQL> ALTER TABLE sales INMEMORY NO INMEMORY (prod_id);
```

- If enabled at the tablespace level, then new materialized views and tables in the tablespace will be enabled by default.

```
SQL> ALTER TABLESPACE ts_data DEFAULT INMEMORY;
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Not all of the objects in an Oracle database need to be populated in the IM column store. The IM column store should be populated with the most performance-critical data in the database. Less performance-critical data can reside on lower cost flash or disk. Of course, if your database is small enough, you can populate all of your tables into the IM column store. In-Memory adds a new `INMEMORY` attribute for tables and materialized views. Only objects with the `INMEMORY` attribute are populated into the IM column store. The `INMEMORY` attribute can be specified on a tablespace, table, (sub)partition, or materialized view. If it is enabled at the tablespace level, then all new tables and materialized views in the tablespace will be enabled for the IM column store by default.

By default, all of the columns in an object with the `INMEMORY` attribute will be populated into the IM column store. However, it is possible to populate only a subset of columns if desired using the `NO INMEMORY` clause. This clause can also be used to indicate an object is no longer a candidate and remove it from the IM store:

```
ALTER TABLE sales MODIFY PARTITION SALES_Q2_2008 NO INMEMORY;
```

Prioritization of In-Memory Population

- The order in which objects are populated is controlled by the keyword **PRIORITY**, which has five levels.

Priority	Description
CRITICAL	Object is populated immediately after database is opened
HIGH	Object is populated after all CRITICAL objects have been populated, if space remains available in the IM column store
MEDIUM	Object is populated after all CRITICAL and HIGH objects have been populated, and space remains available in the IM column store
LOW	Object is populated after all CRITICAL, HIGH, and MEDIUM objects have been populated, if space remains available in the IM column store
NONE	Objects only populated after they are scanned for the first time (Default), if space is available in the IM column store

```
SQL> ALTER TABLE customers INMEMORY PRIORITY CRITICAL;
```

ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

You can specify that the database populates objects in the IM column store either at database instance startup or when INMEMORY objects are accessed. The population algorithm also varies depending on whether you use single-instance or Oracle RAC.

DDL statements include an INMEMORY PRIORITY subclause that provides more control over the population queue.

Objects are populated into the IM column store either in a prioritized list immediately after the database is opened or after they are scanned (queried) for the first time. The order in which objects are populated is controlled by the keyword **PRIORITY**, which has five levels (see above). The default **PRIORITY** is **NONE**, which means an object is populated only after it is scanned for the first time. All objects at a given priority level must be fully populated before the population for any objects at a lower priority level can commence. However, the population order can be superseded if an object without a **PRIORITY** is scanned, triggering its population into IM column store.

In-Memory Column Store and Oracle RAC

- Each node in a RAC environment has its own IM column store.
- By default, populated objects are distributed across all IM column stores in the cluster.
- For any RAC node that does not require an IM column store, set the `INMEMORY_SIZE` parameter to 0.
- The distribution of objects across the IM column stores in a cluster is controlled by two clauses to the `INMEMORY` attribute:
 - `DISTRIBUTE`
 - `DUPLICATE` (only applicable on Oracle Engineered Systems)
- The `DISTRIBUTE` clause supports `BY ROWID`, `BY PARTITION`, `BY SUBPARTITION`, and `FOR SERVICE`.

```
SQL> ALTER TABLE lineorder INMEMORY DISTRIBUTE BY PARTITION;
SQL> ALTER TABLE sales INMEMORY DISTRIBUTE FOR SERVICE sales_ebiz;
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Each node in an Oracle RAC environment has its own In-Memory (IM) column store. Oracle recommends that you equally size the IM column stores on each Oracle RAC node. For any Oracle RAC node that does not require an IM column store, set the `INMEMORY_SIZE` parameter to 0.

It is possible to have completely different objects populated on every node, or to have larger objects distributed across all of the IM column stores in the cluster. It is also possible to have the same objects appear in the IM column store on every node (on engineered systems, only). The distribution of objects across the IM column stores in a cluster is controlled by two additional sub-clauses to the `INMEMORY` attribute; `DISTRIBUTE` and `DUPLICATE`.

The `DISTRIBUTE` clause can be used to specify how an object is distributed across the cluster. By default (`DISTRIBUTE AUTO`), the type of partitioning used (if any) determines how the object is distributed. If the object is not partitioned it is distributed by `rowid` range.

Alternatively, you can specify the `DISTRIBUTE BY` clause to over-ride the default behavior.

The `DUPLICATE` clause is used to control how an object should be duplicated across the IM column stores in the cluster. If you specify just `DUPLICATE`, then one mirrored copy of the data is distributed across the IM column stores in the cluster. If you want to duplicate the entire object in each IM column store in the cluster, then specify `DUPLICATE ALL`.

Note: When you deploy Oracle RAC on a non-Engineered System, the `DUPLICATE` clause is treated as `NO DUPLICATE`.

In-Memory FastStart

- The FastStart enhancement to In-Memory column store was introduced in Oracle Database 12.2.
- The In-Memory column store is populated whenever a database instance restarts.
 - This can be a slow operation that is I/O and CPU intensive.
- IM FastStart optimizes object population in the IM column store by storing IM compression units (IMCU) directly on disk.
 - This results in faster repopulation during instance restarts
- If the database re-opens after being closed, then the database reads columnar data from the FastStart area.
 - It is then populated it into the IM column store, ensuring that all transactional consistencies are maintained.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

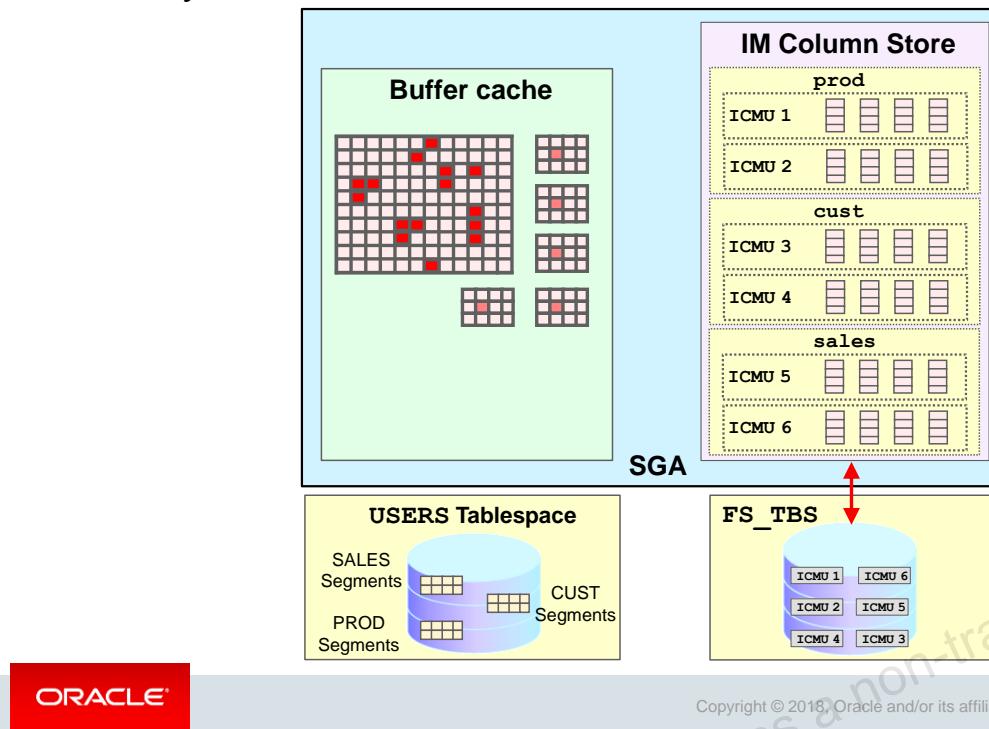
The IM column store is populated whenever a database instance restarts, which can be a slow operation that is I/O-intensive and CPU-intensive.

In-Memory FastStart optimizes the population of database objects in the In-Memory column store by storing In-Memory compression units directly on disk.

When IM FastStart is enabled, the database periodically saves a copy of columnar data to disk for faster repopulation during instance restarts. If the database re-opens after being closed, then the database reads columnar data from the FastStart area, and then populates it into the IM column store, ensuring that all transactional consistencies are maintained.

An IM FastStart tablespace requires intermittent I/O while the database is open and operational. The performance gain occurs when the database re-opens because the database avoids the CPU-intensive compression and formatting of data.

In-Memory FastStart Architecture



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

ORACLE®

During the first population after the FastStart area is enabled, the database creates the FastStart area. The database manages the FastStart area automatically as follows:

- Whenever population or repopulation of an object occurs, the database writes its columnar data to the FastStart area.

The Space Management Worker Processes (`Wnnn`) write IMCUs to the SecureFiles LOB named `SYSDBinstance_name_LOBSEG$`. The database writes FastStart metadata to the `SYSAUX` tablespace, which must be online.

Depending on how much DML activity occurs for a CU, a lag can exist between the CUs in the FastStart area and the CUs in the IM column store. The “hotter” a CU is, the less frequently the database populates it in the IM column store and writes it to the FastStart area. If the database crashes, then some CUs that were populated in the IM column store may not exist in the FastStart area.

- If the attribute of a populated object is changed to `NOINMEMORY`, then the database automatically removes its IMCUs from the FastStart area.
- If the FastStart tablespace runs out of space, then the database uses an internal algorithm to drop the oldest segments, and continues writing to the FastStart area. If no space remains, then the database stops writing to the FastStart area.

The figure above shows PROD, CUST, and SALES populated in the IM column store.

When the FastStart area is enabled, the database also writes the IMCUs for these segments to the FastStart area in the `fs_tbs`. If the database re-opens or if the instance restarts, then the database can validate the IMCUs for modifications to ensure the transactional consistency, and reuse the IMCUs. Regardless of whether the FastStart area is enabled, the database stores data blocks and segments on disk in the users tablespace.

Enabling In-Memory FastStart

- A FastStart area is a designated tablespace where IM FastStart stores and manages data for INMEMORY objects.
- Oracle Database manages the FastStart tablespace without DBA intervention.
- Use the DBMS_INMEMORY_ADMIN.FASTSTART_ENABLE procedure to enable a FastStart tablespace.

```
SQL> exec DBMS_INMEMORY_ADMIN.ENABLE_FASTSTART ('fs_tbs')
```

- Only one FastStart area, and one designated FastStart tablespace, is allowed for each PDB or non-CDB.
- You cannot alter or drop the tablespace while it is the designated IM FastStart tablespace.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

FastStart Area in Oracle RAC

- The FastStart area is shared across all Oracle RAC nodes.
- Only one copy of an IMCU resides in the FastStart area.
- Assuming a `DUPLICATE ALL` is specified for an object in a four-node cluster:
 - Four copies of the object would exist in the IM column stores
 - But the database saves only one copy to the FastStart area
- Any database instance in a RAC cluster can use an IMCU in the FastStart area.
- This improves performance of instance restarts in a RAC environment



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

How the Database Reads from the FastStart Area

- The FastStart area defines what data is loaded when the database reopens, but not when it is loaded.
- Data population is controlled by the priority settings.
- When the database reopens, the standard PRIORITY rules determine population:
 - The database populates objects with PRIORITY NONE on demand
 - Objects with priority CRITICAL are higher in the automatic population queue than objects with priority LOW.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Explain the goals, benefits, and architecture of the In-Memory Column Store
- Implementing the In-Memory Column Store in a database
- Explain and Implement In-Memory FastStart



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Practice 12: Overview

This practice covers the following topic:

- Reconfiguring the Practice Environment



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Multitenant Architecture and RAC



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to do the following:

- Describe the multitenant architecture in a non-RAC environment
- Describe the multitenant architecture in a RAC environment
- Create a RAC multitenant container database (CDB)
- Create a pluggable database (PDB) in a RAC CDB
- Use default CDB and PDB services
- Create PDB services to associate PDB services with server pools
- Drop a pluggable database (PDB) from a RAC CDB



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

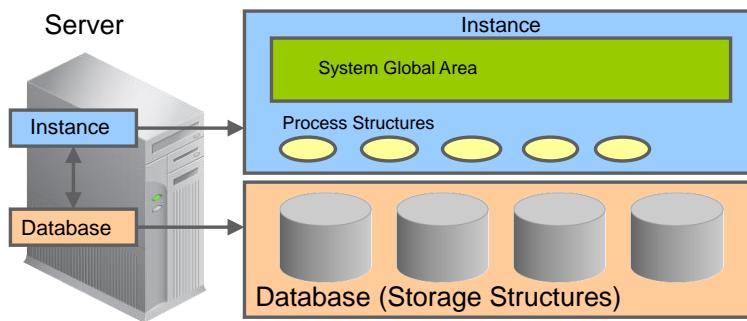
Note: For a complete understanding of Oracle Multitenant new option and usage, refer to the following guides in Oracle documentation:

- *Oracle Database Administrator's Guide 12c Release 1 (12.2)*
- *Oracle Database Concepts 12c Release 1 (12.2)*

Refer to other sources of information available:

- Under Oracle Learning Library: *Oracle Database 12c New Features Demo Series* demonstrations:
 - *Multitenant Architecture*
 - *Basics of CDB and PDB Architecture*
- Under Oracle Learning Library: *Oracle By Example (OBE)*:
 - *Performing Basic Tasks on Multitenant Container Databases and Pluggable Databases*
- Under My Oracle Support:
 - *Oracle Multitenant Option - 12c : Frequently Asked Questions (Doc ID 1511619.1)*
- The courses entitled *Oracle Database 12c: New Features for Administrators* and *Oracle Database 12c: Managing Multitenant Architecture*

Non-CDB Architecture



- Multiple non-CDBs share nothing:
 - Too many background processes
 - High shared/process memory
 - Many copies of Oracle metadata

ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In Oracle Database 12c, there are two possible configurations, **non-CDB** and **multitenant container database**.

In Oracle 11g database, the only kind of database that is supported is a non-CDB. The old architecture is referred to as the non-CDB architecture—the term *non-CDB* will be used as a shorthand for an occurrence of a pre-12.1 database that uses the pre-12.1 architecture—that requires its own instance and, therefore, its own background processes, and memory allocation for the SGA. It also needs to store the Oracle metadata in its data dictionary. The database administrator can still create Oracle 12c non-CDBs with the same pre-12.1 architecture. These databases are non-CDBs.

If there are multiple databases on the same server, then there is a separate and distinct instance for each database, non-CDB or CDB. An instance cannot be shared between a non-CDB and CDB.

When you have to administer small departmental database applications, you have to create as many databases as applications and, therefore, multiply the number of instances, consequently the number of background processes, memory allocation for the SGAs, and provision enough storage for all data dictionaries of these databases.

When you need to upgrade your applications to a new version, you have to upgrade each database, which is time consuming for the DBA.

Multitenant Architecture: Benefits

- Operates **multiple databases in a centrally managed platform** to lower costs:
 - Less instance overhead
 - Less storage cost
- Reduces DBA resources costs and maintains security
 - No application changes
 - **Fast and easy provisioning**
 - **Time saving for patching and upgrade**
 - **Maintain separation of duties** between:
 - Different application administrators
 - Application administrators and DBA
 - Users within application
- **Provides isolation**



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Consolidating many non-CDB databases onto a single platform reduces instance overhead, avoids redundant copies of data dictionaries, and consequently storage allocation, and benefits from fast provisioning, time saving upgrading, better security through separation of duties and application isolation. The new 12c multitenant architecture that consolidates databases together is a multitenant container database or CDB, and a database consolidated within a CDB, a pluggable database or PDB.

DBA resource costs are reduced with:

- *No application change and very fast provisioning:* A new database can be provisioned very quickly. A clone of a populated database can be created very quickly. A populated database can be quickly unplugged from its CDB on one platform and quickly plugged into a CDB on a different platform. A non-CDB can quickly be plugged into a CDB.
- *Fast upgrade and patching of the Oracle Database version:* The cost (time taken and human effort needed) to upgrade many PDBs is the cost of upgrading a single Oracle Database occurrence. You can also upgrade a single PDB by unplugging it and plugging it into a CDB at a different Oracle Database version.

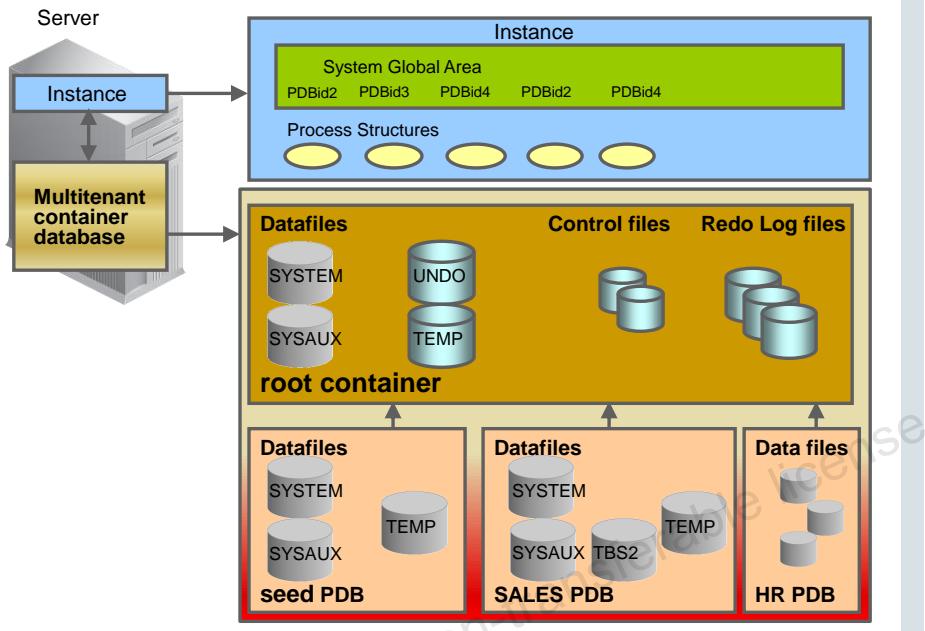
The multitenant architecture maintains:

- *Secure separation of duties:* The administrator of an application can do all the required tasks by connecting to the particular PDB that implements its back end. However, someone who connects to a PDB cannot see other PDBs. To manage PDBs as entities (for example, to create or drop or unplug or plug one), the system administrator needs to connect to the CDB. For these specific tasks, new privileges need to be granted.
- *Isolation of applications* that may not be achieved manually unless using Database Vault for example. A good example of isolation is dictionary separation enabling Oracle Database to manage the multiple PDBs separately from each other and from the CDB itself.

CDB in a Non-RAC Environment

Single DB shares:

- Background processes
- Shared/process memory
- Oracle metadata
- Redo log files
- Control files
- Undo tablespace



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Containers

Two types of containers in V\$CONTAINERS:

- The root container
 - The first **mandatory** container created at CDB creation
 - Oracle system-supplied common objects and metadata
 - Oracle system-supplied common users and roles
- Pluggable database containers (PDBs)
 - A container for an application:
 - Tablespaces (permanent and temporary)
 - Schemas/Objects/Privileges
 - Created/cloned/unplugged/plugged
 - Particular seed PDB\$SEED: fast provisioning of a new PDB
 - Limit of 4096 PDBs in a CDB including the seed
 - Limit of 10000 services in a CDB



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

A CDB is an Oracle database that contains the root, the seed, and zero to n PDBs.

What is a PDB in a CDB? A PDB is the lower part of the horizontally partitioned data dictionary plus the user's quota-consuming data.

A non-CDB cannot contain PDBs.

The multitenant architecture enables an Oracle database to contain a portable collection of schemas, schema objects, and nonschema objects that appear to an Oracle Net client as a separate database. For the PDBs to exist and work, the CDB requires a particular type of container, the root container, generated at the creation of the CDB. The root is a system-supplied container that stores common users, which are users that can connect to multiple containers, and system-supplied metadata and data. For example, the source code for system-supplied PL/SQL packages is stored in the root. If the root container exists, you can create containers of the other type, the PDB.

There is only one seed PDB in a CDB. The seed PDB is a system-supplied template that is used to create new PDBs.

A CDB can contain up to 4096 PDBs, including the CDB seed, with the services being limited to 10000. The V\$CONTAINERS view displays all containers including the root.

Terminology

- DBA, CDBA, and PDBA

- Common Vs Local:

- Users

- A **common user** can only be created in the root container.
 - A **local user** can only be created and known in a PDB.

- Roles

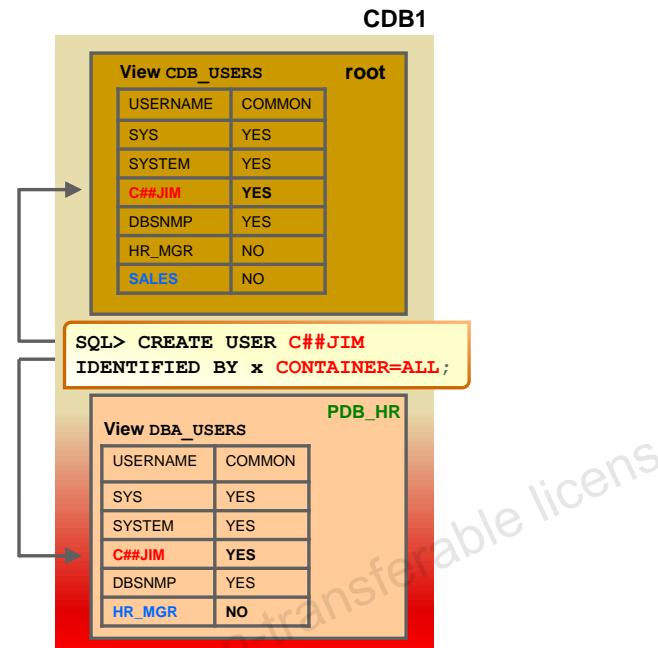
- A **common role** can only be created in the root container.

```
SQL> CREATE ROLE C##manager
CONTAINER=ALL;
```

- A **local role** can only be created and known in a PDB.

- Privileges

- CDB Vs PDB level



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.



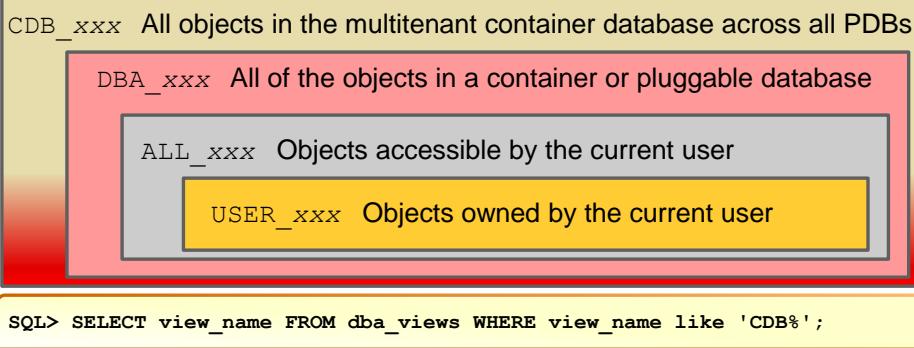
There are three types of database administrators.

- In a non-CDB, the DBA is responsible for all administrative tasks at the database level.
- In a CDB, there are two levels of administration:
 - A CDBA is responsible for administering the CDB instance and the root container.
 - A PDBA is responsible for administering its own PDB.

There is a new terminology for new entities.

- Common vs local:
 - Common users/roles versus local users/roles: A common user is a user that has the same username and authentication credentials across multiple PDBs, unlike the local user which exists in only one PDB. A local user is a traditional user, created in a PDB and known only in its own PDB. A role created across all containers is a common role. A role created in a specific PDB is local.
 - Common privileges versus local privilege: A privilege becomes common or local based on the way it is granted. A privilege granted across all containers is a common privilege. A privilege granted in a specific PDB is local.
- CDB vs PDB level:
 - CDB resource management works at CDB level, and PDB resource management at PDB level.
 - Unified audit policies can be created at CDB level and PDB level.

Data Dictionary Views



`SQL> SELECT view_name FROM dba_views WHERE view_name like 'CDB%';`

- `CDB_pdbs`: All PDBs within CDB
- `CDB_tablespaces`: All tablespaces within CDB
- `CDB_users`: All users within CDB (common and local)

DBA dictionary views providing information within PDB:

`SQL> SELECT table_name FROM dict WHERE table_name like 'DBA%';`

ORACLE®

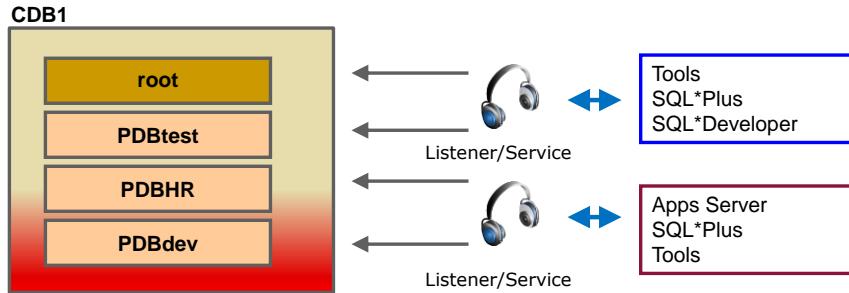
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

For backward-compatibility, DBA views show the same results in a PDB as in a non-CDB: `DBA_OBJECTS` shows the objects that exist in the PDB from which you run the query. This implies, in turn, that although the PDB and the root have separate data dictionaries, each data dictionary view in a PDB shows results fetched from both of these data dictionaries. The `DBA_xxx` views in the root show, even in a populated CDB, only the Oracle-supplied system—as is seen in a freshly created non-CDB. This is another advantage of the new architecture.

To support the duties of the CDB administrator, a new family of data dictionary views is supported with names such as `CDB_xxx`. Each `DBA_xxx` view has a `CDB_xxx` view counterpart with an extra column, `Con_ID`, which shows from which container the listed facts originate. Query the `CDB_xxx` views from the root and from any PDB. The `CDB_xxx` views are useful when queried from the root because the results from a particular `CDB_xxx` view are the union of the results from the `DBA_xxx` view counterpart over the root and all currently open PDBs. When a `CDB_xxx` view is queried from a PDB, it shows only the information that it shows in its `DBA_xxx` view counterpart. If you connect to the root and query `CDB_USERS`, you get the list of users, common and local, of each container. Now if you query `DBA_USERS`, you get the list of common users (you are aware that in the root, only common users exist). Now if you connect to a PDB, and query `CDB_USERS` or `DBA_USERS`, you get the same list of users, common and local, of the PDB.

The same backward-compatibility principle applies also to each of the familiar `v$views`.

Connection to a Non-RAC CDB



1. Every PDB has a default service created at PDB creation.

```
SQL> SELECT name, pdb FROM cdb_services;
```

2. Service name has to be unique across CDBs.

```
SQL> CONNECT / AS SYSDBA
SQL> CONNECT sys@"hostname:1525/CDB1" AS SYSDBA
SQL> CONNECT sys@"hostname:1525/PDBtest" AS SYSDBA
SQL> CONNECT local_user1@"hostname/PDBHR"
SQL> CONNECT common_user2@"hostname/PDBdev"
SQL> SHOW CON_NAME
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When you connect to a CDB, you either want to connect to the root or to a specific PDB. Any container in a CDB owns a service name.

- The root container service name is the CDB name given at the CDB creation concatenated with a domain name.
- Each new PDB is assigned a service name: the service name is the PDB name given at PDB creation concatenated with a domain name. If you create or plug a PDBtest PDB, its service name is PDBtest concatenated with a domain name. The container service names must be unique within a CDB, and even across CDBs that register with the same listener.
- You can find service names maintained in a CDB and PDBs in the CDB_SERVICES or V\$SERVICES views. The PDB column shows the PDB to which the services are linked.

To connect to the root, use local OS authentication or the root service name. For example, if you set the ORACLE_SID to the CDB instance name and use the command CONNECT / AS SYSDBA, you are connected to the root as a SYS user with common system privileges to manage and maintain all PDBs.

With the service name, you can use the EasyConnect syntax or the alias from tnsnames.ora.

Using EasyConnect, you would enter the following connect string:

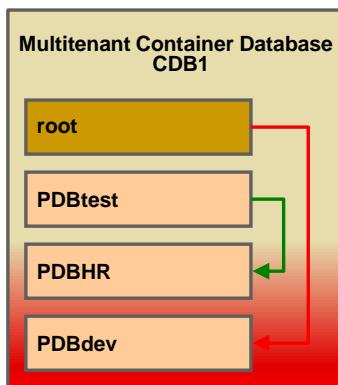
```
SQL> CONNECT username@"hostname:portnumber/service_name"
SQL> CONNECT username@"localhost:portnumber/service_name"
```

Using the tnsnames.ora file, you would enter the following connect string:

```
SQL> CONNECT username@"localhost:net_service_name"
```

To connect to a desired PDB, use either EasyConnect or the alias from the tnsnames.ora file, for example, as shown in the slide. In the examples used here, the net service name in the tnsnames.ora matches the service name.

Switching Connection



Two possible ways to switch connection between containers within a CDB:

- Reconnect

```
SQL> CONNECT / AS SYSDBA
SQL> CONNECT local_user1@PDBdev
```

- Use ALTER SESSION statement:

```
SQL> CONNECT sys@PDBtest AS SYSDBA
SQL> ALTER SESSION SET CONTAINER=PDBHR;
SQL> SHOW CON_NAME
SQL> ALTER SESSION SET CONTAINER=CDB$ROOT;
```

- Using CONNECT allows connection under a common or local user.
- Using ALTER SESSION SET CONTAINER allows connections only under a common user who is granted the new system privilege, SET CONTAINER.
 - AFTER LOGON triggers do not fire.
 - Transactions are still pending after switching containers.

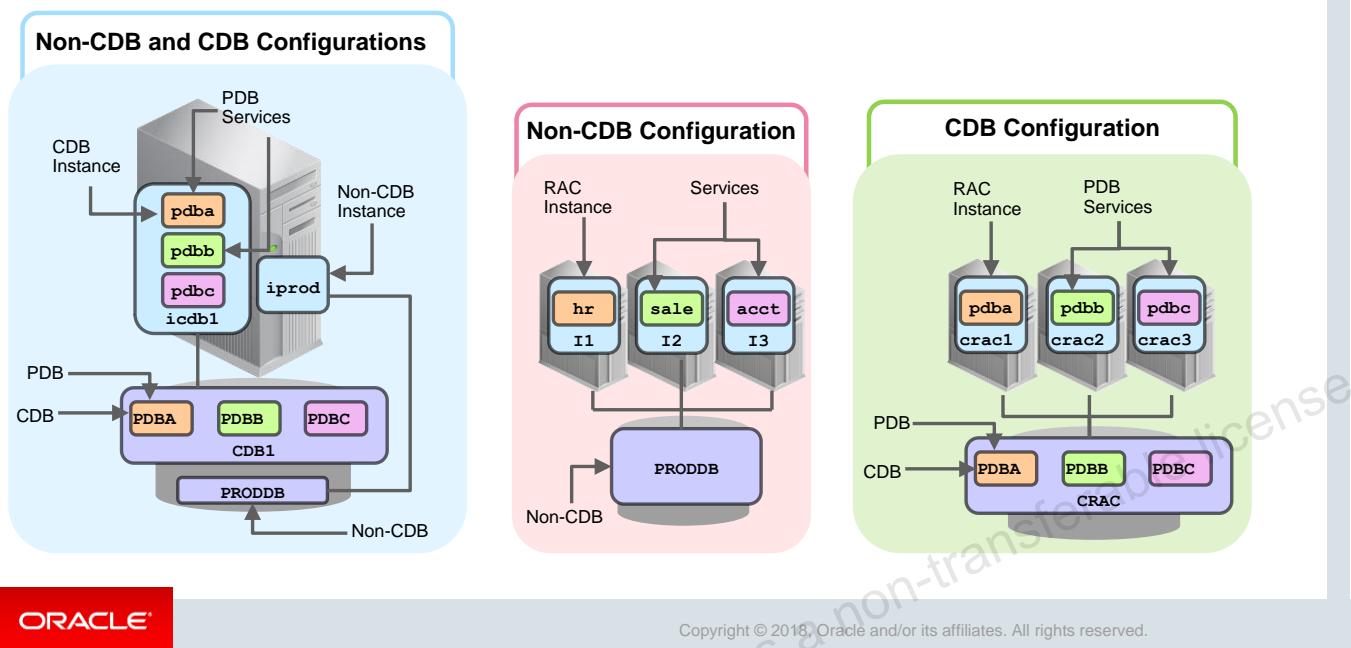


Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

A CDB administrator can connect to any container in the CDB using either CONNECT or ALTER SESSION SET CONTAINER to switch between containers. For example, the CDB administrator can connect to root in one session, and then in the same session switch to the PDBHR container. The requirement here being a common user known in both containers, and a system privilege SET CONTAINER.

- Using the command CONNECT allows connections under common or local users.
- Using ALTER SESSION SET CONTAINER allows connections only under a common user who is granted the new system privilege SET CONTAINER.
 - Ensure that the AFTER LOGON trigger does not fire.
 - Transactions that are neither committed, nor rolled back in the original container are still in a pending state while switching to another container and when switching back to the original container.

Oracle RAC and Multitenant Configuration



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In Oracle Database 12c, there are now three possible configuration options:

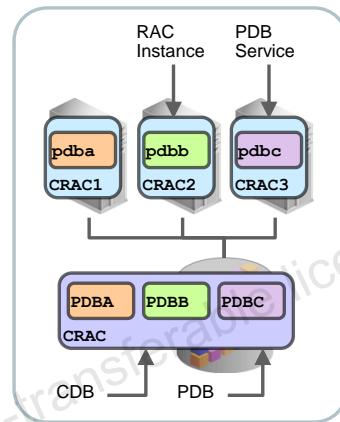
- **Non-CDB:** The old Oracle Database 11g architecture
- **Single-tenant configuration:** The special case of the new architecture, which does not require the licensed option (Oracle Multitenant option)
- **Multitenant configuration:** Typically more than one pluggable database (PDB) per multitenant container database (CDB), but can hold zero, one, or many PDBs at any one time, taking advantage of the full capabilities of the new architecture, which requires the licensed Oracle Multitenant option

Which are the possible instance/database configurations in Oracle Database 12c?

- In a non-RAC environment, each database instance can be associated with only one non-CDB or CDB.
- In a RAC environment, several instances can be associated with a non-CDB or CDB.
- An instance is associated with an entire CDB.

Oracle RAC and Multitenant Architecture

- In a multitenant architecture, there are:
 - Several instances per CDB
 - In Shared Undo mode, One UNDO tablespace per CDB instance.
 - In Local Undo mode, every container in the CDB uses local UNDO tablespace.
 - At least two groups of redo log files per CDB instance
- Each container is exposed as a service:
 - The root
 - Each PDB
- Management of services:
 - Default database services
 - Policy-managed databases:
 - Singletons
 - Uniform across all servers in a server pool



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

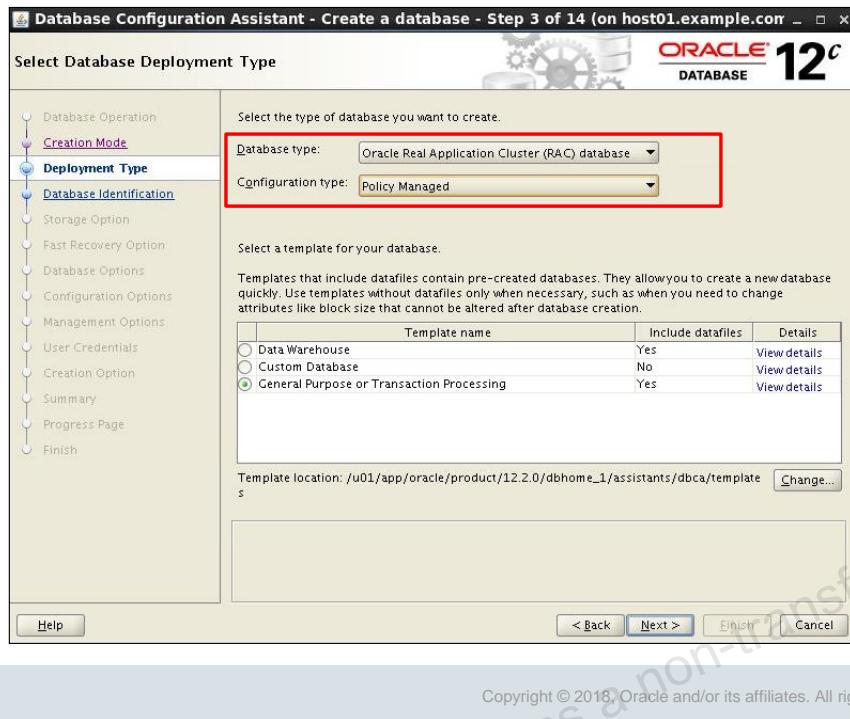
A RAC CDB, exactly as a RAC non-CDB, contains:

- One UNDO tablespace per instance in shared undo mode.
- In local undo mode, every container in the CDB uses local undo
- At least two groups of redo log files per instance
- Internal services at CDB level: Like in a RAC non-CDB, `SYS$BACKGROUND` is used by background processes only. `SYS$USERS` is the default service for user sessions that are not associated with any application service. Both internal services support all the workload management features and neither one can be stopped or disabled. A special Oracle CDB service is created by default for the Oracle RAC CDB with the same name as that of the CDB.

With the multitenant architecture, each container is exposed as a service. Each PDB and the root are assigned a database service when the container is created.

With clusterware policy-managed databases, each PDB service can be exposed across all RAC instances, across a subset of instances, or on a single instance (as shown in the slide). PDB Services can be associated with server pools. Because any server in the server pools within the cluster can run any of the CDBs, you do not have to create and maintain CDB instance-to-node-name mappings or PDB service-to-node-name mappings.

Creating a RAC CDB



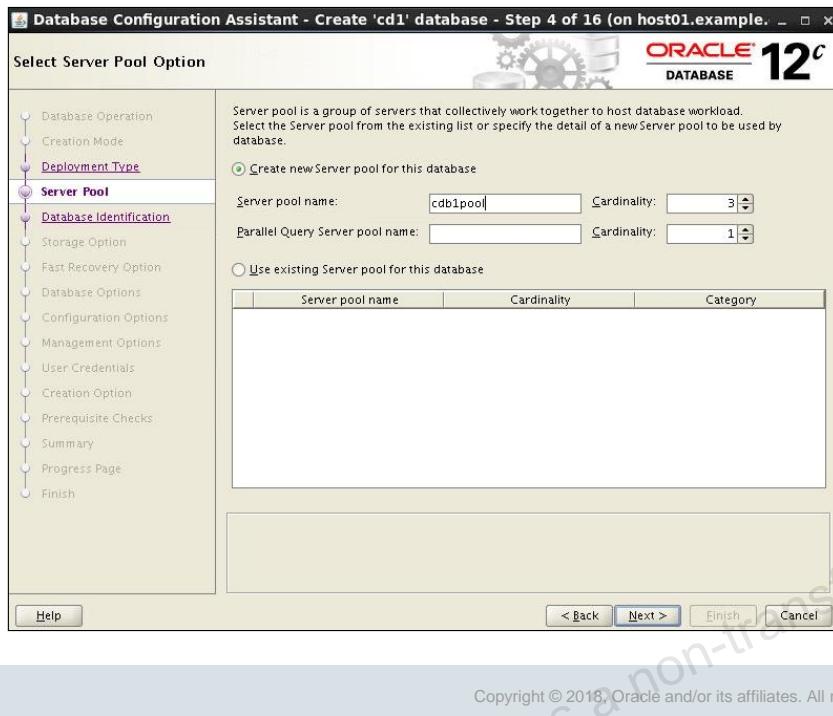
When creating a RAC CDB with DBCA, define the type of database, like you do when you create a RAC non-CDB:

- An Oracle single instance database
- An Oracle Real Application Clusters (RAC) database
- An Oracle RAC One Node database

In the same step, choose, like you do when you create a RAC non-CDB, between two types of management styles:

- Policy-Managed
- Admin-Managed

Hosting a RAC CDB in Server Pools



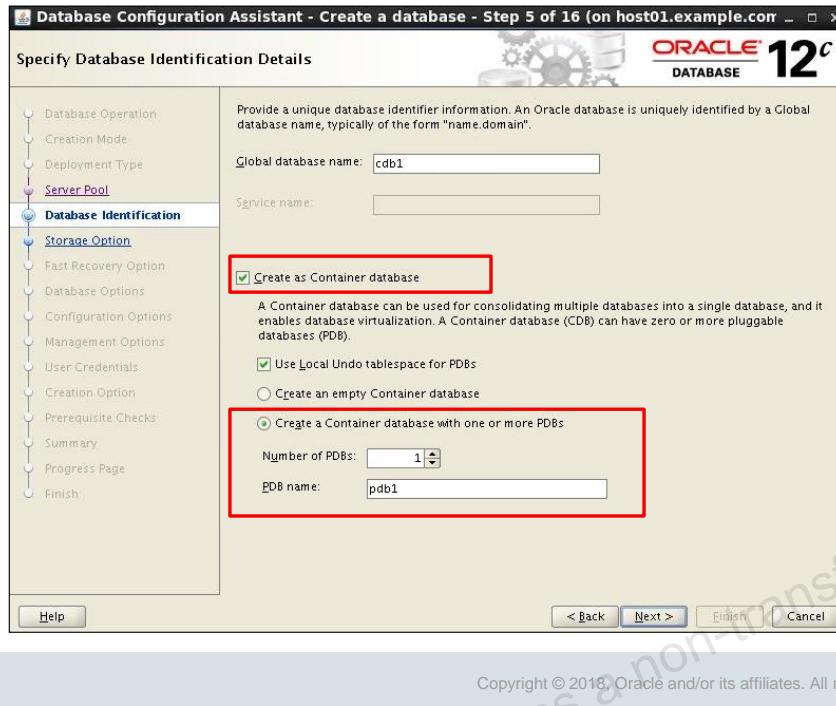
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Because policy-managed configuration was selected, the server pool to host the new CDB must be defined. Choose between using an existing server pool or creating a new one and specifying the detail of the new server pool to be used by the CDB.

Policy-managed deployments facilitate consolidation. In the case of schema consolidation, where multiple applications are being hosted in a single database (CDB) separated into PDBs, since server pools determine which services run together or separately, you can configure and maintain required affinity or isolation of PDB services.

Managing a policy-managed database requires less configuration and reconfiguration steps than an administrator-managed one with respect to creation, sizing, patching, and load balancing. Also, because any server in the server pools within the cluster can run any of the CDB instances and PDB services, you do not have to create and maintain CDB instance-to-node-name mappings and PDB service-to-node-name mappings.

Creating a RAC CDB Including PDBs



Select the “Create As Container Database” check box to create the database as a CDB (otherwise, the database is created as a non-CDB).

Starting with 12cR2, a CDB can run in local undo mode or shared undo mode. Local undo mode means that every container in the CDB uses local undo. Shared undo mode means that there is one active undo tablespace for a single-instance CDB. For an Oracle RAC CDB, there is one active undo tablespace for each instance.

Provide a pluggable database (PDB) name when using the “Create a database with default configuration” option. If the “Advanced Mode” option is selected (as shown in the slide), an empty CDB can be created with only the root and seed containers or a CDB can be created with one or several PDBs. If the CDB contains only one PDB, don’t provide an existing PDB name of another CDB. If the CDB contains several PDBs, a suffix is requested to generate the PDB names.

In Advanced Mode, the CDB can be registered with Enterprise Manager Cloud Control or configured for Enterprise Manager Database Express. The passwords for the SYS and SYSTEM users can also be set.

After CDB Creation

- Check the status of the CDB instances.

```
host01 $ srvctl status database -db cdb2
Instance cdb2_1 is running on node host01
Instance cdb2_2 is running on node host02
Instance cdb2_3 is running on node host03
```

- Check the UNDO tablespaces.

- Check the redo log files.

```
host01 $ export ORACLE_SID=cdb2_1
host01 $ sqlplus / as sysdba
SQL> SELECT tablespace_name, con_id FROM cdb_tablespaces
      WHERE contents='UNDO';

TABLESPACE_NAME          CON_ID
-----
UNDOTBS1                  1
UNDOTBS2                  1
UNDOTBS3                  1
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

After the CDB is created, use SRVCTL to verify that there are as many CDB instances running in the server pool as nodes defined in the server pool.

The UNDO tablespaces are created in the root container and there are as many UNDO tablespaces as CDB instances.

The groups of redo log files are displayed with the V\$LOGFILE view:

```
SQL> select group#, con_id from v$logfile;
```

GROUP#	CON_ID
2	0
1	0
5	0
6	0
3	0
4	0

Note that there are six redo log groups because there are three CDB instances. Note also that the container ID is 0, referring to the CDB, whereas the container ID is 1 for the data files of the UNDO tablespaces referring to the specific root container.

Connecting Using CDB/PDB Services

- The services for the root and each PDB are started on each node of the server pool: **PDB2** can be accessed on any CDB instance.

```
host01 $ lsnrctl status
...
Service "pdb2" has 1 instance(s).
  Instance "cdb2_3", status READY, has 1 handler(s) for this
service...
```

- To connect to the root container on one of the three CDB instances:

```
SQL> connect system@"host01/cdb2" → root
```

- To connect to a PDB on one of the three CDB instances:

```
SQL> connect system@"host01/pdb2" → PDB
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Use LSNRCTL to verify that there are as many CDB instances running on the nodes of the server pool as nodes in the server pool, and as many PDB services as PDBs managed by the CDB.

```
$ lsnrctl status
```

...

Service "cdb2" has 1 instance(s). → *CDB service*

Instance "cdb2_3", status READY, has 1 handler(s) for this
 service... → *One of the three CDB instances*

Service "pdb2" has 1 instance(s). → *PDB service*

Instance "cdb2_3", status READY, has 1 handler(s) for
 this service... → *Attached to one of the three CDB instances*

To connect to the root container to perform CDB administrative tasks, use the CDB service. To connect to a PDB to perform PDB administrative tasks, use one of the PDB services. Either use an EasyConnect string or a net service name declared in the `tnsnames.ora` file as shown in the examples in the slide.

Opening a PDB in a RAC CDB

Start the CDB instances.

```
host01 $ srvctl start database -db cdb2
```

- 1 Open a PDB in the current instance:

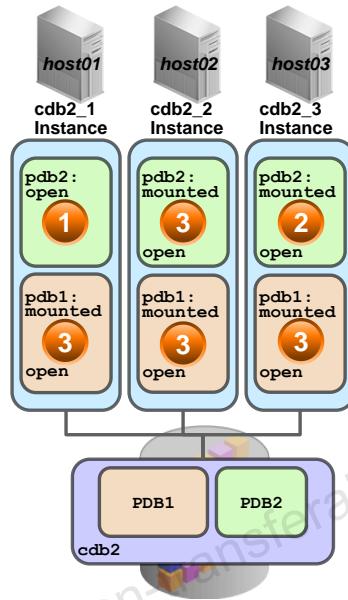
```
SQL> CONNECT sys@host01/cdb2 AS SYSDBA
SQL> ALTER PLUGGABLE DATABASE pdb2 OPEN;
```

- 2 Open a PDB in some instances:

```
SQL> CONNECT sys@host03/cdb2 AS SYSDBA
SQL> ALTER PLUGGABLE DATABASE pdb2 OPEN
      INSTANCES = ('cdb2_3');
```

- 3 Open a PDB in all instances:

```
SQL> ALTER PLUGGABLE DATABASE ALL OPEN
      INSTANCES=ALL;
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.



When the CDB is started with SRVCTL, the following operations occur:

- The CDB instances are started on all nodes of the server pool.
- The CDB control files are opened for the instances: the root and PDB containers are mounted.
- The root is opened in `READ WRITE` mode for all instances, the seed is opened in `READ ONLY` mode for all instances, and the PDBs remain mounted (closed).

To open one or several PDBs on one or some or all of the CDB instances, connect to the root of any CDB instance and issue an `ALTER PLUGGABLE DATABASE OPEN` statement, specifying the following clauses:

- The `INSTANCES` clause to modify the state of the PDB in the specified Oracle RAC instances. If you omit this clause, then the state of the PDB is modified only in the current instance as shown in the first example of the slide where the `pdb2` PDB is opened in the `cdb2_1` instance on `host01`, but remains mounted in the other two CDB instances. Use `instance_name` to specify one or more instance names in a comma-separated list enclosed in parentheses to modify the state of the PDB in those instances as shown in the second example of the slide where the `pdb2` PDB is opened in the `cdb2_3` instance on `host03`, but remains mounted in the `cdb2_2` instance.
- Specify `ALL` to modify the state of the PDB in all instances. Specify `ALL EXCEPT` to modify the state of the PDB in all instances except the specified instances. If the PDB is already open in one or more instances, then you can open it in additional instances, but it must be opened in the same mode as in the instances in which it is already open. This operation opens the data files of the PDBs and provides availability to users.

Use the `open_mode` column from the `V$PDBS` view to verify that all PDBs are in `READ WRITE` open mode except the seed being still in `READ ONLY` open mode.

You can also open a PDB while connected as `SYSDBA` within the PDB. In this case, it is not necessary to name the PDB to open.

Closing a PDB in a RAC CDB

Shut down the instances of a RAC CDB:

```
host01 $ srvctl stop database -db cdb2
```

- All PDBs closed
- CDB closed/CDB dismounted/Instance shut down

Close a PDB:

- In the current instance only:

```
SQL> CONNECT sys@host01/cdb2 AS SYSDBA
SQL> ALTER PLUGGABLE DATABASE pdb2 CLOSE;
```

- In some or all instances of the CDB:

```
SQL> ALTER PLUGGABLE DATABASE pdb2 CLOSE INSTANCES='cdb2_3';
SQL> ALTER PLUGGABLE DATABASE pdb2 CLOSE INSTANCES = ALL;
```

- In the current instance and reopen it in another instance:

```
SQL> ALTER PLUGGABLE DATABASE pdb2 CLOSE RELOCATE TO 'cdb2_3';
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When all instances of a CDB are shut down, the data files of the root and seed containers and all PDBs are closed, all the control files are closed, and the instances shut down.

A PDB can be closed with the `INSTANCES` clause as shown in the slide. In the first example with the `ALTER PLUGGABLE DATABASE` command, the PDB is closed in the current instance of the CDB. In the second example, the PDB is closed in the `cdb2_3` instance of the CDB. In the last example, the PDB is closed in all the CDB instances.

If you need to close all PDBs in all the CDB instances but keep the root opened for maintenance purposes, use the following command:

```
ALTER PLUGGABLE DATABASE ALL CLOSE INSTANCES = ALL;
```

When the PDB is closed on all the CDB instances, the data files of the PDB are closed.

To display the PDB open mode, use the `V$PDBS` view `OPEN_MODE` column in each of the CDB instances. It may display `MOUNTED` in some of the instances and `READ WRITE` in other instances.

To instruct the database to reopen the PDB on a different Oracle RAC instance, use the `RELOCATE` clause. Specify `RELOCATE` to reopen the PDB on a different instance that is selected by Oracle Database or `RELOCATE TO 'instance_name'` to reopen the PDB in the specified instance.

Types of Services

There are two types of services:

- Internal services: SYS\$BACKGROUND, SYS\$USERS
- Application services:
 - A special Oracle CDB service: Created and available by default on all CDB instances
 - PDB (application) services: Limit of 10000 services per CDB

```
SQL> SELECT name, con_id FROM cdb_services;
NAME          CON_ID
-----        -----
SYS$BACKGROUND      1
SYS$USERS          1
cdb1XDB            1
cdb1              1
pdb1              3
```



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

There are two types of services:

- **Internal services:** The RDBMS supports two internal services. SYS\$BACKGROUND is used by the background processes only. SYS\$USERS is the default service for user sessions that are not associated with any application service. Both internal services support all the workload management features and neither one can be stopped or disabled.
- **Application services:** CDB and PDB services.
 - A special Oracle CDB service is created by default for the Oracle RAC CDB. This default service is always available on all instances in an Oracle RAC environment, unless an instance is in restricted mode. This service is useful to connect on another node instance than the instance the session is connected to.
 - Each PDB is assigned a default service. The name of the service is the PDB name. The service is the only method to connect to a PDB.

Managing Services

There are two categories of services to manage:

- Default PDB services: one service per PDB
 - Created at PDB creation
 - Not managed by the clusterware
 - Started at PDB opening
- Dynamic PDB services:
 - Useful to start, stop, and place PDBs across the CDB instances
 - Uniformly managed across all nodes in the server pool
 - Running as a singleton service in the same server pool
 - Manually assigned to PDBs with SRVCTL
 - Manually started with SRVCTL
 - Automatically restored to its original state by clusterware
 - PDB automatically opened when the service is started



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

There are two categories of services:

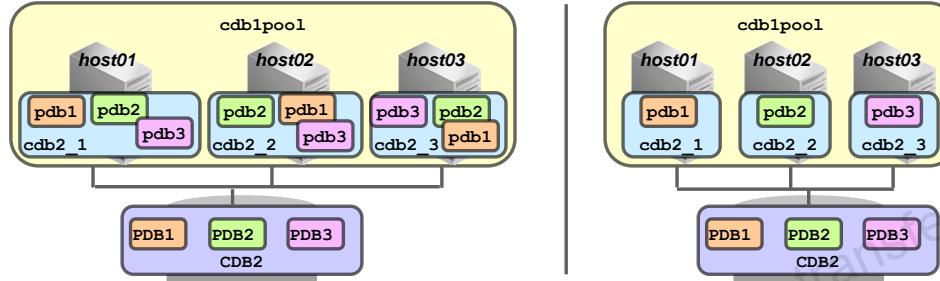
- **Default PDB services:** Each new PDB is assigned a default service. The name of the service is the PDB name. This service is started when the PDB is opened. When the CDB is opened and an AFTER STARTUP ON DATABASE trigger triggers an ALTER PLUGGABLE DATABASE ALL OPEN, the services are started.
- **Dynamic PDB services:** Assign one dynamic database service to each PDB to coordinate start, stop, and placement of PDBs across instances in a RAC CDB, using SRVCTL. When a dynamic PDB service is started by the clusterware, the PDB is automatically opened. There is no need to create an AFTER STARTUP ON DATABASE trigger to open the PDB. Because PDBs can be managed using dynamic services, typical Oracle RAC-based management practices apply. So, if a PDB service is in the ONLINE state when Oracle Clusterware is shut down on a server hosting this service, then the service will be restored to its original state after the restart of Oracle Clusterware on this server. This way, opening PDBs is automated as with any other Oracle RAC database.

Affinitizing PDB Services to Server Pools

- A PDB can be assigned several services.
- Each PDB service can be exposed on some or all of the RAC instances within server pools.

```
host01$ srvctl add service -db cdb2 -pdb pdb1 -service pdb1srv
          -policy automatic -serverpool cdb1pool
          -cardinality uniform | singleton
```

```
host01$ srvctl start service -db cdb2 -service pdb1srv
```



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Policy-managed databases facilitate the management of services, because the services are assigned to a single server pool and run as singletons or uniform across all servers in the pool. You no longer have to create or maintain explicit preferred and available CDB instance lists for each PDB service. If a server moves into a server pool because of manual relocation or a high availability event, all uniform PDB services in the CDB instances are automatically started. If a server hosting one or more singleton services goes down, those services will automatically be started on one or more of the remaining servers in the server pool.

To assign a PDB service to a CDB, use SRVCTL as shown in the example in the slide using the –CARDINALITY UNIFORM parameter. The PDB service will be uniformly managed across all nodes in the server pool as shown in the graphic on the left side of the slide. In this case, each PDB service is available in each of the CDB instances.

If you want to have the PDB service running as a singleton service in the same server pool, use the –CARDINALITY SINGLETON parameter as shown in the graphic on the right side of the slide. In this case, the pdb1 service is available in one of the CDB instances.

Once the PDB service is created, start the service using SRVCTL.

If the pdb1 service is in the ONLINE state when Oracle Clusterware is shut down on a server hosting this service, then the service will be restored to its original state after the restart of Oracle Clusterware on this server.

Adding a PDB to a RAC CDB

- Create a new PDB from the seed or clone it from another PDB:
 - Connect to the root as a common user with the CREATE PLUGGABLE DATABASE system privilege:

```
SQL> CREATE PLUGGABLE DATABASE pdb1
  2 ADMIN USER admin1 IDENTIFIED BY p1 ROLES=(CONNECT)
  3 FILE_NAME_CONVERT = ('PDB$SEEDdir', 'PDB1dir');
```
 - Open the new PDB in the CDB instances.
- The default PDB service is automatically started in the CDB instances.



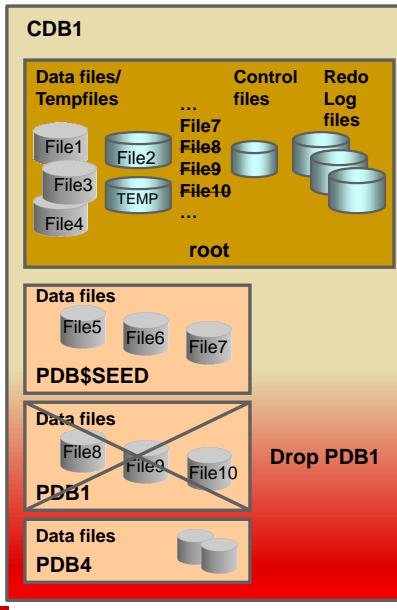
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To create a new PDB from the seed (the template PDB), connect to the root as a common user with the CREATE PLUGGABLE DATABASE system privilege and execute the CREATE PLUGGABLE DATABASE statement as shown in the first example in the slide. The FILE_NAME_CONVERT clause designates first the source directory of the seed datafiles and second the destination directory for the new PDB datafiles.

To create a new PDB from another PDB, connect to the root as a common user with the CREATE PLUGGABLE DATABASE system privilege and execute the CREATE PLUGGABLE DATABASE statement as shown in the second example in the slide. Before proceeding with the CREATE PLUGGABLE DATABASE statement, the source PDB needs to be in READ ONLY mode in all CDB instances.

When the statement completes, open the PDB in the required CDB instances. This starts the default service created for the new PDB in the CDB instances.

Dropping a PDB from a RAC CDB



ORACLE®

1. Remove the dynamic PDB services.

```
host01$ srvctl remove service
-db cdb1
-service mypdb1serv
```

2. Close the PDB in all instances.

```
SQL> ALTER PLUGGABLE DATABASE
  2  pdb1 CLOSE INSTANCES=ALL;
```

3. Drop the PDB.

```
SQL> DROP PLUGGABLE DATABASE
  2  pdb1 [INCLUDING DATAFILES];
```

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When you no longer need the data in a PDB, you can drop the PDB.

When you drop a PDB specifying INCLUDING DATAFILES, all of its datafiles listed in the control files are deleted.

With or without the clause INCLUDING DATAFILES, the DROP PLUGGABLE DATABASE statement modifies the control files to eliminate all references to the dropped PDB.

KEEP DATAFILES is the default behavior to keep the data files, useful in scenarios where an unplugged PDB is plugged into another CDB or replugged into the same CDB.

Because the dynamic PDB services are not removed, remove them before dropping the PDB.

Quiz

Which of the following are true?

- a. Only one SYSTEM tablespace per CDB
- b. Only one instance per PDB
- c. A set of redo log files per PDB
- d. Only one UNDO tablespace per CDB instance
- e. One SYSAUX tablespace per PDB



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Describe the multitenant architecture in a non-RAC environment
- Describe the multitenant architecture in a RAC environment
- Create a RAC multitenant container database (CDB)
- Create a pluggable database (PDB) in a RAC CDB
- Use default CDB and PDB services
- Create PDB services
- Associate PDB services with server pools
- Drop a PDB from a RAC CDB



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Practice 13: Overview

This practice covers the following topics:

- Exploring CDB Architecture and Structures in RAC
- Cloning a PDB in a RAC CDB
- Affinitizing PDB services to CDB instances
- Dropping a PDB from a RAC CDB



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

GANG LIU (gangl@baylorhealth.edu) has a non-transferable license
to use this Student Guide.

Quality of Service Management



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to describe:

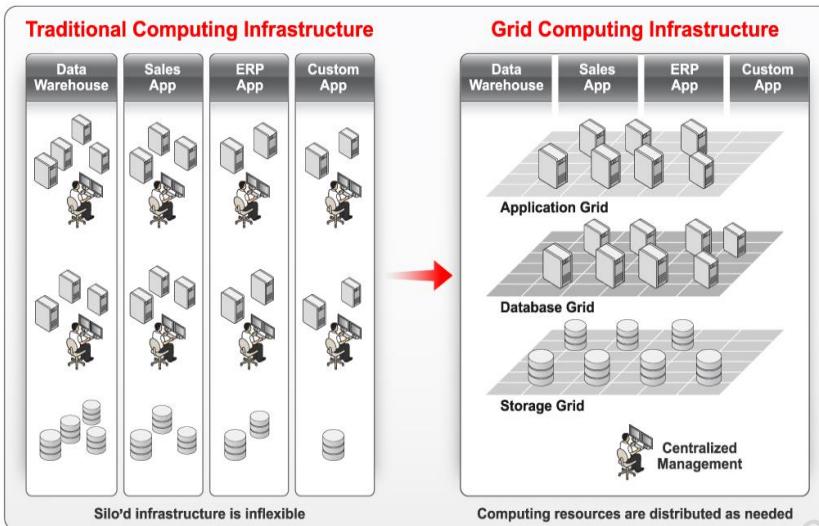
- The purpose of Oracle Database Quality of Service (QoS) Management
- The benefits of using Oracle Database QoS Management
- The components of Oracle Database QoS Management
- The operation of Oracle Database QoS Management



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

QoS Management Background



ORACLE®

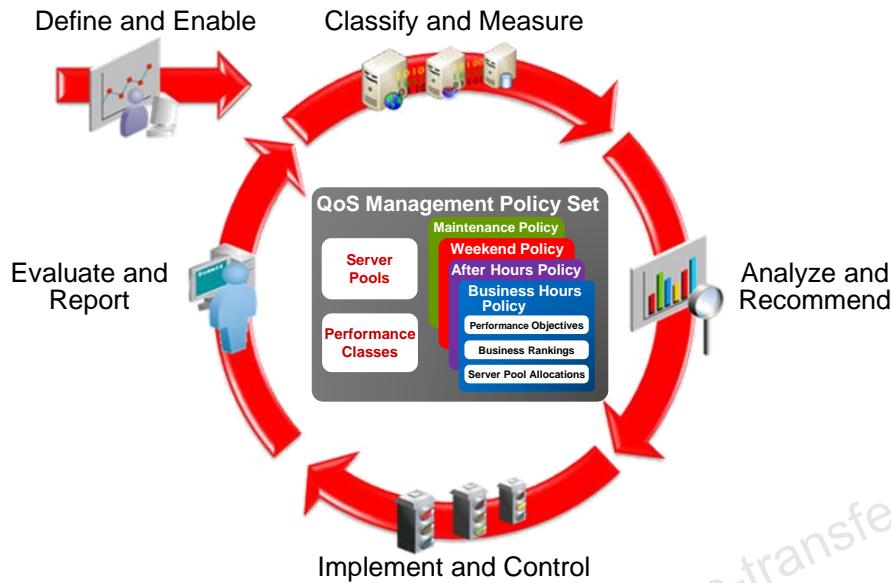
Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In previous Oracle Database releases, you could use services for workload management and isolation. For example, a group of servers might be dedicated to data warehouse work, while another is dedicated to your sales application, a third group is used for ERP processing, and a fourth group to a custom application. Using services, the database administrator can allocate resources to specific workloads by manually changing the number of servers on which a database service is allowed to run. The workloads are isolated from each other, so that demand spikes, failures, and other problems in one workload do not affect the other workloads. The problem with this type of deployment is that each workload needs to be separately provisioned for peak demand because resources are not shared.

You could also define services that shared resources by overlapping server allocations. However, even with this capability, you had to manually manage the server allocations and each service was mapped to a fixed group of servers.

Starting with Oracle Database 11g, you can use server pools to logically partition a cluster and provide workload isolation. Server pools provide a more dynamic and business-focused way of allocating resources because resource allocations are not dependant on which servers are up. Rather, the server pool allocations dynamically adjust when servers enter and leave the cluster to best meet the priorities defined in the server pool policy definitions.

QoS Management Overview



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Many companies are consolidating and standardizing their data center computer systems. In parallel with this, the migration of applications to the Internet has introduced the problem of managing demand surges that cannot be fully anticipated. In this type of environment, it is necessary to pool resources and have management tools that can detect and resolve bottlenecks in real time. Policy-managed server pools provide a foundation for dynamic workload management. However, they can only adjust resource allocations in response to server availability changes.

QoS Management is an automated, policy-based workload management (WLM) system that monitors and adjusts the environment to meet business-level performance objectives. Based on resource availability and workload demands, QoS Management identifies resource bottlenecks and provides recommendations for how to relieve them. It can make recommendations for the system administrator to move a server from one server pool to another, or to adjust access to CPU resources using the Database Resource Manager, in order to satisfy the current performance objectives. Using QoS Management enables the administrator to ensure the following:

- When sufficient resources are available to meet the demand, business-level performance objectives for each workload are met, even if the workloads change.
- When sufficient resources are not available to meet all demands, QoS Management attempts to satisfy more critical business objectives at the expense of less critical ones.

QoS Management and Exadata Database Machine

In its initial form, QoS Management is a feature of the Oracle Database product family:

- Introduced in Oracle Database 11g release 2
- Associated with Oracle RAC software
- Released exclusively on Exadata Database Machine
- Focused on environments supporting multiple OLTP workloads
- Not Exadata-specific technology
- The first step along the road towards a broader solution



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

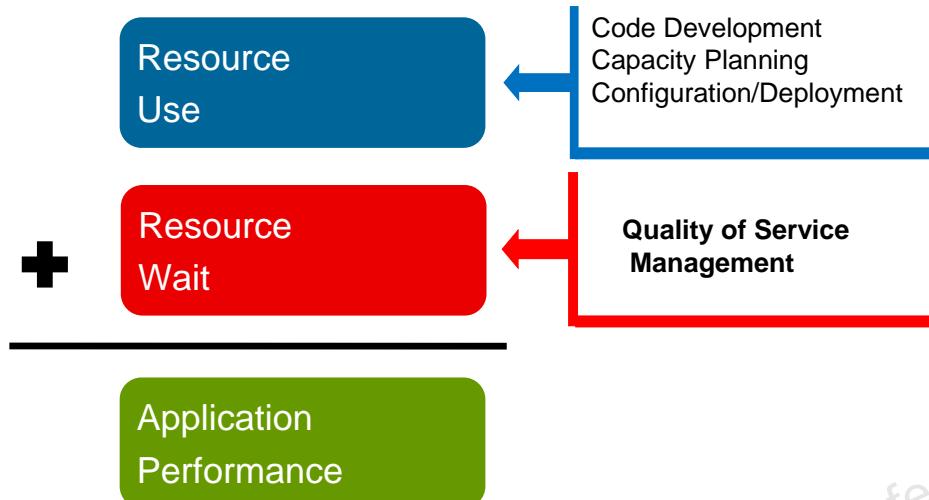
The initial incarnation of QoS Management is as a feature of the Oracle Database product family in association with Oracle Real Application Clusters (RAC) software. It was first introduced in Oracle Database 11g release 2.

The initial set of features and benefits associated with QoS Management are exclusively available to Exadata Database Machine customers and are best suited to customers using Database Machine predominantly as a consolidation platform for OLTP applications.

QoS Management software can operate on non-Exadata environments where Oracle Database 11g release 2 is available. Commencing with version 11.2.0.3, a subset of QoS Management functionality has been released that enables non-Exadata users to monitor performance classes, but not to generate and implement changes in response to the currently observed workload.

In its current form, QoS Management provides a powerful database-focused capability that represents the first step along the road toward a broader workload management solution.

QoS Management Focus



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

QoS Management monitors the performance of each work request on a target system. By accurately measuring the two components of performance, resource use and wait, bottlenecks can be quickly detected and resources reallocated to relieve them, thus preserving or restoring service levels. Changing or improving the execution time generally requires application source code changes. QoS Management, therefore, only observes and manages wait times.

QoS Management bases its decisions on observations of how long work requests spend waiting for resources. Examples of resources that work requests might wait for include hardware resources, such as CPU cycles, disk I/O queues, and global cache blocks.

Other waits can occur within the database, such as latches, locks, pins, and so on. While these database waits are accounted for by QoS Management, they are not broken down by type or managed. Minimizing unmanaged waits requires changes that QoS Management cannot perform, such as application code changes and database schema optimizations. QoS Management is still beneficial in these cases, because the measurement and notification of unmanaged waits can be used as a tool to measure the effect of application optimization activities.

QoS Management Benefits

- Determines where additional resources are needed
- Determines whether additional hardware can be added to maintain acceptable performance
- Reduces the number of critical performance outages
- Reduces the time to resolve performance objective violations
- Improves system stability as the workload changes
- Helps to ensure that SLAs are met
- Facilitates effective sharing of hardware resources



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Some of the benefits of QoS Management include:

- By categorizing and measuring database work, QoS Management can help administrators determine where additional resources are needed.
- QoS Management is Oracle RAC-aware, and it uses this fundamental understanding to determine if additional hardware can be added to maintain acceptable performance.
- QoS Management helps reduce the number of critical performance outages. By reallocating runtime resources to the busiest business-critical applications, those applications are less likely to suffer from a performance outage.
- QoS Management reduces the time needed to resolve performance objective violations. Rather than requiring administrators to understand and respond to changes in performance, much of the work can be automated. Administrators are provided with a simple interface to review and implement the recommended changes.
- Performance stresses can often lead to system instability. By moving resources to where they are most needed, QoS Management reduces the chance that systems will suffer from performance stress and related instability.
- QoS Management allows the administrator to define performance objectives that help to ensure Service Level Agreements (SLAs) are being met. Once the objectives are defined, QoS Management tracks performance and recommends changes if the SLAs are not being met.
- As resource needs change, QoS Management can reallocate hardware resources to ensure that applications make more effective use of those resources. Resources can be removed from applications that no longer require them, and added to an application that is suffering from performance stress.

QoS Management Functional Overview

QoS Management works with Oracle RAC and Oracle Clusterware to:

- Manage database server CPU resources by evaluating CPU wait times to identify workloads that are not meeting performance objectives
 - QoS Management can recommend:
 - Adjustments to the size of server pools
 - Alterations to consumer group mappings
 - Adjustments to the CPU resources allocated to different database instances within a server pool
- Manage memory pressure due to number of sessions or runaway workloads
 - QoS Management restricts new sessions from being established on servers that are suffering from memory stress.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

QoS Management works with Oracle RAC, Oracle Clusterware, and Cluster Health Monitor (CHM) to manage database resources to meet service levels and manage memory pressure for managed servers.

Typically, database services are used to group related work requests and for measuring and managing database work. For example, a user-initiated query against the database might use a different service than a report generation application. To manage the resources used by a service, some services may be deployed on several Oracle RAC instances concurrently, while others may be deployed on only one instance. In an Oracle RAC database, QoS Management monitors the nodes on which user-defined database services are offered. Services are created in a specific server pool and the service runs on all servers in the server pool. If a singleton service is required because the application cannot effectively scale across multiple RAC servers, the service can be hosted in a server pool with a maximum size of one.

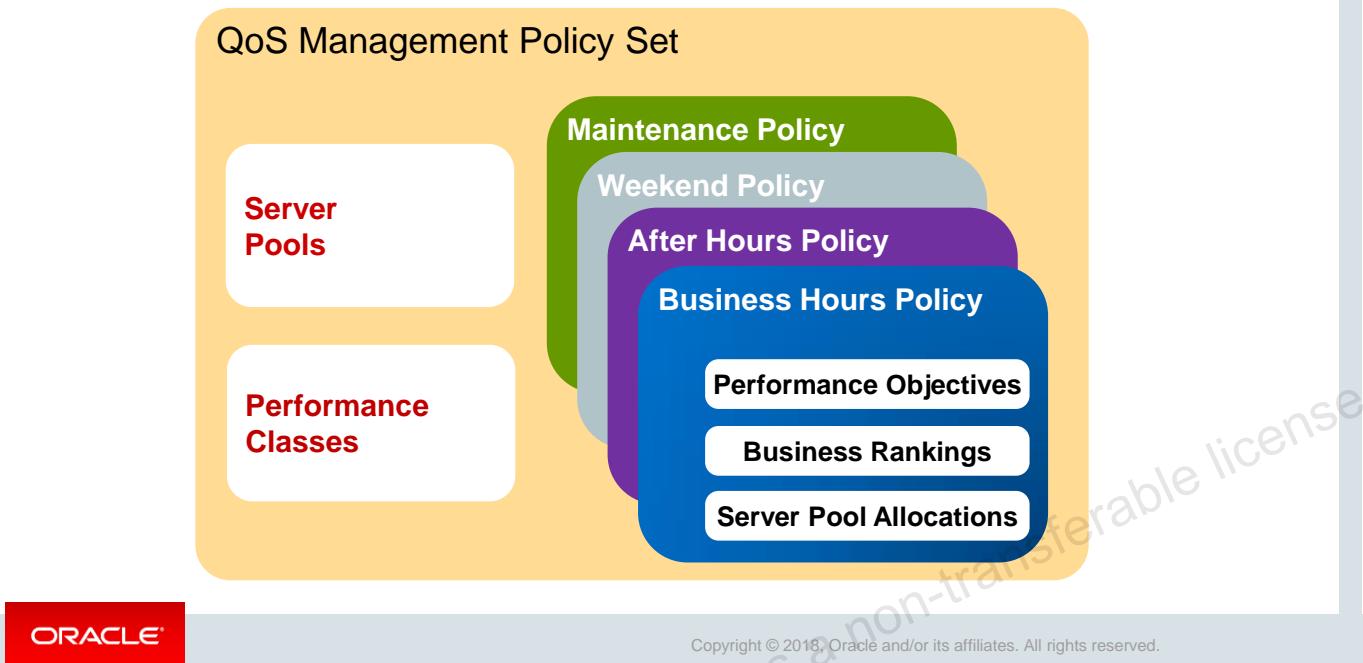
QoS Management periodically evaluates database server CPU wait times to identify workloads that are not meeting performance objectives. If needed, QoS Management provides recommendations for adjusting the size of the server pools or alterations to Database Resource Manager (DBRM) consumer group mappings. Starting with Oracle Database release 11.2.0.3, QoS Management also supports moving CPUs between databases within the same server pool.

DBRM is an example of a resource allocation mechanism; it can allocate CPU shares among a collection of resource-consumer groups based on a resource plan specified by an administrator. A resource plan allocates the percentage of opportunities to run on the CPU. QoS Management does not adjust DBRM plans; it activates a shared multi-level resource plan and then, when implementing a recommendation, it moves workloads to specific resource-consumer groups to meet performance objectives for all the different workloads.

Enterprise database servers can run out of available memory due to too many open sessions or runaway workloads. Running out of memory can result in failed transactions or, in extreme cases, a reboot of the server and loss of valuable resources for your applications. QoS Management eases memory pressure by temporarily shutting down the services for database instances on a server suffering from memory stress. This causes new sessions to be directed to lighter loaded servers. Rerouting new sessions protects the existing workloads and the availability of the memory-stressed server.

When QoS Management is enabled and managing an Oracle Clusterware server pool, it receives a metrics stream from Cluster Health Monitor that provides real-time information about memory resources for a server, including the amount of available memory, the amount of memory currently in use, and the amount of memory swapped to disk for each server. If QoS Management determines that a node is under memory stress, the Oracle Clusterware managed database services are stopped on that node preventing new connections from being created. After the memory stress is relieved, the services are restarted automatically and the listener can send new connections to the server. The memory pressure can be relieved in several ways (for example, by closing existing sessions or by user intervention).

QoS Management Policy Sets



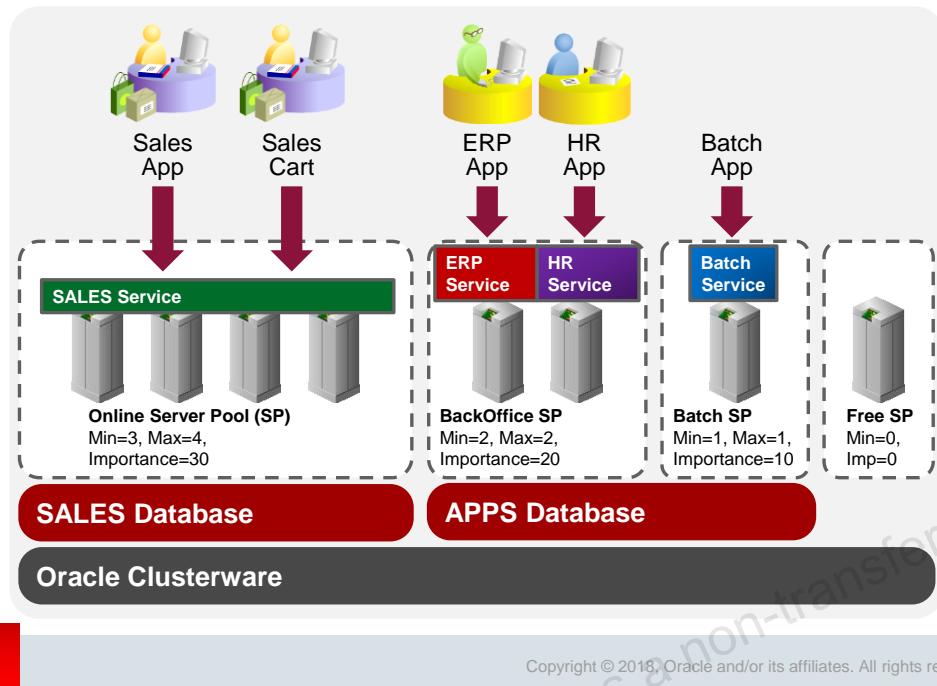
A central concept in QoS Management is the policy set. A policy set allows you to specify your resources, performance classes (workloads), and a collection of performance policies that specify the performance objective for each performance class and sets constraints for resource availability. QoS Management uses a system-wide policy set that defines performance objectives based upon the classes of work and the availability of resources. Specific performance policies can be enabled based upon a calendar schedule, maintenance windows, events, and so on. Only one performance policy can be in effect at any time.

To maintain the current performance objectives, QoS Management makes resource reallocation recommendations and predicts their effect. The recommendations can be easily implemented with a single button click.

A policy set consists of the following:

- The server pools that are being managed by QoS Management
- Performance classes, which are work requests with similar performance objectives
- Performance policies, which describe how resources should be allocated to the performance classes by using performance objectives and server pool directive overrides. Within a performance policy, performance objectives are ranked based on business importance, which enables QoS Management to focus on specific objectives when the policy is active.

Server Pools



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

A server pool is a logical division of a cluster. Server pools facilitate workload isolation within a cluster while maintaining agility and allowing users to derive other benefits associated with consolidation. Administrators can define server pools, which are typically associated with different applications and workloads. An example is illustrated in the slide. QoS Management can assist in managing the size of each server pool and also by managing the allocation of resources within a server pool.

When Oracle Grid Infrastructure is first installed, a default server pool, called the Free pool, is created. All servers are initially placed in this server pool. Specific server pools can then be created for each workload that needs to be managed. When a new server pool is created, the servers assigned to that server pool are automatically moved out of the Free pool and placed into the newly created server pool.

After a server pool is created, a database can be configured to run on the server pool, and cluster-managed services can be established for applications to connect to the database.

For an Oracle RAC database to take advantage of the flexibility of server pools, the database must be created using the policy-managed deployment option, which places the database in one or more server pools.

A key attribute of policy-based management is the allocation of resources to server pools based on cardinality and importance.

When the cluster starts or when servers are added, all the server pools are filled to their minimum levels in order of importance. After the minimums are met, server pools continue to be filled to their maximums in order of importance. If there are any left-over servers, they are allocated to the Free pool.

If servers leave the cluster for any reason, a server reallocation may take place. If there are servers in the Free pool and another server pool falls below its maximum value, a free server is allocated to the affected server pool. If there are no free servers, then server reallocation takes place only if a server pool falls below its minimum level. If that occurs, a server will be sourced from one of the following locations in the following order:

1. The server pool with the lowest importance that has more than its minimum number of servers
2. The server pool with the lowest importance that has at least one server and has lower importance than the affected server pool

Using these mechanisms, server pools can maintain an optimal level of resources based on the current number of servers that are available.

Consider the example shown in the slide. If one of the servers in the Online server pool failed, the server currently residing in the Free server pool would automatically move to the Online server pool.

Now, if one of the servers from the BackOffice server pool failed, there would be no servers to allocate from the Free server pool. In this case, the server currently servicing the Batch server pool would be dynamically reallocated to the BackOffice server pool, because the failure would cause the BackOffice server pool to fall below its minimum and it has a higher importance than the Batch server.

If one node is later returned to the cluster, it will be allocated to the Batch pool in order to satisfy the minimum for that server pool.

Any additional nodes added to the cluster after this point will be added to the Free pool, because all the other pools are filled to their maximum level.

Performance Classes

- A performance class is a group of work requests whose service level needs to be managed.
- Work requests are defined by performance classifiers containing the database service name and optional session parameters.
- An initial set of performance classifiers is automatically discovered and created from cluster-managed services.
- Performance objectives are defined on performance classes.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Performance classes are used to categorize workloads with similar performance requirements. A set of classification rules are evaluated against work requests when they arrive at the edge of the system. These rules allow value matching against attributes of the work request; when there is a match between the type of work request and the criteria for inclusion in a performance class, the work request is classified into that performance class.

This classification of work requests applies the user-defined name, or tag, that identifies the performance class (PC) to which the work request belongs. All work requests that are grouped into a particular PC have the same performance objectives. In effect, the tag connects the work request to the performance objective that applies to it. Tags are carried along with each work request so that every component of the system can take measurements and provide data to QoS Management for evaluation against the applicable performance objectives.

QoS Management supports user-defined combinations of connection parameters called classifiers to map performance classes to the actual workloads running in the database.

These connection parameters fall into two general classes and can be combined to create fine-grained Boolean expressions:

- **Configuration Parameters:** The supported configuration parameters are `SERVICE_NAME` and `USERNAME`. Each classifier in a performance class must include one or more cluster-managed database services. Additional granularity can be achieved by identifying the Oracle Database user that is making the connection from either a client or the middle tier. The advantage of using these classifiers is that they do not require application code changes to define performance classes.

- **Application Parameters:** The supported application parameters are MODULE, ACTION, and PROGRAM. These are optional parameters set by the application as follows:
 - OCI: Use OCI_ATTR_MODULE and OCI_ATTR_ACTION.
 - ODP.NET: Specify the ModuleName and ActionName properties on the OracleConnection object.
 - JDBC: Set MODULE and ACTION in SYS_CONTEXT.

The PROGRAM parameter is set or derived differently for each database driver and platform. Please consult the appropriate Oracle Database developer's guide for further details and examples.

To manage the workload for an application, the application code directs database connections to a particular service. The service name is specified in a classifier, so all work requests that use that service are tagged as belonging to the performance class created for that application. If you want to provide more precise control over the workload generated by various parts of the application, you can create additional performance classes and use classifiers that include MODULE, ACTION, or PROGRAM in addition to SERVICE_NAME or USERNAME.

The performance classes used in an environment can change over time. A common scenario is to replace a single performance objective with multiple, more specific performance objectives, dividing the work requests into additional performance classes. For example, application developers can suggest performance classes for QoS Management to use. In particular, an application developer can define a collection of database classifiers using the MODULE and ACTION parameters, and then put them in separate performance classes so each type of work request is managed separately.

Classification and Tagging

- Each session is classified:
 - The classification is determined by evaluating session parameters against performance class classifiers.
 - Evaluation occurs only when a session is established or when session parameters change.
 - This minimizes the overhead associated with classification.
- Each work request is tagged:
 - The tag is based on the current session classification.
 - The tag connects the work request with a performance class.
 - It enables measurements associated with the work request to be recorded against the appropriate performance class.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To enable QoS Management, work requests must be classified and tagged.

When a database session is established, the session parameters are evaluated against the performance class classifiers to determine a classification. Work associated with the session is then tagged based on the session classification until the session ends or the session parameters change. If the session parameters change, the classification is re-evaluated. Thus the overhead associated with the classification is very small, because the classification is only evaluated when a session is established or when session parameters change.

Tags are permanently assigned to each work request so that all the measurements associated with the work request can be recorded against the appropriate performance class. In effect, the tag connects the work request to a performance class and its associated performance objective.

Performance Policies

- Performance policies are named sets of performance objectives and server pool overrides to meet business objectives.
 - Performance objectives can be ranked according to their importance.
- Only one policy is active at any time.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

To manage various performance objectives, a QoS Management administrator defines one or more performance policies. For example, the administrator might define a performance policy for normal business hours, another for weekday non-business hours, one for weekend operations, and another to be used during processing for the quarter-end financial closing. Note that at any time, only one performance policy is in effect.

A performance policy has a collection of performance objectives in effect; one or more for each application that is being managed on the system. Some performance objectives are always more critical to the business than others, while other performance objectives might be more critical at certain times, and less critical at other times. The ability to define multiple performance policies inside the policy set provides QoS Management with the flexibility required to implement different priority schemes when they are required.

Performance Class Ranks

- Performance class ranks assign a relative level of business criticality to each performance class within a performance policy:
 - Highest
 - High
 - Medium
 - Low
 - Lowest



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Performance Objectives

- Performance objectives can be derived from your SLAs. They specify:
 - A business requirement
 - The performance class to which the business requirement applies
- Average response time per database call is currently the only performance objective type.
 - Response time is the total time from the time the database receives the request to when the response leaves the server.
 - Response time does not include network traffic time.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

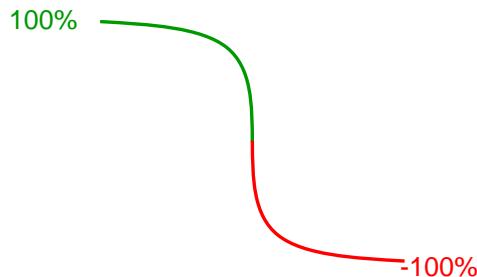
You create a performance objective for each performance class to specify the desired performance level for that performance class. A performance objective specifies both a business requirement, and the work to which it applies (the performance class). For example, a performance objective might say that database work requests that use the SALES service should have an average response time of less than 60 milliseconds.

Each performance policy includes a performance objective for each and every performance class, unless the performance class is marked measure-only. In this release, QoS supports only one type of performance objective, average response time.

Response time is based upon database client calls from the point that the database server receives the request over the network until the request leaves the server. Response time does not include the time it takes to send the information over the network to or from the client. The response time for all database client calls in a performance class is averaged and presented as the average response time.

Performance Satisfaction Metrics

- Different performance objectives can be compared using the Performance Satisfaction Metric (PSM).
- The PSM quickly shows how the system is coping with the objective.



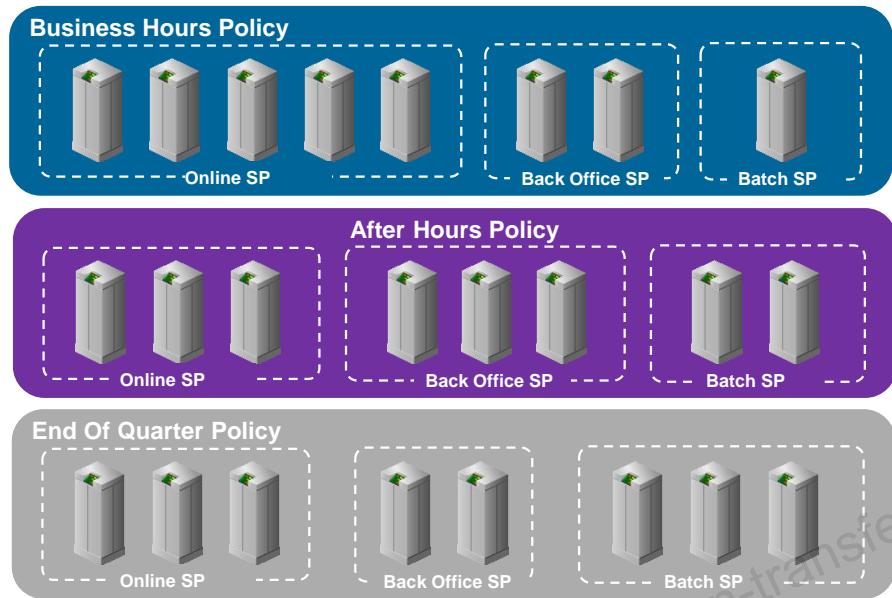
ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Different performance objectives are used to measure the performance of different workloads. QoS Management currently supports only OLTP workloads and uses only the average response time performance objective. When configuring QoS Management, you can have very different performance objectives for each performance class. For example, one performance objective may specify that a Checkout call should complete within 1 millisecond, while another performance objective may specify that a Browse call should complete within 1 second. As more performance objectives are added to a system, it can be difficult to compare them quickly.

Because of this, it is useful to have a common and consistent numeric measure indicating how the current workload for a performance class is measuring up against its current performance objective. This numeric measure is called the Performance Satisfaction Metric. The Performance Satisfaction Metric is thus a normalized numeric value (between +100% and -100%) that indicates how well a particular performance objective is being met, and which allows QoS Management to compare the performance of the system for widely differing performance objectives.

Server Pool Directive Overrides



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Overview of Metrics

- QoS Management uses a standardized set of metrics.
- There are two metric types:
 - Performance metrics give an overview of where time is spent in the system.
 - Resource metrics measure the time that work requests use a resource or wait for a resource.
- Metrics are used to identify bottlenecked resources and to determine the best corrective action:
 - For a performance class, the bottlenecked resource is the resource that contributes the largest average wait time.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

QoS Management uses a standardized set of metrics collected by all the servers in the system. There are two types of metrics: performance metrics and resource metrics. These metrics enable direct observation of the use and wait time incurred by work requests in each performance class, for each resource requested, as it traverses the servers, networks, and storage devices that form the system.

Performance metrics are collected at the entry point to each server in the system. They give an overview of where time is spent in the system and enables comparisons of wait times across the system. Data is collected periodically and forwarded to a central point for analysis, decision-making, and historical storage.

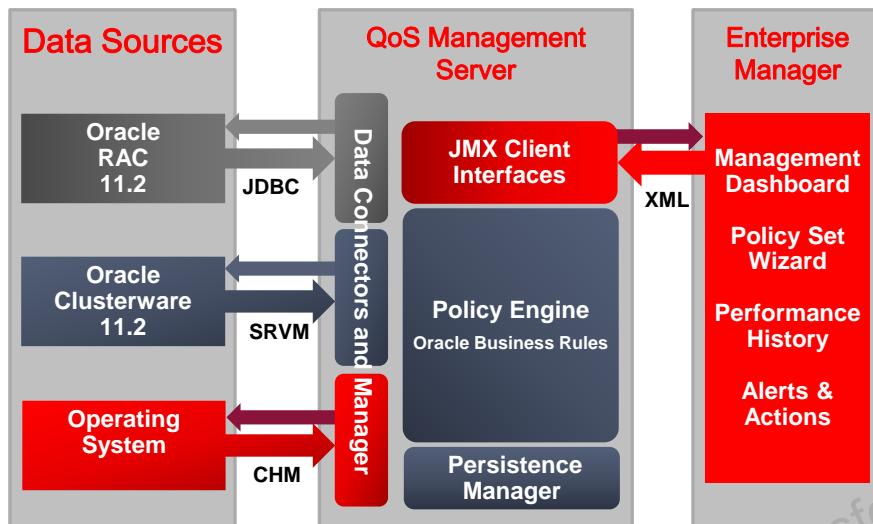
A key performance metric is response time, or the difference between the time a request comes in and the time a response is sent out. The response time for all database calls in a performance class is averaged and presented as the average response time. Another important performance metric is the arrival rate of work requests. This provides a measure of the demand associated with each performance class.

Resource metrics exist for the following resources; CPU, Storage I/O, Global Cache, and Other (database waits). Two resource metrics are provided for each resource:

- **Resource usage time:** Measures how much time is spent using the resource
- **Resource wait time:** Measures the time spent waiting to get the resource

QoS Management metrics provide the information needed to systematically identify performance class bottlenecks in the system. When a performance class is violating its performance objective, the bottleneck for that performance class is the resource that contributes the largest average wait time for each work request in that performance class.

QoS Management Architecture



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

QoS Management retrieves metrics data from each database instance running in managed server pools and correlates the data by performance class every 5 seconds. The data includes many metrics; for example, call arrival rate and CPU, I/O and Global Cache use, and wait times. The data is combined with the current topology of the cluster and the health of the servers in the Policy Engine to determine the overall performance profile of the system with regard to the current performance objectives established by the active performance policy.

The performance evaluation occurs once a minute and results in a recommendation if there is a performance class not meeting its objective. The recommendation specifies what resource is bottlenecked. Specific corrective actions are included, if possible, along with the projected impact on all performance classes in the system. The slide shows the collection of data from various data sources by the data connectors component of QoS Management:

- Oracle RAC 11.2 communicates with the data connector using JDBC.
- Oracle Clusterware 11.2 communicates with the data connector using the SRVM component of Oracle Clusterware.
- The server operating system communicates with the data connector using Cluster Health Monitor (CHM).

Enterprise Manager displays the information in a variety of ways (for example, on the Management Dashboard, Policy Set Wizard, Performance History, and Alerts and Actions pages).

QoS Management Recommendations

- If performance objectives are not being met, QoS Management makes a recommendation.
- Each recommendation focuses on improving the highest ranked performance class exceeding its performance objective.
- Recommendations may include:
 - Changing consumer group mappings
 - Reprioritize work within existing resource boundaries.
 - Moving servers between server pools
 - Reprioritize resources between server pools to meet workload demands.
 - Moving CPUs between databases within a server pool
 - Reprioritize CPU resources within existing server pool boundaries.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

If your business experiences periodic demand surges, then to retain performance levels for your applications you can acquire additional hardware to be available when needed, and sit idle when not needed. Rather than have extra servers sit idle for most of the time, you might decide to use those servers to run other application workloads. However, if the servers are busy running other applications when a demand surge hits, your main business applications are not able to perform as expected. QoS Management helps to manage such situations.

When you implement a performance policy, QoS Management continuously monitors the system and manages it using an iterative process. When one or more performance objectives are not being met, each iteration seeks to improve the performance of a single performance objective; the highest ranked performance objective that is currently not being met. When all performance objectives are being met, QoS Management makes no further recommendations.

The recommendations take the form of moving servers between server pools, changing consumer group mappings, or moving CPUs between databases within a server pool.

Changing consumer group mappings may involve promoting a specific workload so that it gets a greater share of resources, or it may involve demoting a competing workload as a way of making additional resources available to the target performance class. In both cases, workloads are reprioritized within existing resource boundaries.

Moving servers between server pools is another approach used by QoS Management. This approach alters the distribution of servers to meet workload demands.

Commencing with Oracle Database release 11.2.0.3, QoS Management can also move CPU resources between databases within the same server pool. This alters the distribution of CPU resources between database instances using instance caging and provides additional control for environments where multiple databases are consolidated within the same Exadata Database Machine environment.

Implementing Recommendations

Action: Rank 1: Promote sales cart from Consumer Group 2 to Consumer Group 0.

Rationale: All potential single mapping changes have been analyzed. Changes evaluated and rejected are listed below.

Evaluation: The beneficiary's PSM value is expected to change by 3.764 percentage points. The sum of all PSM values is expected to change by -12.314 percentage points. This action is a candidate for recommendation.

Performance Class	Projected (%)	Projected Change (%)	Optimum Value (sec)	Current Value (sec)	Promised Value (sec)
Default	100	0.0	0.00000	0.01158	0.01106
sales app	77	0.0	0.01000	0.02232	0.00233
sales cart	34	-16.1	0.00900	0.02364	0.05039
eip app	42	0.0	0.00500	0.02231	0.00291

Projected Results:

Situation Analysis: Donor Performance Classes. Quality of Service Management could help sales cart at the expense of sales app. sales app is another PC using resource cpu in Server Pool online. sales app's Performance Objective is of lesser rank than sales cart's Performance Objective. Quality of Service Management could move servers from Server Pool backoffice to Server Pool online. The current size of Server Pool backoffice is larger than its configured minimum size. Default is not currently violating its Performance Objective. Default's Performance Objective is of lesser rank than sales cart's Performance Objective.

Donor Server Pools: Quality of Service Management could move servers from Server Pool backoffice to Server Pool online. The current size of Server Pool backoffice is larger than its configured minimum size.

Implement:



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

When QoS Management is working to improve the performance of a particular performance class, it recommends to add more of the bottleneck resource (such as CPU time) for that performance class, or to make the bottleneck resource available more quickly to work requests in the performance class.

Implementing a recommendation makes the resource less available to other performance classes. The negative impact on the performance classes from which the resource is taken may be significantly smaller than the positive impact on the service that is getting better access, resulting in a net win for the system as a whole. Alternatively, the performance class being penalized may be less business critical than the one being helped.

When generating recommendations, QoS Management evaluates the impact to the system performance as a whole. If the improvement for one performance class is rather small, but the negative impact on another performance class is large, then QoS Management might report that the performance gain is too small, and not recommended. If there is more than one way to resolve the bottleneck, then QoS Management advises the best overall recommendation factoring in variables such as the calculated impact on all the performance classes along with the predicted disruption and settling time associated with the action. Using Oracle Enterprise Manager, you can view the current recommendation and the alternative recommendations.

Performance data is sent to Oracle Enterprise Manager for display on the QoS Management Dashboard and Performance History pages. Alerts are generated to drive notifications that one or more performance objectives are not being met or that a problem has developed that prevents one or more server pools from being managed. As a result of these notifications the administrator can implement the recommendation.

In this release, QoS Management does not implement the recommendations automatically. It suggests a way of improving performance, which must then be implemented by the administrator by clicking the Implement button. After implementing a recommendation, the system is allowed to settle before any new recommendations are made. This is to ensure stable data is used for further evaluations and also to prevent recommendations that result in oscillating actions.

QoS Support For Admin-Managed RAC Databases

- In Oracle Database 12.2, you can use QoS with RAC on systems in full management mode in both policy- and admin-managed deployments.
- QoS now supports full management of multitenant RAC databases in both policy- and admin-managed deployments.
- The ability to expand or shrink the number of instances by changing the server pool size is not available for administrator-managed databases.



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.



Quiz

Oracle Database Quality of Service Management helps to meet performance objectives by reducing resource usage.

- a. True
- b. False



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.



Quiz

Oracle Database Quality of Service Management recommendations can include:

- a. Moving servers between server pools
- b. Adding spindles to improve I/O performance
- c. Changing consumer group mappings
- d. Recommending partitioning strategies to ease global cache bottlenecks



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to describe:

- The purpose of Oracle Database Quality of Service (QoS) Management
- The benefits of using Oracle Database QoS Management
- The components of Oracle Database QoS Management
- The operation of Oracle Database QoS Management



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle Database Exadata Cloud Service Overview



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

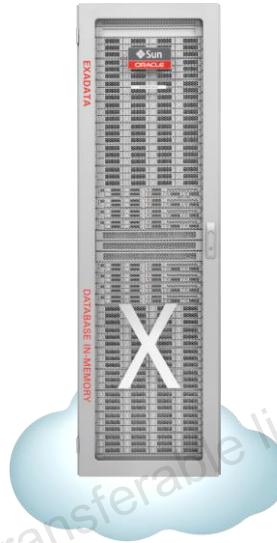
- Describe the architecture and capabilities of Exadata Cloud Service
- Compare and contrast between Exadata Cloud Service and an on-premise Exadata implementation



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Introducing Exadata Cloud Service

- Oracle Database with all features and options:
 - Industry-leading database for mission-critical OLTP and analytics
- On Exadata Database Machine:
 - The fastest and most available database cloud platform
- In the Oracle Cloud:
 - No capital expenditure, just a simple monthly subscription
 - Oracle deploys and manages the infrastructure
 - Fast, elastic, web-driven service provisioning
 - Complete service isolation with no over-provisioning
 - 100% compatibility with on-premises applications and Oracle database



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Oracle Database Exadata Cloud Service enables you to leverage the power of Oracle Exadata Database Machine inside the Oracle Cloud. With Exadata Cloud Service you get:

- Oracle Database with all features and options:

Each Exadata Cloud Service database deployment is provisioned with a complete Oracle Database installation that includes all the features of Oracle Database Enterprise Edition, plus all the database enterprise management packs and all the Enterprise Edition options, such as Oracle Real Application Clusters (RAC), Oracle Database In-Memory, and Oracle Multitenant.

- On Exadata Database Machine:

The foundation for Exadata Cloud Service is Exadata Database Machine. Deployed at thousands of sites around the world, Exadata is established as the highest performing and most available platform for running Oracle Database. With a fault-tolerant architecture featuring scale-out database servers and scale-out intelligent storage connected with a fully redundant high-performance InfiniBand fabric, Exadata is an ideal cloud platform.

Exadata Cloud Service delivers all of the advanced features of Exadata, including SQL offload, Smart Flash Cache, Storage Index, and Hybrid Columnar Compression.

- In the Oracle Cloud:

Customers pay a simple monthly subscription fee for Exadata Cloud Service. There is no initial capital cost and no data center costs.

All supporting infrastructure for Exadata Cloud Service is deployed, maintained and managed by Oracle, including datacenter networking, private Exadata InfiniBand networks, physical Exadata database servers and storage servers, firmware, and Exadata Storage Server software.

Exadata Cloud Service includes easy-to-use web-based wizards through which you can quickly provision an Exadata system and associated database deployments. The wizards are available through Oracle's Cloud Portal, <http://cloud.oracle.com>.

Exadata Cloud Service features complete service isolation with no over-provisioning of hardware to ensure that response times and throughput are predictable for critical business processes. This contrasts with some cloud service delivery models that silently overprovision hardware, and consequently may not be able to deliver the expected resources during busy periods.

Exadata Service is 100% compatible with on-premises Oracle databases and all existing applications. Exadata Cloud Service enables existing on-premises Exadata customers to easily embark on a journey to the cloud – without compromising the database performance and availability levels they enjoy with their on-premises Exadata deployments. While existing Oracle Database customers who have not yet experienced Exadata can easily start enjoying the performance, availability and scalability benefits of Exadata without compromising any of the database functionality that they rely on. With Exadata Cloud Service, organizations can easily deploy a hybrid cloud environment that uses on-premises databases as well as databases in the Cloud.

Service Configuration Options

	Quarter Rack	Half Rack	Full Rack
Number of Database Servers	2	4	8
Number of CPU Cores ¹	16 - 68	56 - 136	112 – 272
Total RAM Capacity	496 GB	992 GB	1984 GB
Number of Exadata Storage Servers ²	3	6	12
Total Flash Capacity	19.2 TB	38.4 TB	76.8 TB
Total Usable Disk Capacity ³	42 TB	84 TB	168 TB
Maximum SQL Flash Bandwidth ⁴	30 GB/sec	60GB/sec	120GB/sec
Maximum SQL Flash Read IOPS ⁵	900,000	1,800,000	3,600,000
Maximum SQL Flash Write IOPS ⁶	500,000	1,000,000	2,000,000
Maximum SQL Disk Bandwidth ⁴	4.5 GB/sec	9 GB/sec	20 GB/sec
Maximum SQL Disk IOPS ⁵	7,000	14,000	28,000
Maximum Data Load Rate ⁷	5 TB/hour	10 TB/hour	10 TB/hour



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

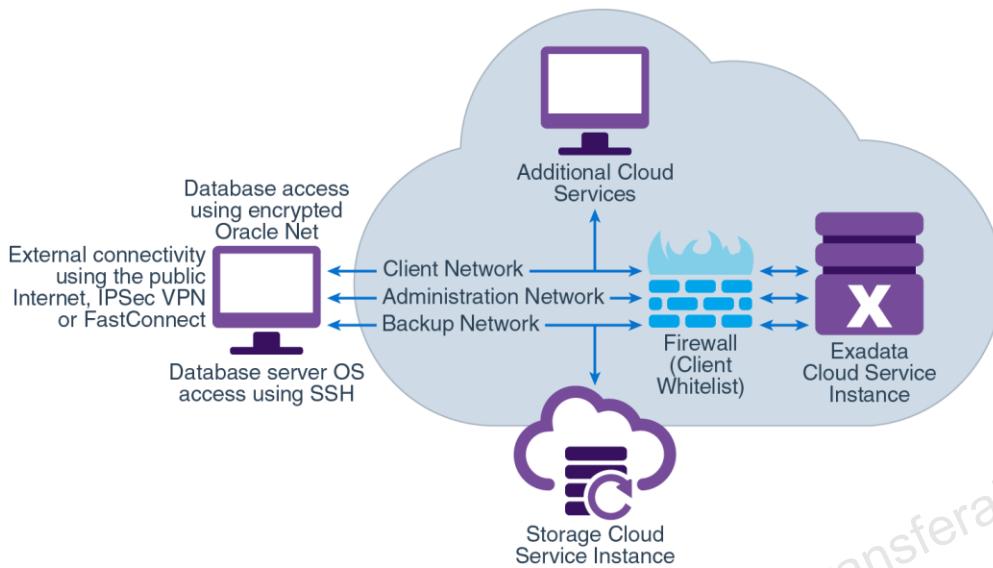
Each Exadata Cloud Service instance is based on an Exadata configuration that contains a predefined number of database servers and a predefined number of Exadata Storage Servers, all tied together by a high-speed, low-latency InfiniBand network and intelligent Exadata software. Three configurations are offered; Quarter Rack, Half Rack and Full Rack. The table in the slide outlines the vital statistics for each system configuration.

Note the following:

1. Each Exadata Cloud Service configuration is equipped with a fixed amount of memory, storage and network resources. However, you can choose how many database server CPU cores are enabled, within the minimum and maximum limits listed in the table. This enables you to scale an Exadata Cloud Service configuration to meet workload demands, and only pay for the database server resources that you need. Each database server must contain the same number of enabled CPU cores.
2. Note that Exadata Cloud Service Half Rack and Full Rack configurations differ from on-premises Half Rack and Full Rack. The Exadata Cloud Service Half Rack contains six storage servers, and the Exadata Cloud Service Full Rack contains twelve storage servers. Consequently, an Exadata Cloud Service Half Rack is twice the size of a Quarter Rack, and a Full Rack is twice the size of a Half Rack.
3. The usable storage capacity is the storage that available for Oracle Database files after taking into account high-redundancy ASM mirroring (triple mirroring), which is used to provide highly resilient database storage on all Exadata Cloud Service configurations. The usable storage capacity does not factor in the effects of Exadata compression capabilities, which can be used to increase the effective storage capacity.

4. Bandwidth is the peak physical scan bandwidth that is achieved by running SQL, and assuming that there is no database compression. Effective user data bandwidth is higher when database compression is used. In all cases, actual performance varies by application.
5. Based on 8K Oracle Database I/O requests. Note that the I/O size greatly affects Flash IOPS, so IOPS based on smaller I/Os is not relevant for databases.
6. Based on 8K Oracle Database I/O requests. Flash write I/Os are measured at the storage servers after ASM mirroring, which issues multiple storage I/Os to maintain redundancy.
7. Load rates are typically limited by database server CPU capacity, not I/O. Rates vary based on a variety of factors, including the load method and data types used, and the use of indexes, compression and partitioning.

Service Connection Options



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide outlines the primary ways of connecting to Exadata Cloud Service.

You can connect directly from cloud clients, which are applications already running in the Oracle Cloud. This includes java applications connecting through JDBC running on the Java Cloud Service, or client applications connecting through Oracle Net (SQL*Net) running on a Compute Cloud Service instance.

You can also connect to Exadata Cloud Service from your on-premises applications by using Oracle Net. On Exadata Cloud Service, Oracle Net is configured by default to use native encryption and integrity capabilities to secure data in transit.

Access to the database server operating system is provided using Secure Shell (SSH). This is primarily used for administration purposes.

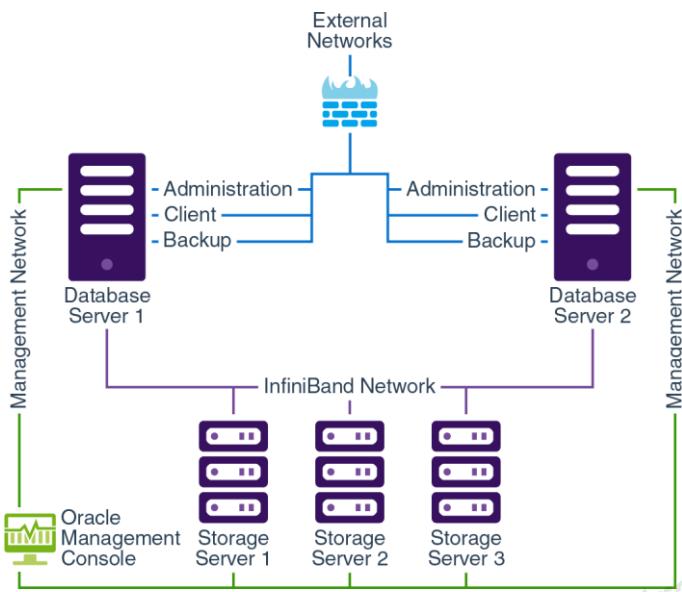
A backup network is also provided. This network keeps high-load activities separate from application connections and is primarily used when Exadata Cloud Service database deployments are backed up to an Oracle Storage Cloud Service container.

The Oracle Cloud network is firewall-protected and you must specify the clients that can access the service. The permitted clients are registered in the firewall using a whitelist.

To provide connectivity between your network and the Oracle Cloud you can:

- Use the public Internet and secure protocols, such as SSH and encrypted Oracle Net.
- Configure an IPSec VPN to enable secure connectivity between your network and the Oracle Public Cloud over the Internet.
- Use Oracle Network Cloud Service - FastConnect Partner Edition, which is a secure, high-bandwidth, point-to-point networking solution that is integrated with Oracle Cloud.

Service Architecture



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide illustrates the Exadata Cloud Service architecture for a Quarter Rack service instance. Larger service instances are principally the same, except that they contain more database servers and Exadata Storage Servers.

The architecture is essentially the same as for an on-premises implementation of Exadata, with clustered database servers connected to a grid of Exadata Storage Servers through a high-speed, low-latency, InfiniBand network.

Application users and administrators can connect only to the database servers, using the supplied administration, client and backup network interfaces.

Oracle manages all hardware, firmware, and the Exadata Storage Server software by using a separate management network.

Service Availability

- Exadata Cloud Service inherent HA capabilities:
 - Exadata platform full hardware redundancy:
 - Database and storage servers
 - InfiniBand and Ethernet networking
 - Power supplies and PDUs
 - Oracle software HA capabilities:
 - RAC protects against database server failures
 - ASM data mirroring protects against storage failures
 - Exadata Cloud Service used 3-way mirroring (ASM high redundancy)
 - Plus Flashback technologies, Online DDL, In-memory fault-tolerance, and so on...
- Implementation best-practices are derived from thousands of mission-critical deployments worldwide



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Exadata Cloud Service inherits all of the high availability (HA) capabilities that are an integral part of the Exadata platform. Exadata has full hardware redundancy including redundant InfiniBand networking, redundant network ports, redundant Power Distribution Units (PDUs), redundant power supplies, redundant database servers, and redundant storage servers.

Exadata Cloud Service inherits all of the Oracle Database HA capabilities. Oracle RAC protects against database server failures. Automatic Storage Management (ASM) provides data mirroring to protect against disk or storage server failures. In Exadata Cloud Service, high redundancy (3-way mirroring) is used by default to protect data. Oracle Recovery Manager (RMAN) provides extremely fast and efficient backups to disk or cloud storage. Oracle Database Flashback technologies allow user errors to be backed out at the database, table or row level. Online Data Definition Language (DDL) operations enhance availability during data maintenance operations. In-memory fault-tolerance mirrors the contents of the in-memory data store, which enables surviving stores to automatically and transparently satisfy user queries if a database instance fails.

Because of its HA pedigree, Exadata is deployed worldwide at thousands of mission-critical deployments in leading banks, airlines, telecommunications companies, stock exchanges, government agencies and e-commerce sites. The best practices guiding the implementation of Exadata Cloud Service are derived from the experience of those deployments.

Management Responsibilities

Oracle Managed (No Customer Access)	Customer Managed (No Oracle Access)
Initial configuration and installation	Database server DomU - including OS
Exadata Storage Server software and objects	Oracle Database, Grid Infrastructure, ASM
Database server Dom0	Database and OS updates*
InfiniBand switches, HCAs, and partitioning	Database and OS monitoring*
Management and ILOM networks	Database backup and recovery*
All hardware, firmware, and BIOS	
Client access VLANS and IP addresses	* Oracle tooling provided



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Storage Configuration

- Preconfigured ASM Disk Groups:
 - High redundancy disk groups, which consume nearly all of the storage
 - DATA contains Oracle Database data files
 - RECO contains the Fast Recovery Area (FRA)
 - System disk groups, which are comparatively very small
 - DBFS contains shared clusterware files
 - ACFS disk groups contain Oracle Database binaries and patch files
- Configuration options for space allocation:
 - Provision for Local Backups: 40% DATA, 60% RECO
 - No Provision for Local Backups: 80% DATA, 20% RECO

	Quarter Rack	Half Rack	Full Rack
Total Usable Disk Capacity	42 TB	84 TB	168 TB
Disk Group Sizes with Provision for Local Backups	DATA: 16.8 TB RECO: 25.2 TB	DATA: 33.6 TB RECO: 50.4 TB	DATA: 67.2 TB RECO: 100.8 TB
Disk Group Sizes without Provision for Local Backups	DATA: 33.6 TB RECO: 8.4 TB	DATA: 67.2 TB RECO: 16.8 TB	DATA: 134.4 TB RECO: 33.6 TB



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

As part of provisioning each Oracle Exadata Database Machine environment, the storage space inside the Exadata Storage Servers is provisioned for use by Oracle Automatic Storage Management (ASM). By default, the following ASM disk groups are created:

- Two high redundancy disk groups consume nearly all of the available storage space:
 - The DATA disk group is intended for the storage of Oracle Database data files.
 - The RECO disk group is primarily used for storing the Fast Recovery Area (FRA), which is an area of storage where Oracle Database can create and manage various files related to backup and recovery, such as RMAN backups and archived redo log files.
- The system disk groups support various operational purposes.
 - The DBFS disk group is primarily used to store the shared clusterware files (Oracle Cluster Registry and voting disks).
 - The ACFS disk groups are primarily used to store Oracle Database binaries and to facilitate patching.

Compared to the DATA and RECO disk groups, the system disk groups are very small, consuming less than 1% of the total available disk space. The system disk groups are configured to use normal redundancy. You should not store Oracle Database data files or backups inside the system disk groups.

Note that the disk group names contain a short identifier string that is associated with your Exadata Database Machine environment. For example, the identifier could be C2, in which case the DATA disk group would be named DATA_C2, the RECO disk group would be named RECO_C2, and so on.

As an input to the provisioning process, you must decide if you intend to perform backups to the local storage within your Exadata Database Machine. Your backup storage choice profoundly affects how storage space in the Exadata Storage Servers is allocated to the ASM disk groups.

If you choose to provision for local backups, approximately 40% of the available storage space is allocated to the DATA disk group and approximately 60% is allocated to the RECO disk group. If you choose not to provision for local backups, approximately 80% of the available storage space is allocated to the DATA disk group and approximately 20% is allocated to the RECO disk group.

After the Exadata Database Machine is activated, the only way to adjust the storage allocation is by lodging a Service Request with Oracle. For details see My Oracle Support Note 2007530.1.

The table in the slide outlines how the usable storage capacity is allocated to the DATA and RECO disk groups for each configuration option. The usable storage capacity is the storage that available for Oracle Database files after taking into account high-redundancy ASM mirroring (triple mirroring), which is used to provide highly resilient database storage on all Exadata Cloud Service configurations. The usable storage capacity does not factor in the effects of Exadata compression capabilities, which can be used to increase the effective storage capacity. The usable disk capacity does not include the space allocated to the system disk groups.

Storage Management Details

- Storage is preconfigured and allocated to ASM disk groups
- Customers manage database objects inside ASM
- Oracle manages the Exadata storage, including Exadata Storage Server software updates
- Exadata Storage Servers are configured using best practices:
 - One cell disk on each physical storage device
 - One set of grid disks for each ASM disk group
 - Space allocation depends on backup configuration
 - Disk groups for database files use high redundancy
 - Grid disk names are prefixed with the corresponding disk group name
 - Flash cache and flash log are preconfigured
 - Flash cache mode is write-through
 - IORM plan is active, IORM objective is set to balanced
- Customers have no direct access to the Exadata Storage Servers
 - Can request custom configurations for grid disks and IORM



Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

With Exadata Cloud Service, storage is preconfigured and allocated to ASM disk group, so customers must manage database objects inside ASM. This includes management of tablespaces and the data they contain. Customers must also monitor and maintain ASM to ensure continued availability.

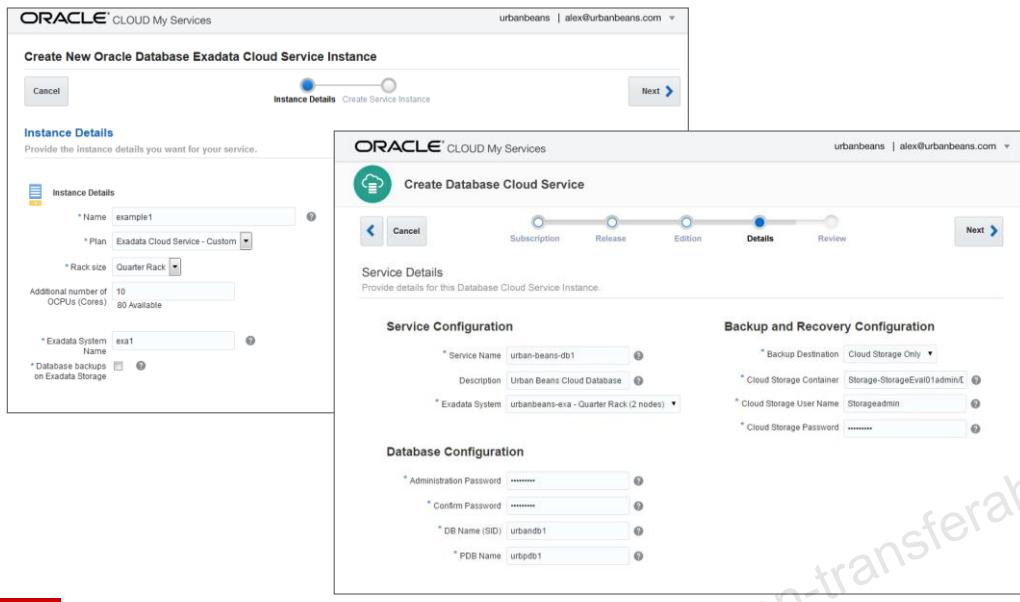
Oracle manages the Exadata Storage Servers, including Exadata Storage Server software updates. The Exadata Storage Servers are configured in accordance with best practices:

- The cell disks, and grid disks to support the DATA, RECO and system disk groups are preconfigured and use standard Exadata default settings.
- The flash cache and flash log are preconfigured. For Exadata Cloud Service, the flash cache is configured in write-back mode by default.
- The I/O Resource Management (IORM) plan is active and the objective is set to balanced.

No direct access is provided to the Exadata Storage Servers. However, you can lodge a Service Request (SR) for:

- Storage reconfiguration to redistribute the space amongst the existing grid disks, or to create additional grid disks.
- IORM configuration to a custom specification.

Simple Web-Based Provisioning



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

Exadata Cloud Service provides simple web-based provisioning interfaces, which you can use to quickly and easily provision an Exadata system and subsequently create their Oracle databases.

The screenshots in the slide show examples of the key screens in the Exadata Cloud Service instance provisioning wizard, and the wizard used to create databases on Exadata Cloud Service. In both cases, only a few key inputs are required and either task can be completed in only a few minutes.

Simple Web-Based Management

The screenshot shows the Oracle Cloud My Services dashboard for the 'urbanbeans' Exadata system. At the top, it displays system statistics: Nodes (2), OCPUs (28), Memory (480 GB), and Storage (42 TB). Below this, the 'Exadata System: urbanbeans-exa' and 'Cluster: urbanbeans-exa' are listed. Under 'Instances', two instances are shown: 'urbandb11' (Public IP: 10.128.13.168) and 'urbandb12' (Public IP: 10.128.13.169). Each instance has a 'Start', 'Stop', and 'Restart' button. The 'Additional Information' section provides detailed deployment details, including the Connect String and Database Configuration. The bottom of the slide features the Oracle logo and a copyright notice: 'Copyright © 2018, Oracle and/or its affiliates. All rights reserved.'

Exadata Cloud Service also provides a set of web-based management interfaces, which you can use to quickly and easily accomplish various management tasks including:

- Database server VM lifecycle activities (start, stop, restart).
- SSH key management (add, update).
- View service instance details.
- View database deployment details.
- Configure an Exadata I/O Resource Management (IORM) inter-database plan.
- Scale within an Exadata system (add or remove database server CPU cores).

The screenshot in the slide show a simple example of the interface that displays database deployment details along with the options to start, stop, or restart one of the database server virtual machines.

REST APIs

- REST APIs provide programmatic management and control:
 - Create or Delete a Database Deployment
 - Stop, Start or Restart a Database Server VM
 - View Details
 - View a Database Deployment
 - View All Database Deployments
 - View Database Servers
 - View the Status of an Operation

```
$ curl --include --request POST --cacert ~/cacert.pem  
--user serviceadmin:Pa55_word  
--header "X-ID-TENANT-NAME:usexample"  
--header "Content-Type:application/json"  
--data '{ \"lifecycleState\" : \"stop\", \"vmName\" : \"node02\" }'  
https://dbaas.oraclecloud.com/paas/service/dbcs/api/v1.1/instances/usexample/db12c
```

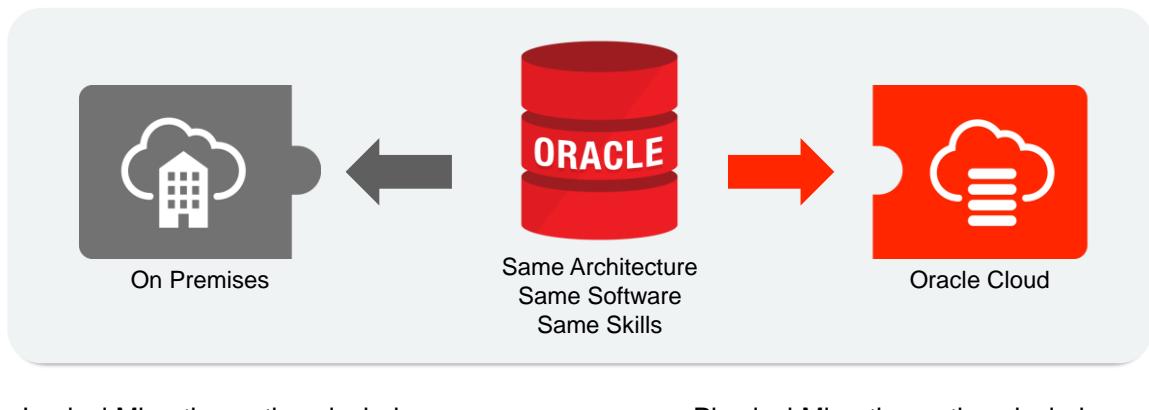


Copyright © 2018, Oracle and/or its affiliates. All rights reserved.

In addition to the web-management interfaces, Exadata Cloud Service provides a series of REST APIs. The Exadata Cloud Service REST APIs enable you to perform various management operations programmatically, such as create and delete database instances, start and stop compute nodes, and view status information. The slide shows an example that uses the cURL utility to invoke the REST API call to stop a database server virtual machine.

See <http://docs.oracle.com/cloud/latest/exadataacs/EXARS/index.html> for details about the Exadata Cloud Service REST APIs.

Migrating to Exadata Cloud Service



- Logical Migration options include:
 - Oracle Data Pump
 - Oracle GoldenGate
 - Physical Migration options include:
 - Recovery from an RMAN backup
 - Transportable Tablespaces
 - Oracle Data Guard

Full compatibility between existing on-premises Oracle databases and Oracle databases on Exadata Cloud Service makes migration easy and low risk. Following established best practices for Oracle Database, two types of migration methodologies are supported:

- Logical Migration allows data reorganization as part of migration. Database solutions that can be used for this purpose include Oracle Data Pump and Oracle GoldenGate.
 - Physical Migration, which involves a byte-to-byte copy of the data, offers the simplest way to migrate databases. Solutions that can be used for this purpose include RMAN backup, Transportable Tablespaces, and Oracle Data Guard. You can also restore from a backup stored on the Oracle Public Cloud through the Oracle Database Backup Service.

Summary

In this lesson, you should have learned how to:

- Describe the architecture and capabilities of Exadata Cloud Service
- Compare and contrast between Exadata Cloud Service and an on-premise Exadata implementation



ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.