



Hardware and Software
Engineered to Work Together



Oracle University and its partners. You are not a Valid Partner user only

Oracle Database 12c: RAC Administration

Student Guide
D81250GC10
Edition 1.0 | February 2014 | D85494

Learn more from Oracle University at oracle.com/education/

Authors

Jim Womack
Dominique Jeunot

**Technical Contributors
and Reviewers**

Allan Graves
Gerlinde Frenzen
Branislav Valny
Herbert Bradbury
Ira Singer
Harald Van Breederode
Joel Goodman
Sean Kim
Andy Fortunak
Al Flournoy
Markus Michalewicz
Maria Billings
Mark Scardina
Ron Soltani

Editors

Anwesha Ray
Raj Kumar

Graphic Designer

Divya Thallap

Publishers

Syed Imtiaz Ali
Pavithran Adka

Copyright © 2014, 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
- What Is a Cluster? 1-3
- What Is Clusterware? 1-4
- Oracle Clusterware 1-5
- Oracle Flex Clusters 1-6
- Flex Cluster Scalability 1-7
- Clusterware Architecture and Cluster Services 1-8
- Goals for Oracle Clusterware 1-9
- Oracle Clusterware Networking 1-10
- Oracle Clusterware Initialization 1-12
- GPnP Architecture: Overview 1-13
- How GPnP Works: Cluster Node Startup 1-15
- Grid Naming Service (GNS) 1-16
- Single-Client Access Name 1-17
- Client Database Connections 1-19
- What Is Oracle ASM? 1-20
- ASM CloudFS and ACFS 1-21
- Oracle Flex ASM 1-23
- ASM Features and Benefits 1-24
- Quiz 1-25
- Summary 1-27
- Practice 1: Overview 1-28

2 RAC Databases Overview & Architecture

- Objectives 2-2
- Overview of Oracle RAC 2-3
- Typical Oracle RAC Architecture 2-5
- RAC One Node Single-Instance High Availability 2-6
- Oracle RAC One Node 2-7
- Oracle RAC One Node and Oracle Clusterware 2-8
- Cluster-Aware Storage Solutions 2-9
- Oracle Cluster File System 2-10
- Oracle RAC and Network Connectivity 2-11
- Benefits of Using RAC 2-12

Clusters and Scalability	2-13
Levels of Scalability	2-14
Scaleup and Speedup	2-15
Speedup/Scaleup and Workloads	2-16
I/O Throughput Balanced: Example	2-17
Necessity of Global Resources	2-18
Additional Memory Requirement for RAC	2-19
Parallel Execution with RAC	2-20
Summary	2-22
3 Installing and Configuring Oracle RAC	
Objectives	3-2
Installing the Oracle Database Software	3-3
Creating the Cluster Database	3-8
Database Type Selection	3-9
Database Identification	3-11
Cluster Database Management Options	3-12
Passwords for Database Schema Owners	3-13
Storage Locations	3-14
Database Content	3-15
Initialization Parameters	3-16
Create the Database	3-17
Monitoring Progress	3-18
Postinstallation Tasks	3-19
Background Processes Specific to Oracle RAC	3-20
Single Instance-to-RAC Conversion	3-22
Considerations for Converting Single-Instance Databases to Oracle RAC	3-23
Single-Instance Conversion Using DBCA	3-24
Conversion Steps	3-25
Single-Instance Conversion Using rconfig	3-28
Quiz	3-29
Summary	3-31
Practice 3: Overview	3-32
4 Oracle RAC Administration	
Objectives	4-2
Cluster Database Home Page	4-3
Cluster Database Instance Home Page	4-5
Cluster Home Page	4-6
Topology Viewer	4-7
Enterprise Manager Alerts and RAC	4-8

Enterprise Manager Metrics and RAC	4-9
Enterprise Manager Blackouts and RAC	4-10
Redo Log Files and RAC	4-11
Automatic Undo Management and RAC	4-13
Starting and Stopping RAC Instances	4-14
Starting and Stopping RAC Instances with srvctl	4-15
Starting and Stopping RAC Instances with SQL*Plus	4-16
Starting and Stopping Pluggable Databases in Oracle RAC	4-17
Switch Between Automatic and Manual Policies	4-19
RAC Initialization Parameter Files	4-21
SPFILE Parameter Values and RAC	4-22
Parameter File Search Order in Oracle RAC	4-23
EM and SPFILE Parameter Values	4-24
RAC Initialization Parameters	4-26
Parameters That Require Identical Settings	4-28
Parameters That Require Unique Settings	4-29
Quiescing RAC Databases	4-31
Terminating Sessions on a Specific Instance	4-32
How SQL*Plus Commands Affect Instances	4-33
Transparent Data Encryption and Keystores in RAC	4-34
Quiz	4-36
Summary	4-38
Practice 4: Overview	4-39

5 Managing Backup and Recovery for RAC

Objectives	5-2
RAC and Instance Recovery	5-3
Instance Recovery and Database Availability	5-5
Instance Recovery and RAC	5-6
Protecting Against Media Failure	5-8
Media Recovery in Oracle RAC	5-9
Parallel Recovery in RAC	5-10
RAC and the Fast Recovery Area	5-11
RAC Backup and Recovery Using EM	5-12
Configuring RAC Recovery Settings with EM	5-13
Archived Redo File Conventions in RAC	5-14
Configuring RAC Backup Settings with EM	5-15
Oracle Recovery Manager	5-16
Configuring RMAN Snapshot Control File Location	5-17
Configuring Control File and SPFILE Autobackup	5-18
Crosschecking on Multiple RAC Clusters Nodes	5-19

Channel Connections to Cluster Instances 5-20
RMAN Channel Support for the Grid 5-21
RMAN Default Autolocation 5-22
Distribution of Backups 5-23
Managing Archived Redo Logs Using RMAN 5-24
Noncluster File System Local Archiving Scheme 5-25
Configuring Non-Cluster, Local Archiving 5-26
ASM and Cluster File System Archiving Scheme 5-27
Configuring the CFS Archiving Scheme 5-28
Restoring and Recovering 5-29
Quiz 5-30
Summary 5-32
Practice 5: Overview 5-33

6. Global Resource Management Concepts

Objectives 6-2
Need for Global Concurrency Control 6-3
Global Resource Directory (GRD) 6-4
Global Resource Management 6-5
Global Resource Remastering 6-6
Global Resource Recovery 6-7
Global Resource Background Processes 6-8
Global Resource Access Coordination 6-10
Global Enqueues 6-11
Instance Locks 6-12
Global Cache Management: Overview 6-13
Global Cache Management Components 6-14
Global Cache Buffer States 6-15
Global Cache Management Scenarios for Single Block Reads 6-16
Global Cache Scenarios: Overview 6-17
Scenario 1: Read from Disk 6-18
Scenario 2: Read-Write Cache Fusion 6-22
Scenario 3: Write-Write Cache Fusion 6-26
Scenario 4: Write-Read Cache Fusion 6-30
Global Cache Management Scenarios for Multi-Block Reads 6-34
Useful Global Resource Management Views 6-35
Quiz 6-36
Summary 6-37

7 RAC Database Monitoring and Tuning
Objectives 7-2
CPU and Wait Time Tuning Dimensions 7-3
RAC-Specific Tuning 7-4
Analyzing Cache Fusion Impact in RAC 7-5
Typical Latencies for RAC Operations 7-6
Wait Events for RAC 7-7
Wait Event Views 7-8
Global Cache Wait Events: Overview 7-9
Global Enqueue Waits 7-11
Session and System Statistics 7-12
Most Common RAC Tuning Tips 7-13
Index Block Contention: Considerations 7-15
Oracle Sequences and Index Contention 7-16
Undo Block Considerations 7-17
High-Water Mark Considerations 7-18
Concurrent Cross-Instance Calls: Considerations 7-19
Monitoring RAC Database and Cluster Performance 7-20
Cluster Database Performance Page 7-21
Determining Cluster Host Load Average 7-22
Determining Global Cache Block Access Latency 7-23
Determining Average Active Sessions 7-24
Determining Database Throughput 7-25
Accessing the Cluster Cache Coherency Page 7-27
Viewing the Database Locks Page 7-29
AWR Snapshots in RAC 7-30
AWR Reports and RAC: Overview 7-31
Active Session History Reports for RAC 7-33
Automatic Database Diagnostic Monitor for RAC 7-35
What Does ADDM Diagnose for RAC? 7-37
EM Support for ADDM for RAC 7-38
Quiz 7-39
Summary 7-41
Practice 7: Overview 7-42

8 Managing High Availability of Services

Objectives 8-2
Oracle Services 8-3
Service Usage in an Oracle RAC Database 8-4
Parallel Operations and Services 8-5
Service Characteristics 8-6

Default Service Connections	8-8
Restricted Service Registration	8-9
Creating Service with Enterprise Manager	8-10
Creating Services with SRVCTL	8-11
Managing Services with Enterprise Manager	8-12
Managing Services with EM	8-13
Managing Services with srvctl	8-14
Using Services with Client Applications	8-15
Services and Connection Load Balancing	8-16
Services and Transparent Application Failover	8-17
Using Services with the Resource Manager	8-18
Services and Resource Manager with EM	8-19
Using Services with the Scheduler	8-20
Services and the Scheduler with EM	8-21
Using Distributed Transactions with RAC	8-23
Distributed Transactions and Services	8-24
Service Thresholds and Alerts	8-26
Services and Thresholds Alerts: Example	8-27
Service Aggregation and Tracing	8-28
Top Services Performance Page	8-29
Service Aggregation Configuration	8-30
Service, Module, and Action Monitoring	8-31
Service Performance Views	8-33
Quiz	8-34
Summary	8-36
Practice 8: Overview	8-37

9 High Availability for Connections and Applications

Objectives	9-3
Types of Workload Distribution	9-4
Client-Side Connect-Time Load Balancing	9-5
Fast Application Notification (FAN): Overview	9-6
Fast Application Notification: Benefits	9-7
Implementing FAN Events	9-8
FAN and Oracle Integrated Clients	9-9
FAN-Supported Event Types	9-11
FAN Event Reasons	9-12
FAN Event Status	9-13
FAN Event Format	9-14
Load Balancing Advisory: FAN Event	9-15
Server-Side Callouts Implementation	9-16

Server-Side Callout Parse: Example	9-17
Server-Side Callout Filter: Example	9-18
Server-Side ONS	9-19
Optionally Configuring the Client-Side ONS	9-20
UCP JDBC Fast Connection Failover: Overview	9-21
JDBC/ODP.NET FCF Benefits	9-22
Load Balancing Advisory	9-23
UCP JDBC/ODP.NET Runtime Connection Load Balancing: Overview	9-24
Connection Load Balancing in RAC	9-25
Monitoring LBA FAN Events	9-26
Transparent Application Failover: Overview	9-27
TAF Basic Configuration on Server-Side: Example	9-28
TAF Basic Configuration on a Client-Side: Example	9-29
TAF Preconnect Configuration: Example	9-30
TAF Verification	9-31
FAN Connection Pools and TAF Considerations	9-32
Quiz	9-33
Summary	9-34
Objectives	9-36
The Situation Prior to Application Continuity	9-37
Introducing Transaction Guard and Application Continuity	9-38
RAC and Application Continuity	9-39
Key Concepts for Application Continuity	9-40
Workflow of a Database Request	9-42
What Is Transaction Guard?	9-43
How Transaction Guard Works	9-44
Using Transaction Guard	9-45
Benefits of Transaction Guard	9-46
What Is Application Continuity?	9-47
How Does Application Continuity Work?	9-48
Using Application Continuity	9-49
Application Continuity Processing Phases	9-50
Restrictions	9-52
Potential Side Effects	9-53
Actions That Disable Application Continuity	9-54
When Is Application Continuity Transparent?	9-55
Benefits of Application Continuity	9-56
Application Assessment for Using Application Continuity	9-57
Handling Request Boundaries	9-59
Disabling Replay by Using the disableReplay API	9-60
Connection Initialization Options	9-61

Mutable Objects and Application Continuity	9-63
Keeping Mutable Objects for Replay	9-64
Configuring the JDBC Replay Data Source	9-65
Creating Services for Application Continuity	9-66
Creating Services for Transaction Guard	9-68
Resource Requirements for Application Continuity	9-69
Quiz	9-70
Summary	9-73
Practice 9 Overview: Using Application Continuity	9-74

10 Upgrading and Patching Oracle RAC

Objectives	10-2
Patch and Patch Set Overview	10-3
Types of Patches	10-4
Configuring the Software Library	10-5
Obtaining Oracle RAC Patches	10-6
Downloading Patches	10-9
Reduced Down-Time Patching for Cluster Environments	10-10
Rolling Patches	10-11
Out-of-Place Database Upgrades	10-12
Out-of-Place Database Upgrade with OUI	10-13
OPatch: Overview	10-14
OPatch: General Usage	10-15
Before Patching with OPatch	10-16
OPatch Log and Trace Files	10-20
Queryable Patch Inventory	10-21
Quiz	10-22
Summary	10-23

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	11-6
Oracle RAC One Node Online Migration	11-7
Online Migration Considerations	11-8
Performing an Online Migration	11-9
Online Migration Illustration	11-10
Online Maintenance: Rolling Patches	11-13
Adding an Oracle RAC One Node Database to an Existing Cluster	11-15
Converting a RAC One Node Database to RAC	11-16

Converting a Single Instance Database to RAC One Node	11-18
Converting a RAC Database to RAC One Node	11-19
Quiz	11-20
Summary	11-22
Practice 11: Overview	11-23

12 Quality of Service Management

Objectives	12-2
QoS Management Background	12-3
QoS Management Overview	12-4
QoS Management and Exadata Database Machine	12-5
QoS Management Focus	12-6
QoS Management Benefits	12-7
QoS Management Functional Overview	12-9
QoS Management Policy Sets	12-11
Server Pools	12-12
Performance Classes	12-14
Classification and Tagging	12-16
Performance Policies	12-17
Performance Class Ranks	12-18
Performance Objectives	12-19
Performance Satisfaction Metrics	12-20
Server Pool Directive Overrides	12-21
Overview of Metrics	12-22
QoS Management Architecture	12-24
QoS Management Recommendations	12-25
Implementing Recommendations	12-27
Quiz	12-29
Summary	12-31
Lesson 12 Demonstrations	12-32

13 Multitenant Architecture and RAC

Objectives	13-2
Non-CDB Architecture	13-3
Multitenant Architecture: Benefits	13-4
CDB in a Non-RAC Environment	13-6
Containers	13-7
Terminology	13-8
Data Dictionary Views	13-9
Connection to a non-RAC CDB	13-10
Switching Connection	13-12

Oracle RAC and Multitenant Configuration	13-13
Oracle RAC and Multitenant Architecture	13-14
Creating a RAC CDB	13-15
Creating a RAC CDB Including PDBs	13-16
Hosting a RAC CDB in Server Pools	13-17
After CDB Creation	13-18
Connecting Using CDB/PDB Services	13-19
Opening a PDB in a RAC CDB	13-20
Closing a PDB in a RAC CDB	13-22
Types of Services	13-23
Managing Services	13-24
Affinitizing PDB Services to Server Pools	13-25
Adding a PDB to a RAC CDB	13-26
Dropping a PDB from a RAC CDB	13-27
Quiz	13-28
Summary	13-29
Practice 13: Overview	13-30

1

Grid Infrastructure Overview



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

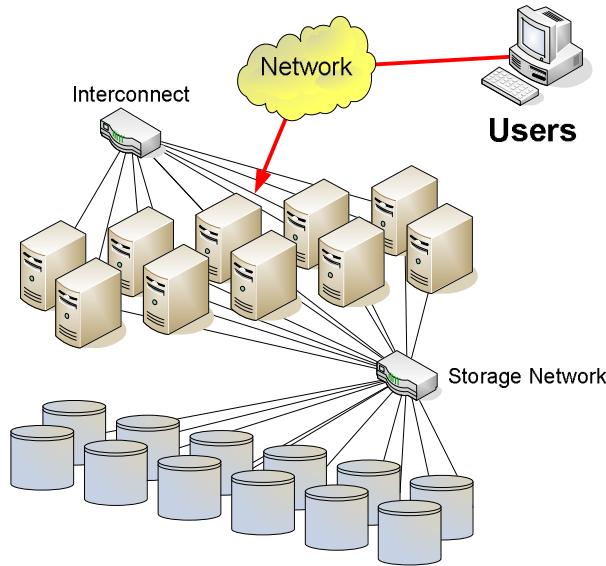
- Explain the principles and purposes of clusters
- Describe the Oracle Clusterware architecture
- Describe how Grid Plug and Play affects Clusterware



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

What Is a 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 © 2014, 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.

What Is Clusterware?

- Clusterware is 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 © 2014, Oracle and/or its affiliates. All rights reserved.

Clusterware is a term used to describe 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.

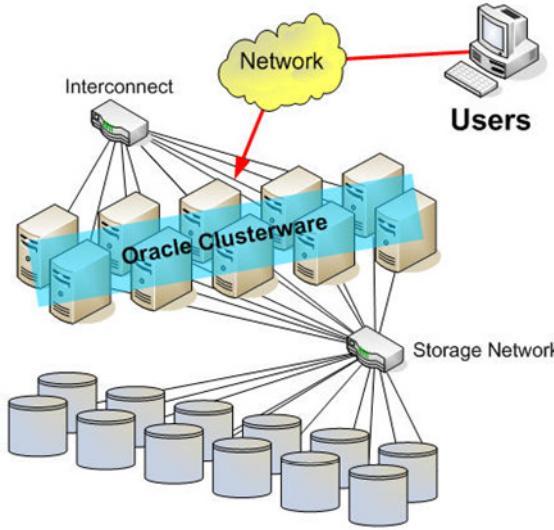
Typically, clusterware includes capabilities that:

- 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 Cloud File System
- A foundation for Oracle Real Application Clusters (RAC)
- A generalized cluster infrastructure for all kinds of applications



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

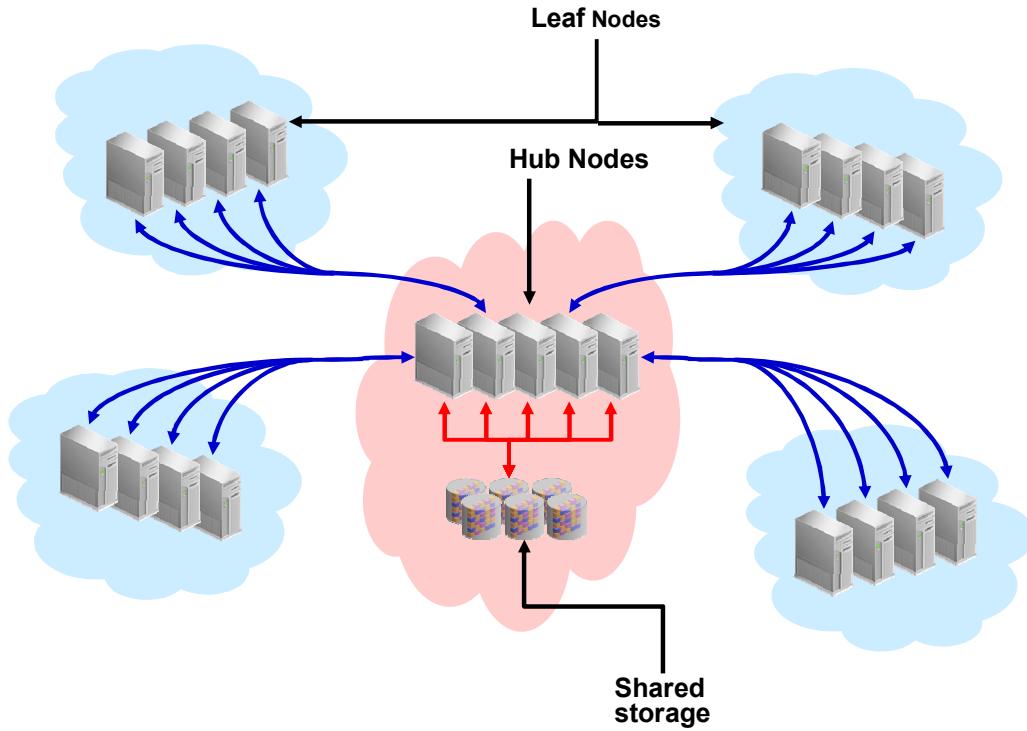
Oracle Clusterware is a key part of Oracle Grid Infrastructure, which also includes Automatic Storage Management (ASM) and the Oracle Cloud File System. Oracle Clusterware can use ASM for all the shared files required by the cluster. Oracle Clusterware is also a foundation for the ASM Cluster File System, 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 © 2014, Oracle and/or its affiliates. All rights reserved.

Previous releases of Oracle Clusterware have been used to build large production clusters containing between 32 and 64 nodes. A few clusters larger than 100 nodes have been successfully deployed. With Oracle Clusterware 12c, a new set of features enables Flex Clusters. In this release, Flex Clusters are designed to scale well up to 2000 nodes.

In release 12.1, you can use Flex Clusters to:

- Manage large pools of application resources with high-availability and failover protection.
- Efficiently support multiple highly available databases and applications running in a single cluster.

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

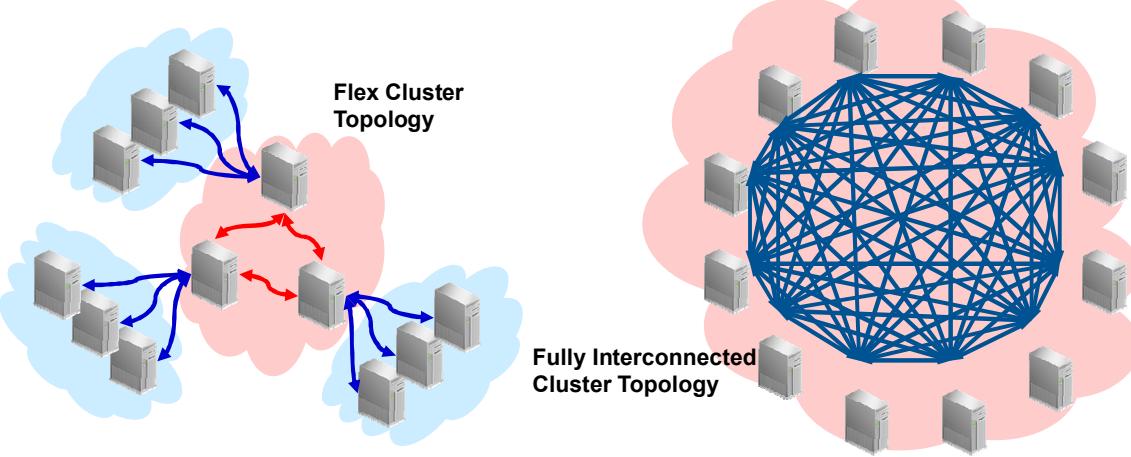
The core of a Flex Cluster is a group of Hub Nodes. The group is essentially the same as a release 11.2 cluster, and can scale up to the size of an existing release 11.2 cluster. There must be one, and only one, group of Hub Nodes in a Flex Cluster deployment, and like a release 11.2 cluster, each Hub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

Flex Cluster Scalability

The Flex Cluster hub-and-spoke topology segments the cluster into more manageable groups of nodes.

- Only the Hub Nodes require direct access to the OCR and voting disks.
- Fewer interactions are required between nodes.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

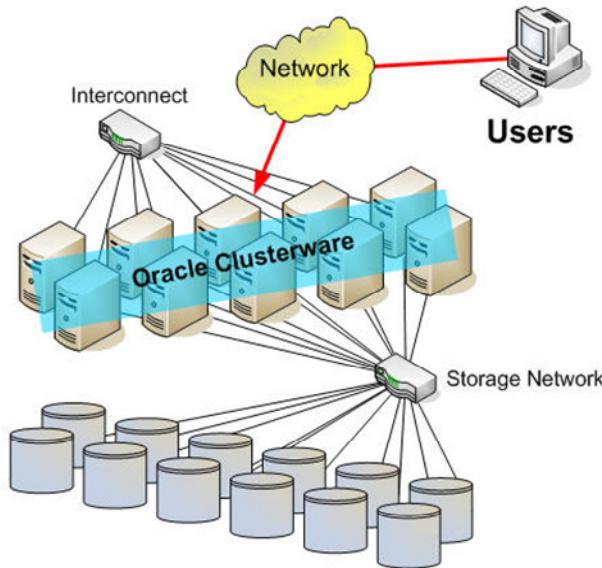
The two-layered hub-and-spoke topology is the key architectural feature that allows a Flex Cluster to scale well beyond previous limits. In essence, the hub-and-spoke topology segments the cluster into groups of nodes, and each group contains a manageable number of nodes. This segmentation has two fundamental impacts:

First, by limiting the size of the hub, contention for key clusterware resources, such as the Oracle Cluster Registry (OCR) and voting disks, does not increase significantly due to the addition of the Leaf Nodes. This is important because contention for the voting disks can lead to nodes being evicted from a cluster.

Second, fewer network interactions are required between nodes in the cluster. Consequently, there is less administrative network traffic, such as heartbeats, exchanged between the nodes. This is illustrated in the diagram on the slide. On the left side, the 12-node Flex Cluster contains 12 interaction paths. On the right side, the fully interconnected 12-node cluster contains 66 possible interaction paths. For a 1000-node cluster, the difference would be far more noticeable. Assuming 40 Hub Nodes, with 24 Leaf Nodes per Hub Node, a Flex Cluster contains 1740 possible interaction paths. In comparison, a 1000-node fully interconnected cluster contains 499500 interaction paths.

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

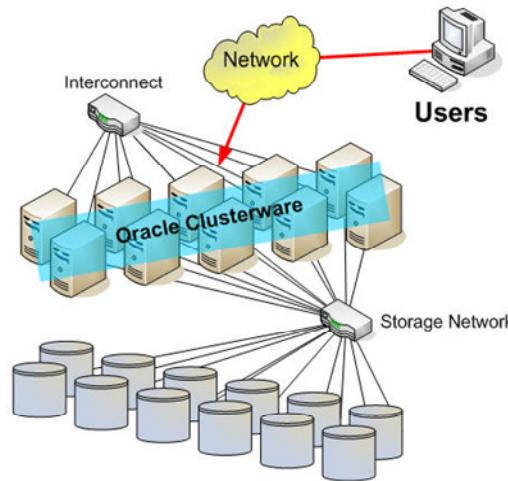
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 publishes 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)

Goals for 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 © 2014, Oracle and/or its affiliates. All rights reserved.

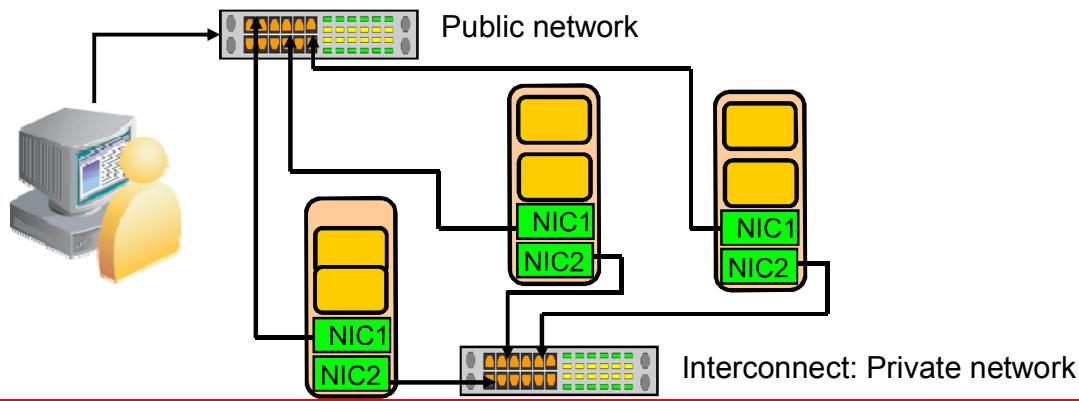
Oracle Clusterware has become the required clusterware for Oracle Real Application Clusters (RAC). Oracle Database 12c builds on the tight integration between Oracle Clusterware and RAC by extending the integration with Automatic Storage Management (ASM). The result is that now all the shared data in your cluster can be managed 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).



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

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 new 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 OL5, Oracle Clusterware installation modifies `/etc/inittab` to restart `ohasd` in the event of a crash.

```
# cat /etc/inittab
..
h1:35:respawn:/etc/init.d/init.ohasd run >/dev/null 2>&1 </dev/null
```

- 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

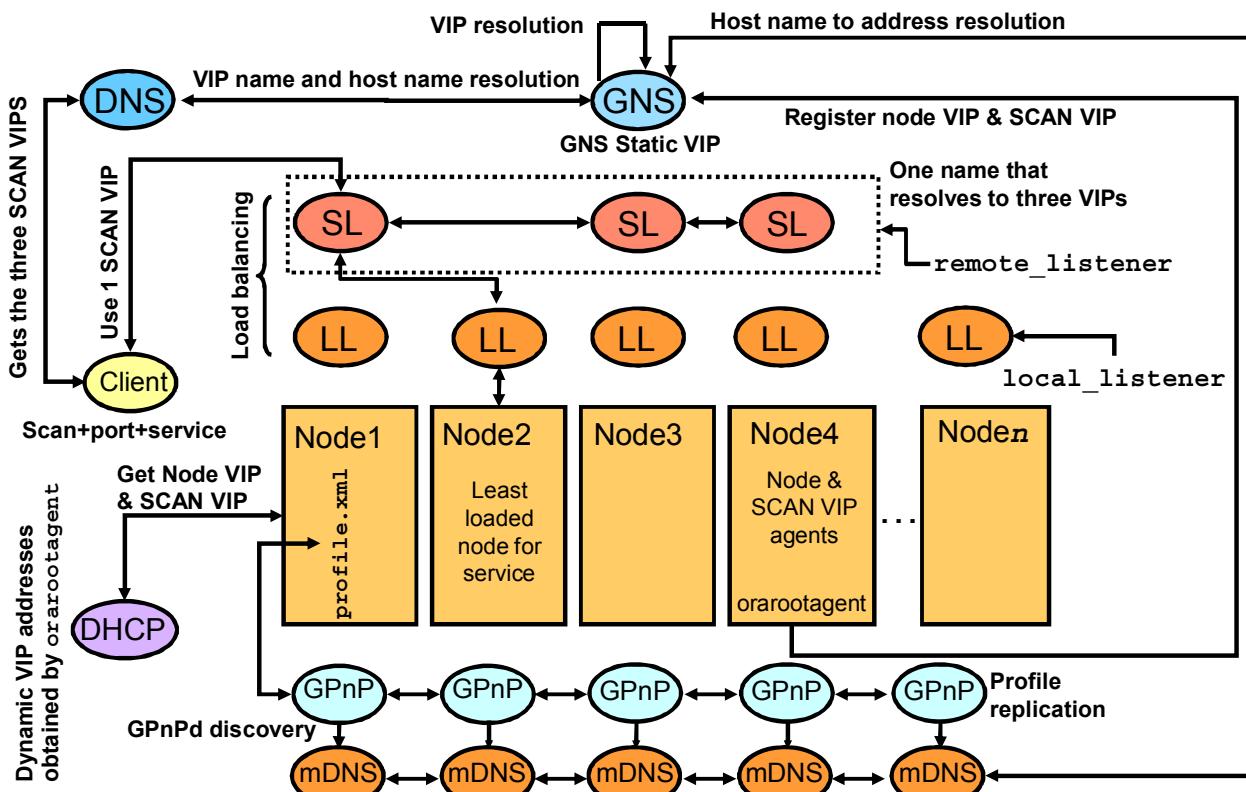
start on runlevel [35]
stop on runlevel [!35]
respawn
exec /etc/init.d/init.ohasd run >/dev/null 2>&1 </dev/null
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

On OL5 systems, Clusterware startup and crash behavior is handled by `init` based on an entry in the `/etc/inittab` file. On OL6 systems, Clusterware startup and crash behavior is still handled by `init`, but the `/etc/inittab` only defines the default runlevel for that node. Service startup is controlled by the Upstart boot service based on service definition files located in the `/etc/init` directory. The primary advantage of Upstart over the traditional System V init method is boot speed. Unlike System V init, which starts services serially, Upstart is event-driven. Services are started only when they are needed.

GPnP Architecture: Overview



ORACLE

Copyright © 2014, 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 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.

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 © 2014, 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, then disable any third-party mDNS daemons on your system.



Copyright © 2014, 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 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 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 © 2014, 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.

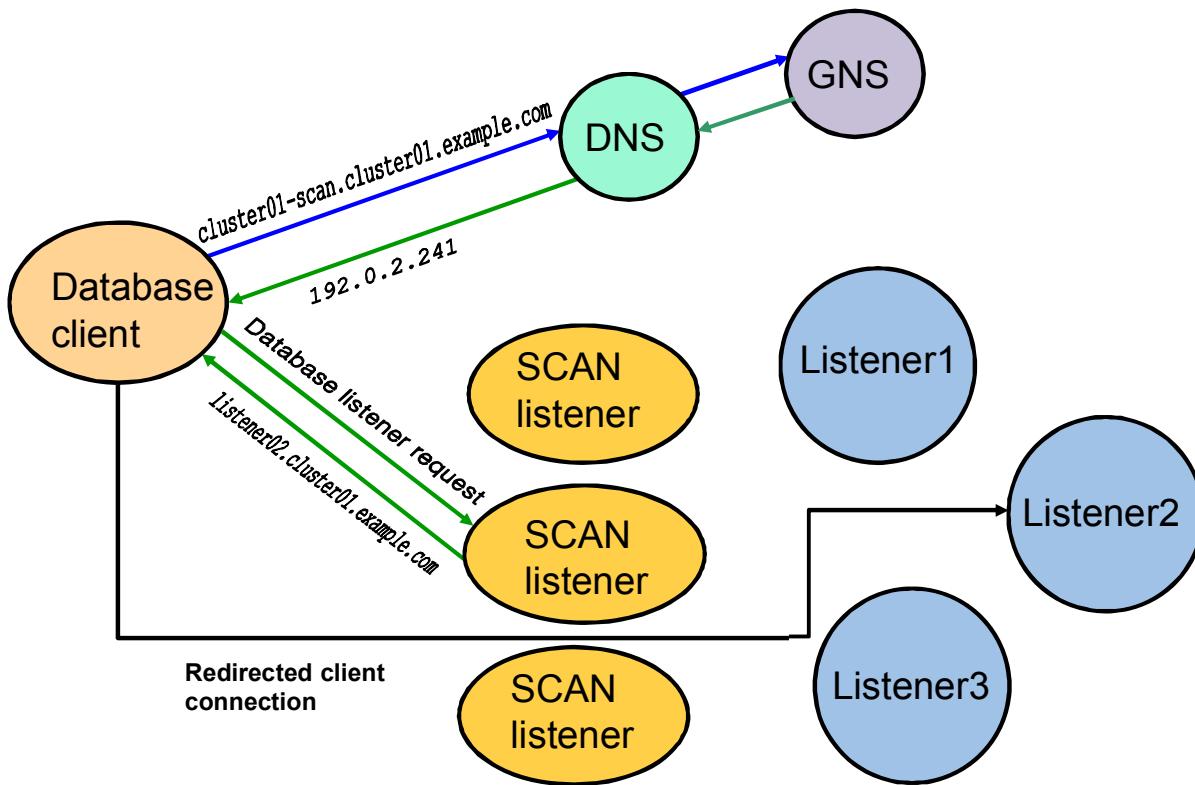
During installation, listeners are created on each node for the SCAN IP addresses. Oracle Clusterware routes application requests to the cluster SCAN to the least loaded instance providing the service.

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

Client Database Connections



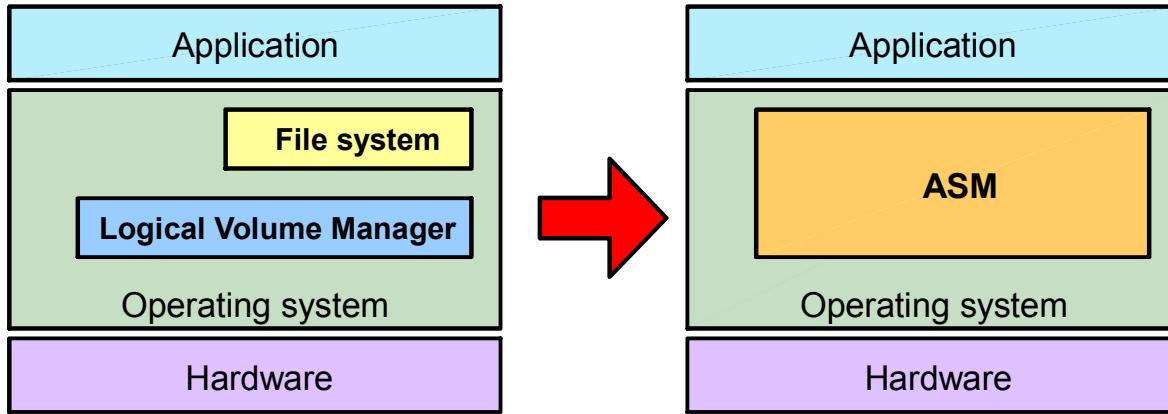
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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.

What Is Oracle ASM?



ORACLE

Copyright © 2014, 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. Oracle ASM is Oracle's recommended storage management solution that provides an alternative to conventional volume managers, file systems, and raw devices..

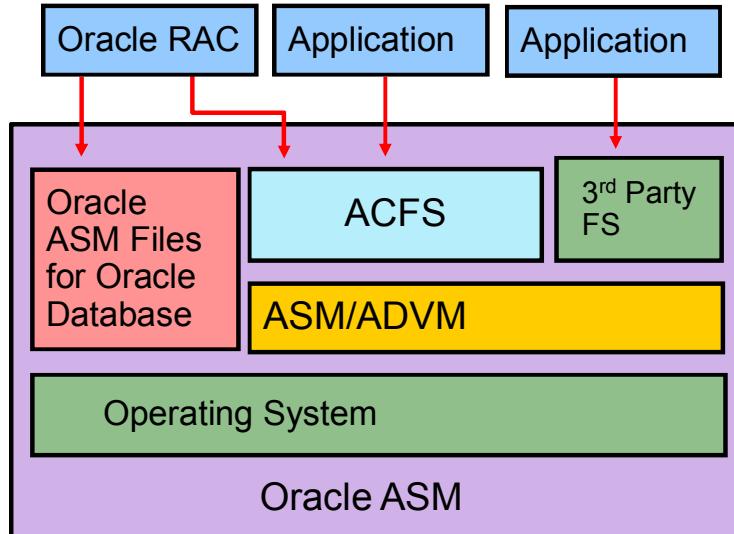
Combining volume management functions with a file system allows a level of integration and efficiency that would not otherwise be possible. For example, ASM is able to avoid the overhead associated with a conventional file system and achieve native raw disk performance for Oracle data files and other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

ASM CloudFS and ACFS

- ASM manages Oracle database files.
- ACFS manages other files.
- Spreads data across disks to balance load
- Provides integrated mirroring across disks



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Cloud File System (Oracle CloudFS) is designed to help organizations deploy their applications, databases, and storage in private clouds. It delivers a cloud infrastructure that provides network access, rapid elasticity, and provisioning for pooled storage resources that are the key requirements for cloud computing. Customers can use Oracle CloudFS to manage and store all database file types, including general purpose files. Oracle CloudFS includes Oracle Automatic Storage Management Cluster File System (Oracle ACFS) and Oracle ASM Dynamic Volume Manager (Oracle ADVM).

Oracle Automatic Storage Management Cluster File System (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 above. 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 © 2014, 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 © 2014, 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. ASM 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.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: c

Statement c is not correct. The GNS VIP address must be statically defined.

Quiz

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

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

This statement is false. Actually, each cluster node's public Ethernet adapter should support TCP/IP. The private adapter should support UDP or RDS.

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 © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 1: Overview

This practice covers installing Oracle Grid Infrastructure.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

2

RAC Databases Overview & Architecture

ORACLE

Copyright © 2014, 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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Overview of Oracle RAC

- 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 © 2014, 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.

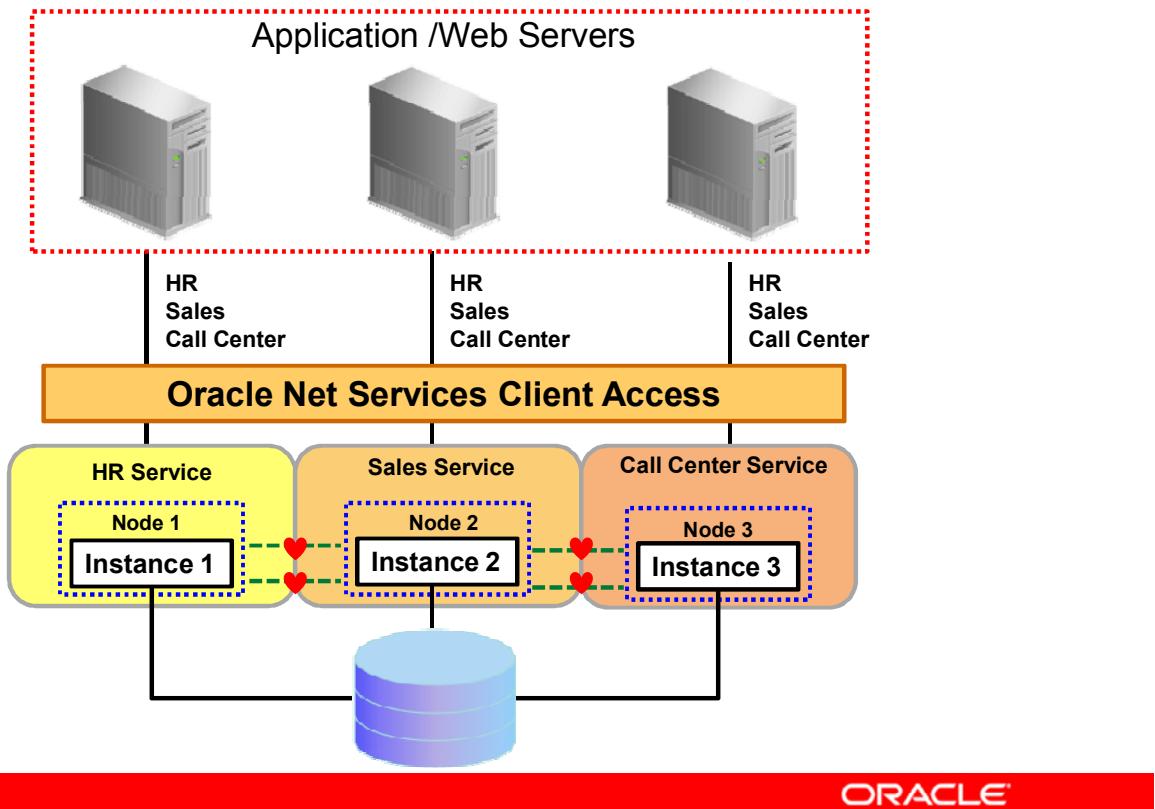
Noncluster Oracle databases have a one-to-one relationship between the Oracle database and the instance. Oracle RAC environments, however, have a one-to-many relationship between the database and instances. An Oracle RAC database can have up to 100 instances, all of which access one database. All database instances must use the same interconnect, which can also be used by Oracle Clusterware.

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

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.

RAC One Node Single-Instance High Availability

- The Oracle RAC One Node option is a single instance of Oracle RAC running on one node in a cluster.
- This option adds to the flexibility that Oracle offers for consolidation.
- Many databases can be consolidated into a single cluster with minimal overhead while providing:
 - High-availability benefits of failure protection
 - Online rolling patch application
 - Rolling upgrades for the operating system and Oracle Clusterware.
- Oracle RAC One Node is supported on all platforms on which Oracle RAC is certified.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle RAC One Node is an option to Oracle Database Enterprise Edition available since Oracle Database 11g release 2 (11.2). Oracle RAC One Node is a single instance of Oracle RAC running on one node in a cluster. This option adds to the flexibility that Oracle offers for consolidation. Many databases can be consolidated into a single cluster with minimal overhead while providing the high availability benefits of failure protection, online rolling patch application, as well as rolling upgrades for the operating system and Oracle Clusterware.

Oracle RAC One Node is supported on all platforms on which Oracle RAC is certified. Oracle RAC One Node is certified on Oracle Virtual Machine (Oracle VM). Oracle RAC One Node provides the following benefits:

- Always available single-instance database services
- Built-in cluster failover for high availability
- Live migration of instances across servers
- Online rolling patches and rolling upgrades for single-instance databases
- Online upgrade from single-instance to multi-instance Oracle RAC
- Better consolidation for database servers
- Enhanced server virtualization
- Lower-cost development and test platform for full 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 © 2014, Oracle and/or its affiliates. All rights reserved.

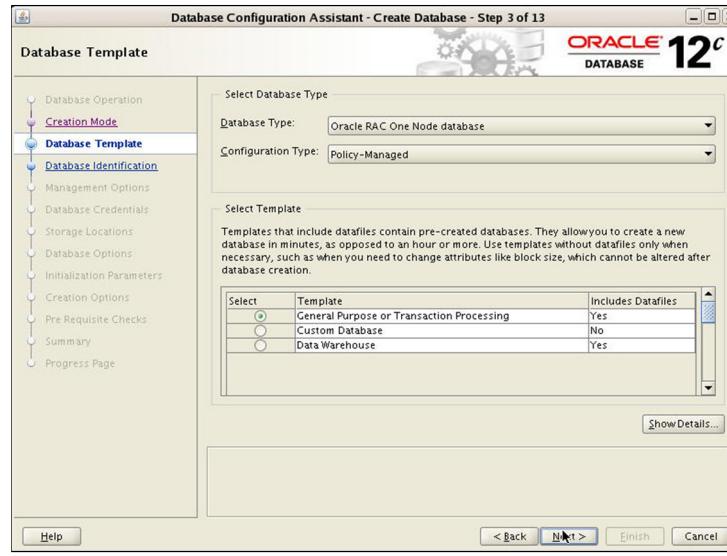
Using Oracle RAC One Node online database relocation, you can relocate the Oracle RAC One Node instance to another server, if the current server is running short on resources or requires maintenance operations, such as operating system patches. You can use the same technique to relocate Oracle RAC One 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.

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, in order 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.



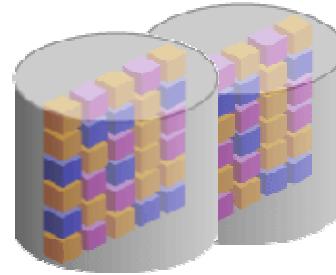
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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.

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.
- Oracle Database provides the following file storage options for Oracle RAC:
 - Oracle Automatic Storage Management
 - Oracle Cluster File System and OCFS2 (Linux)
 - Certified cluster file system or cluster-aware volume manager
 - Certified NFS file servers



ORACLE

Copyright © 2014, 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 the following file storage options for Oracle RAC:

- Oracle Automatic Storage Management (Oracle ASM). Oracle recommends this solution to manage your storage.
- A certified cluster file system, including Oracle OCFS2 (Linux only) and IBM GPFS (IBM AIX only).
- Certified network file system (NFS) file servers

Oracle Cluster File System

- It is a shared disk cluster file system for Linux (OCFS2).
- It provides open solution on the operating system side
- OCFS2 can be downloaded from OTN:
 - <http://oss.oracle.com/projects/ocfs2/> (Linux)
- OCFS2 1.6 is the latest version and is now available with the Oracle Linux 5.
- With this release, there are three supported releases of the file system:
 - OCFS2 1.2
 - OCFS2 1.4
 - OCFS2 1.6



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

OCFS2 is a general-purpose shared-disk cluster file system for Linux capable of providing both high performance and high availability. As it provides local file system semantics, it can be used with almost all applications. Cluster-aware applications can make use of cache-coherent parallel I/Os from multiple nodes to scale out applications easily. Other applications can make use of the file system facilities to fail-over running application in the event of a node failure.

The file system is currently being used in virtualization (Oracle VM) in both the management domain, to host virtual machine images, and in the guest domain, to allow Linux guests to share a file system. It is also being used in database clusters (Oracle RAC), middleware clusters (Oracle E-Business Suite), appliances (SAP's Business Intelligence Accelerator), etc.

OCFS2 1.6 is the latest version and is now available with the Oracle Linux 5. It is bundled with Oracle's Unbreakable Enterprise Kernel. With this release, there are three supported releases of the file system, viz., OCFS2 1.2, OCFS2 1.4 and OCFS2 1.6.

OCFS2 1.2 is available with Oracle Linux 4, OCFS2 1.4 with Oracle Linux 5 (Default Kernel), and OCFS2 1.6 with Oracle Linux 5 (Unbreakable Enterprise Kernel). OCFS2 1.6 is fully compatible with OCFS2 1.4. Users can upgrade the cluster to the new release one-node-at-a-time (rolling upgrade).

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 © 2014, 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). Network attached storage can be typical NAS devices, such as NFS filers, or can be storage that is connected 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 © 2014, Oracle and/or its affiliates. All rights reserved.

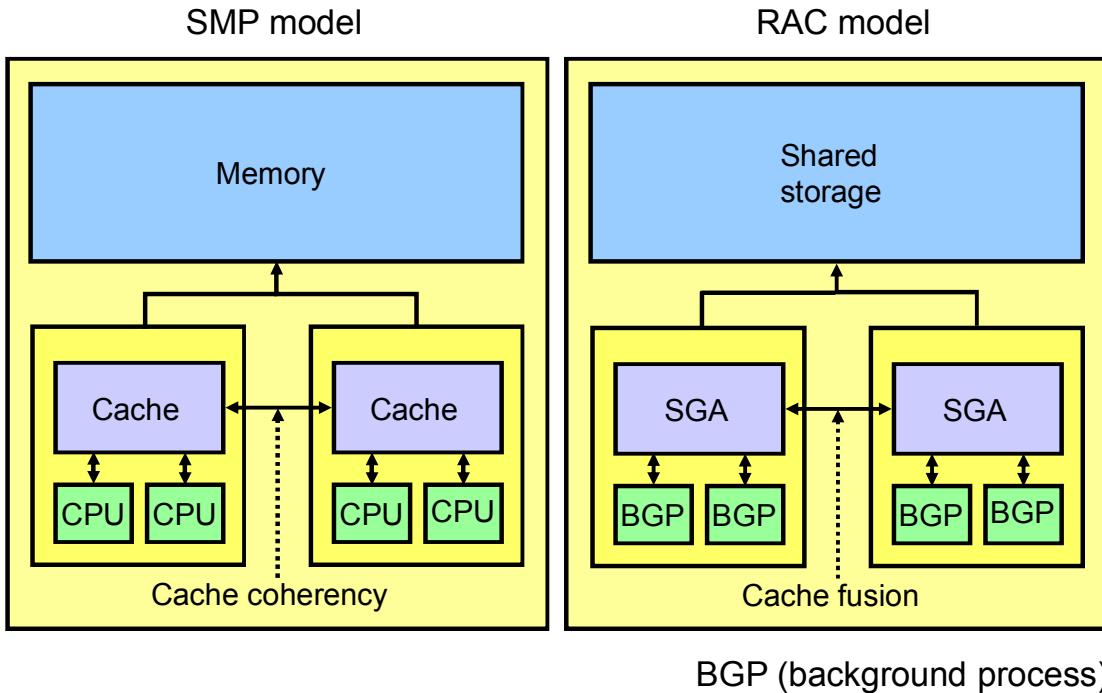
Oracle Real Application Clusters (RAC) enables high utilization of a cluster of standard, low-cost modular servers such as blades.

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 © 2014, 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 or 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 © 2014, Oracle and/or its affiliates. All rights reserved.

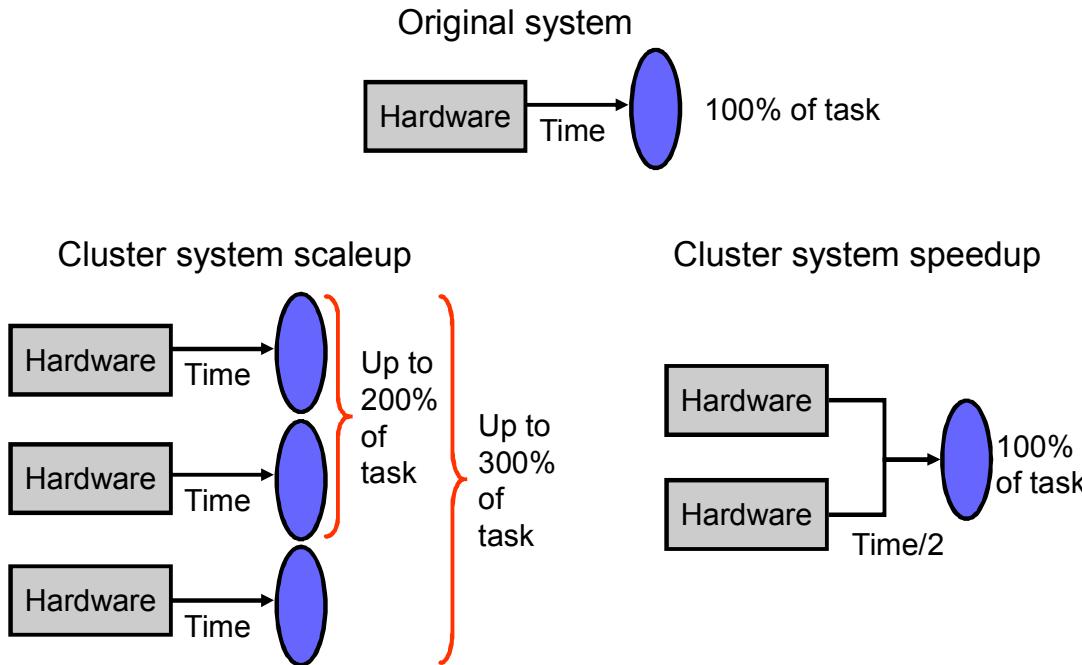
Successful implementation of cluster databases requires optimal scalability on four levels:

- **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 © 2014, 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 ten minutes to process, and running in parallel reduces the time to five 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 © 2014, Oracle and/or its affiliates. All rights reserved.

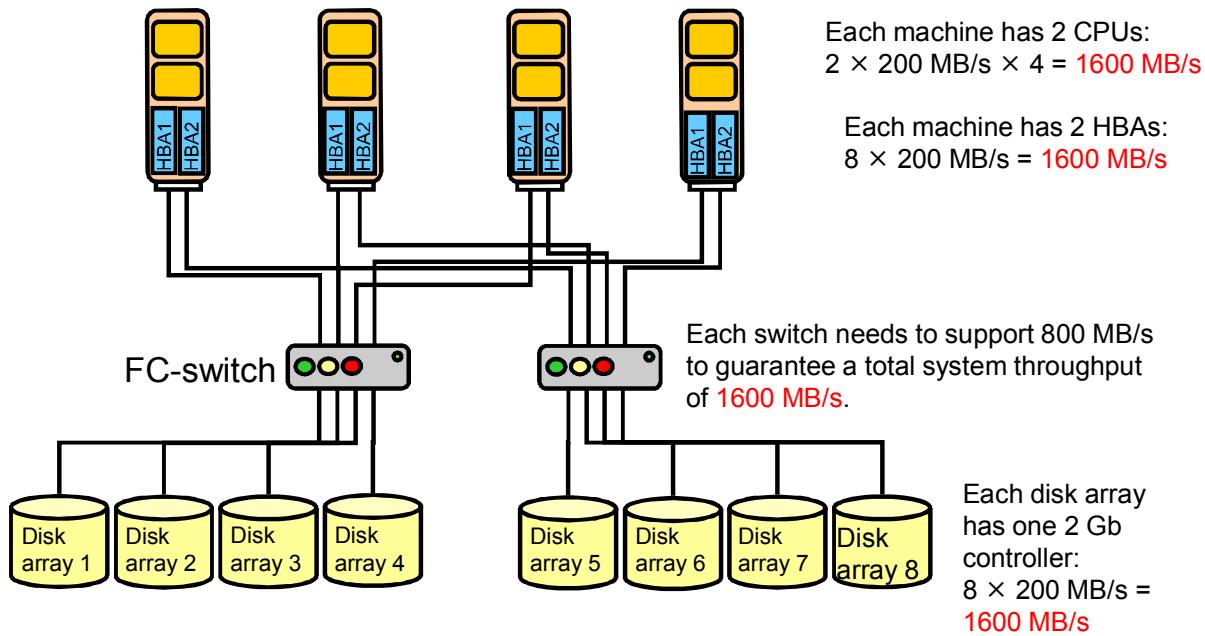
The type of workload determines whether scaleup or speedup capabilities can be achieved using parallel processing.

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 © 2014, Oracle and/or its affiliates. All rights reserved.

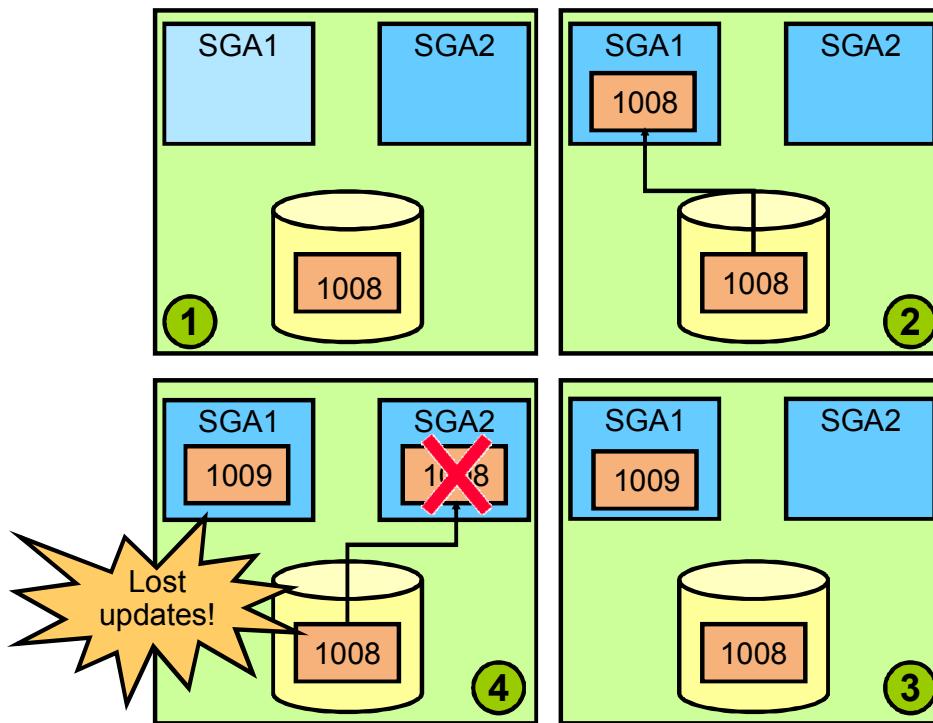
To make sure that a system delivers the I/O demand that is required, all system components on the I/O path need to be orchestrated to work together.

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. And the total number of ports on a switch limits the number of HBAs and disk controllers.

Necessity of Global Resources



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In single-instance environments, locking coordinates access to a common resource such as a row in a table. Locking prevents two processes from changing the same resource (or row) at the same time.

In RAC environments, internode synchronization is critical because it maintains proper coordination between processes on different nodes, preventing them from changing the same resource at the same time. Internode synchronization guarantees that each instance sees the most recent version of a block in its buffer cache.

The slide shows you what would happen in the absence of cache coordination. RAC prevents this problem. Resource coordination is performed for the buffer cache, library cache, row cache, and results cache and will be explored in later lessons.

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 © 2014, 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 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 © 2014, 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 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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Installing and Configuring Oracle RAC

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

Objectives

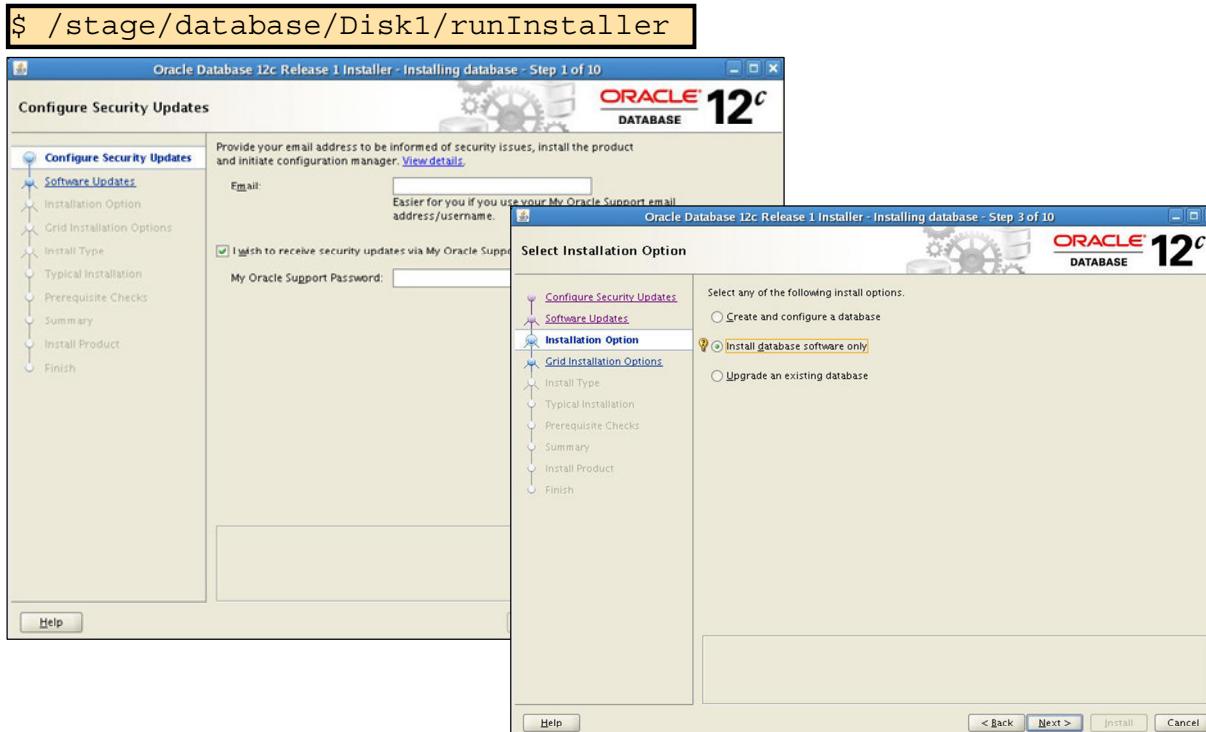
After completing this lesson, you should be able to:

- Install the Oracle database software
- Create a cluster database
- Perform post-database-creation tasks
- Convert a single-instance Oracle database to RAC



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Installing the Oracle Database Software



ORACLE

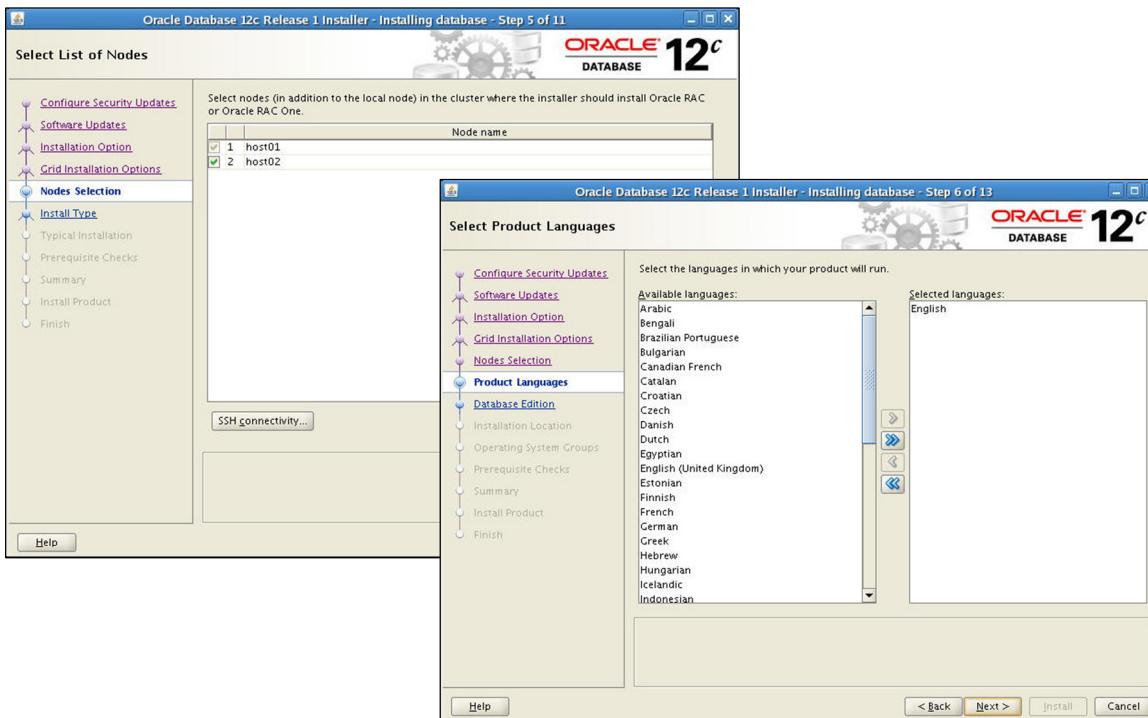
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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



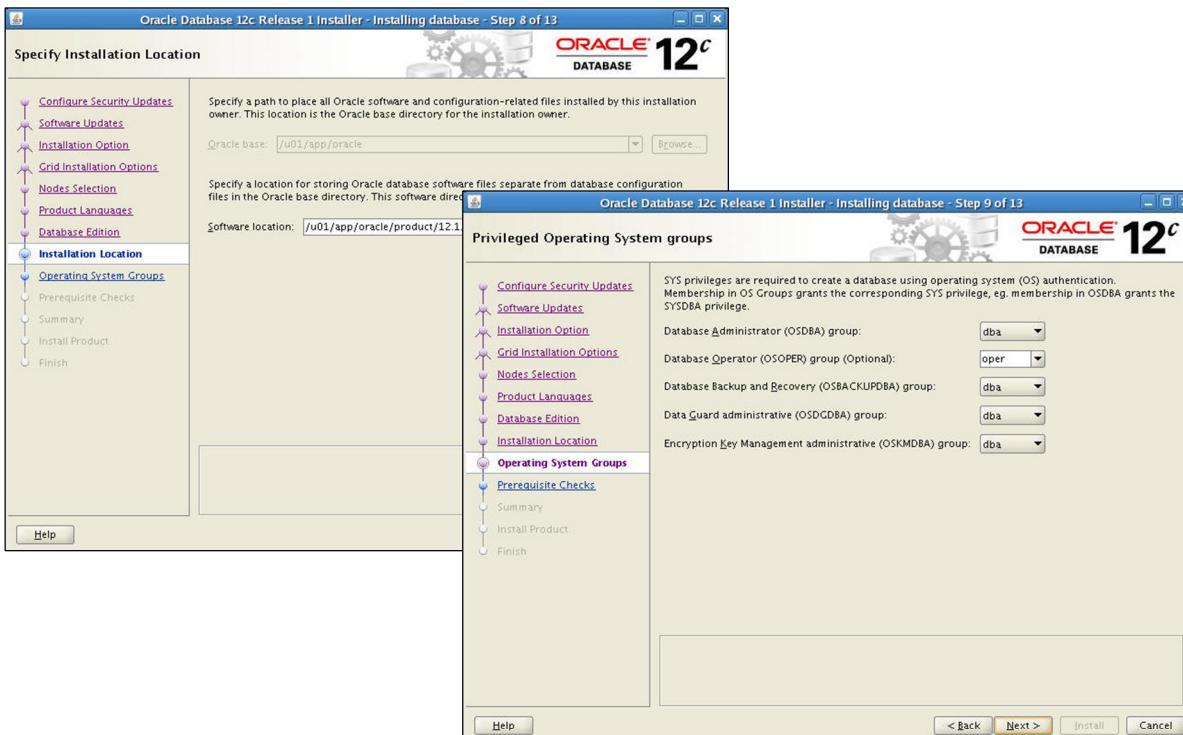
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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. on the Select List of Nodes page. 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, then click Test. When finished, click Next to continue. In the “Select product languages” window, select your desired languages from the Available Languages list and click the right arrow to promote the selected languages to the Selected Languages list. Click Next to continue.

In the “Select database edition” window (not shown in the slide), 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



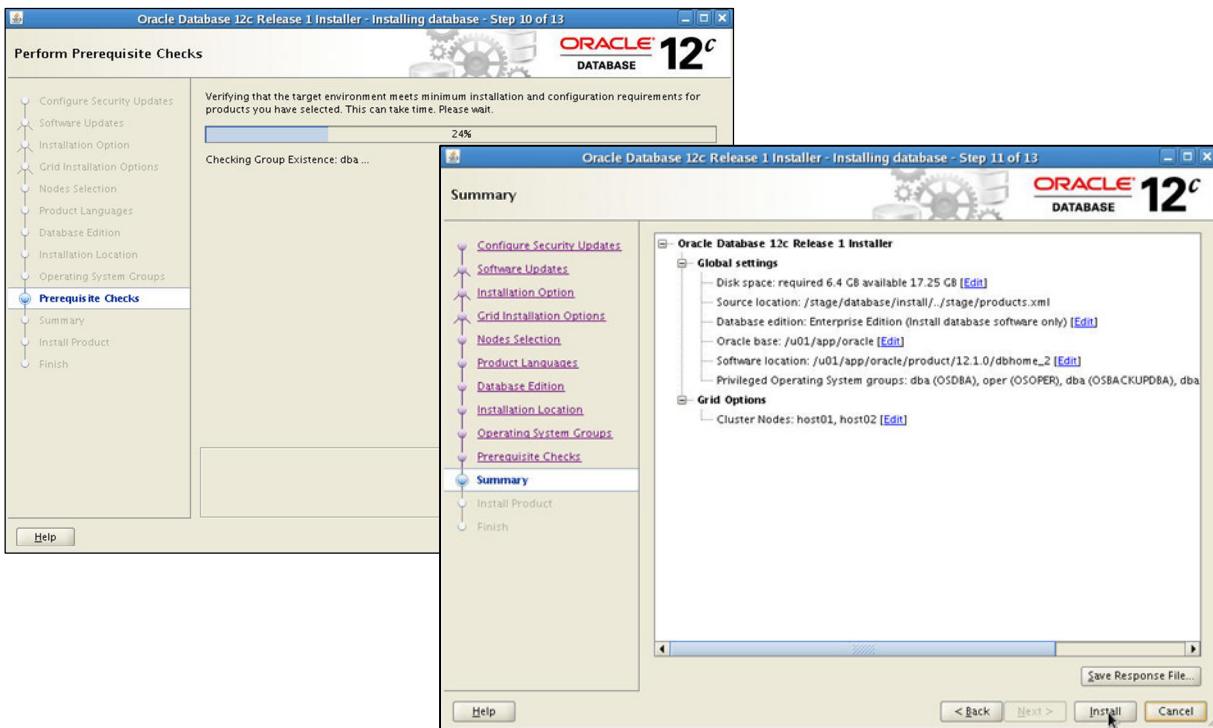
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

On the Select Database edition page, you can select to install either the Enterprise Edition or Standard Edition. Click Next to continue.

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/12.1.0/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 group that will act as the OSDBA group, and the group that will act as the OSOPER group. In addition you can assign operating system groups to the database backup and recovery group, the Data Guard administrative group, and the encryption key management administrative group. Click Next to continue.

Installing the Oracle Database Software



ORACLE

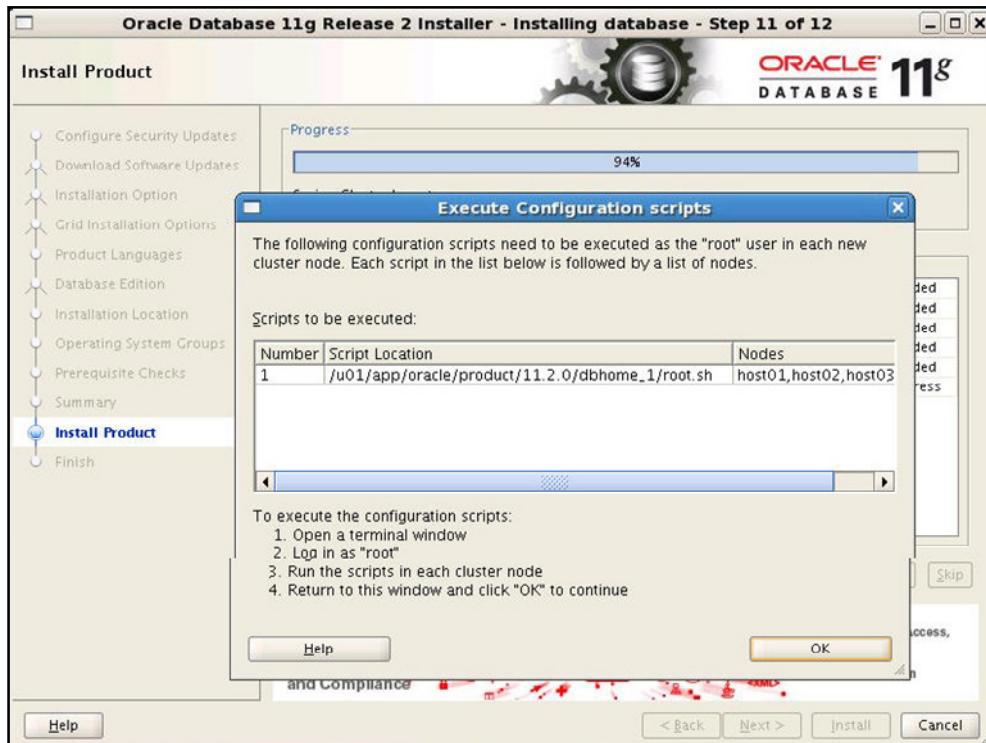
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

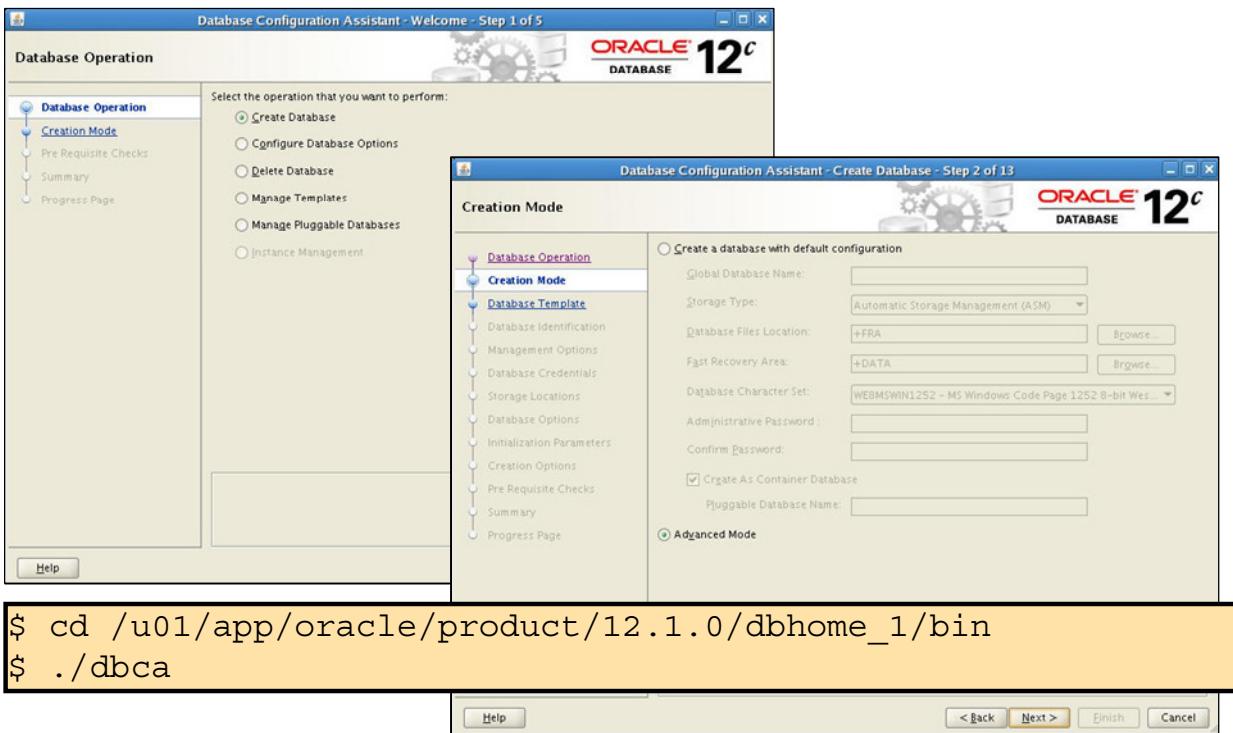


ORACLE

Copyright © 2014, 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



ORACLE

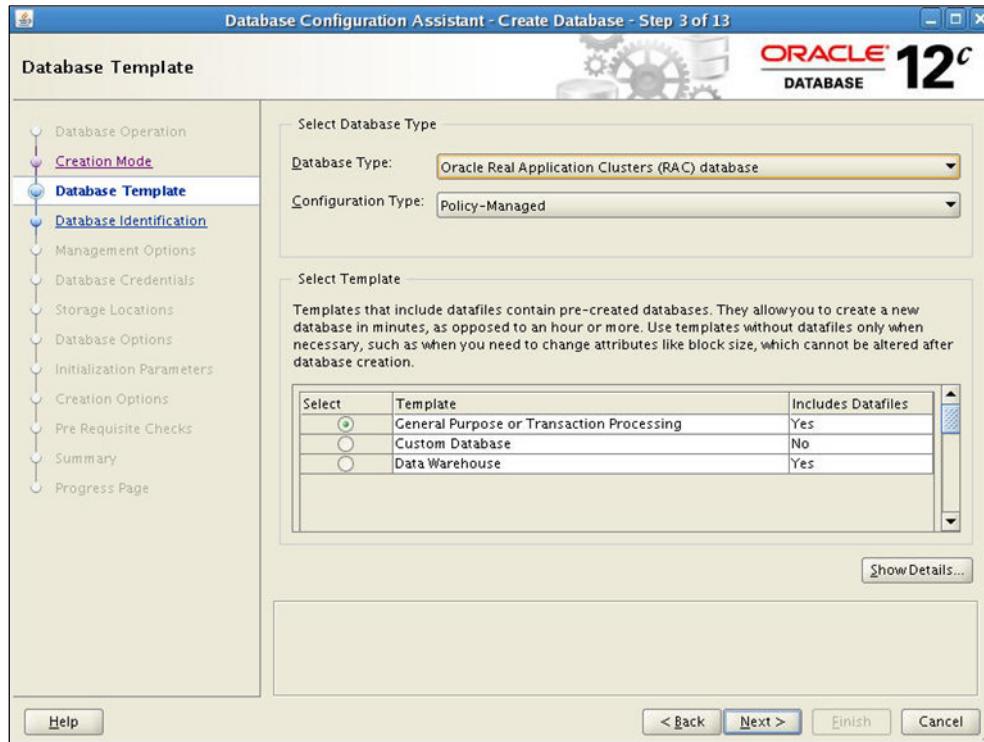
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

```
$ cd /u01/app/oracle/product/12.1.0/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 using the default configuration or you can choose Advanced Mode. Select Advanced Mode and click Next to continue.

Database Type Selection



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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.

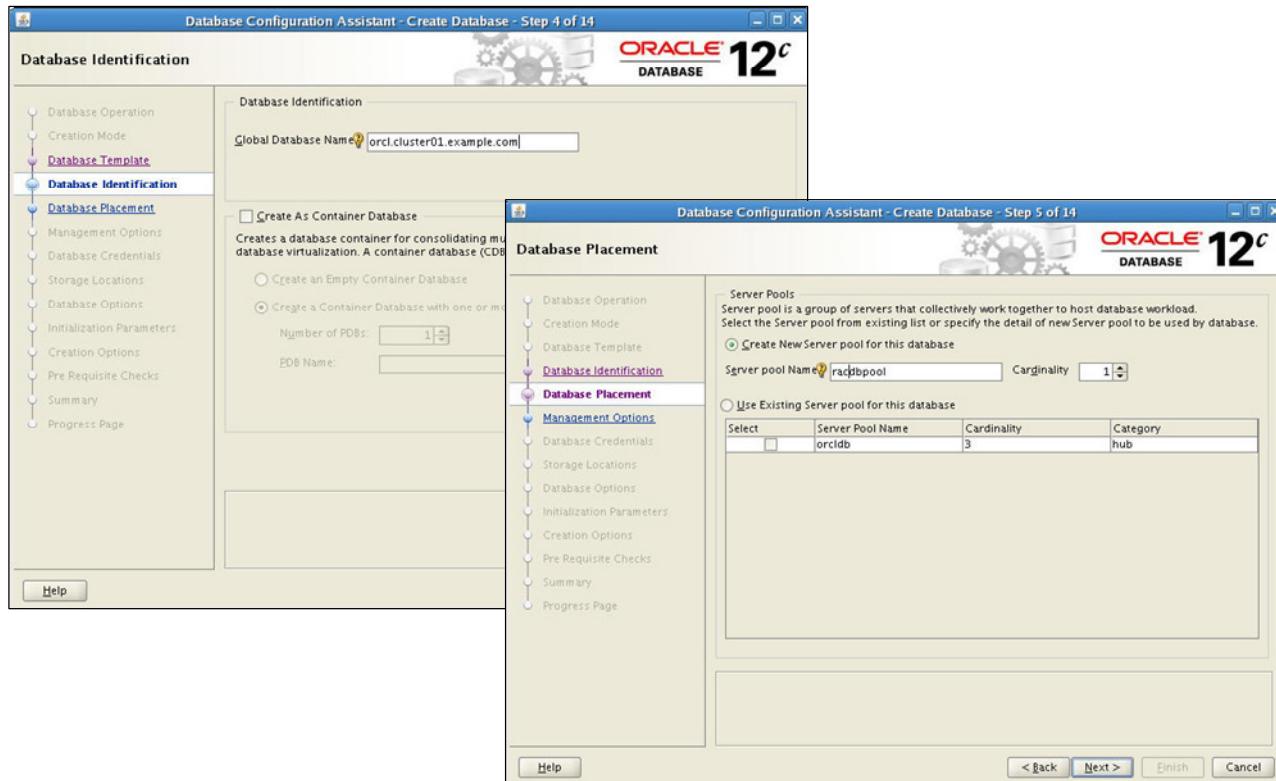
Policy-based management breaks the explicit association between services, instances, and cluster nodes. Policy-based management employs server pools, which are logical divisions of a cluster that are dynamically allocated based on relative importance. Database services are associated with server pools, 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 using `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:

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

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

Database Identification



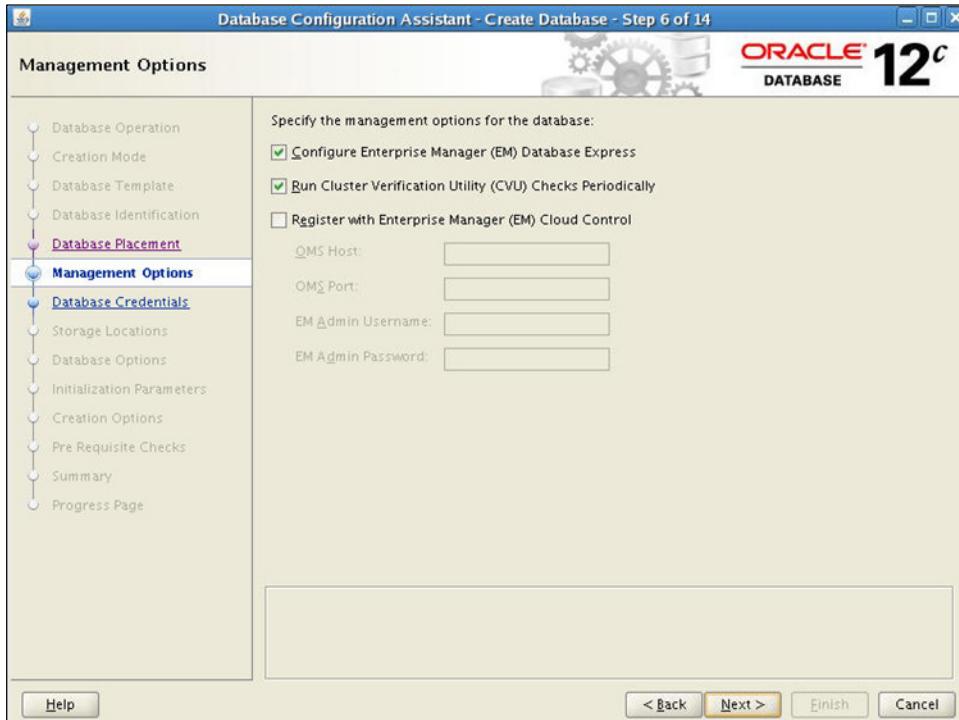
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

In the Database Identification 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 checkbox. 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.

On the Database Placement 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.

Cluster Database Management Options



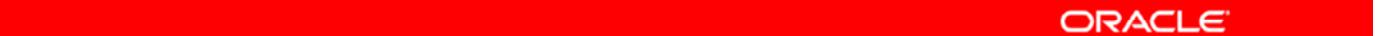
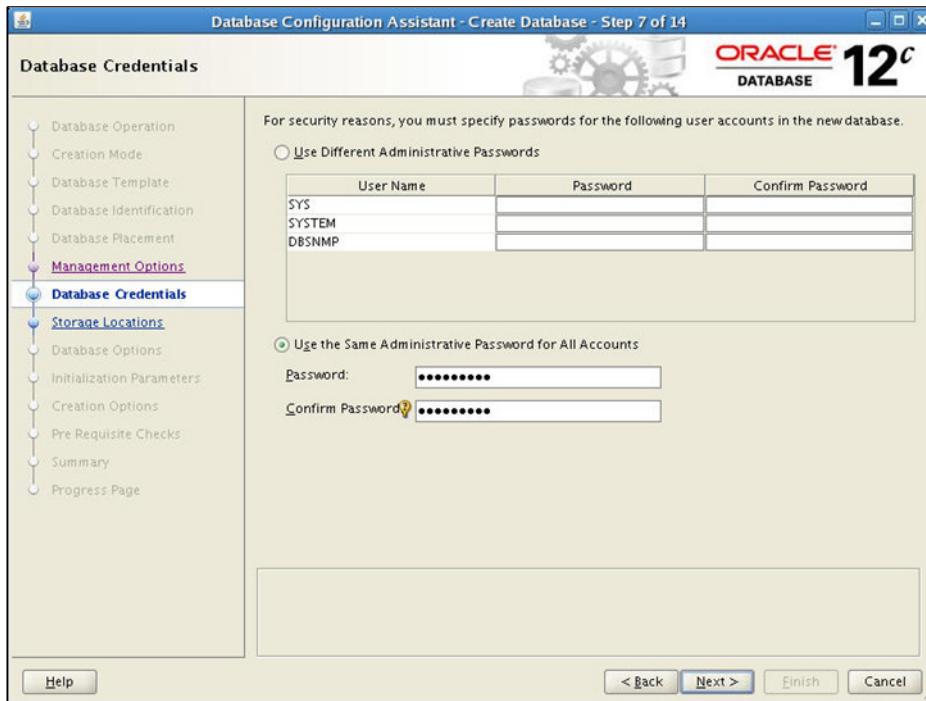
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

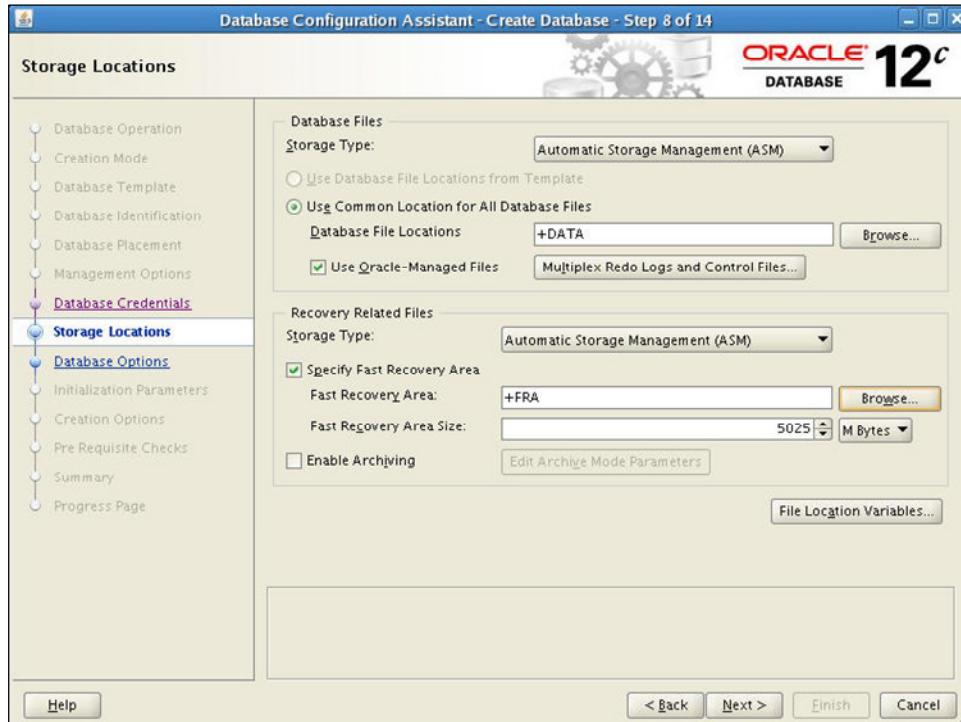
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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.

Storage Locations



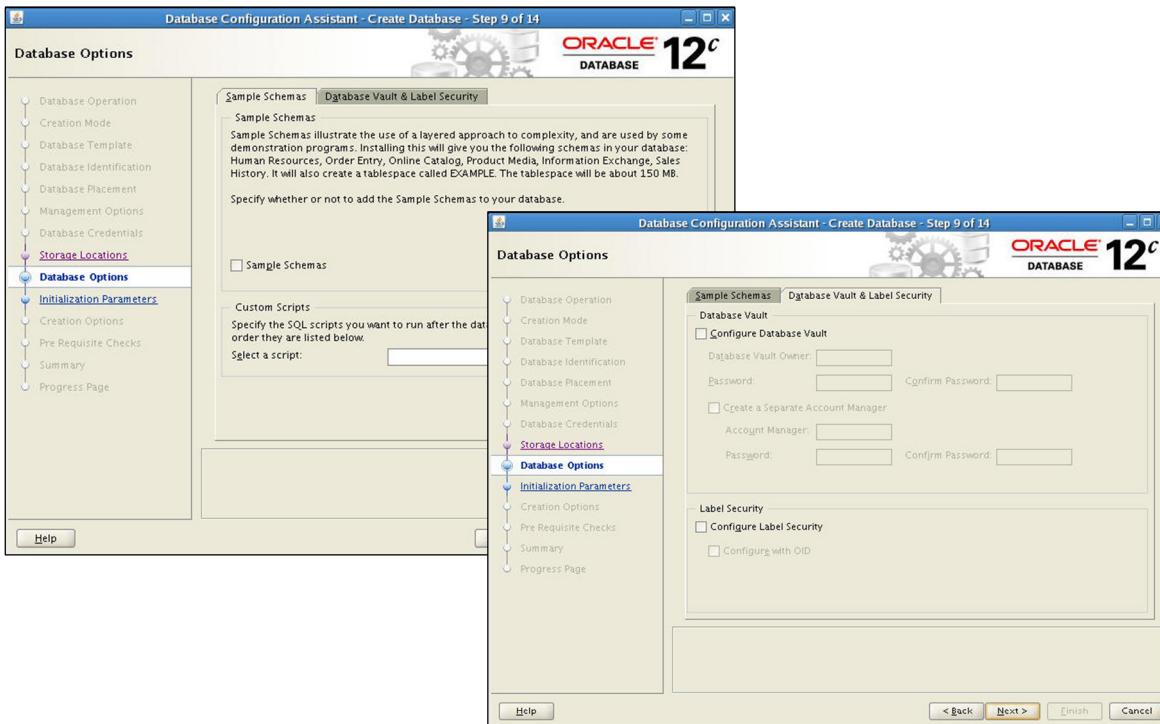
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the Database File Locations 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 Oracle Managed Files. 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 Recovery Related Files section, 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 5025 megabytes in this example, 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

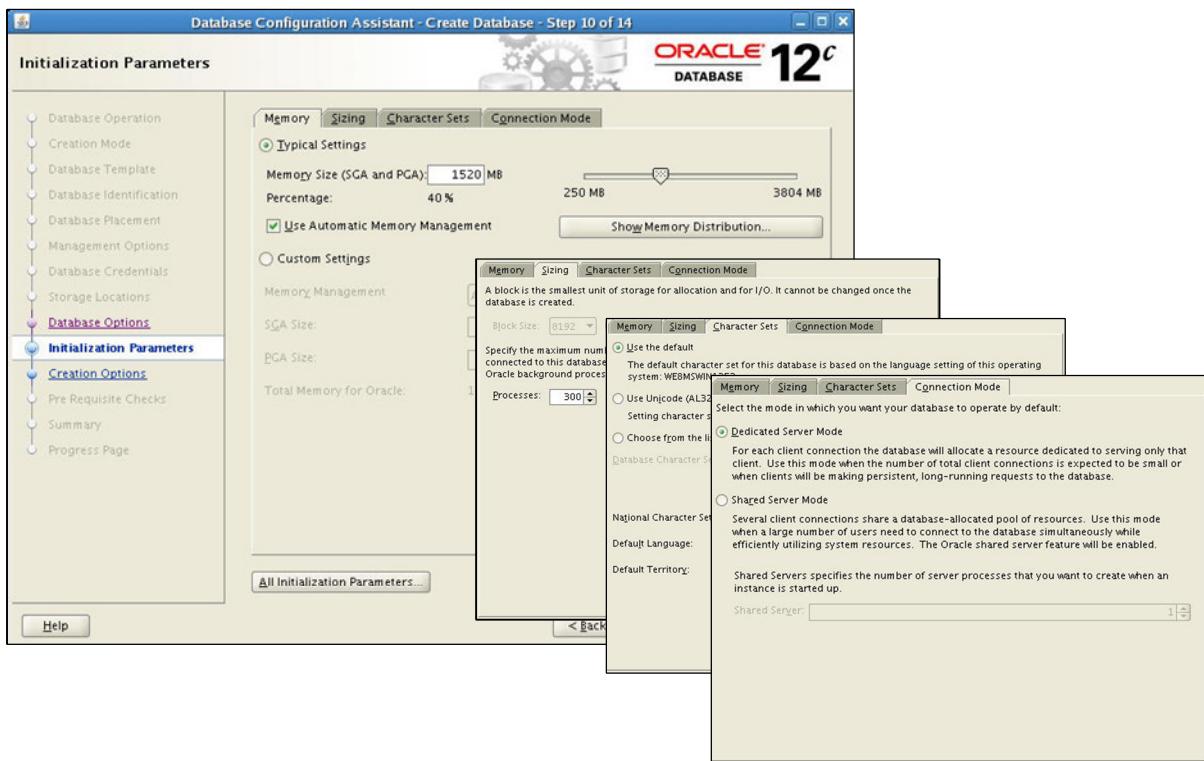


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the Database Content window, you can choose to install the Sample Schemas included with the database distribution. In the Custom Scripts section, you can choose to run your own scripts as part of the database creation process. Click the Database Vault & Label Security tab to configure these items. 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.

Initialization Parameters



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

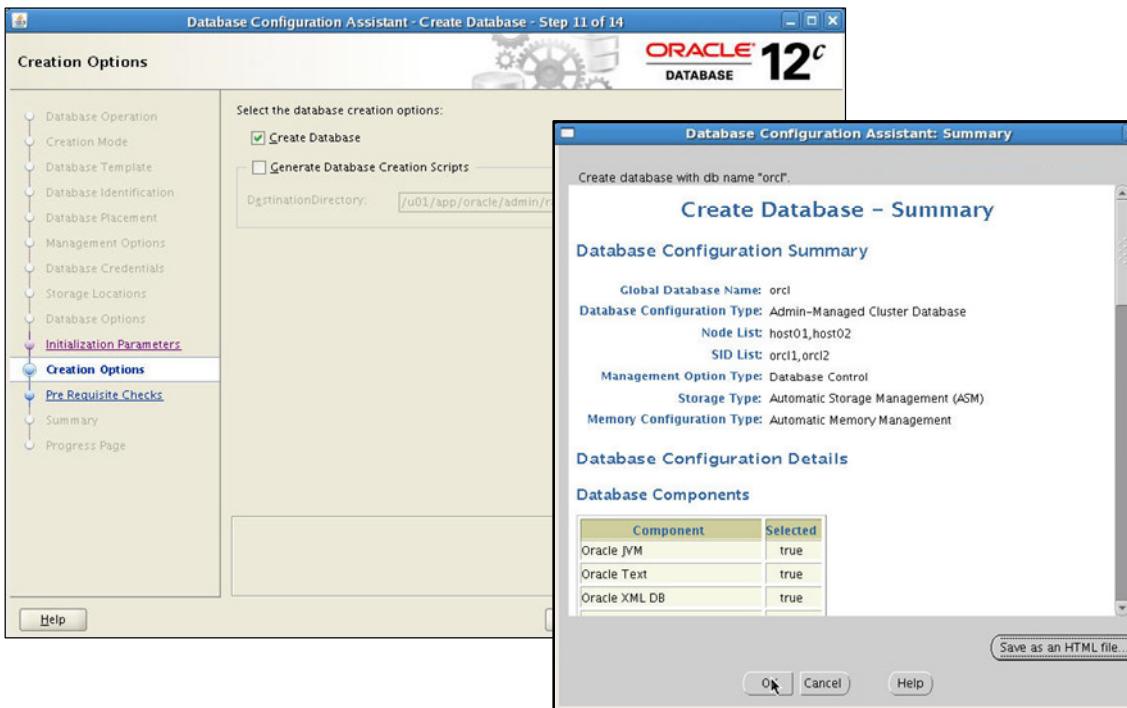
In the Initialization Parameters window, you can set important database parameters. The parameters are grouped under four tabs:

- Memory
- Sizing
- Character Sets
- Connection Mode

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. 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 Memory Management button and click Next to continue.

Create the Database



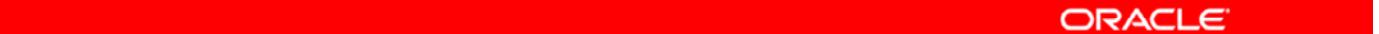
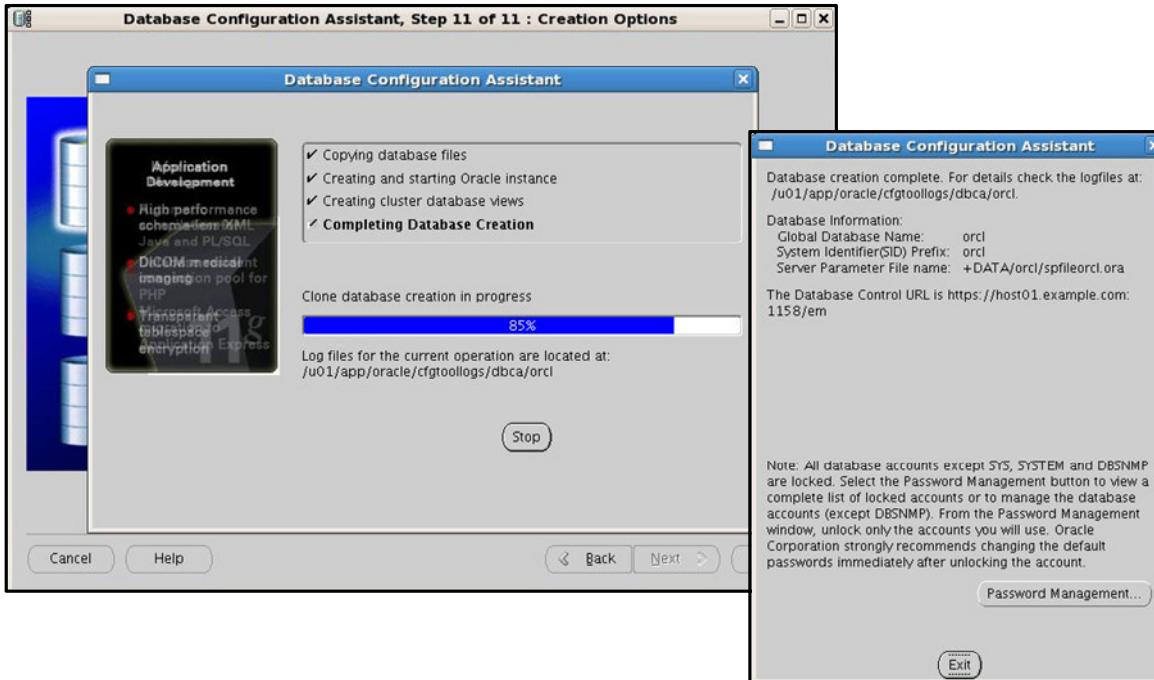
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

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 OK.

Monitoring Progress

ORACLE

Copyright © 2014, 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 a dialog box 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.

Postinstallation 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/12.1.0/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
Start concurrency:
Stop concurrency:
Database is policy managed
```



Copyright © 2014, 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 © 2014, Oracle and/or its affiliates. All rights reserved.

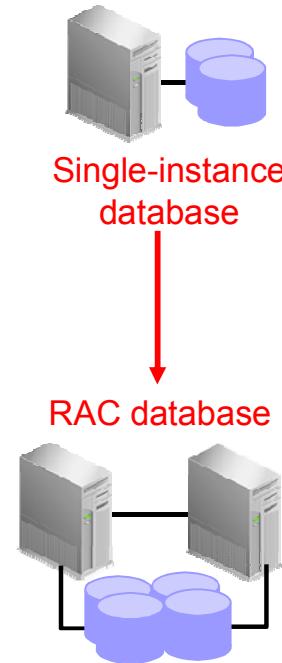
An Oracle RAC database has the same processes and memory structures as a single-instance Oracle database and additional process and memory structures that are specific to Oracle RAC. The global cache service and global enqueue service processes, and the global resource directory (GRD) collaborate to 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 LMS_n 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 (RMS_n)** : The RMS_n processes perform manageability tasks for Oracle RAC. Tasks accomplished by an RMS_n 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.

Single Instance-to-RAC Conversion

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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can use the Database Configuration Assistant (DBCA) to convert single-instance Oracle databases to RAC. The 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. It also configures Oracle Net Services, Oracle Clusterware resources, and the configuration for RAC database management for use by Oracle Enterprise Manager or the `srvctl` utility.

Before you use the DBCA to convert a single-instance database to a RAC database, ensure that your system meets the conditions:

- It is a supported hardware and operating system configuration.
- It has shared storage. A supported Cluster 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.

You can also use Enterprise Manager and the `rconfig` utility to perform the single instance-to-RAC conversion.

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 © 2014, 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. 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 OFA rules, and reduces human error with data file management.

Single-Instance Conversion Using DBCA

Conversion steps for a single-instance database on *nonclustered* hardware:

1. Back up the original single-instance database.
2. Complete the Oracle Grid Infrastructure installation.
3. Validate the cluster.
4. Copy the preconfigured database image.
5. Install the Oracle Database 12c software with RAC.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

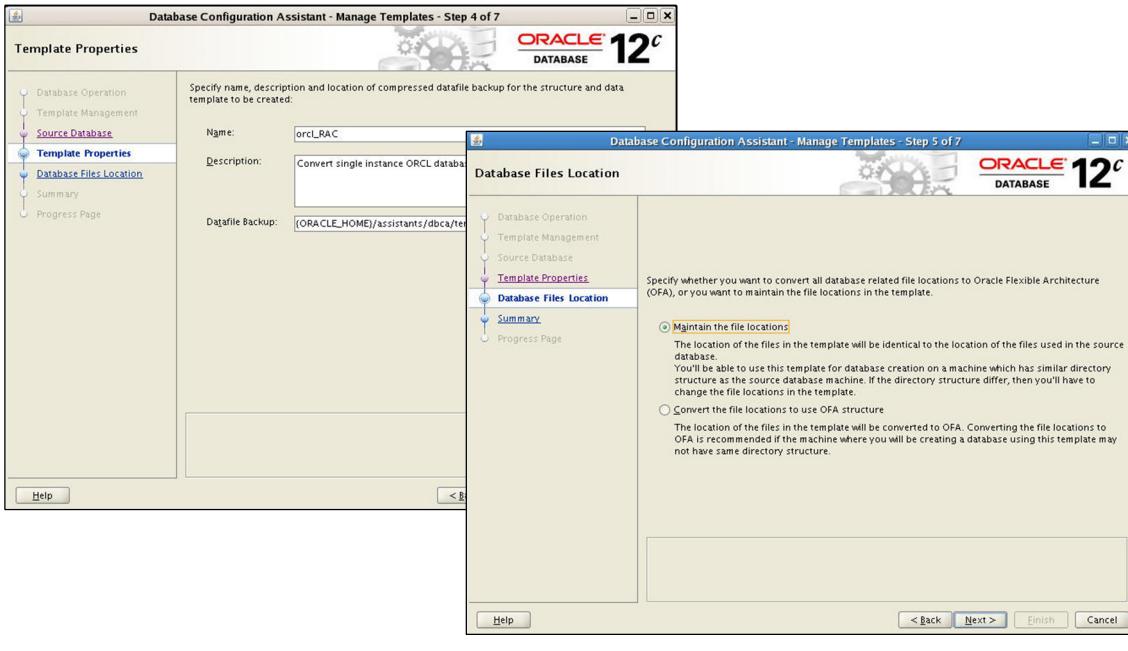
You can use Database Configuration Assistant (DBCA) to convert from single-instance Oracle databases to Oracle RAC or Oracle RAC One Node databases. DBCA automates the configuration of the control file attributes, creates the undo tablespaces and the redo logs, and creates the initialization parameter 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 noncluster computer to a RAC database, perform the following steps:

1. Back up the original single-instance database.
2. Complete the Oracle Grid Infrastructure installation.
3. Validate the cluster.
4. Copy the preconfigured database image.
5. Install the Oracle Database 12c software with RAC.

Conversion Steps

1. Back up the original single-instance database.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

1. Back Up the Original 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 (`template_name.dbc`) and a database preconfigured image file (`template_name.dbf`).

Conversion Steps

- 2. Perform the preinstallation steps.**
 - Tasks include kernel parameter configuration, hardware setup, network configuration, and shared storage setup.
- 3. 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`.
- 4. Copy the preconfigured database image.**
 - The database structure * .dbc file
 - The preconfigured database image * .dfb file
- 5. Install the Oracle Database 12c software with RAC.**



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

2. Perform the Preinstallation Steps.

Complete the Oracle Clusterware installation, as described in the *Oracle Grid Infrastructure Installation Guide* for your platform.

3. 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.

4. Copy the Preconfigured Database Image.

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.

5. Install the Oracle Database 12c Software with RAC.

1. Run the 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 postinstallation 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.

Single-Instance Conversion Using rconfig

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

```
$ cd $ORACLE_HOME/assistants/rconfig/sampleXMLs  
$ vi ConvertToRAC_PolicyManaged.xml  
... Saved as my_rac_conversion.xml  
$ rconfig my_rac_conversion.xml
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can use the rconfig command-line utility to convert a single-instance database to RAC. To use this feature, perform the following steps:

1. 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.
2. 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.
3. Assuming that you save your XML file as my_rac_conversion.xml, navigate to the \$ORACLE_HOME/bin directory, and use the following syntax to run the rconfig command:
`$./rconfig my_rac_conversion.xml`

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 prerequisite checks.

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

The statement is true.

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 © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a, c

Choices a and c are correct.

Summary

In this lesson, you should have learned how to:

- Install the Oracle database software
- Create a cluster database
- Perform post-database-creation tasks
- Convert a single-instance Oracle database to RAC



Copyright © 2014, 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 © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle RAC Administration

4

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

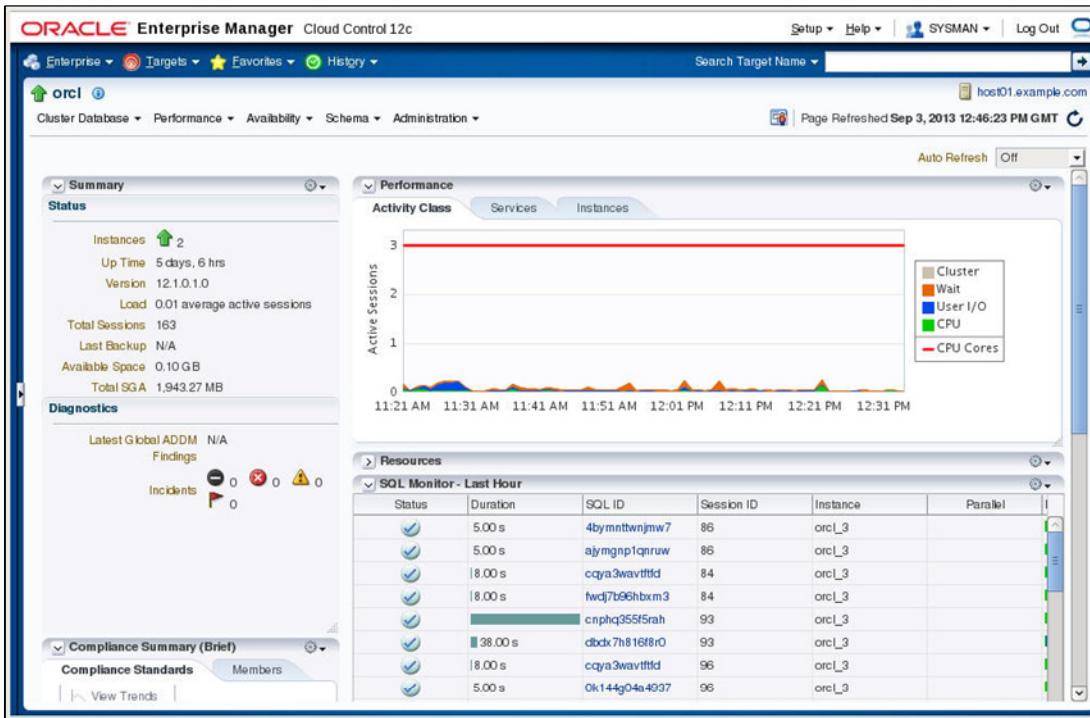
After completing this lesson, you should be able to:

- Use Enterprise Manager Cluster Database pages
- Define redo log files in a RAC environment
- Define undo tablespaces in a RAC environment
- Start and stop RAC databases and instances
- Modify initialization parameters in a RAC environment



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Cluster Database Home Page



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The 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, etc. Below, 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 'orc1'. 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.

Compliance Standards pane: Shows 'Members' with three entries: 'orc1_3' (status: up, host: host01.example.com), 'orc1_3' (status: up, host: host01.example.com), and 'orc1_3' (status: up, host: host01.example.com). A 'View Trends' button is available.

Instances pane: Displays two instances: 'orc1_orcl_3' (status: up, host: host01.example.com) and 'orc1_orcl_1' (status: up, host: host02.example.com). It also lists an ASM instance: '+ASM1_host01.example.com' (status: up, host: host01.example.com).

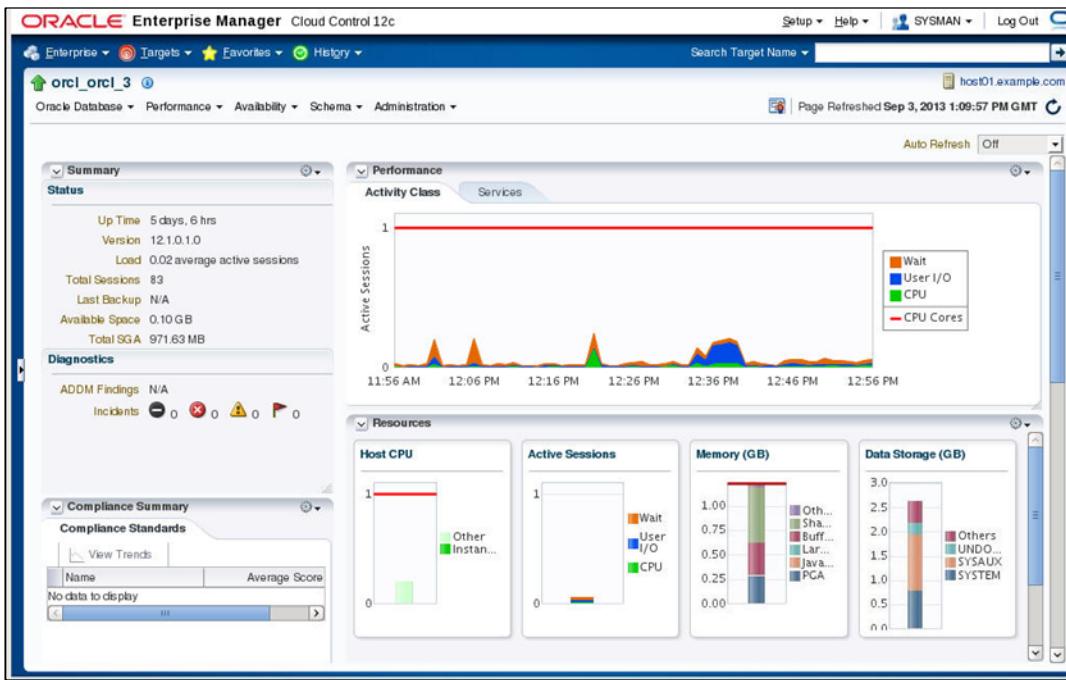
Incidents and Problems pane: Shows a summary table with columns: Summary, Target, Severity, Status, Escalation level, Type, and Time since last update. It indicates 'No matching incidents or problems found.'

Copyright © 2014, 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



Copyright © 2014, 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 Cloud Control 12c Cluster Home Page for a cluster named 'cluster01'. The page is divided into several sections:

- General:** Shows the overall status as 'Up' with 3 hosts (2 green, 1 yellow), availability at 100.0% (last 24 hours), Cluster Name 'cluster01', Clusterware Status 'Up' (2 green, 1 red), Clusterware Version '12.1.0.1.0', Oracle Home '/u01/app/12.1.0/grid', Cluster Mode 'Flex Cluster', and Reconfiguration Activities '11'. A link 'View All Properties' is also present.
- Status Summary:** Details Hub Nodes (3 hosts, 2 green, 1 yellow), Clusterware on Hub Nodes (3 hosts, 2 green, 1 red), Listeners on Hub Nodes (1 host, 1 red), and SCAN Listeners (7 hosts, 3 green, 3 yellow, 1 red).
- Diagnostic Summary:** Shows Interconnect Events (0) and Problem Resources (0).
- Cluster Databases:** Displays a table with one row for 'orcl' with a green arrow icon, status 'Up', 0 incidents, and 100% compliance.
- Cluster ASM:** Displays a table for 'No Cluster ASM' with 0 ASM Instances and 0 ASM IO Servers.
- Incidents:** A table showing 'No Incidents' found.
- Hosts:** A table listing three hosts:

Name	Status	Clusterware Status	Incidents	Compliance Score (%)	Actions
host01.example.com	Up	has_host01.example.com	1 0 0 0	100	+ASM1_host01.example.com
host02.example.com	Up	has_host02.example.com	0 1 0 0	100	
host03.example.com	Down	host03.example.com	1 1 0 0	100	

ORACLE

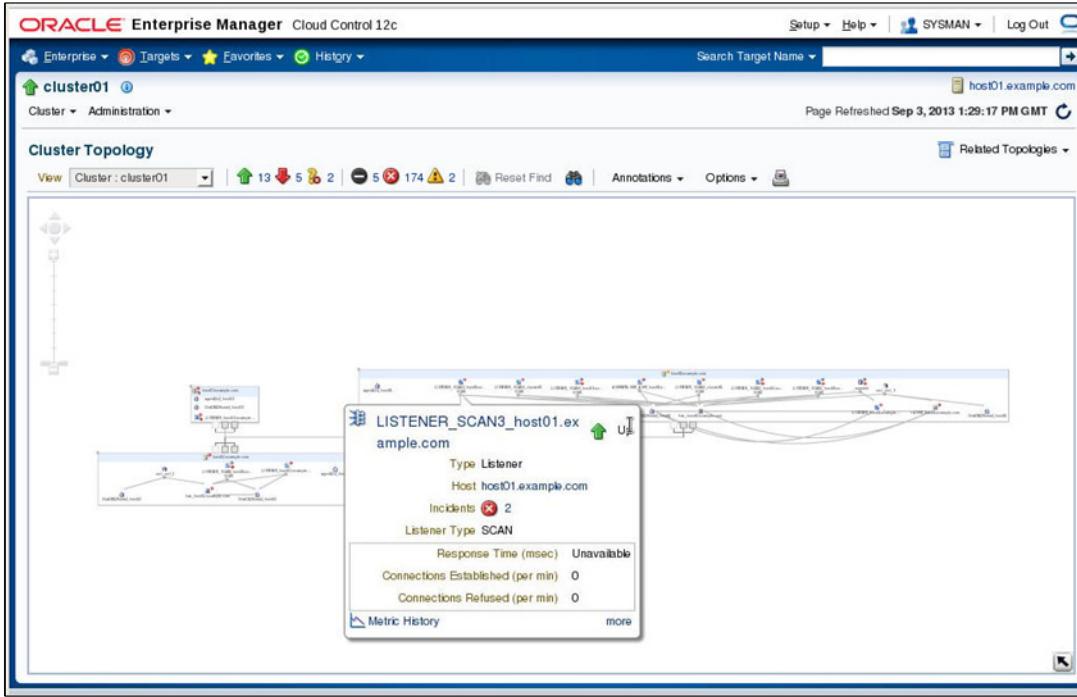
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

Severity	Summary	Target	Priority	Status	Last Updated	Owner	Acknow	Escala	Type	Category
✖	[ctssd1162]CRS-2412:The Cluster Time Synchro has_host02	None	None	New	Sep 3, 2013 1:11:41 PM	-	No	No	Incident	Error
✖	TNS-1189. Please check log for details.	LISTENER_1	None	New	Sep 3, 2013 12:54:19 PM	-	No	No	Incident	
✖	[ctssd1162]CRS-2412:The Cluster Time Synchro has_host02	None	None	New	Sep 3, 2013 12:41:35 PM	-	No	No	Incident	Error
✖	[ctssd1162]CRS-2412:The Cluster Time Synchro has_host02	None	None	New	Sep 3, 2013 12:11:31 PM	-	No	No	Incident	Error
✖	TNS-1189. Please check log for details.	LISTENER_1	None	New	Sep 3, 2013 11:54:02 AM	-	No	No	Incident	
✖	[ctssd1162]CRS-2412:The Cluster Time Synchro has_host02	None	None	New	Sep 3, 2013 11:41:26 AM	-	No	No	Incident	Error
✖	[ctssd1162]CRS-2412:The Cluster Time Synchro has_host02	None	None	New	Sep 3, 2013 11:11:20 AM	-	No	No	Incident	Error
✖	TNS-1190. Please check log for details.	LISTENER_1	None	New	Sep 3, 2013 11:04:53 AM	-	No	No	Incident	
✖	TNS-1190. Please check log for details.	LISTENER_1	None	New	Sep 3, 2013 10:54:09 AM	-	No	No	Incident	
✖	TNS-1190. Please check log for details.	LISTENER_1	None	New	Sep 3, 2013 10:53:57 AM	-	No	No	Incident	
✖	[ctssd1162]CRS-2412:The Cluster Time Synchro has_host02	None	None	New	Sep 3, 2013 10:41:14 AM	-	No	No	Incident	Error
✖	TNS-1190. Please check log for details.	LISTENER_1	None	New	Sep 3, 2013 10:39:28 AM	-	No	No	Incident	

Row count 185

ORACLE®

Copyright © 2014, 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 main menu has 'Cluster Database: orcl' selected. Underneath, a sub-menu titled 'Metric and Collection Settings' is open. On the left, a tree view shows 'Metrics' and 'Other Collected Items' under 'orcl'. The right side displays a table of metrics with thresholds for the 'orcl_orcl_3' instance. One row is expanded to show alert log details like 'Archiver Hung Alert Log Error' and 'Data Block Corruption Alert Log Error'.

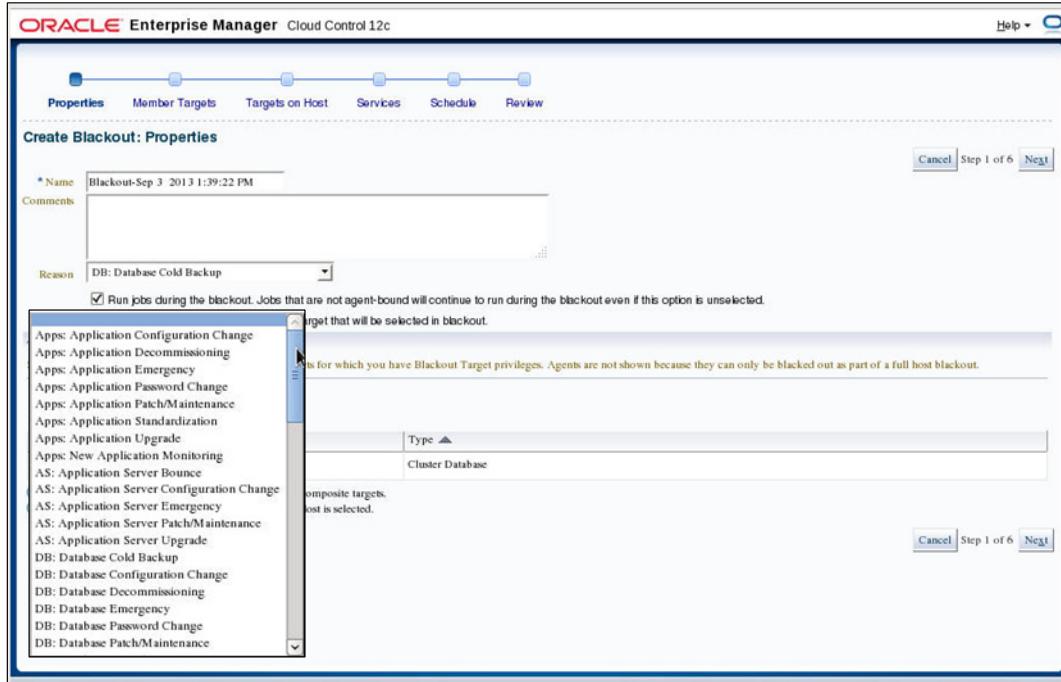
Metric	Operator	Warning Threshold	Critical Threshold	Corrective Actions	Collection Schedule	Edit
orcl_orcl_3					Disabled	
Alert Log						
Archiver Hung Alert Log Error	Contains	<input type="text"/>	ORA-	None		
Data Block Corruption Alert Log Error	Contains	<input type="text"/>	ORA-	None		
Generic Alert Log Error	Matches	<input type="text"/>	ORA-0>600?	None		
Media Failure Alert Log Error	Contains	<input type="text"/>	ORA-	None		
Session Terminated Alert Log Error	Contains	<input type="text"/>	ORA-	None		
Archive Area	>	<input type="text"/> 80	<input type="text"/>	None	Every 15 Minutes	
Database Job Status	>	<input type="text"/> 0	<input type="text"/>	None	Every 30 Minutes	
Database Service Status	Matches	<input type="text"/> Up With Warn	<input type="text"/> Down	None	Every 5 Minutes	
Database Vault Configuration Issues - Command Rules	>	<input type="text"/>	<input type="text"/> 0	None	Every 1 Hour	
Database Vault Configuration Issues Count - Command Rules	>	<input type="text"/>	<input type="text"/> 0	None	Every 1 Hour	
Database Vault Configuration Issues - Realms	>	<input type="text"/>	<input type="text"/> 0	None		
Database Vault Configuration Issues Count - Realms	>	<input type="text"/>	<input type="text"/> 0	None		

ORACLE

Copyright © 2014, 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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

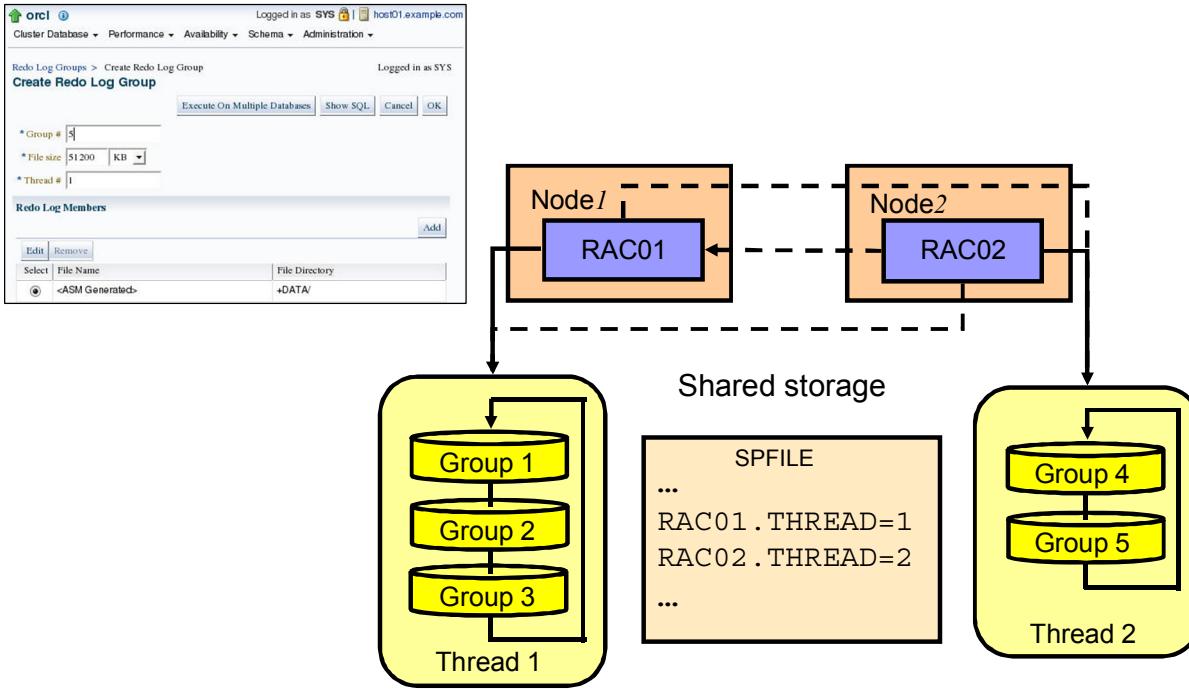
You can use Enterprise Manager to define blackouts for all managed targets of your RAC database to prevent alerts from being recorded. Blackouts are useful when performing scheduled or unscheduled maintenance or other tasks that might trigger extraneous or unwanted events. You can define blackouts for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

Redo Log Files and RAC



ORACLE

Copyright © 2014, 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.

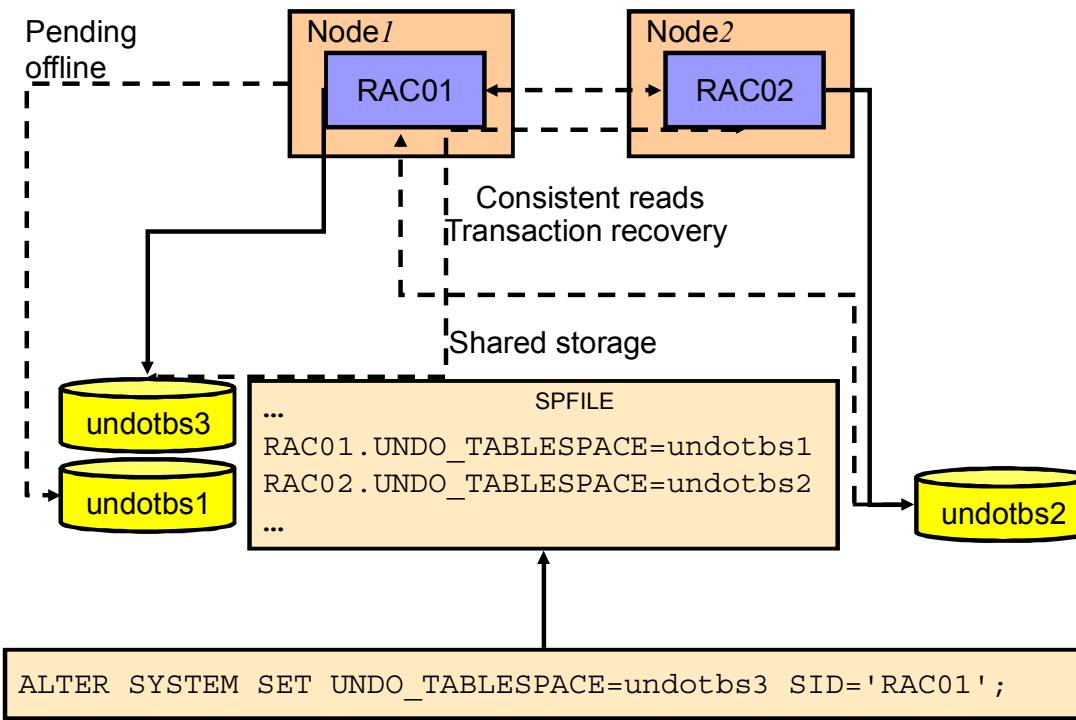
When the current group fills, an instance begins writing to the next log file group. If your database is in ARCHIVELOG mode, then each instance must save filled online log groups as archived redo log files that are tracked in the control file. During database recovery, all enabled instances are checked to see if recovery is needed. If you remove an instance from your Oracle RAC database, then 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.

Automatic Undo Management and RAC



ORACLE

Copyright © 2014, 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.

Starting and Stopping RAC Instances

- Multiple instances can open the same database simultaneously.
- Shutting down one instance does not interfere with other running instances.
- 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 © 2014, 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. Also, shutting down one instance does not interfere with the operation of other running instances.

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.

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 instance -db db_unique_name -node node_name  
-instance instance_name_list [-startoption  
open|mount|nomount|normal|transactional|immediate|abort]
```

```
srvctl start|stop database -db <db_name> -eval  
[-startoption  
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 open
```

```
$ srvctl start instance -db orcl -node host01
```

```
*** This command will start a Policy-Managed database***
```



Copyright © 2014, 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. In order 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, then 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
```

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



Copyright © 2014, 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 PDB 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 © 2014, 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 pluggable database (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.

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/12.1.0/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 © 2014, 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]
```

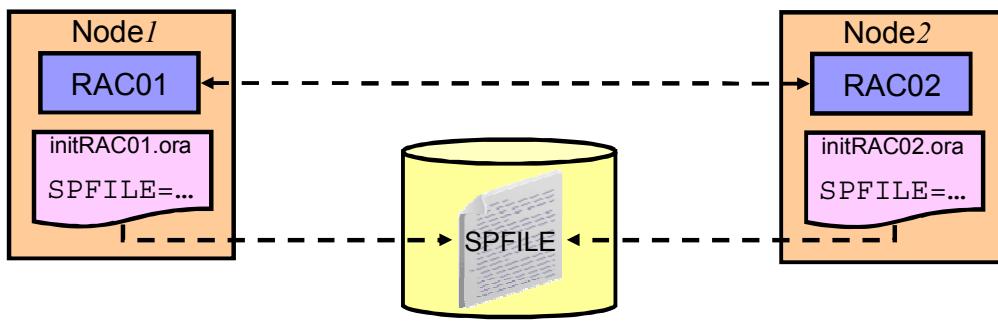
When you add a new database using the `srvctl add database` command, you can use the `-policy` parameter to specify the management policy:

```
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 © 2014, 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 © 2014, 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>.<dparam> entry. This allows for the current *.<dparam> entry to be used, or for the next created *.<dparam> entry to be taken into account on that particular SID.

Using the last example, you can remove a line from your SPFILE.

Note: When you start an instance with an SPFILE, the default for SQL*Plus is SCOPE=BOTH.

Parameter File Search Order in Oracle RAC

- On Linux and UNIX platforms, the search order is as follows:
 1. \$ORACLE_HOME/dbs/spfilesid.ora
 2. \$ORACLE_HOME/dbs/spfile.ora
 3. \$ORACLE_HOME/dbs/initsid.ora
- On Windows platforms, the search order is as follows:
 1. %ORACLE_HOME%\database\spfilesid.ora
 2. %ORACLE_HOME%\database\spfile.ora
 3. %ORACLE_HOME%\database\initsid.ora



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Database searches for your parameter file in a particular order depending on your platform. For Oracle RAC databases, you can easily determine the location of the parameter file by using the `srvctl config database` command.

Oracle recommends that you do not use the default SPFILE names because all instances must use the same file and they all have different SIDs. Instead, store the SPFILE on Oracle ASM. If you store the SPFILE on a cluster file system, then use the following naming convention for the SPFILE: \$ORACLE_HOME/dbs/spfiledb_unique_name.ora. Create a PFILE named \$ORACLE_HOME/dbs/initsid.ora that contains the name SPFILE=ORACLE_HOME/dbs/spfiledb_unique_name.ora.

EM and SPFILE Parameter Values

The screenshot shows the Oracle Enterprise Manager interface for a Cluster Database named 'orcl'. The 'Administration' tab is selected, and the 'Initialization Parameters' option is highlighted with a red box. A red arrow points from this box down to the table below. The table displays various initialization parameters with their current values, types, and categories. At the bottom of the table, there are buttons for 'Execute On Multiple Databases', 'Show SQL', 'Revert', and 'Apply'.

Select	Instance	Name	Value	Comments	Type	Basic	Modified	Dynamic	Category
<input checked="" type="radio"/>	*	open_cursors	300		Integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Cursors and Library Cache
<input type="radio"/>	*	open_links	4		Integer				Distributed, Replication and Snapshot
<input type="radio"/>	*	open_links_per_instance	4		Integer				Distributed, Replication and Snapshot
<input type="radio"/>	*	read_only_open_delayed	FALSE		Boolean				Memory
<input type="radio"/>	*	session_max_open_files	10		Integer				Objects and LOBs

ORACLE

Copyright © 2014, 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`.

EM and SPFILE Parameter Values

The parameter values listed here are from the SPFILE +DATA/orcl/spfile/orclora

Name	Basic	Dynamic	Category
open	All	All	All
Filter on a name or partial name:			
<input checked="" type="checkbox"/> Apply changes in SPFILE mode to the current running instance(s). For static parameters, you must restart the database.			
<input type="button" value="Add"/> <input type="button" value="Reset"/>			
Select	Instance	Name	Comments
<input checked="" type="radio"/>	*	open_cursors	300
Type	Constraint	Basic	Dynamic
Integer	None	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Cursors and Library Cache			

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The SPFILE tabbed page displays the current values stored in your SPFILE.

As on the Current tabbed page, you can add or reset parameters. However, if you select the “Apply changes in SPFILE mode” check box, the ALTER SYSTEM command uses SCOPE=BOTH. If this check box is not selected, SCOPE=SPFILE is used.

Click Apply to accept and generate your changes.

RAC Initialization Parameters

Initialization Parameters

Current **SPFile**

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

Filter on a name or partial name

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	i		TRUE		Boolean	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		Cluster Database
<input type="radio"/>	*	cluster_database_instances	i		3		Integer		<input checked="" type="checkbox"/>		Cluster Database
<input type="radio"/>	*	cluster_interconnects	i				String				Cluster Database

Execute On Multiple Databases **Show SQL** **Revert** **Apply**

Database Identification

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

Shared Server

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

Miscellaneous

<input checked="" type="radio"/>	orcl_1	instance_name	i	orcl_1	String			<input checked="" type="checkbox"/>		Instance Identification
<input type="radio"/>	orcl_3	instance_name		orcl_3	String					Instance Identification

Copyright © 2014, 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 © 2014, 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 © 2014, 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.

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 © 2014, 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 if a clean shutdown were 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 © 2014, 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 above 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 wish to terminate the session without being replayed, use the `-noreplay` option:

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

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



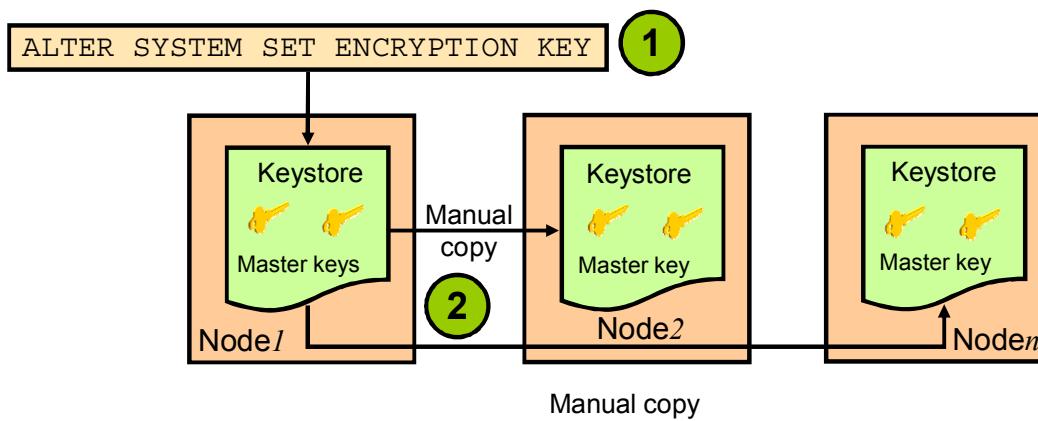
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Most SQL statements affect the current instance. You can use SQL*Plus to start and stop instances in the Oracle RAC database. You do not need to run SQL*Plus commands as root on Linux and UNIX systems or as Administrator on Windows systems. You need only the proper database account with the privileges that you normally use for a noncluster Oracle database. Some examples of how SQL*Plus commands affect instances are:

- ALTER SYSTEM CHECKPOINT LOCAL affects the current instance.
- ALTER SYSTEM CHECKPOINT or ALTER SYSTEM CHECKPOINT GLOBAL affect all instances in the cluster database.
- ALTER SYSTEM SWITCH LOGFILE affects only the current instance.
 - To force a global log switch, use the ALTER SYSTEM ARCHIVE LOG CURRENT statement.
 - The INSTANCE option of ALTER SYSTEM ARCHIVE LOG enables you to archive each online redo log file for a specific instance
- The INSTANCE option of ALTER SYSTEM ARCHIVE LOG enables you to archive each online redo log file for a specific instance.

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 © 2014, 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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

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.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: c

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
- Start and stop RAC databases and instances
- Modify initialization parameters in a RAC environment



Copyright © 2014, 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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

Managing Backup and Recovery for RAC



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

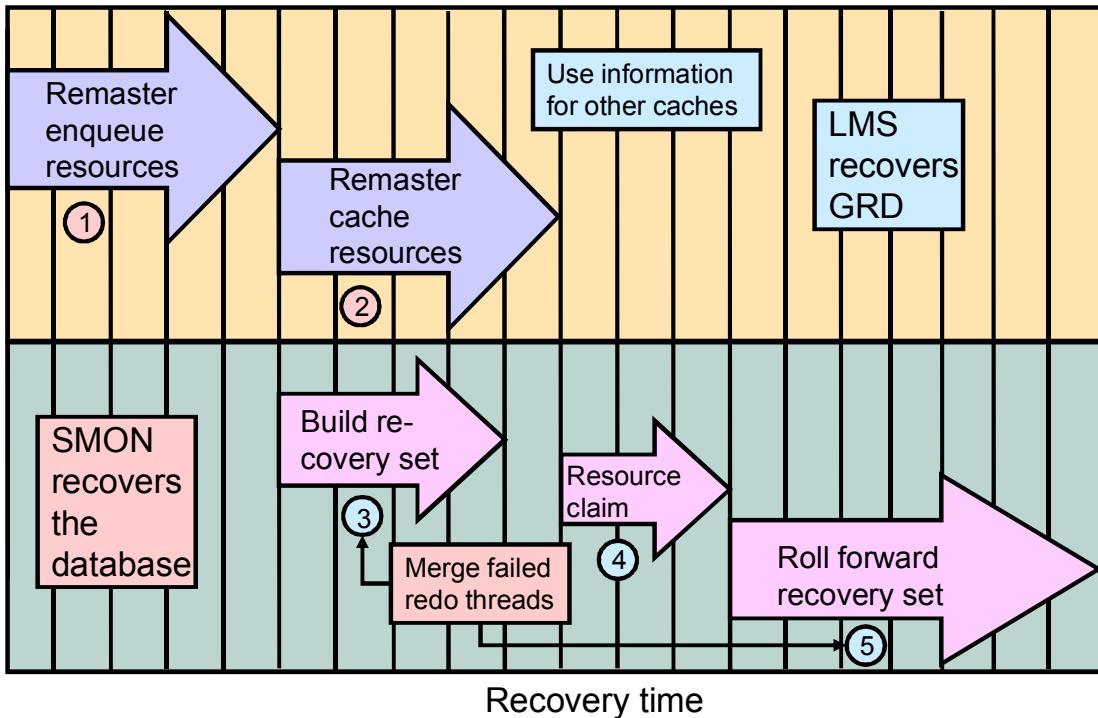
After completing this lesson, you should be able to configure the following:

- The RAC database to use ARCHIVELOG mode and the fast recovery area
- RMAN for the RAC environment



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

RAC and Instance Recovery



ORACLE

Copyright © 2014, 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.

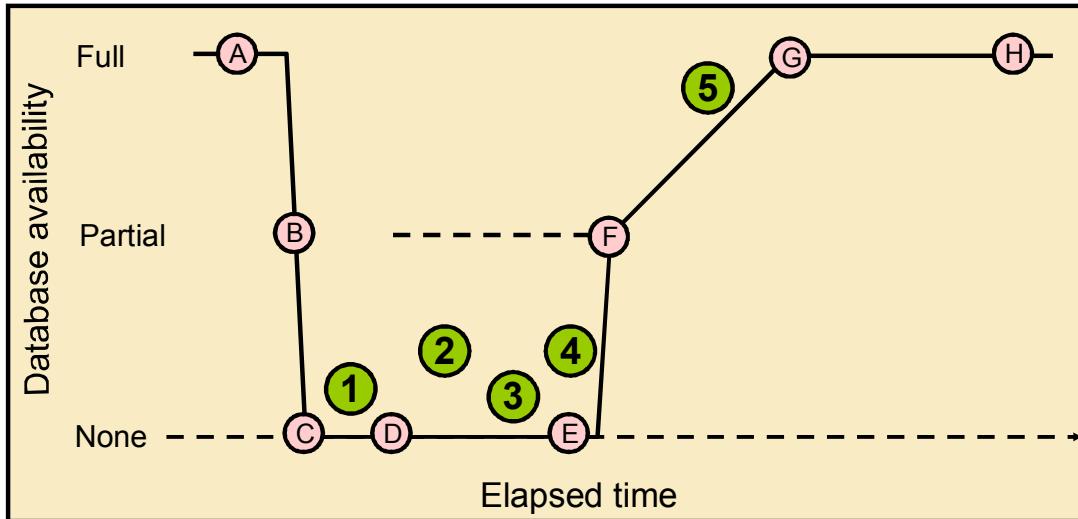
4. Buffer space for recovery is allocated and the resources that were identified in the previous reading of the redo logs are claimed as recovery resources. This is done to avoid other instances to access those resources.
5. All resources required for subsequent processing have been acquired and the 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

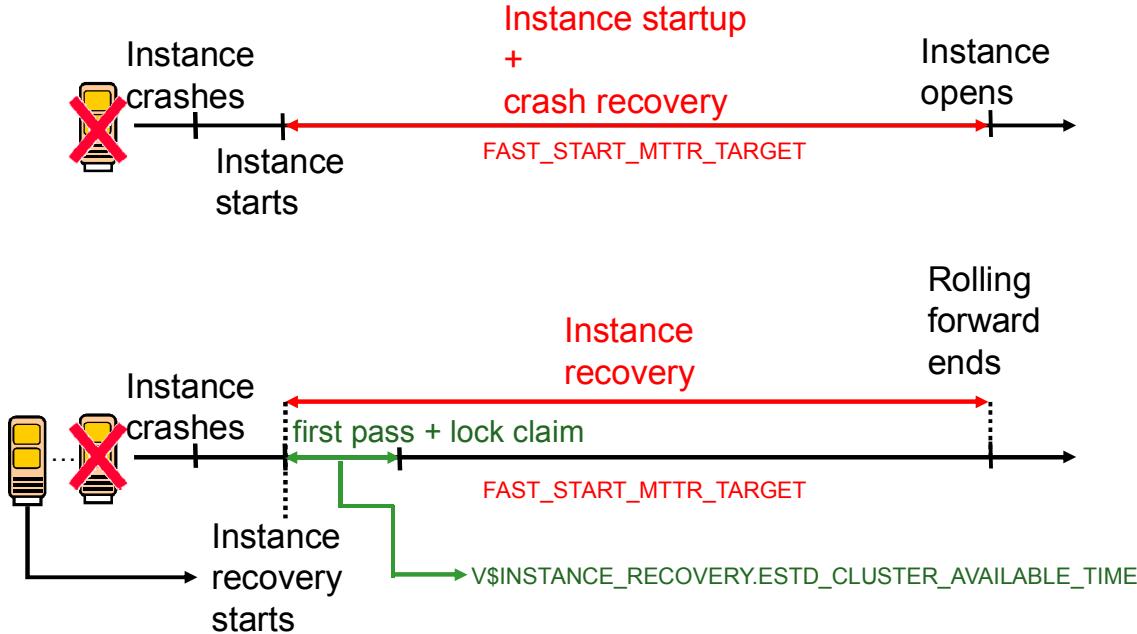
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The graphic illustrates the degree of database availability during each step of Oracle instance recovery:

- Real Application Clusters is running on multiple nodes.
- Node failure is detected.
- The enqueue part of the GRD is reconfigured; resource management is redistributed to the surviving nodes. This operation occurs relatively quickly.
- 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.
- 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.
- 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.
- The Oracle server performs rollback recovery. Undo blocks are applied to the database for all uncommitted transactions.
- 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 © 2014, 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 GCD 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.

Note: 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.

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.

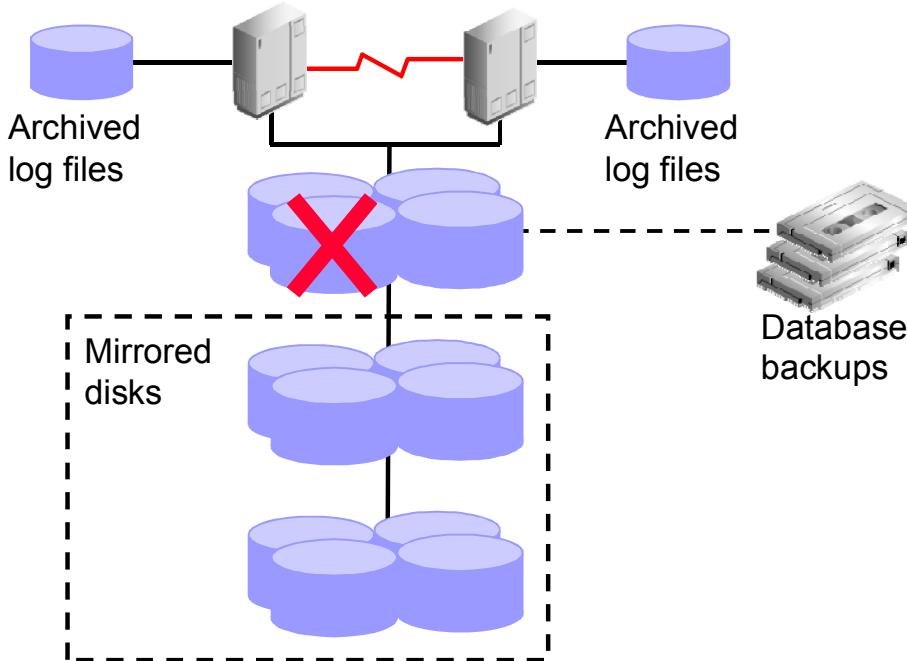


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Here are some guidelines you can use to make sure that instance recovery in your RAC environment is faster:

- Use parallel instance recovery by setting RECOVERY_PARALLISM.
- Set PARALLEL_MIN_SERVERS to CPU_COUNT - 1. This will prespawn recovery slaves at startup time.
- If a system fails when there are uncommitted parallel DML or DDL transactions, you can speed up transaction recovery during startup by setting the FAST_START_PARALLEL_ROLLBACK parameter.
- Using asynchronous I/O is one of the most crucial factors in recovery time. The first-pass log read uses asynchronous I/O.
- Instance recovery uses 50 percent of the default buffer cache for recovery buffers. If this is not enough, some of the steps of instance recovery will be done in several passes. You should be able to identify such situations by looking at your alert.log file. In that case, you should increase the size of your default buffer cache.

Protecting Against Media Failure



ORACLE

Copyright © 2014, 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 © 2014, 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 © 2014, Oracle and/or its affiliates. All rights reserved.

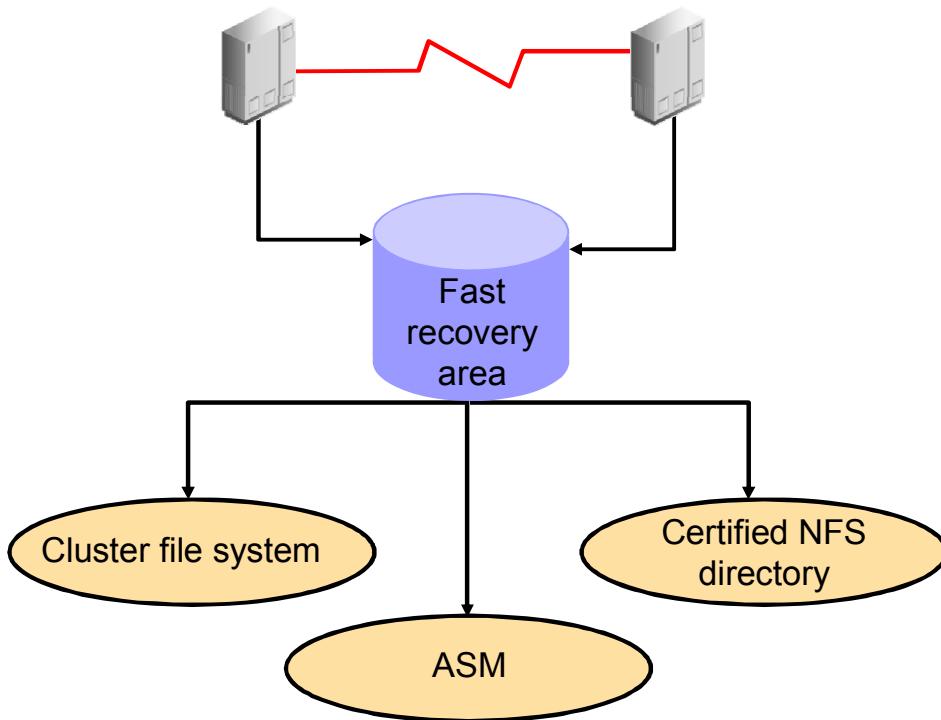
Oracle Database automatically selects the optimum degree of parallelism for instance and crash recovery. Oracle Database applies archived redo logs using an optimal number of parallel processes based on the availability of CPUs. With RMAN's RESTORE and RECOVER commands, Oracle Database automatically 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

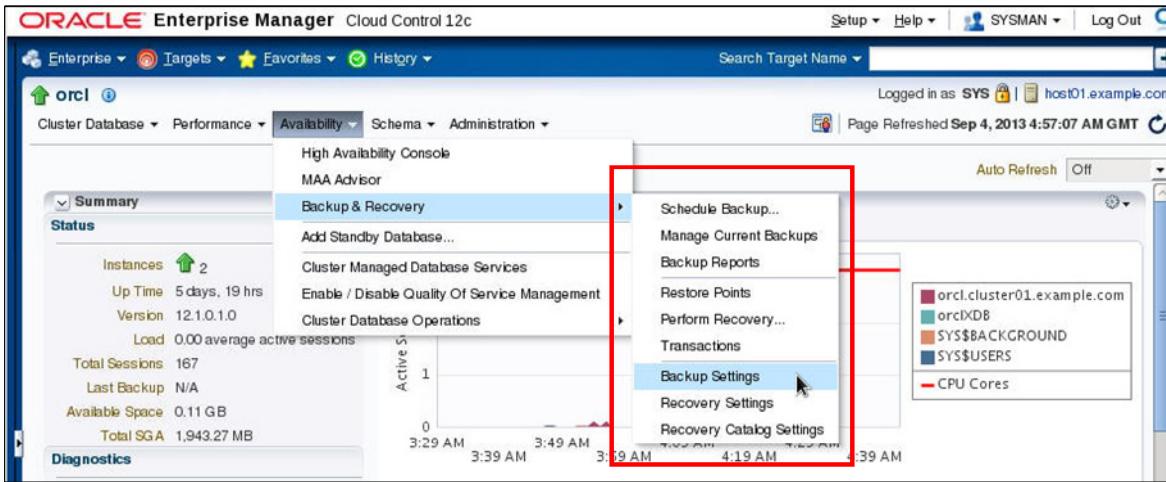
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

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

Media Recovery

The database is currently in NOARCHIVELOG mode. Archived redo log files. If you change the database to ARCHIVELOG mode, all archived redo log files will be lost in the event of database recovery.

ARCHIVELOG Mode

Log Archive Filename Format: %t_%s_%r

Number	Archived Redo Log Destination
1	[USE_DB_RECOVERY_FILE_DEST]

Add Another Row

Fast Recovery Area Usage

File Type	Usage (%)
Redo Log	5.7%
Control File	0.4%
Archived Redo Log	0%
Backup Piece	0%
Image Copy	0%
Flashback Log	0%
Auxiliary Datafile Copy	0%
Usable	94%

ORACLE

Copyright © 2014, 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
%t	Thread number, not padded	log_1
%T	Thread number, left-zero-padded	log_0001
%s	Log sequence number, not padded	log_251
%S	Log sequence number, left-zero-padded	log_000000251
%r	Resetlogs identifier	log_23452345
%R	Padded resetlogs identifier	log_0023452345
%t_%s_%r	Using multiple variables	log_1_251_23452345



Copyright © 2014, 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 is divided into several sections:

- Disk Settings:** Shows 'Parallelism' set to 1 and 'Disk Backup Location' as the current disk backup location. It includes options for 'Backup Set' (selected), 'Compressed Backup Set', and 'Image Copy'.
- Tape Settings:** Shows 'Tape Drives' set to 1 and 'Tape Backup Type' selected as 'Backup Set'. It includes options for 'Compressed Backup'.
- Oracle Secure Backup Domain:** Shows 'Version on Database Server' as Unknown and 'Oracle Secure Backup Domain Target' as Not Defined. It includes a 'Configure' button for 'Backup Storage Selectors'.
- Media Management Settings:** A note stating 'If you need to configure a media manager from a third-party vendor, specify the library parameters.' It includes a 'Media Management Vendor Library Parameters' section.
- Host Credentials:** A note stating 'Supply operating system login credentials to access the target database.' It includes options for 'Credential' (Preferred selected), 'Preferred Credential Name' (Database Host Credentials selected), and 'Credential Details' (Default preferred credentials are not set).

ORACLE

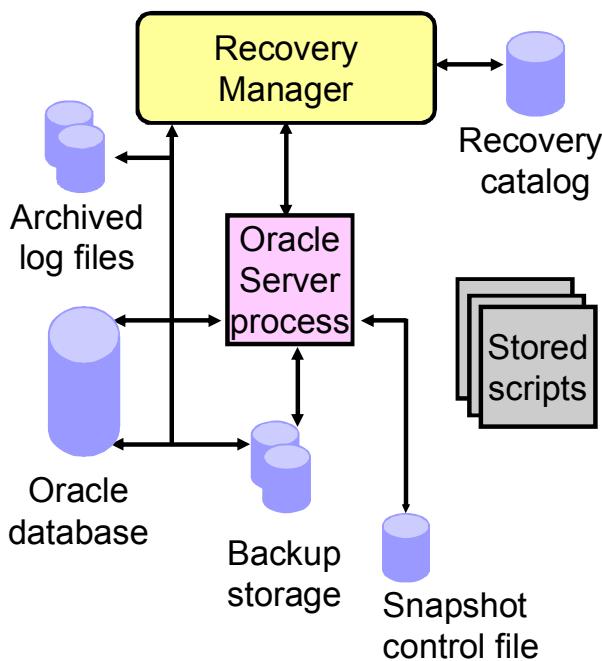
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 © 2014, 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.

If you use different Oracle Home locations for your RAC instances on each of your nodes, create a snapshot control file in a location that exists on all your nodes. The snapshot control file is needed only on the nodes on which RMAN performs backups.

You can use either a cluster file or a local directory that exists on each node in your cluster. Here is an example:

```
RMAN> CONFIGURE SNAPSHOT CONTROLFILE TO
'/oracle/db_files/snaps/snap_prod1.cf';
```

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/12.1.0/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 © 2014, 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 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 © 2014, 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 © 2014, 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 © 2014, 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 © 2014, 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 © 2014, 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 © 2014, 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 a 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 © 2014, Oracle and/or its affiliates. All rights reserved.

When a node generates an archived redo log, Oracle Database always records the filename 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 © 2014, 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_ARCH_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 database -d mydb -n node1 -i sid1  
$ srvctl modify database -d mydb -n node2 -i sid2  
$ srvctl modify database -d mydb -n node3 -i sid3
```



Copyright © 2014, 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_ARCH_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 database -d mydb -n node1 -i sid1  
$ srvctl modify database -d mydb -n node2 -i sid2  
$ srvctl modify database -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 © 2014, 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 list below 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 © 2014, 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 © 2014, 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.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

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 © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 5: Overview

This practice covers the following topics:

- Configuring the archive log mode
- Configuring specific instance connection strings
- Configuring RMAN and performing parallel backups



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

Global Resource Management Concepts



ORACLE®

Copyright © 2014, 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 © 2014, 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 © 2014, Oracle and/or its affiliates. All rights reserved.

Latches and mutexes may only protect access to memory structures if they are accessed by processes in the same instance.

In RAC, latches and mutexes are still used, but for global concurrency control, some additional global enqueues are used to provide protection across instances.

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 © 2014, Oracle and/or its affiliates. All rights reserved.

A master metadata structure contains information about the state of the related resource for each instance in which that resource resides. A shadow metadata structure only contains information about the state of the related resource in the instance containing the shadow metadata.

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 © 2014, 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



Copyright © 2014, 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.

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 © 2014, 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 enqueues 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 © 2014, 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 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 © 2014, 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 © 2014, 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 © 2014, 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 © 2014, Oracle and/or its affiliates. All rights reserved.

Concurrency control handles situations where the same block has multiple images in the same or different buffer caches.

Reduction in physical I/O is achieved by having a global view of buffer cache resources and potentially satisfying I/O requests from any instances cache, rather than reading from disk or writing the same buffer multiple times to disk from different buffer caches

Parallel queries may result in caching parts of a table in each buffer cache and using cache fusion to avoid repeatedly doing direct reads from disk. This is known as “In Memory Parallel Query”. Parts of the table are cached in separate buffer caches, rather than having the blocks cached multiple times in different caches due to cache fusion block transfer. The parallel execution servers in the different instances serve results over the interconnect for the part of the table in their respective instance caches.

Global Cache Management Components

- The LMS_n processes
- Buffers
- Buffer headers
- Global Cache Master Resources
- Global Cache Shadow Resources



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Multiple LMS processes may be used by Oracle RAC depending on the workload size. There are buffer headers for each buffer in each buffer cache. There may be multiple block images for the same database block in the same or different buffer caches.

The mastering instance for a specific database block will have master metadata in the GRD, maintained by LMS_n, describing all block images, specifying the instance and state of each image. The shadow metadata in the GRD, is maintained by LMS_n in the same instance, containing information for each block in the buffer cache not mastered locally.

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 © 2014, 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 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 in order 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 © 2014, 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

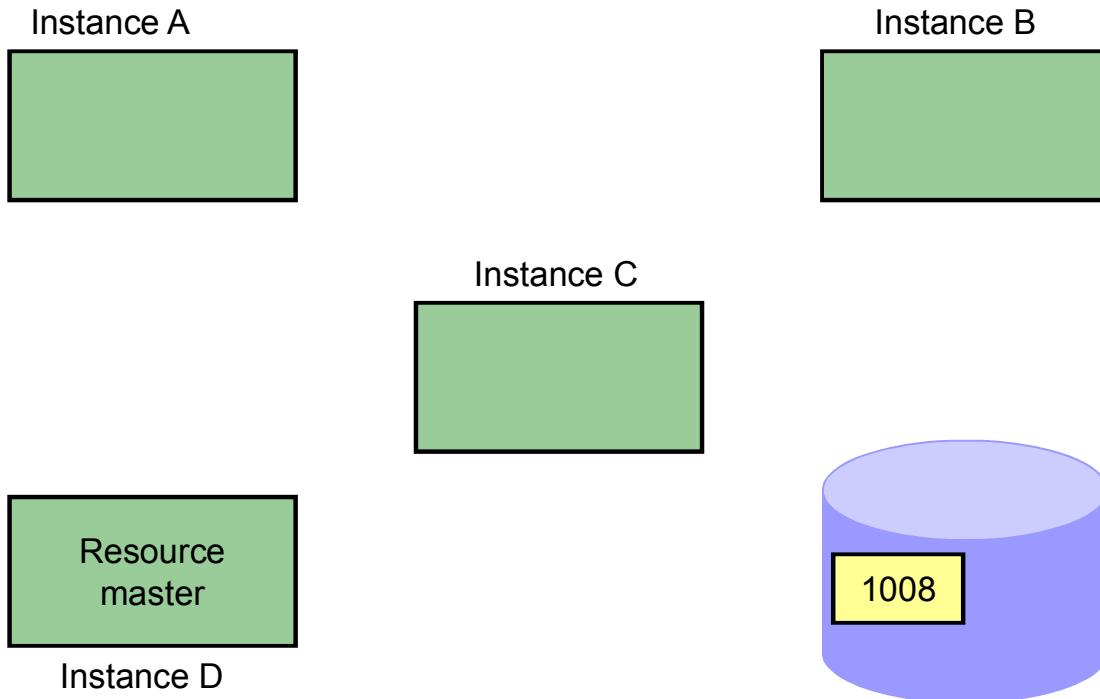
Read – Write: Occurs when a block image exists in at least one buffer cache in shared current state (SCUR), and another instance wishes to access the block for update (XCUR)

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

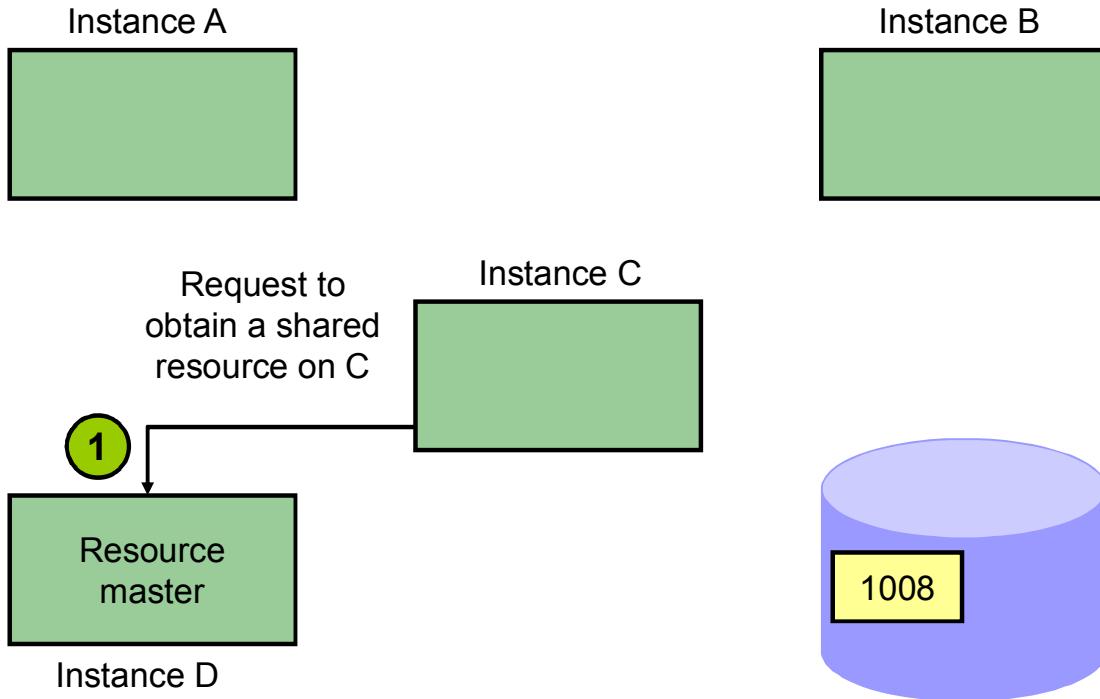
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the following slides, a four-instance database is used to demonstrate various scenarios involving cache fusion processing for a single block.

The mastering instance for this block is D, and the block in the database is at SCN 1008 on disk but with no images in any buffer cache.

Note that the status listed in the boxes representing the instances in the status column from V\$BH and not the GRD metadata, which has its own statuses.

Scenario 1: Read from Disk



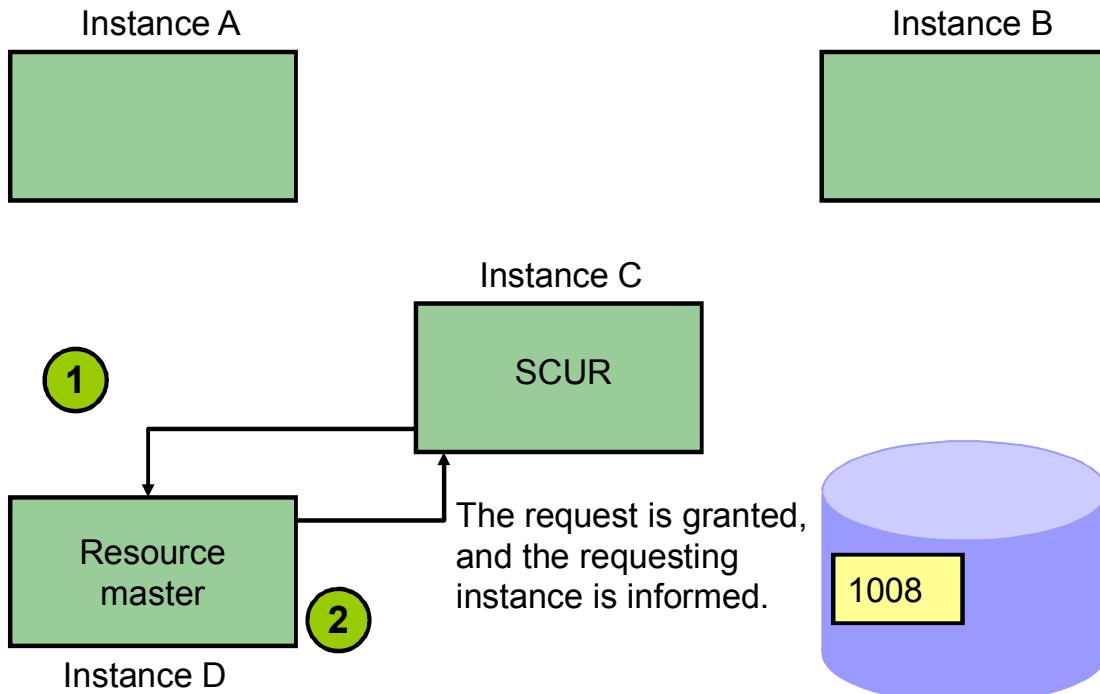
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

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



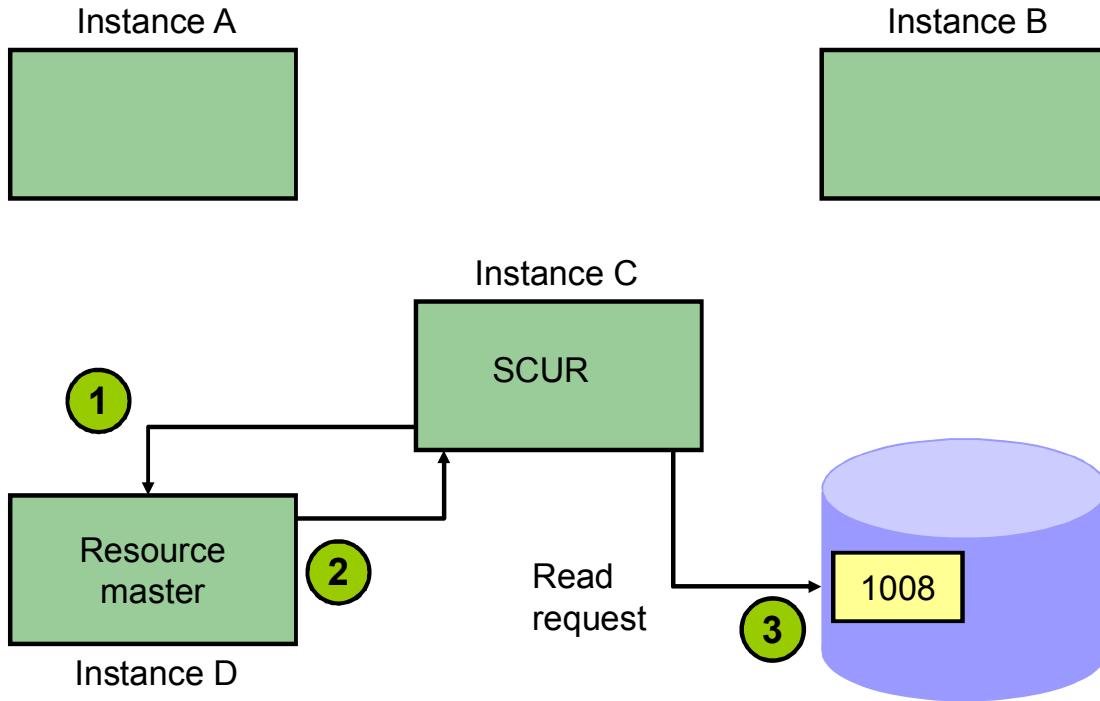
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

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.

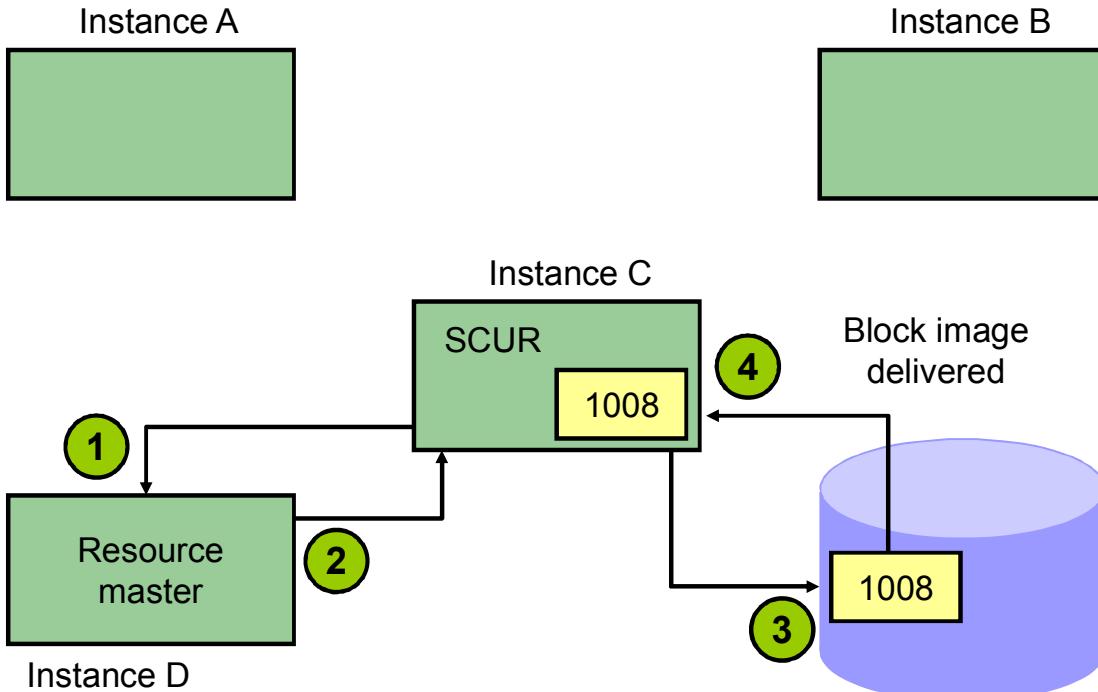
Scenario 1: Read from Disk



Copyright © 2014, 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



ORACLE

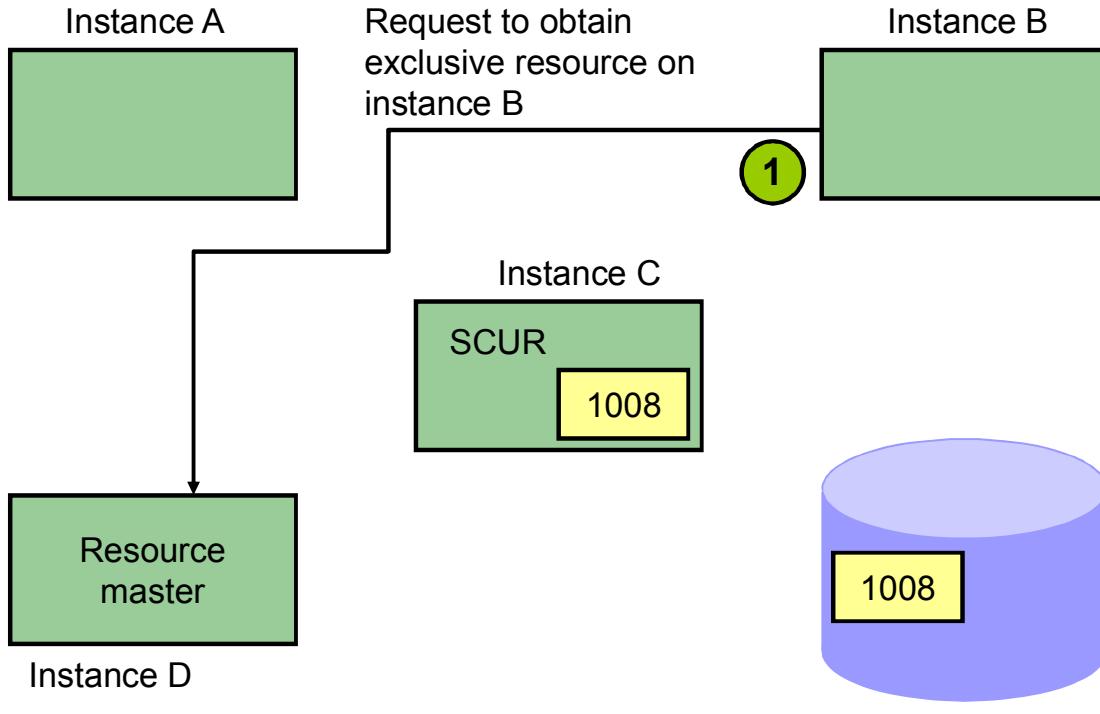
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 wish 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



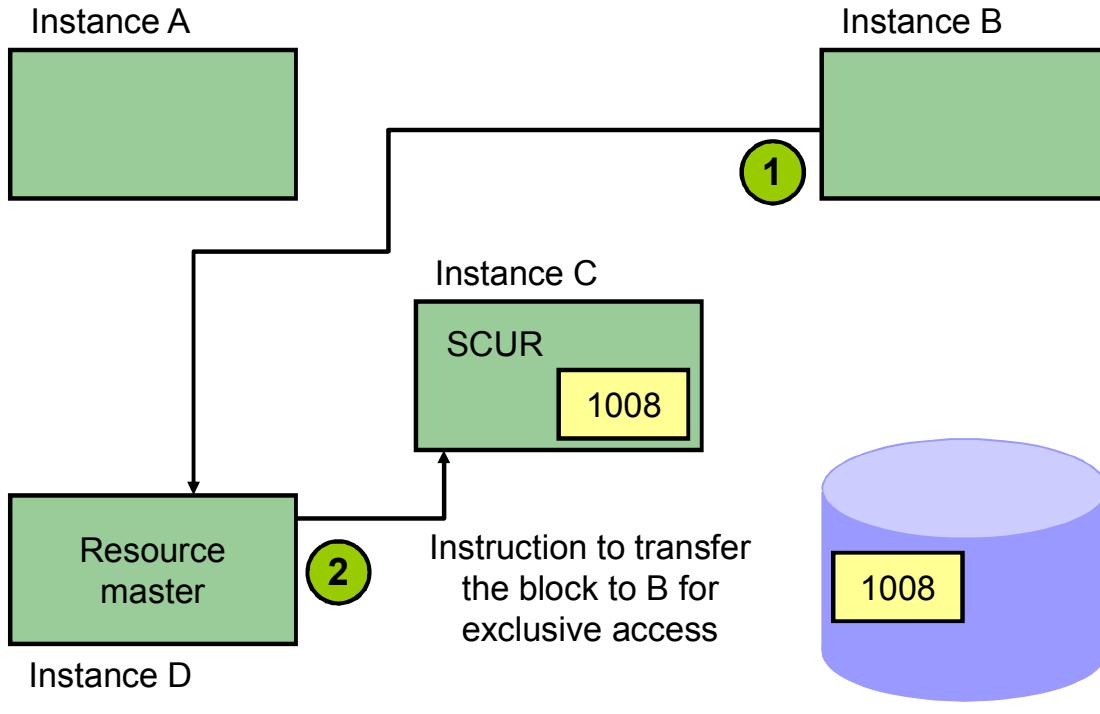
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

Now instance B wishes to read the same block. The server 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 B containing a block image for the block, and there is no shadow metadata for this block in the GRD in instance B.

Scenario 2: Read-Write Cache Fusion



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

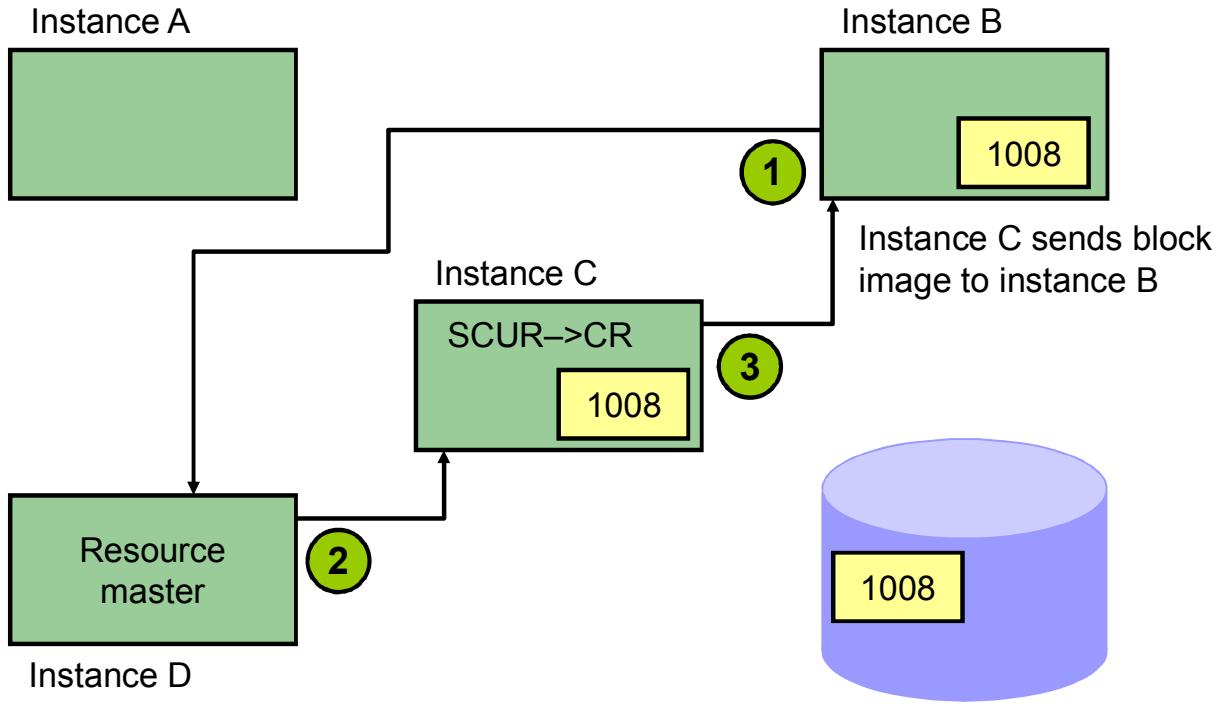
ORACLE

The mastering instance has master metadata, which indicates that instance C has an SCUR image of the block that may be sent to the requesting instance B.

So LMS on instance D sends a request to LMS on instance C, to send an image of the block to instance B over the interconnect.

Note that the instance chosen to serve the block image is determined internally if two or more instances have SCUR images for the same block.

Scenario 2: Read-Write Cache Fusion



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

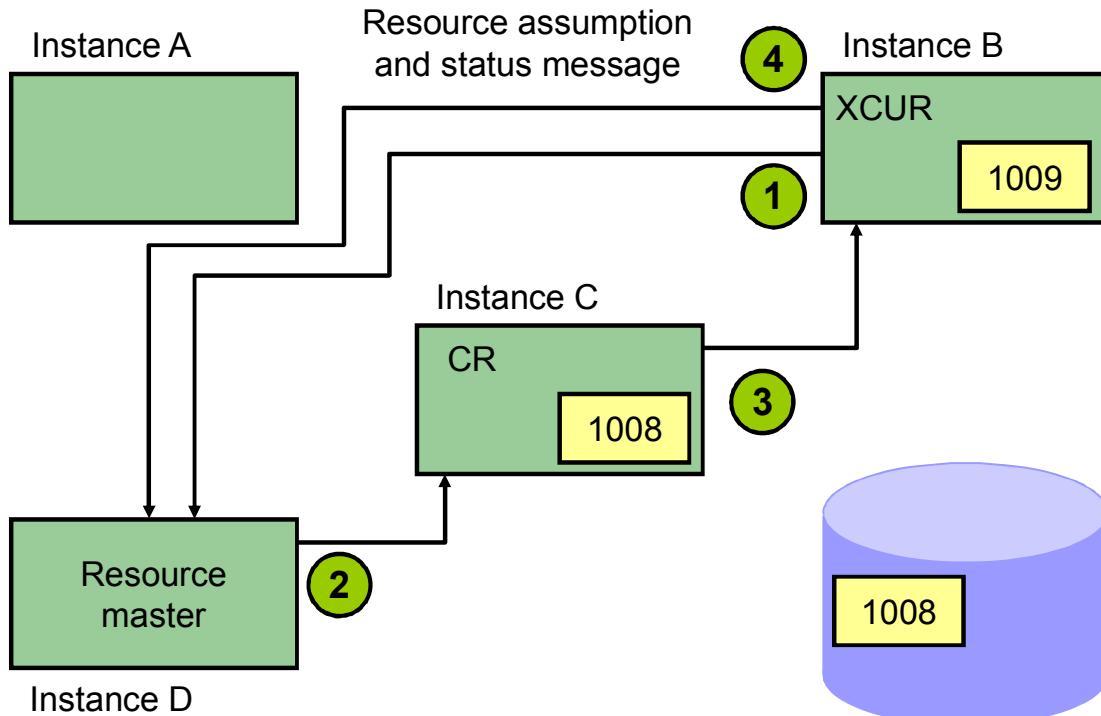
ORACLE

Instance C now sends an image of the block to instance B over the interconnect. The shadow metadata in the GRD in instance C is changed to CR, to reflect that the block image is no longer shared.

Note that if two or more instances have SCUR images for the same block, then they will all be messaged to downgrade them to CR by LMS on the mastering instance D.

Both images have show an SCN of 1008, but this will change when the block image is updated in instance B.

Scenario 2: Read-Write Cache Fusion



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

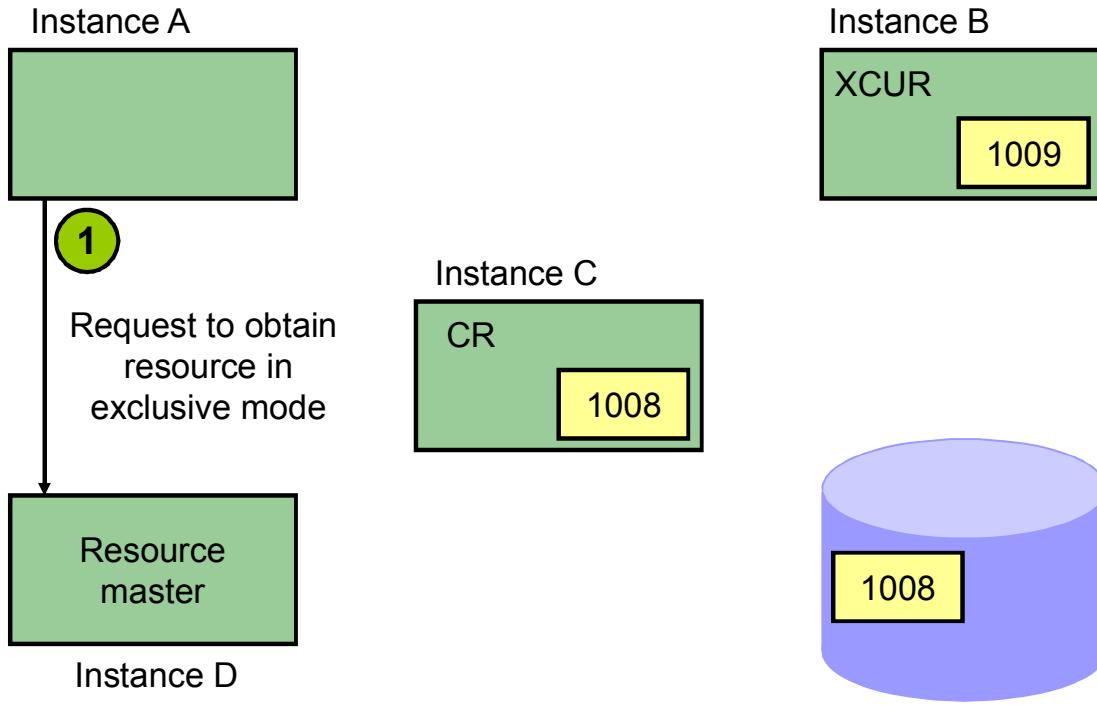
LMS on instance B now sends a status message to mastering instance D, over the interconnect. The shadow metadata in the GRD in instance B reflects that the block image is owned exclusive.

The mastering instance can update the master metadata because there will be two images of the block in different instances in different states.

The image in instance B will be exclusive from the point of view of the GRD and that in instance C will be null, because CR images do not represent blocks that reside on disk currently or that will get written to disk. They are only used for read-consistency purposes.

LMS on instance B now posts the foreground process, and the block is updated. The SCN is now 1009 in instance B but is still 1008 in instance C.

Scenario 3: Write-Write Cache Fusion

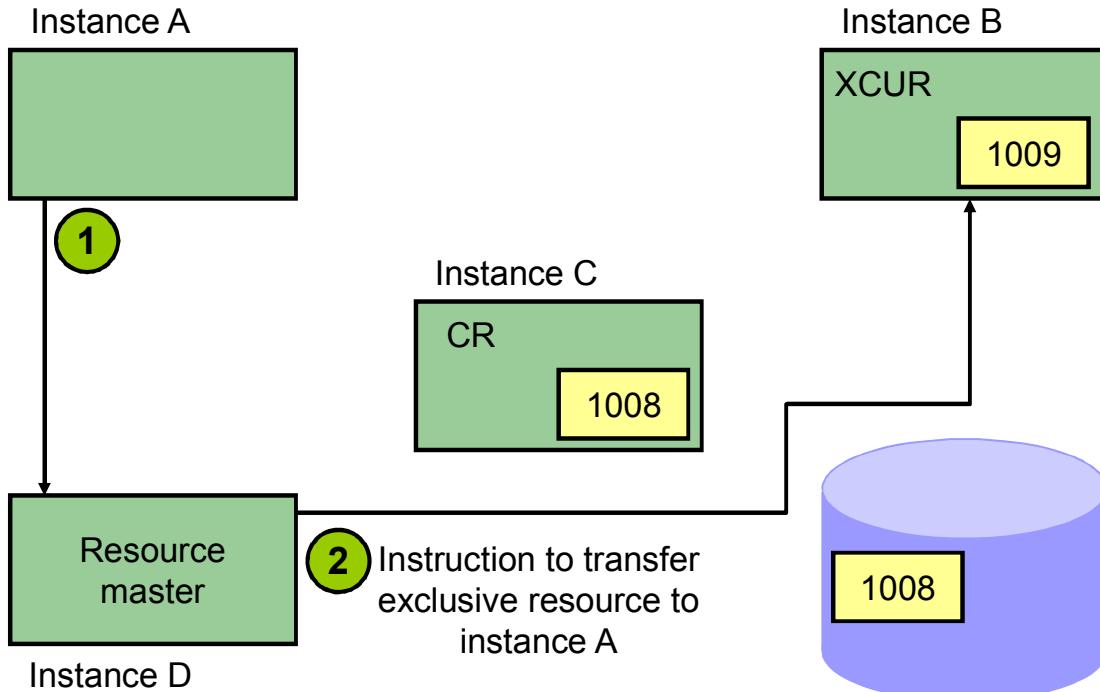


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

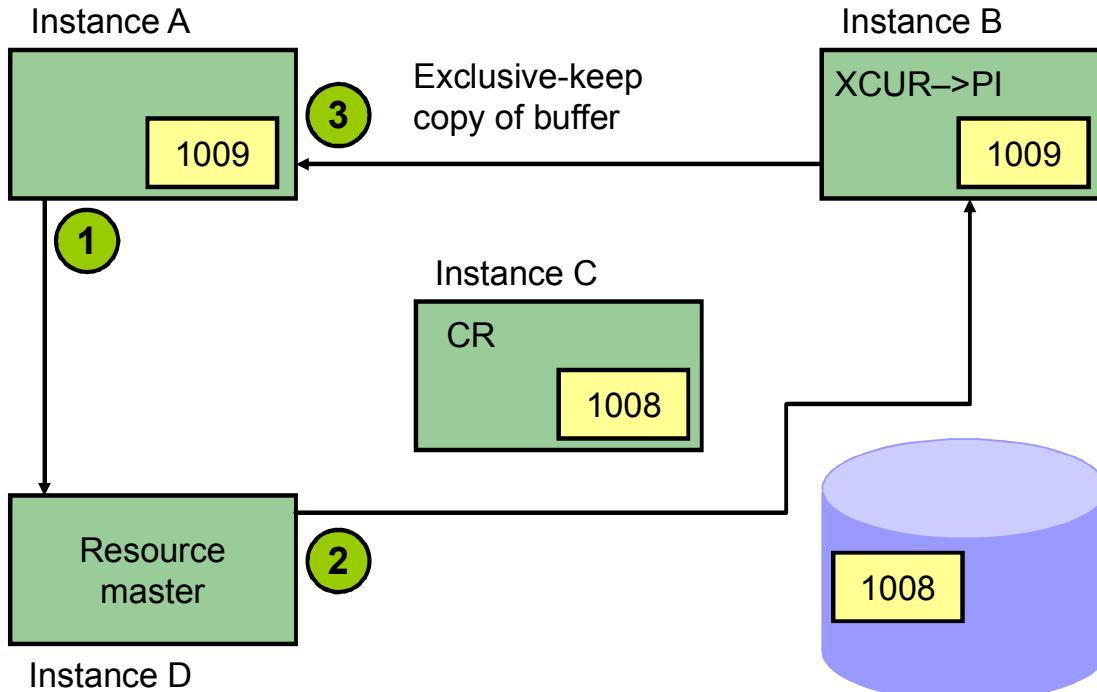
ORACLE

LMS on mastering instance D examines the mastering metadata in the GRD, which indicates that instance B has exclusive ownership of the resource for this block.

LMS on mastering instance D then sends a request to LMS on instance B, requesting that the XCUR image of the block be sent to LMS on instance A, but to keep the past image (PI) of the block.

Instance B must flush redo buffers, if not already written, that contain the change vectors describing the changes made to this block, before sending the block image over the interconnect. This is not reflected in the slide, which only shows the cache fusion. Sending the image to another instance is treated, for recovery purposes, in the same way as writing the block to disk, and the log buffer must be written to permit recovery to work.

Scenario 3: Write-Write Cache Fusion



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

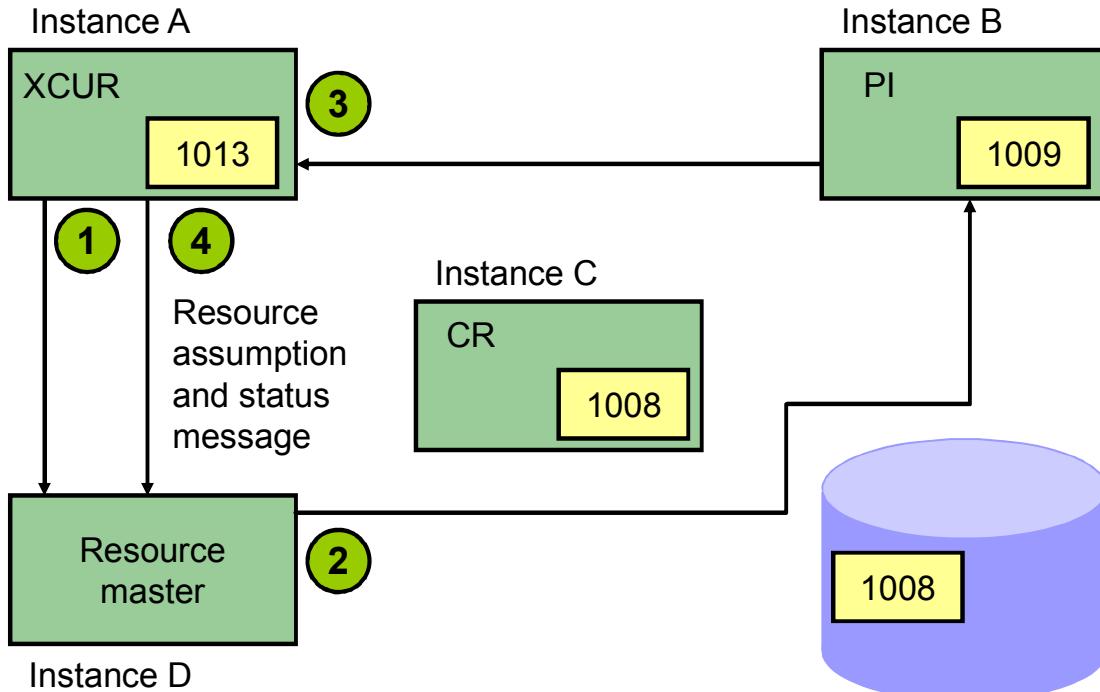
ORACLE

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 instance 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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

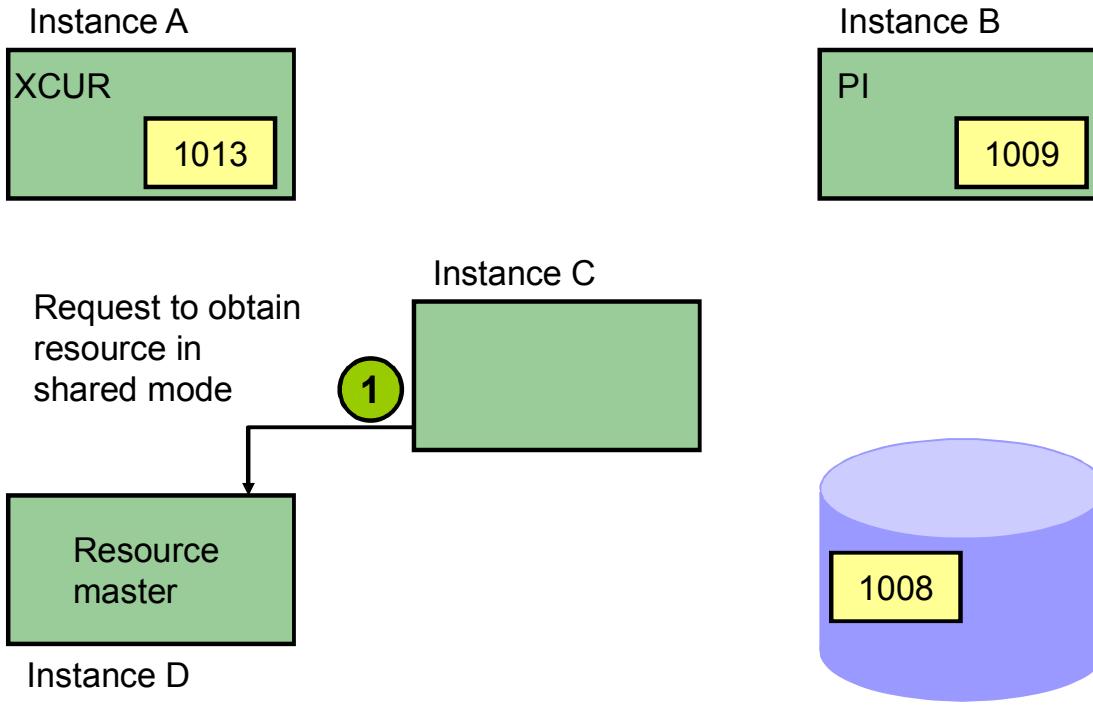
LMS on instance A now sends a status message to the mastering instance D over the interconnect.

The LMS on instance A updates the shadow metadata and posts the foreground process. The status is set to **XCUR** in the buffer header in instance A.

The foreground updates the block and the SCN changes to 1013.

Note that this process may be repeated in a daisy-chain fashion, resulting in an **XCUR** image in one instance and two or more **PI** images for the same block in other instances at different points in time.

Scenario 4: Write-Read Cache Fusion



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

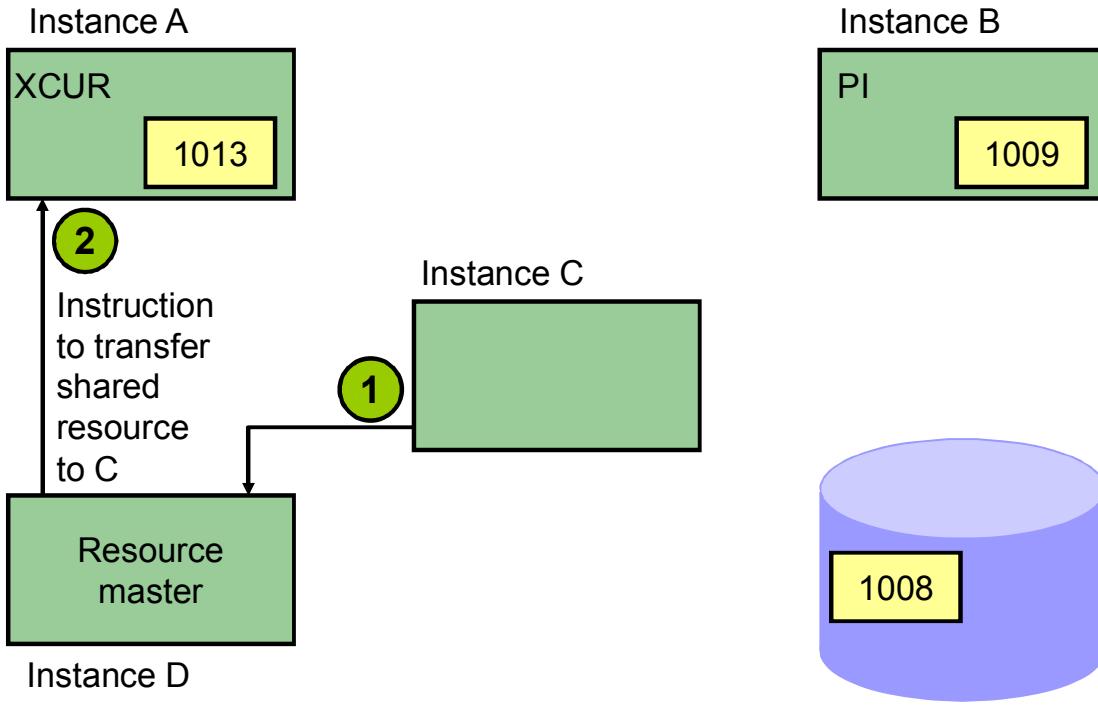
ORACLE

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 DBW n 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 DBW n , 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 LMS n processes in the affected instances communicating with the LMS n 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

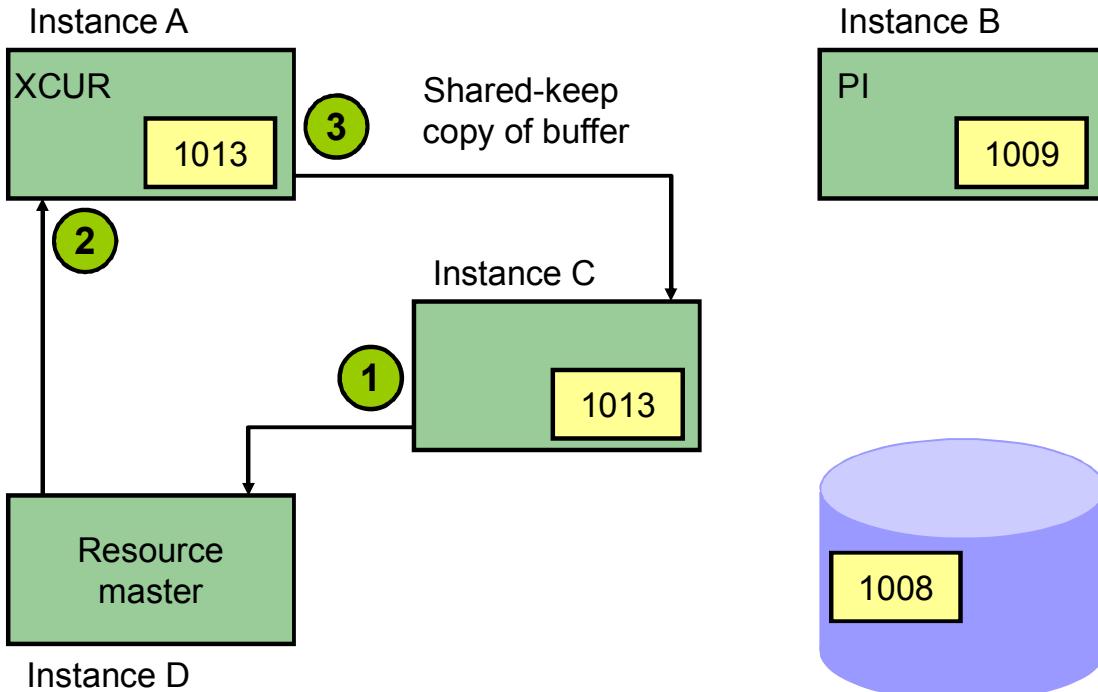


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

LMS on mastering instance D examines the mastering metadata in the GRD, which indicates that instance A has exclusive ownership of the resource for this block.

LMS on mastering instance D then sends a request to LMS on instance A, requesting that an image of the block satisfying the SCN of the query on instance C be sent to instance C.

Scenario 4: Write-Read Cache Fusion



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

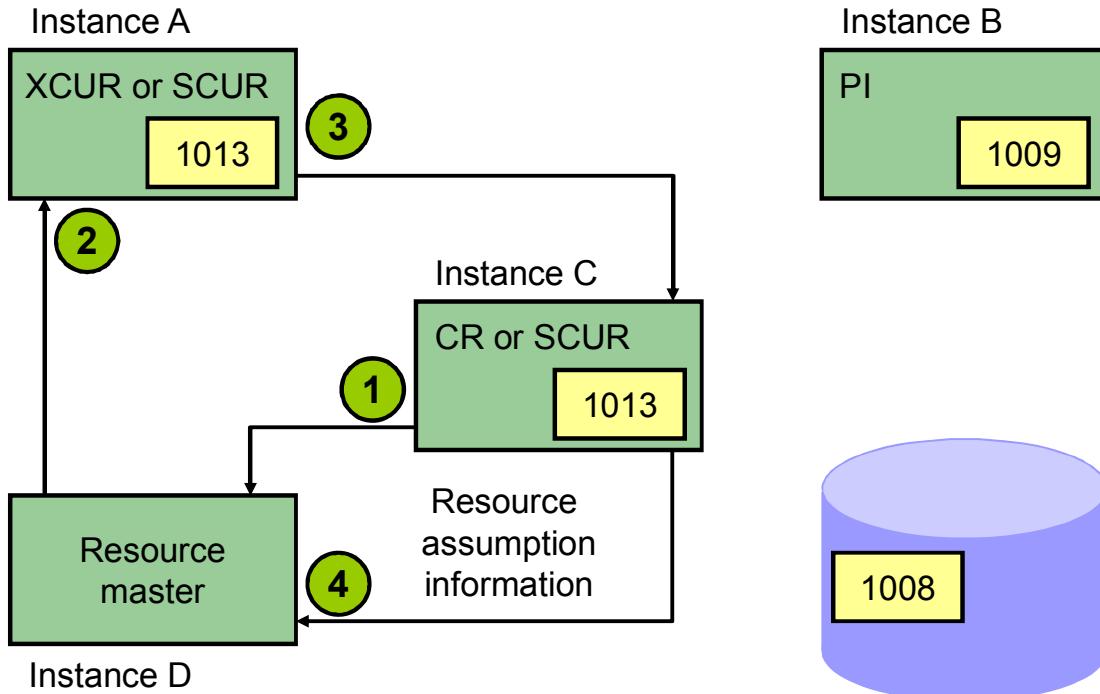
Normally, LMS on instance A builds a CR image by using undo and sends it to instance C. The buffer then remains XCUR in the buffer cache on instance A, even if the XCUR image satisfies the read-consistency requirements. This saves downgrading to SCUR and having to obtain the image in XCUR again if another update occurs in instance A.

But if the block also has PI images, caused by multiple updates on the same block from different instances, it may require multiple rollbacks of undo to create a consistent read image that satisfies the request. If so, then another instance might be requested to build the CR image by the mastering instance.

Please note that in this slide, the SCN for the block images in instance A and C are both 1013, because in this example the XCUR image did not require rolling back to an earlier version of the block.

Note also that the old CR image of this block could still be in the buffer cache of instance C, if it had not aged out yet due to pressure on the replacement list.

Scenario 4: Write-Read Cache Fusion



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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
 - This is done by LMSn coordination from the requesting instance to the LMSn on the mastering instance(s)
 - 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 © 2014, 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
- V\$CR_BLOCK_SERVER
- V\$CURRENT_BLOCK_SERVER
- V\$INSTANCE_CACHE_TRANSFER
- V\$DYNAMIC_REMASTER_STATS
- V\$GCSPFMASTER_INFO



Copyright © 2014, 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.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Statement b is incorrect.

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 © 2014, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

RAC Database Monitoring and Tuning



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

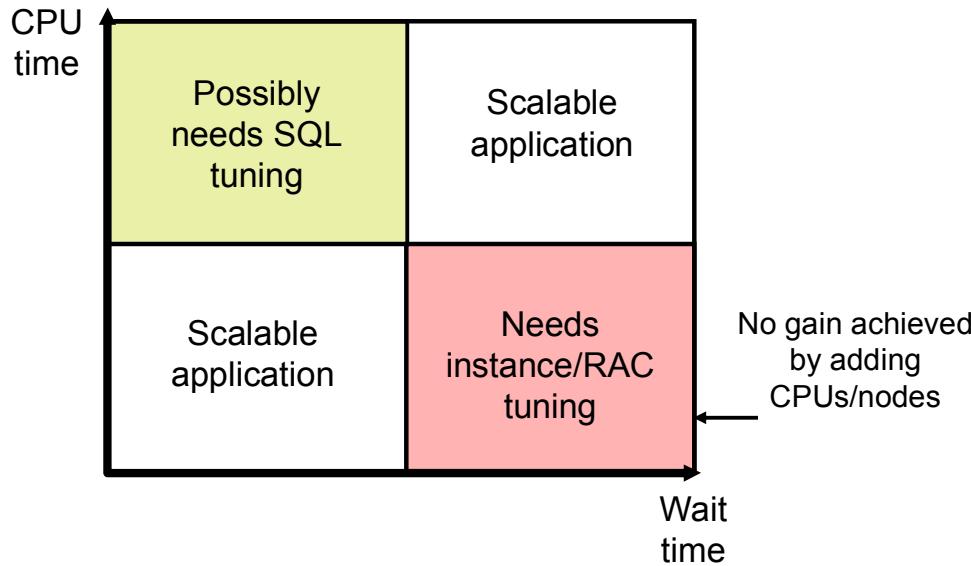
After completing this lesson, you should be able to:

- 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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

CPU and Wait Time Tuning Dimensions



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

When tuning your system, it is important that you compare the CPU time with the wait time of your system. Comparing CPU time with wait time helps to determine how much of the response time is spent on useful work and how much on waiting for resources potentially held by other processes.

As a general rule, the systems where CPU time is dominant usually need less tuning than the ones where wait time is dominant. Alternatively, heavy CPU usage can be caused by badly written SQL statements.

Although the proportion of CPU time to wait time always tends to decrease as load on the system increases, steep increases in wait time are a sign of contention and must be addressed for good scalability.

Adding more CPUs to a node, or nodes to a cluster, would provide very limited benefit under contention. Conversely, a system where the proportion of CPU time to wait time does not decrease significantly as load increases can scale better, and would most likely benefit from adding CPUs or Real Application Clusters (RAC) instances if needed.

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 © 2014, Oracle and/or its affiliates. All rights reserved.

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. At least, this must be your starting point.

Obviously, if you have serialization issues in a single-instance environment, these may be exacerbated with RAC.

As shown in the slide, you have basically the same tuning tools with RAC as with a single-instance system. However, certain combinations of specific wait events and statistics are well-known RAC tuning cases.

In this lesson, you see some of those specific combinations, as well as the RAC-specific information that you can get from the Enterprise Manager performance pages, and Statspack and AWR reports. Finally, you see the RAC-specific information that you can get from the Automatic Database Diagnostic Monitor (ADDM).

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 © 2014, 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 in order 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 © 2014, 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 © 2014, 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	V\$SYSTEM_EVENT
Waits for a wait event class by a session	V\$SESSION_WAIT_CLASS
Waits for an event by a session	V\$SESSION_EVENT
Activity of recent active sessions	V\$ACTIVE_SESSION_HISTORY
Last 10 wait events for each active session	V\$SESSION_WAIT_HISTORY
Events for which active sessions are waiting	V\$SESSION_WAIT
Identify SQL statements impacted by interconnect latencies	V\$SQLSTATS

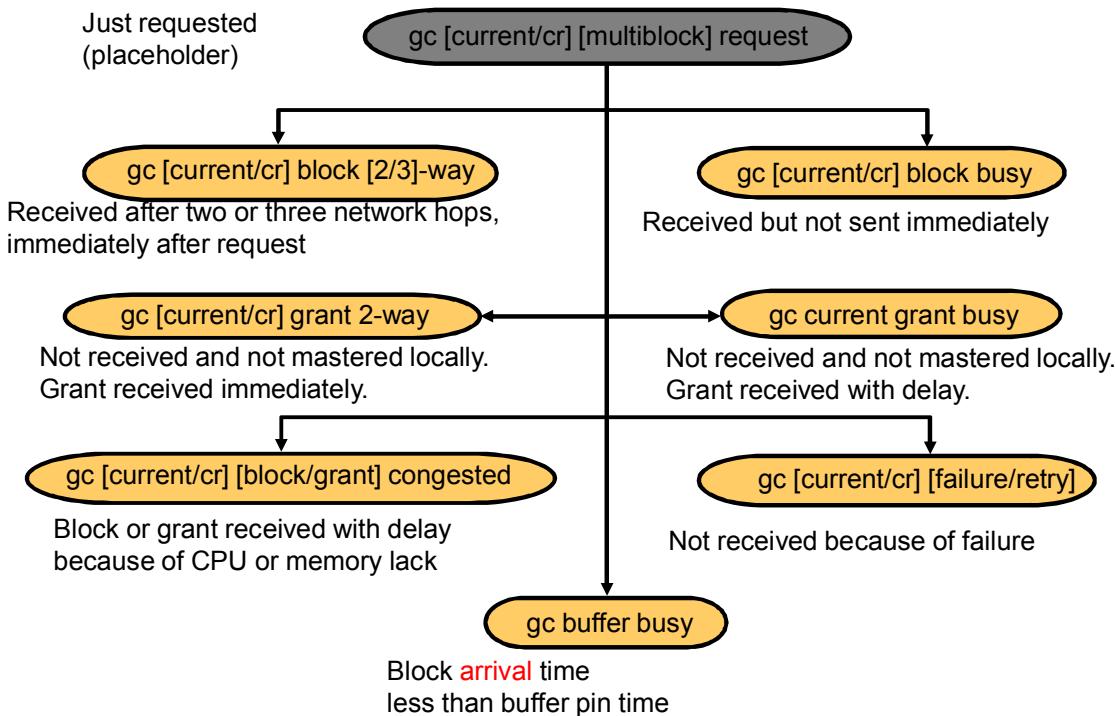
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 V\$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 © 2014, 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.

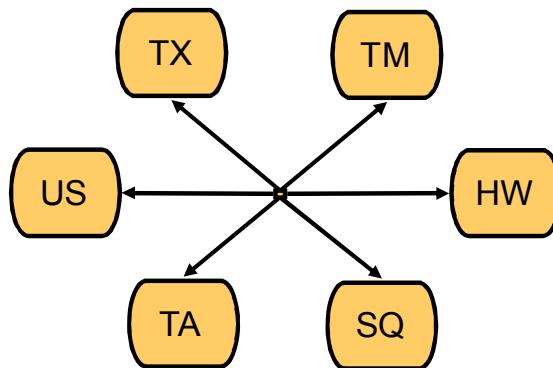
Note: For dynamic remastering, two events are of most importance: `gc remaster` and `gc quiesce`. They can be symptoms of the impact of remastering on the running processes.

- **gc [current/cr] [block/grant] congested:** A current or cr block is requested and a block or grant message 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.

ORACLE

Copyright © 2014, 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 V\$SYSSTAT to characterize the workload.
- Use V\$SESSTAT to monitor important sessions.
- V\$SEGMENT_STATISTICS includes RAC statistics.
- RAC-relevant statistic groups are:
 - Global Cache Service statistics
 - Global Enqueue Service statistics
 - Statistics for messages sent
- V\$ENQUEUE_STATISTICS determines the enqueue with the highest impact.
- V\$INSTANCE_CACHE_TRANSFER breaks down GCS statistics into block classes.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Using system statistics based on V\$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.

In order to drill down to individual sessions or groups of sessions, V\$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 V\$SESSION.

V\$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*

V\$ENQUEUE_STATISTICS can be queried to determine which enqueue has the highest impact on database service times and, eventually, response times.

V\$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.



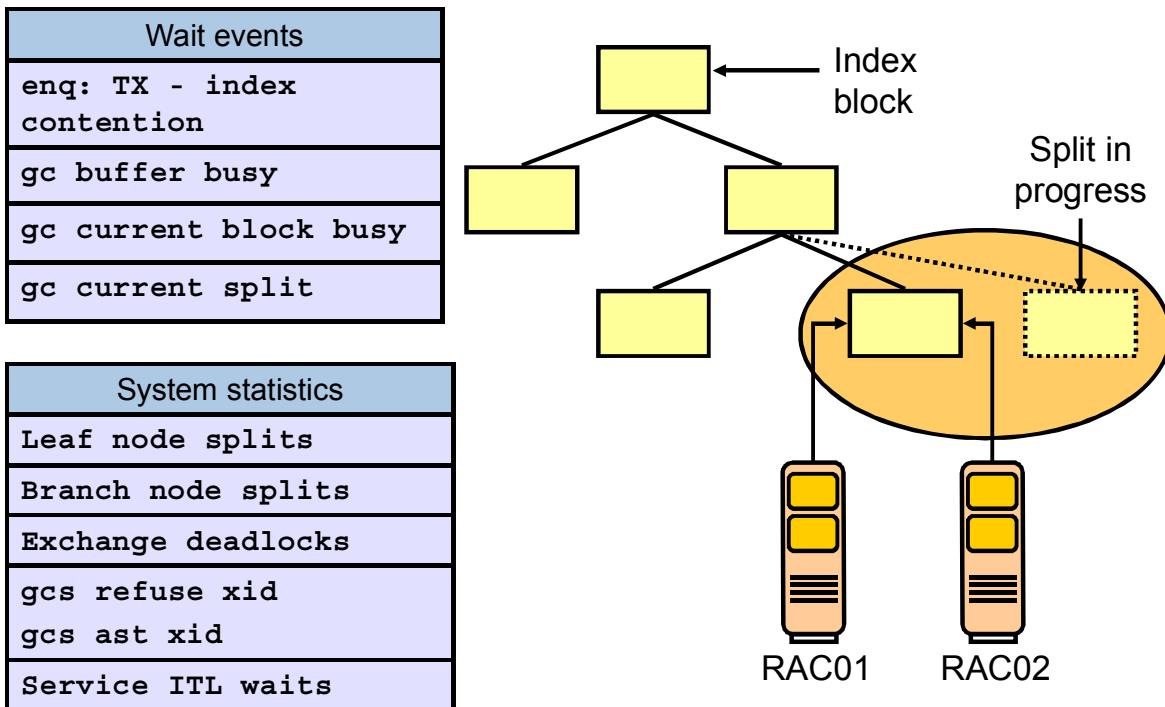
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 © 2014, Oracle and/or its affiliates. All rights reserved.

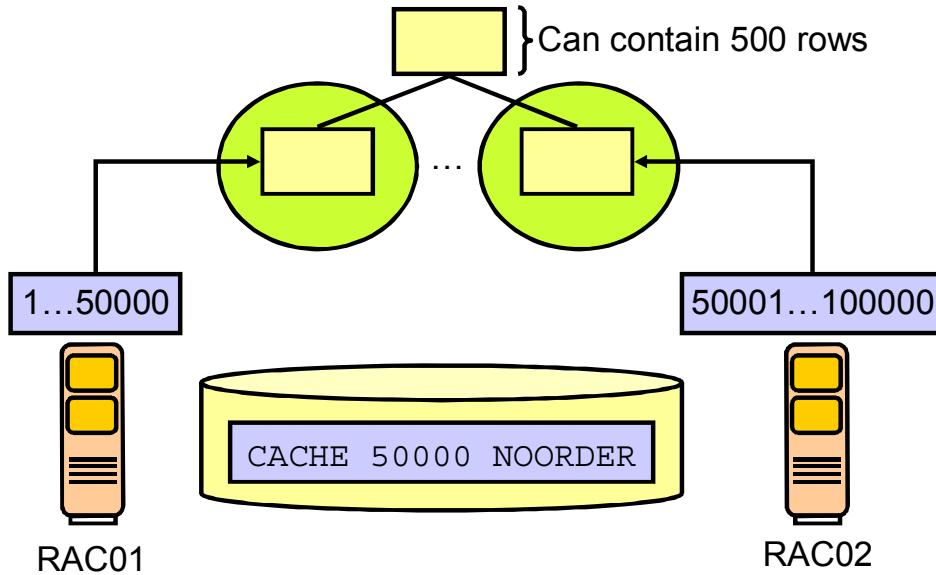
In application systems where the loading or batch processing of data is a dominant business function, there may be performance issues affecting response times because of the high volume of data inserted into indexes. Depending on the access frequency and the number of processes concurrently inserting 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 © 2014, Oracle and/or its affiliates. All rights reserved.

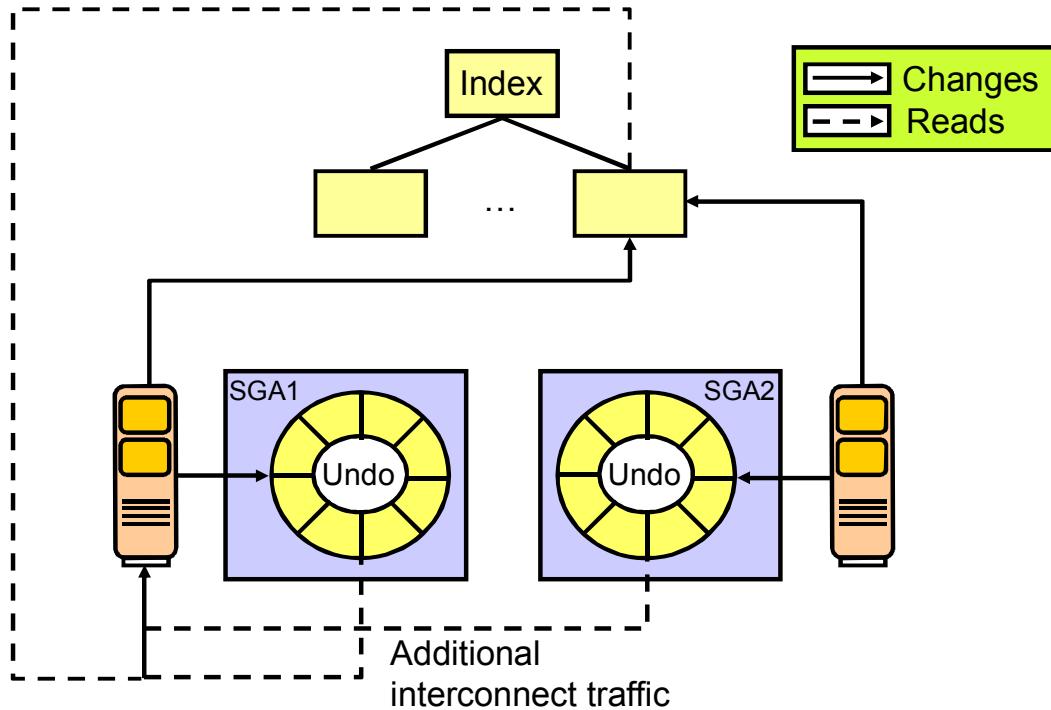
Indexes with key values generated by sequences tend to be subject to leaf block contention when the insert rate is high. That is because the index leaf block holding the highest key value is changed for every row inserted, as the values are monotonically ascending. In RAC, this may lead to a high rate 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.

Undo Block Considerations



ORACLE

Copyright © 2014, 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.

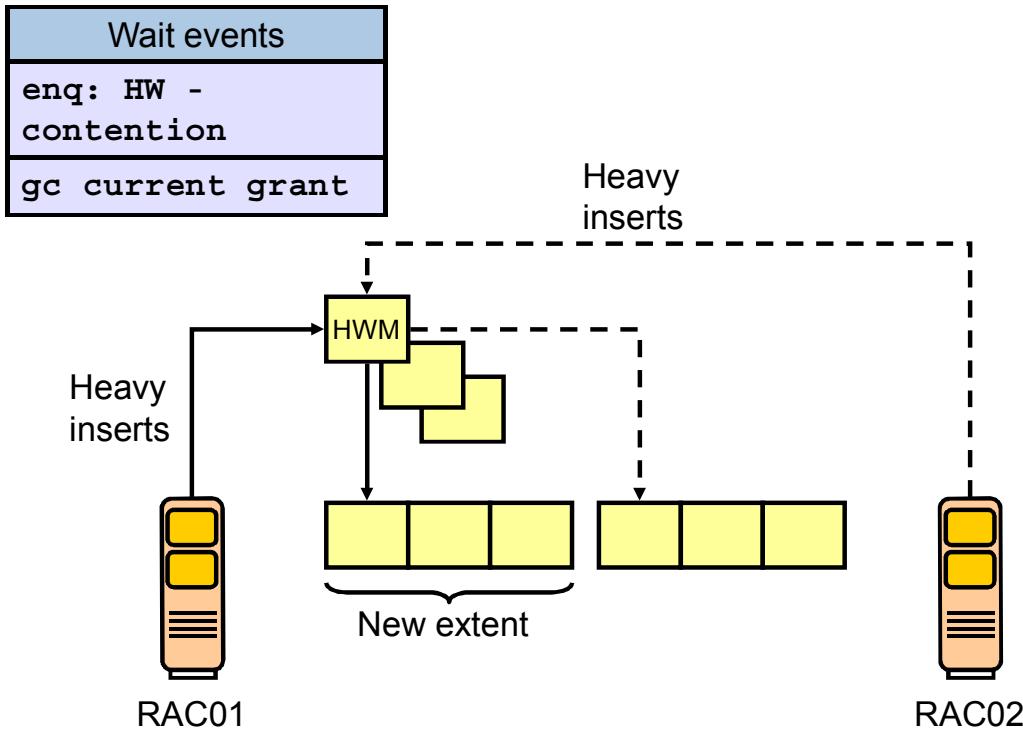
When a `SELECT` statement needs to read a block with active transactions, it has to undo the changes to create a CR version. If the active transactions in the block belong to more than one instance, there is a need to combine local and remote undo information for the consistent read. Depending on the 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 © 2014, Oracle and/or its affiliates. All rights reserved.

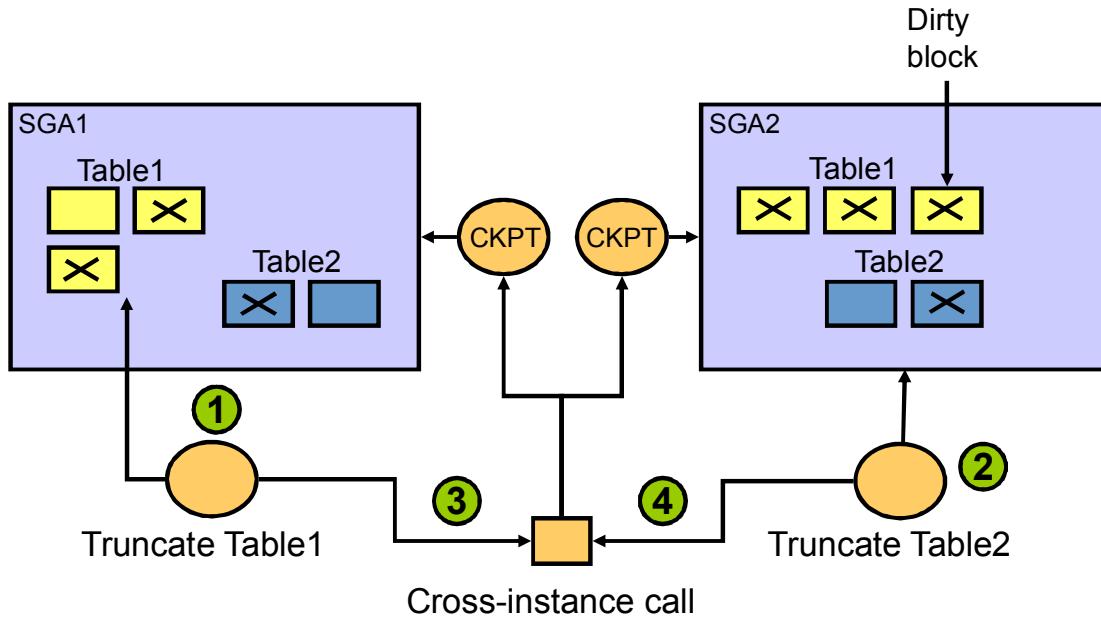
A certain combination of wait events and statistics presents itself in applications where the insertion of data is a dominant business function and new blocks have to be allocated frequently to a segment. If data is inserted at a high rate, new blocks may have to be made available after unfruitful searches 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 © 2014, 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.

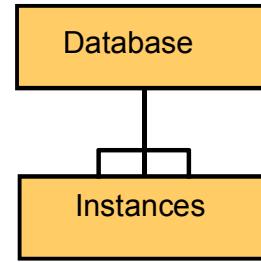
There are different types of cross-instance calls. However, all use the same serialization mechanism.

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.



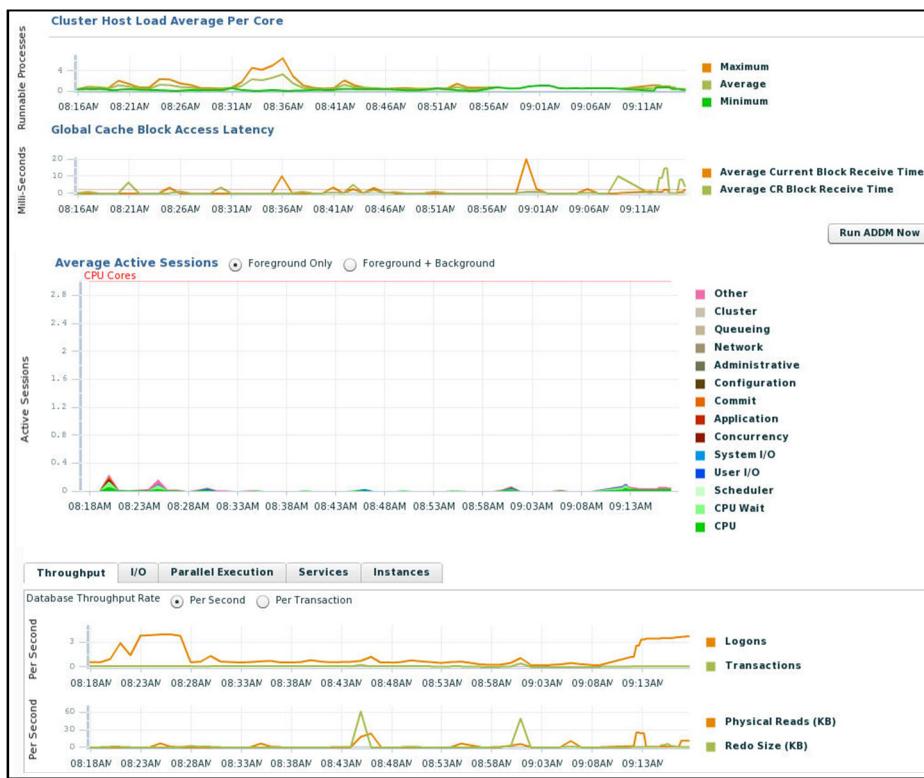
ORACLE

Copyright © 2014, 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 © 2014, Oracle and/or its affiliates. All rights reserved.

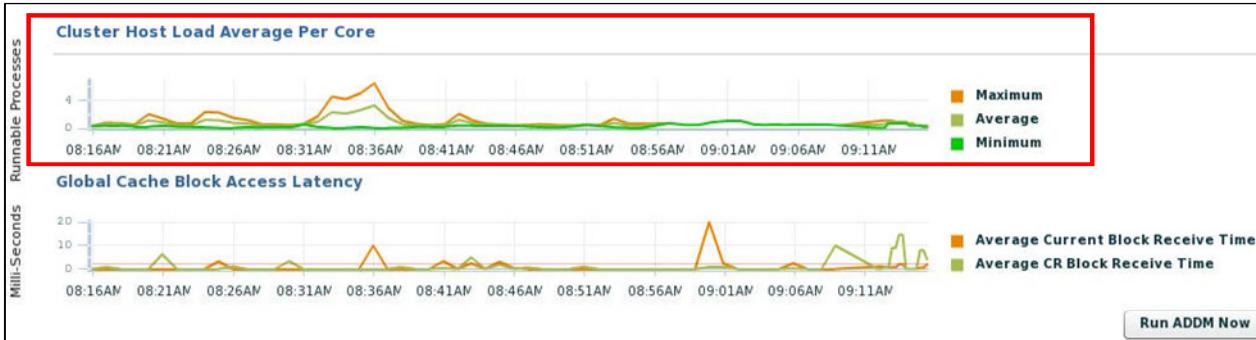
The Cluster Database Performance page provides a quick glimpse of the performance statistics for a database. Enterprise Manager accumulates data from each instance over specified periods of time, called collection-based data. Enterprise Manager also provides current data from each instance, known as 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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

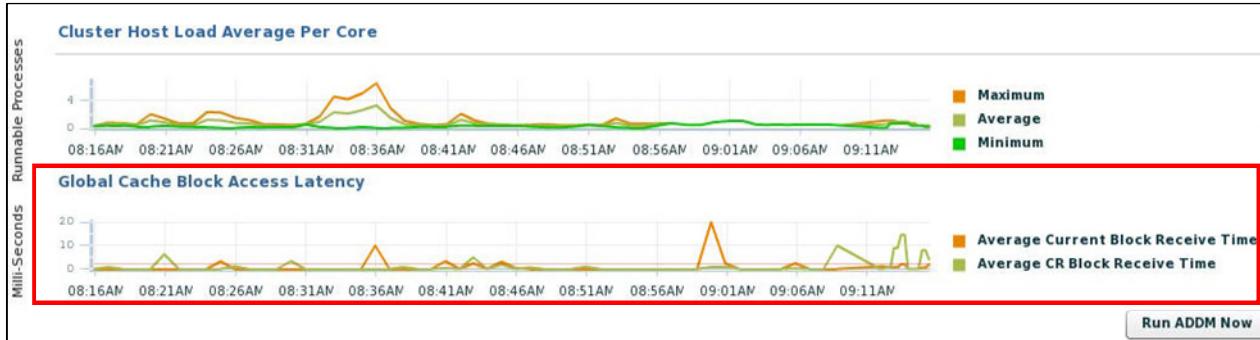
The Cluster Host Load Average chart in the Cluster Database Performance page shows potential problems that are outside the database. The chart shows maximum, average, and minimum load values for available nodes in the cluster for the previous hour.

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



ORACLE

Copyright © 2014, 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 © 2014, 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 in “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

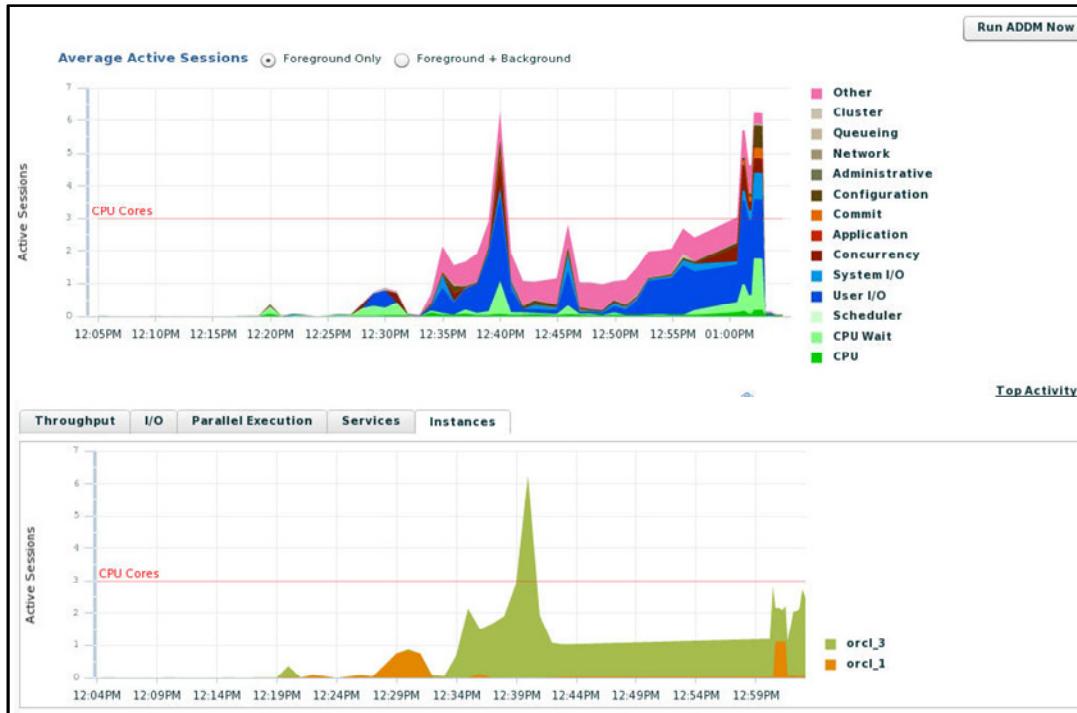
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 Sessions subpage of the Top Consumers page for that instance.

Determining Database Throughput



ORACLE

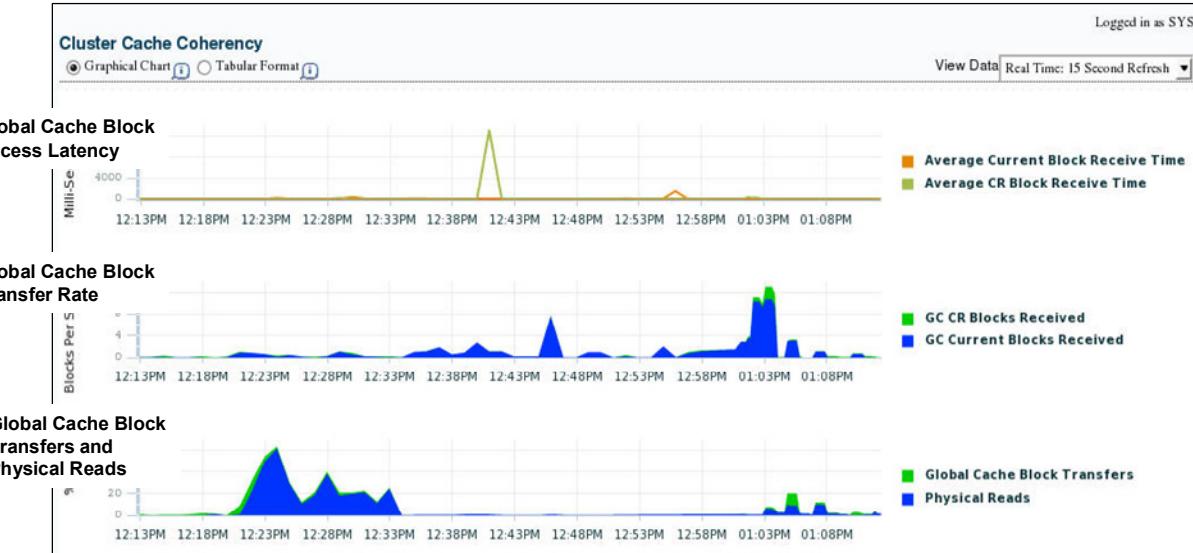
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 © 2014, 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

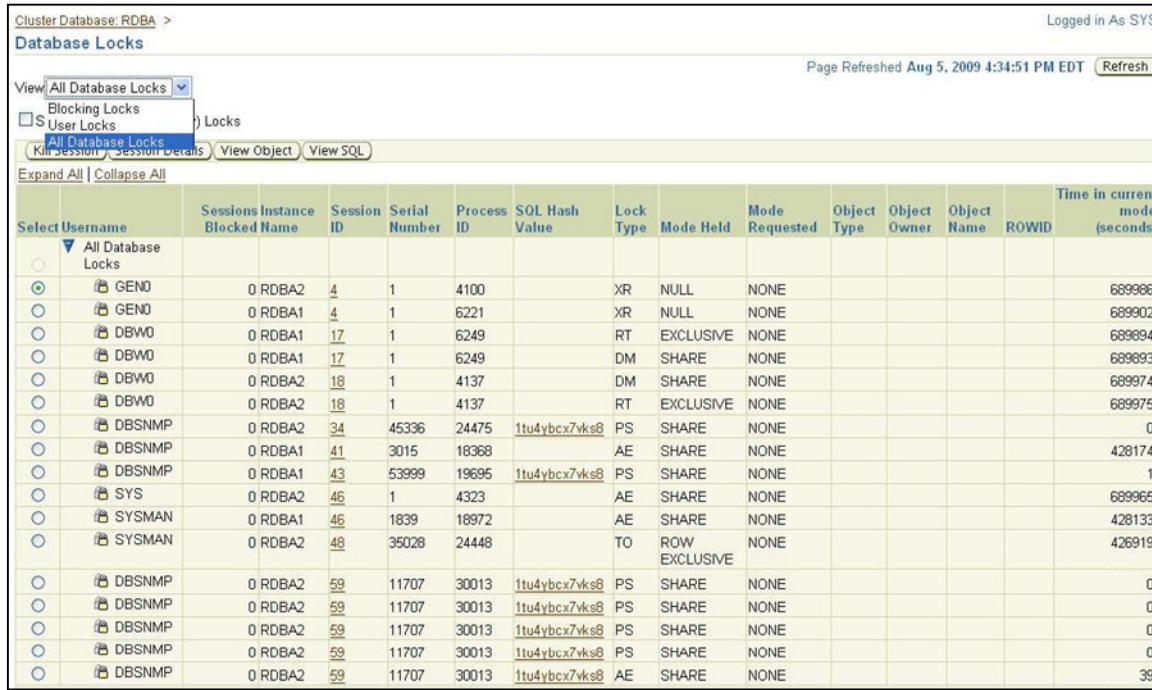


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- **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



The screenshot shows the 'Database Locks' page from the Oracle Database Performance Monitoring interface. The page title is 'Cluster Database: RDBA > Database Locks'. It displays a grid of locks across multiple sessions and instances. The columns include: Select Username, Sessions Blocked, Instance Name, Session ID, Serial Number, Process ID, SQL Hash Value, Lock Type, Mode Held, Mode Requested, Object Type, Object Owner, Object Name, ROWID, and Time in current mode (seconds). The 'All Database Locks' link in the top left is highlighted.

Select Username	Sessions Blocked	Instance Name	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														
GEND	0	RDBA2	4	1	4100		XR	NULL	NONE					689986
GEND	0	RDBA1	4	1	6221		XR	NULL	NONE					689902
DBW0	0	RDBA1	17	1	6249		RT	EXCLUSIVE	NONE					689894
DBW0	0	RDBA1	17	1	6249		DM	SHARE	NONE					689893
DBW0	0	RDBA2	18	1	4137		DM	SHARE	NONE					689974
DBW0	0	RDBA2	18	1	4137		RT	EXCLUSIVE	NONE					689975
DBSNMP	0	RDBA2	34	45336	24475	1tu4ybcx7vks8	PS	SHARE	NONE					0
DBSNMP	0	RDBA1	41	3015	18368		AE	SHARE	NONE					428174
DBSNMP	0	RDBA1	43	53999	19695	1tu4ybcx7vks8	PS	SHARE	NONE					1
SYS	0	RDBA2	46	1	4323		AE	SHARE	NONE					689965
SYSMAN	0	RDBA1	46	1839	18972		AE	SHARE	NONE					428133
SYSMAN	0	RDBA2	48	35028	24448		TO	ROW EXCLUSIVE						426919
DBSNMP	0	RDBA2	59	11707	30013	1tu4ybcx7vks8	PS	SHARE	NONE					0
DBSNMP	0	RDBA2	59	11707	30013	1tu4ybcx7vks8	PS	SHARE	NONE					0
DBSNMP	0	RDBA2	59	11707	30013	1tu4ybcx7vks8	PS	SHARE	NONE					0
DBSNMP	0	RDBA2	59	11707	30013	1tu4ybcx7vks8	PS	SHARE	NONE					0
DBSNMP	0	RDBA2	59	11707	30013	1tu4ybcx7vks8	AE	SHARE	NONE					39

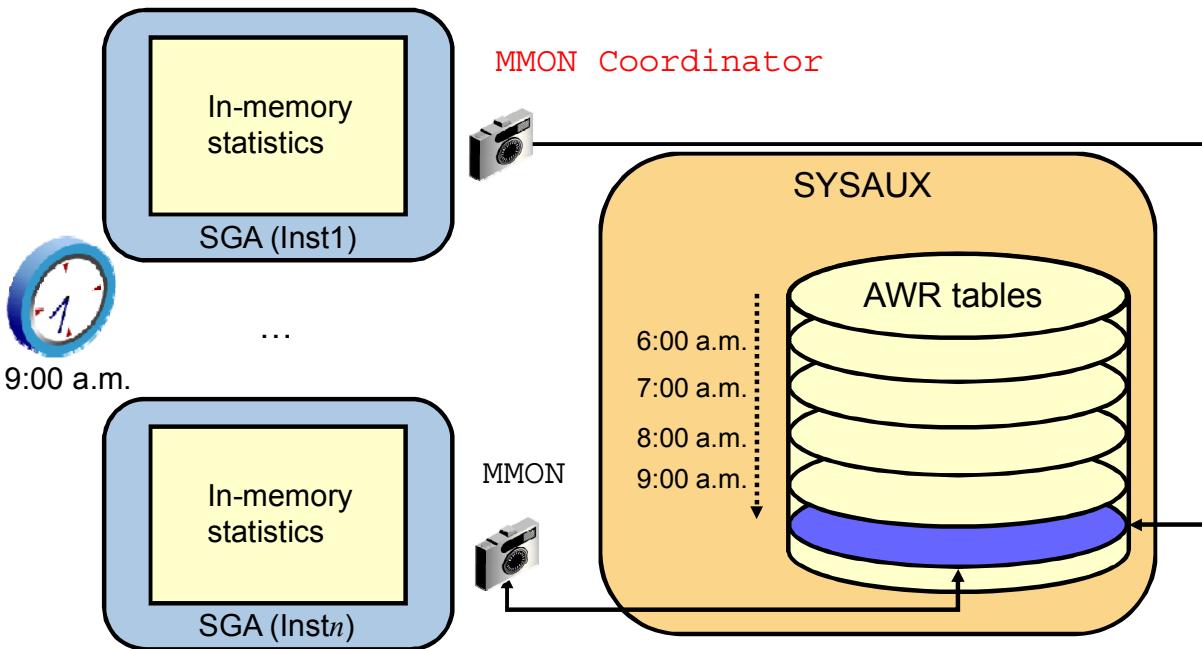


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

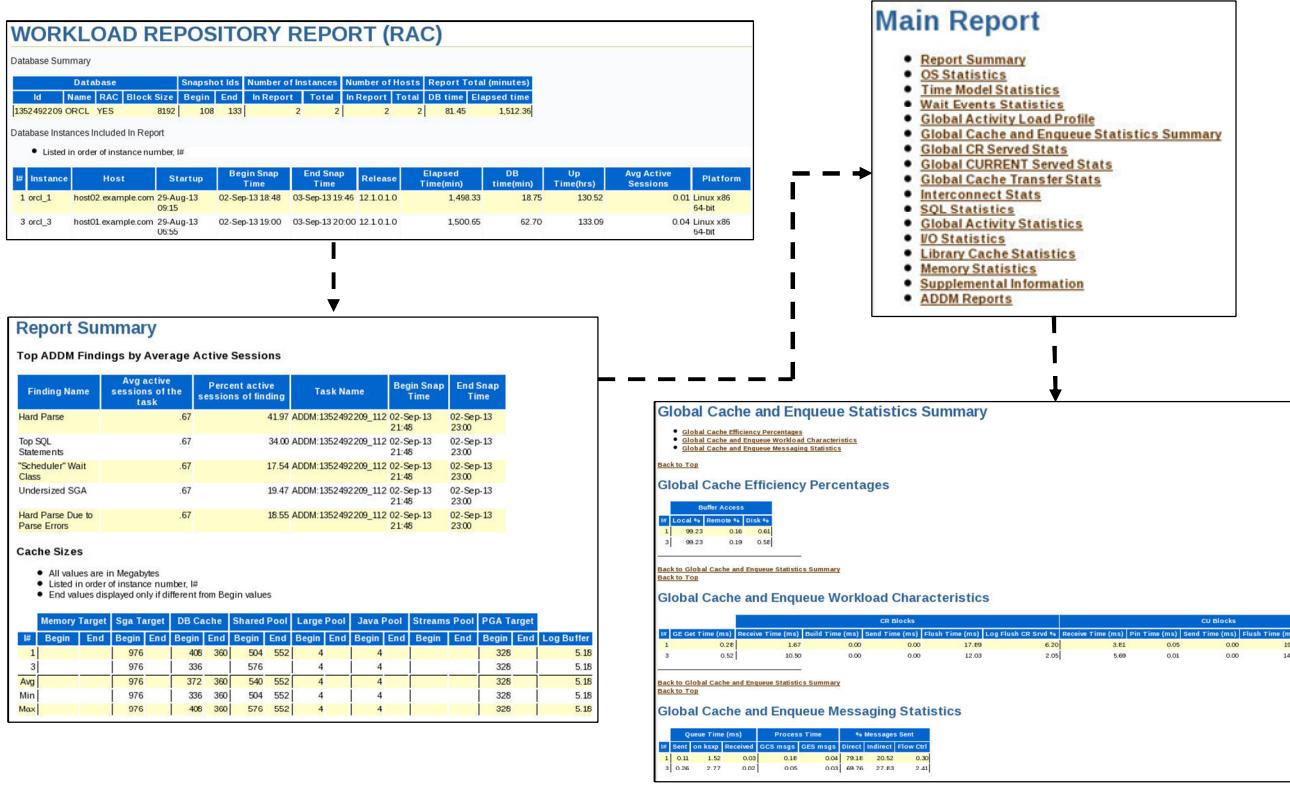
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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	1352492200	orcl_3	1	3.12.1.0.1.0	YES	host01example.com
CPU	SGA Size	Buffer Cache	Shared Pool	ASH Buffer Size		
1	972M (100%)	524M (53.9%)	364M (39.5%)	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 Sessions](#)
- [Top Objects/Files/Latches](#)
- [Activity Over Time](#)

Copyright © 2014, 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.

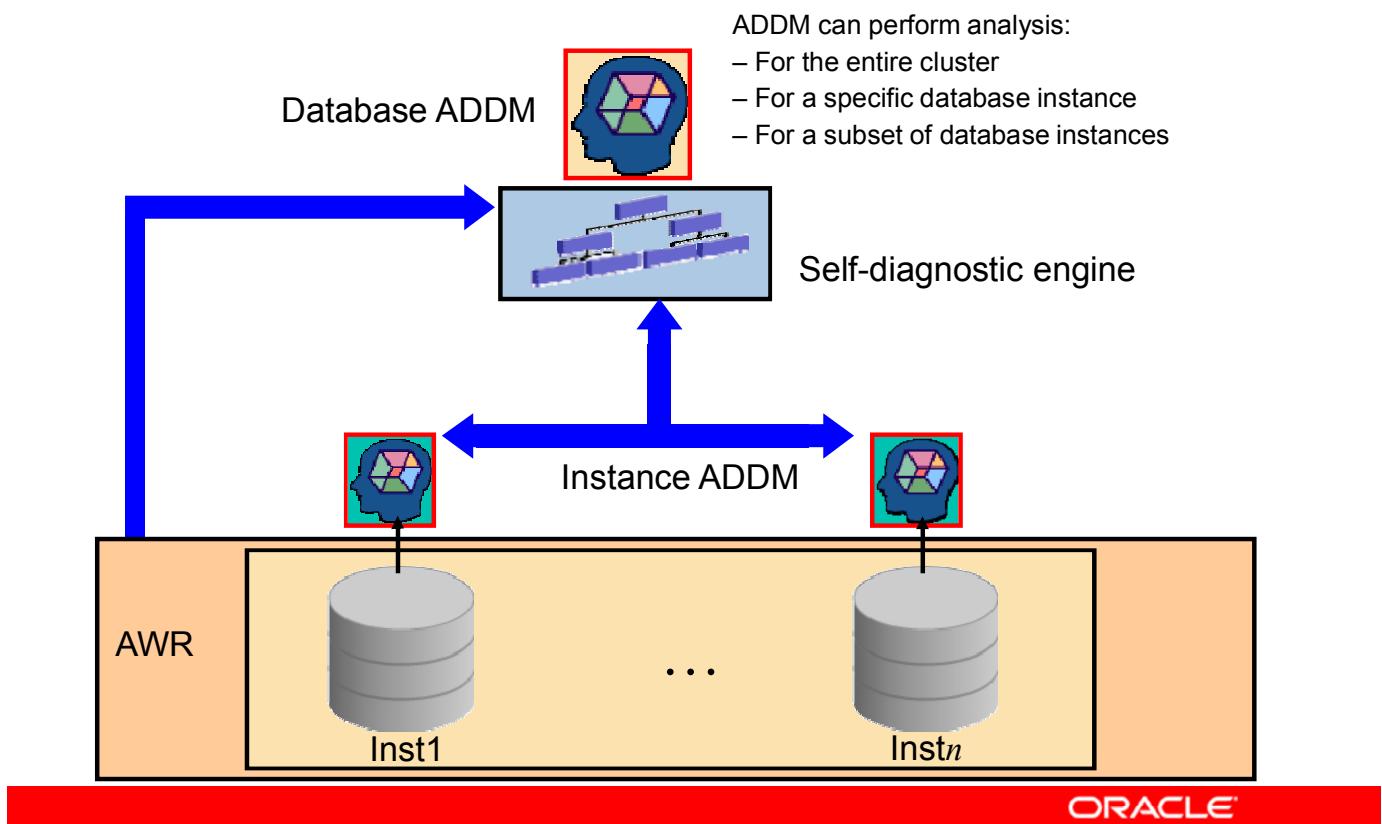
Top Cluster Events

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 using the advisor framework through Advisor Central in Oracle Enterprise Manager, or through the `DBMS_ADVVISOR` 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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can create a period analysis mode for ADDM that analyzes the throughput performance for an entire cluster. When the advisor runs in this mode, it is called *database ADDM*. You can run the advisor for a single instance, which is called *instance ADDM*.

Database ADDM has access to AWR data generated by all instances, thereby making the analysis of global resources more accurate. Both database and instance ADDM run on continuous time periods that can contain instance startup and shutdown. In the case of database ADDM, there may be several instances that are shut down or started during the analysis period. However, you must maintain the same database version throughout the entire time period.

Database ADDM runs automatically after each snapshot is taken. You can also perform analysis on a subset of instances in the cluster. This is called *partial analysis ADDM*.

I/O capacity finding (the I/O system is overused) is a global finding because it concerns a global resource affecting multiple instances. A local finding concerns a local resource or issue that affects a single instance. For example, a CPU-bound instance results in a local finding about the CPU. Although ADDM can be used during application development to test changes to either the application, the database system, or the hosting machines, database ADDM is targeted at DBAs.

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)



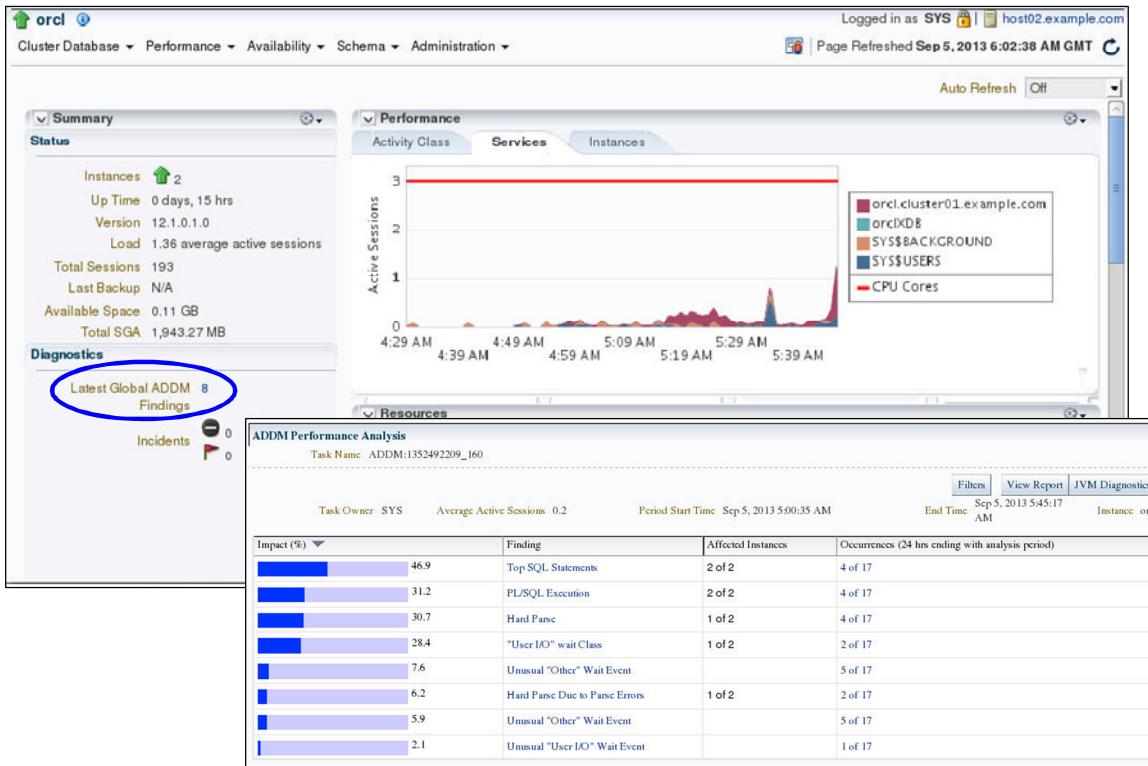
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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



Copyright © 2014, 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.

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

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.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a, b, d, e

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 7: Overview

This practice covers manually discovering performance issues by using the EM performance pages as well as ADDM.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Managing High Availability of Services

8

ORACLE®

Copyright © 2014, 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 © 2014, 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 © 2014, 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 © 2014, 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 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 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 © 2014, 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 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 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 © 2014, 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 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 `-l` 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 © 2014, 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.

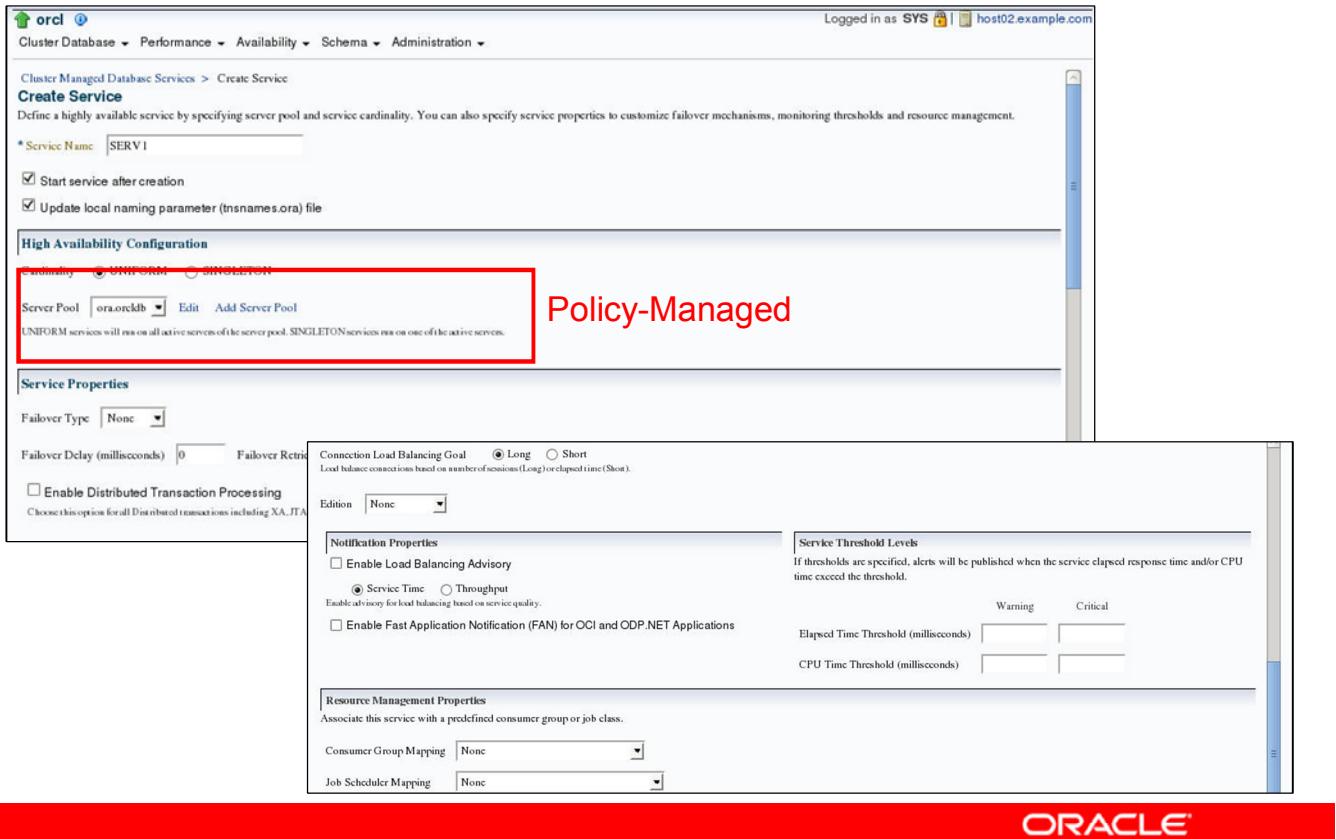


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 using SRVCTL as follows:

```
$ srvctl modify scan_listener -invitednodes node_list -invitedsubnets subnet_list
```

Creating Service with Enterprise Manager



Copyright © 2014, 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
```



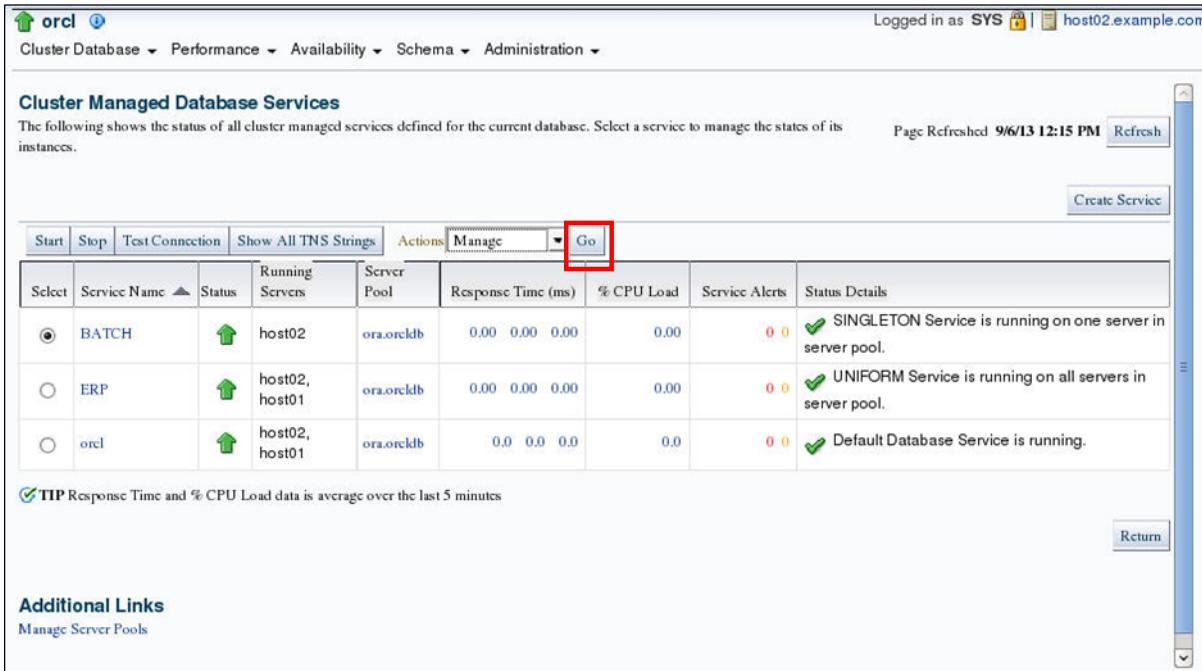
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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



The screenshot shows the 'Cluster Managed Database Services' page. At the top, there are tabs for 'Start', 'Stop', 'Test Connection', 'Show All TNS Strings', 'Actions', 'Manage', and 'Go'. The 'Manage' button is highlighted with a red box. Below the tabs is a table with columns: Select, Service Name, Status, Running Servers, Server Pool, Response Time (ms), % CPU Load, Service Alerts, and Status Details. Three services are listed:

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	0 0	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	0 0	UNIFORM Service is running on all servers in server pool.
<input type="radio"/>	orcl		host02, host01	ora.orcldb	0.0 0.0 0.0	0.0	0 0	Default Database Service is running.

A note at the bottom left says: TIP Response Time and % CPU Load data is average over the last 5 minutes. A 'Return' button is at the bottom right.

ORACLE

Copyright © 2014, 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

Cluster Managed Database Service: ERP

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.

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

Top Consumers Details

Service Properties Edit

Instances							
Select	Node Name	Instance Name	Service Status for Node	Instance Status	Response Time (per user call) (milliseconds)	CPU Time (per user call) (milliseconds)	Status Details
<input checked="" type="radio"/>	host02	orcl_1	✓ Running	✓			✓
<input type="radio"/>	host01	orcl_3	✓ Running	✓	✓ 0.00	✓ 0.00	✓

ORACLE

Copyright © 2014, 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 © 2014, 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=
      (LOAD_BALANCE=on)
      (ADDRESS= (PROTOCOL=TCP) (HOST=cluster01-scan) (PORT=1521)))
      (CONNECT_DATA= (SERVICE_NAME=ERP)) "
```



Copyright © 2014, 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 © 2014, 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 © 2014, 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.

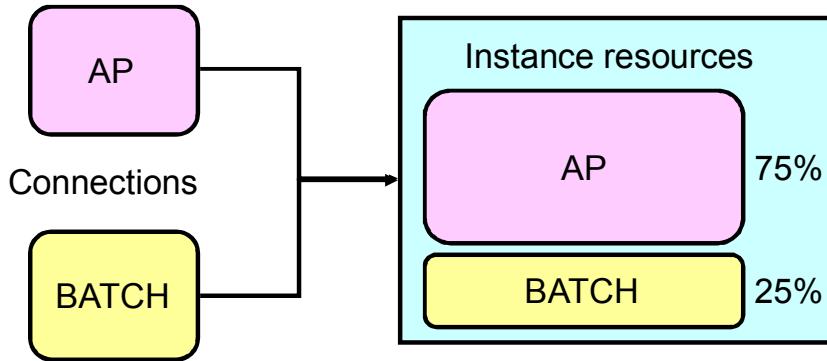
TAF can restart a query after failover has completed but for other types of transactions, such as **INSERT**, **UPDATE**, or **DELETE**, the application must roll back the failed transaction and resubmit the transaction. You must re-execute any session customizations, in other words, **ALTER SESSION** 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

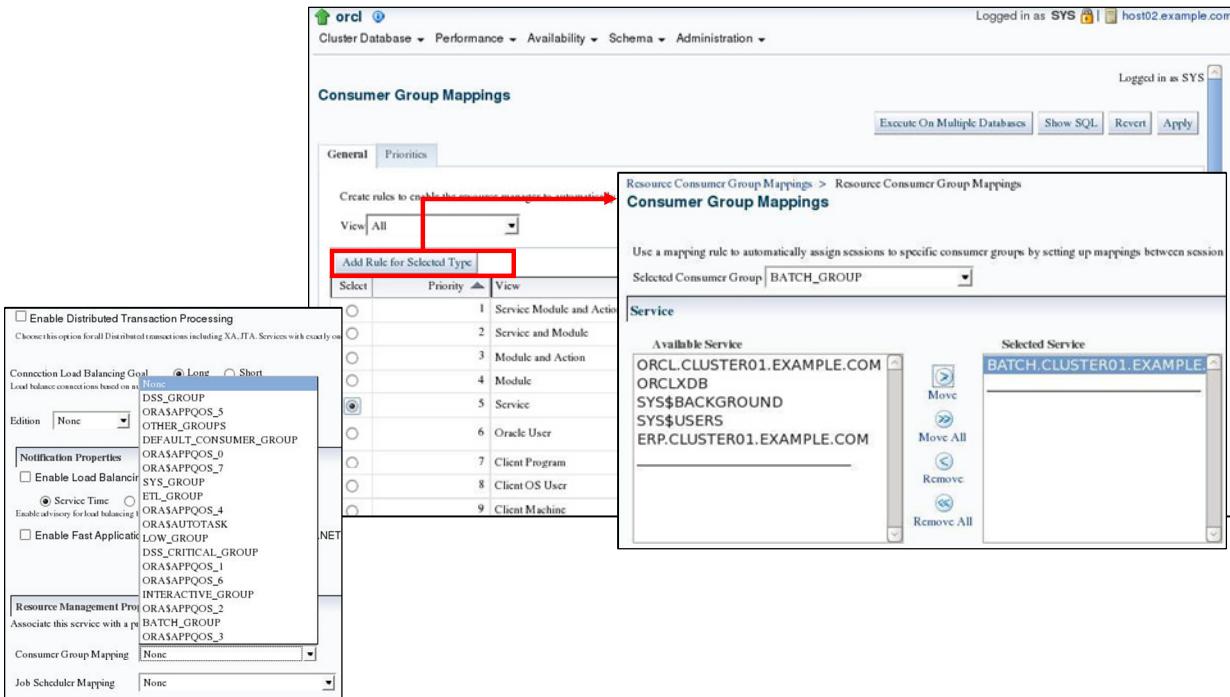
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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.

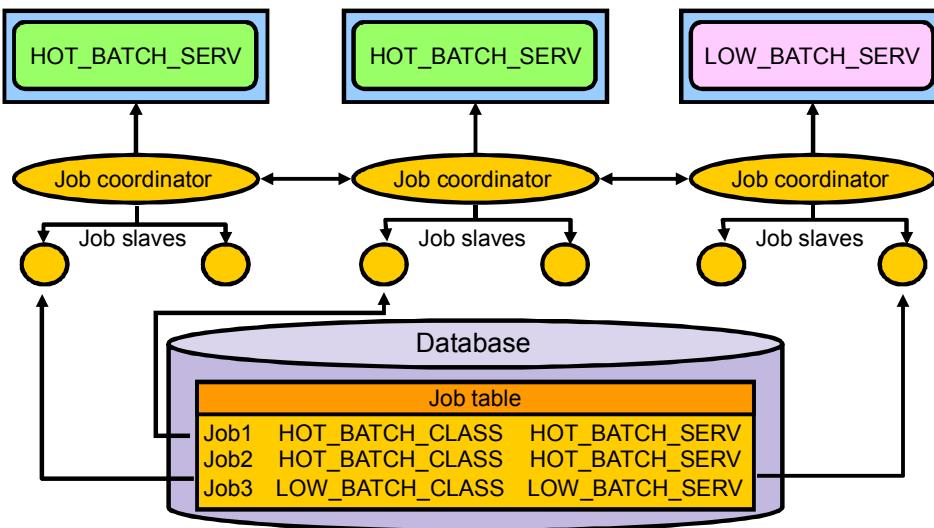
With the ability to map sessions to consumer groups by service, module, and action, you have greater flexibility when it comes to managing the performance of different application workloads.

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



Copyright © 2014, 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 interface for creating a job class. The 'Service Name' field is highlighted with a red box. A dropdown menu is open, listing various Oracle services. The services listed are:

- ORASAT_JCURG_SQL
- ORASAT_JCMED_OS
- XMLDB_NFS_JOBCLASS
- ORASAT_JCNRM_SA
- DEFAULT_JOB_CLASS
- ORASAT_JCURG_SA
- ORASAT_JCNRM_OS
- DBMS_JOBS
- ORASAT_JCURG_OS
- ORASAT_JCMED_SA
- SCHEDS_LOG_ON_ERRORS_CLASS
- ORASAT_JCMED_SQL
- ORASAT_JCNRM_SQL
- AQ\$_PROPAGATION_JOB_CLASS
- Nonc

ORACLE

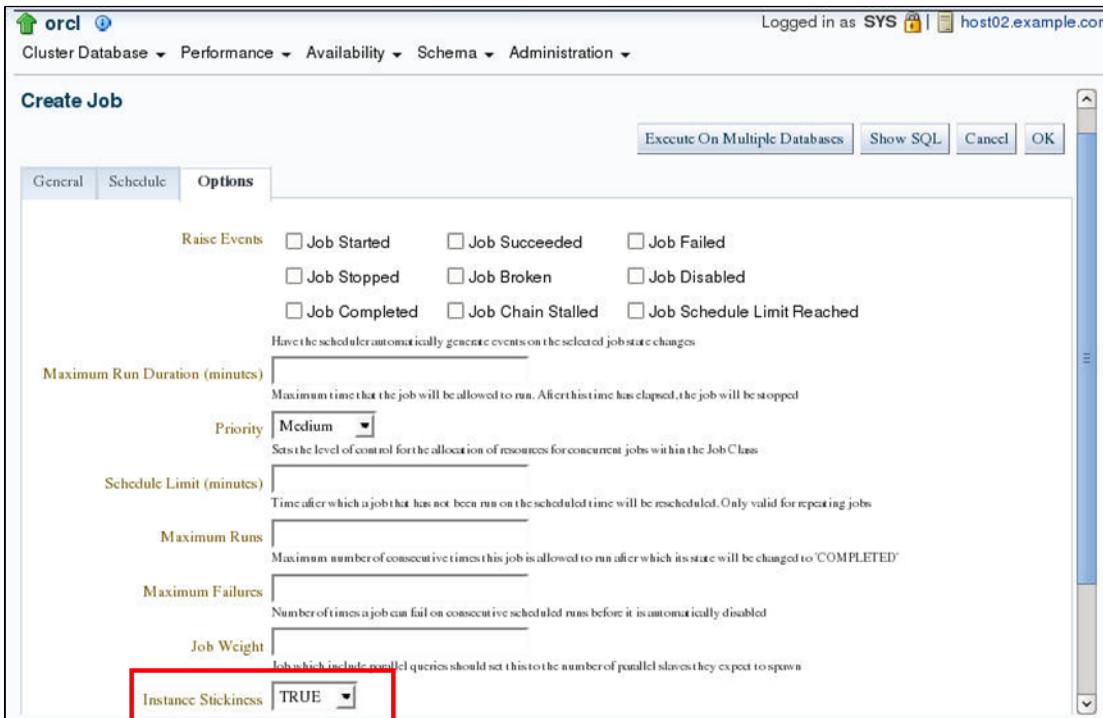
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 © 2014, 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 © 2014, 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 2PC 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
```



Copyright © 2014, 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 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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 © 2014, 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.



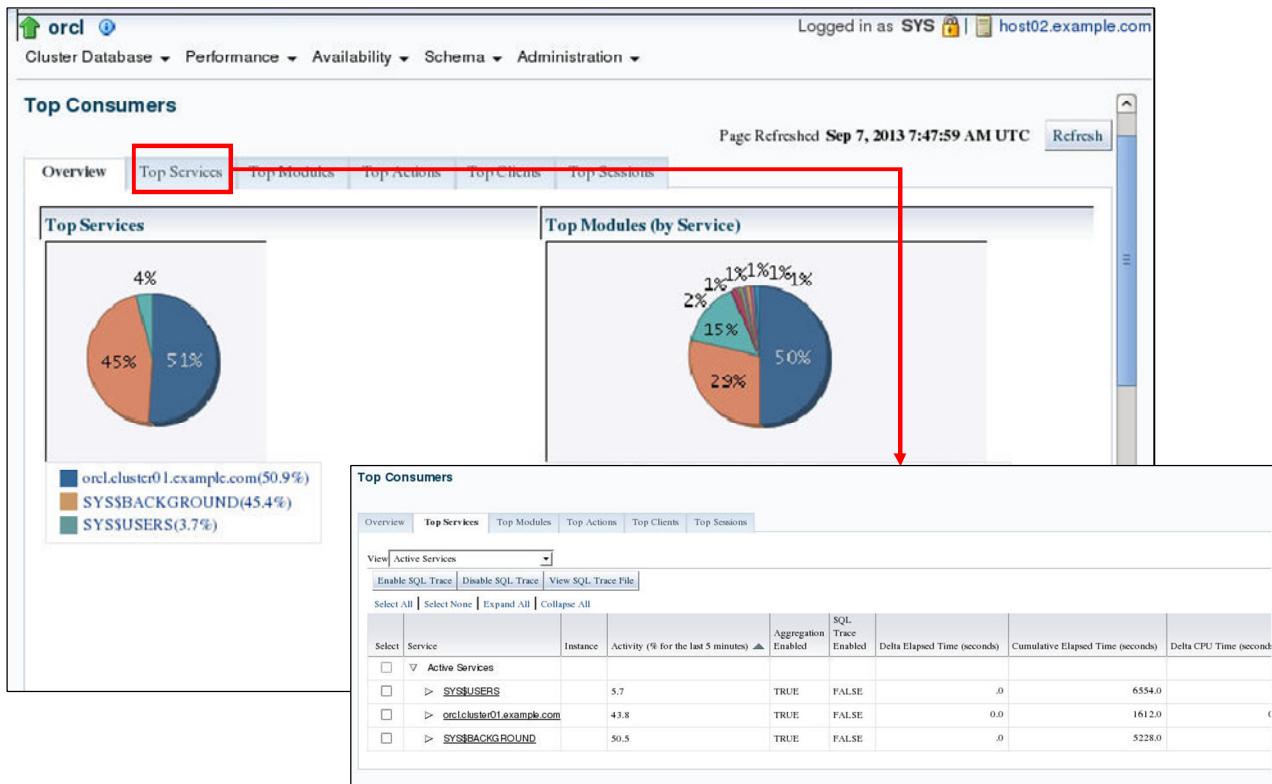
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 © 2014, 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 © 2014, 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 SELECT statement below:

```
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:
 - V\$SESSION
 - V\$ACTIVE_SESSION_HISTORY
- Service performance in:
 - V\$SERVICE_STATS
 - V\$SERVICE_EVENT
 - V\$SERVICE_WAIT_CLASS
 - V\$SERVICEMETRIC
 - V\$SERVICEMETRIC_HISTORY
 - V\$SERV_MOD_ACT_STATS
 - DBA_ENABLED_AGGREGATIONS
 - DBA_ENABLED_TRACES



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The service, module, and action information are visible in V\$SESSION and V\$ACTIVE_SESSION_HISTORY.

The call times and performance statistics are visible in V\$SERVICE_STATS, V\$SERVICE_EVENT, V\$SERVICE_WAIT_CLASS, V\$SERVICEMETRIC, and V\$SERVICEMETRIC_HISTORY.

When statistics collection for specific modules and actions is enabled, performance measures are visible at each instance in V\$SERV_MOD_ACT_STATS.

More than 600 performance-related statistics are tracked and visible in V\$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 © 2014, 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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

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 © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 8: Overview

This practice covers the following topics:

- Creating and managing services using EM
- Using server-generated alerts in combination with services



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

High Availability for Connections and Applications

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Part I

High Availability of Connections



Copyright © 2014, 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
- Use the Load Balancing Advisory (LBA)
- Describe the benefits of Fast Application Notification (FAN)
- Configure server-side callouts
- Configure the server- and client-side ONS
- Configure Transparent Application Failover (TAF)



Copyright © 2014, 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 © 2014, 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 © 2014, 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 © 2014, 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 © 2014, 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.NET clients
- 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.



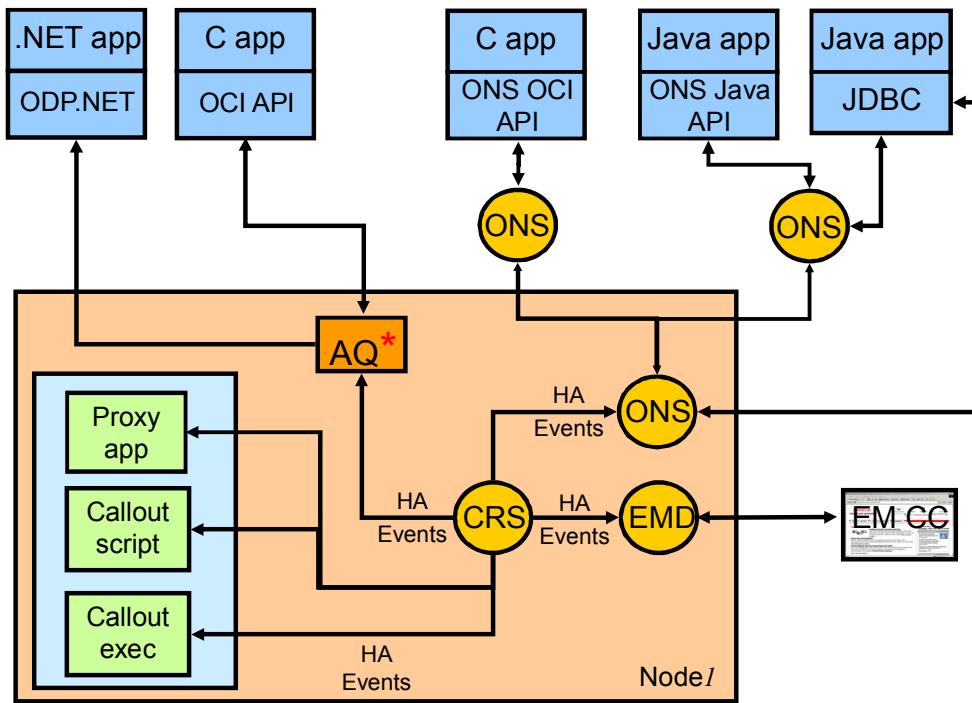
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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.

ORACLE

Copyright © 2014, 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

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 © 2014, Oracle and/or its affiliates. All rights reserved.

FAN delivers events pertaining to the list of managed cluster resources shown in the slide. The table describes each of the resources.

Note: SRV_PRECONNECT and SERVICEMETRICS are discussed later in this lesson.

FAN Event Reasons

Event Reason	Description
USER	User-initiated commands, such as <code>srvctl</code> and <code>sqlplus</code>
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 event is triggered
AUTOSTART	Initial cluster boot: Managed resource has profile attribute <code>AUTO_START=1</code> , 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 © 2014, 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 © 2014, 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 © 2014, 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 SERVICE, DATABASE, and 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 © 2014, 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 programatically 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 © 2014, 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 © 2014, 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 © 2014, 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.

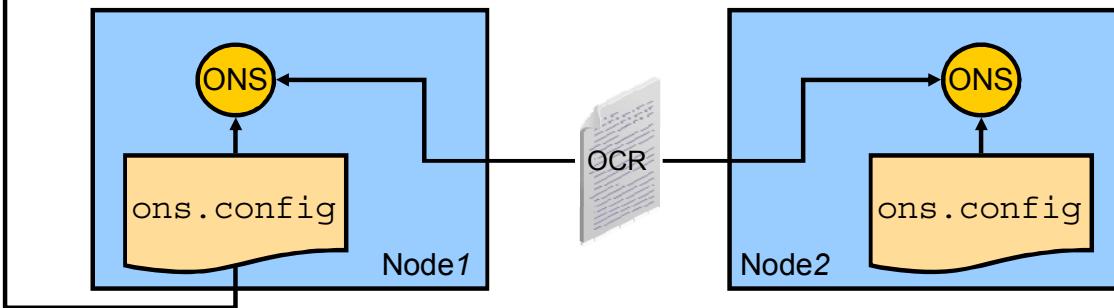
Now that the event characteristics are identified, this script triggers the execution of the trouble-logging program `/usr/local/bin/logTicket` only when the RAC HA framework posts a SERVICE, DATABASE, or NODE event type, with a status set to either `not_restarting` or `restart_failed`, and only for 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

```
usesharedinstall=true  
allowgroup=true  
localport=6100          # line added by Agent  
remoteport=6200          # line added by Agent  
nodes=host01:6200        # line added by Agent
```



ORACLE

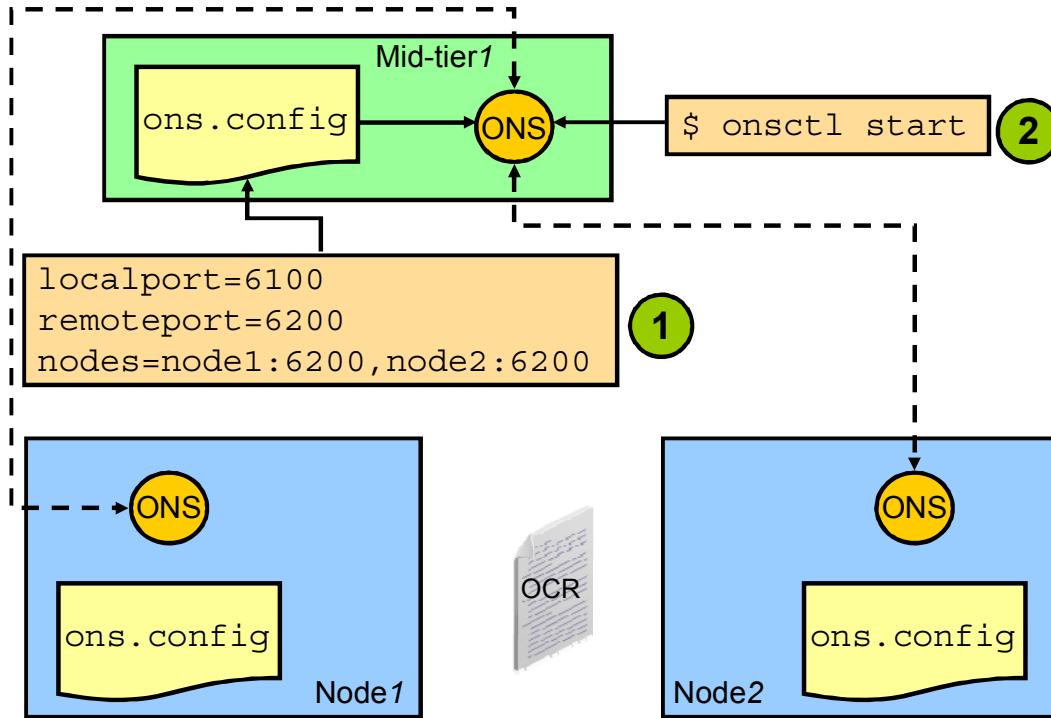
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

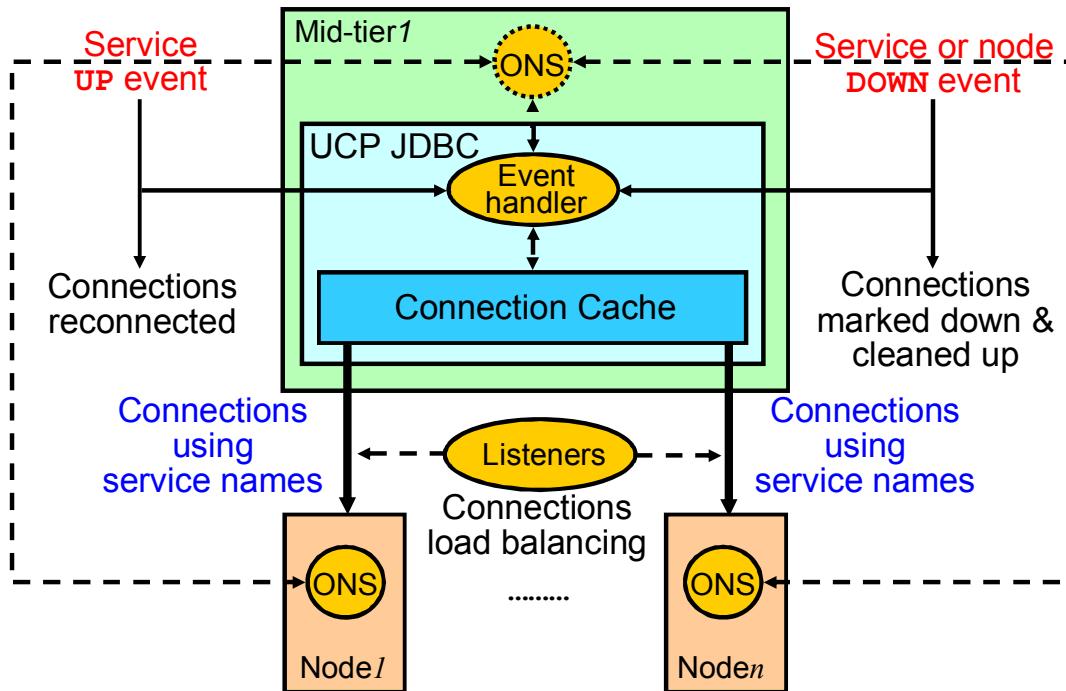
ORACLE

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 is 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 © 2014, Oracle and/or its affiliates. All rights reserved.

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 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 © 2014, 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 © 2014, Oracle and/or its affiliates. All rights reserved.

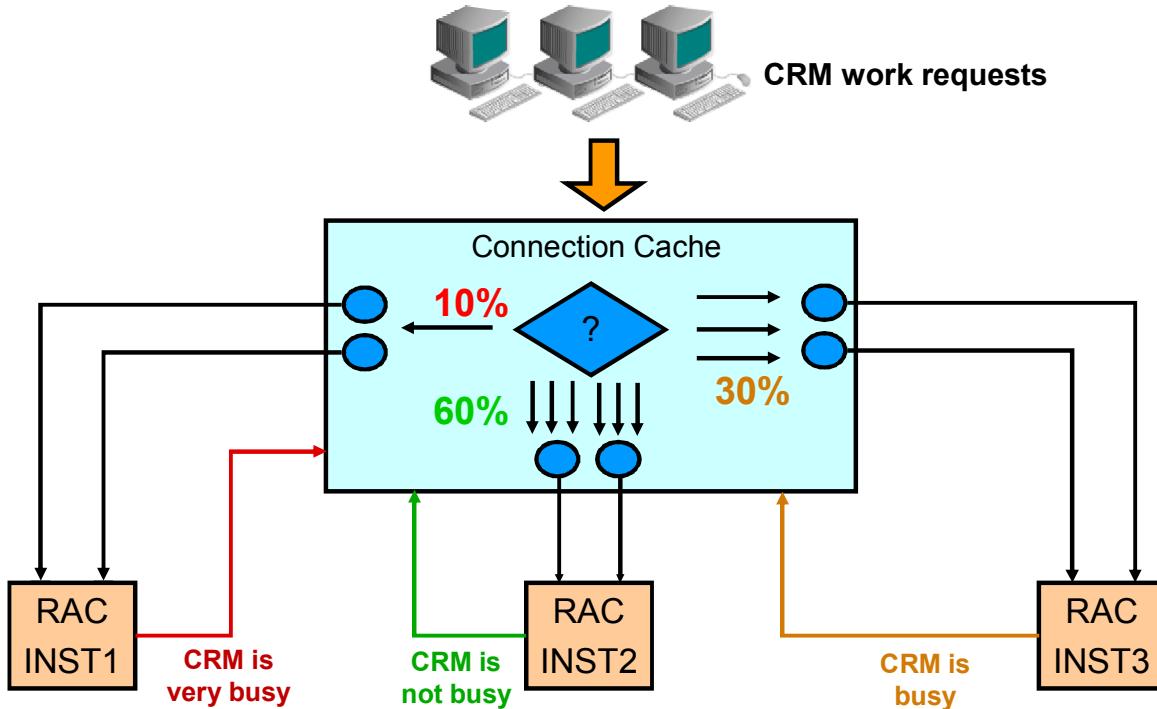
Load balancing distributes work across all of the available RAC database instances. Well-written applications use persistent connections that span the instances of RAC offering a service. Connections are created infrequently and exist for a long duration. Work comes into the system with high frequency, 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 © 2014, 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 © 2014, 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>
```

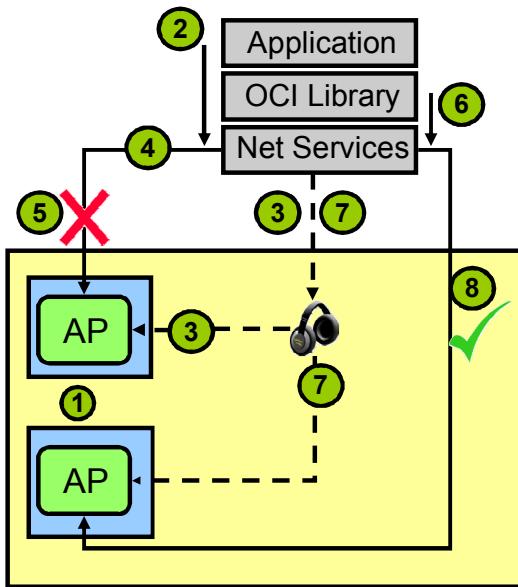


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

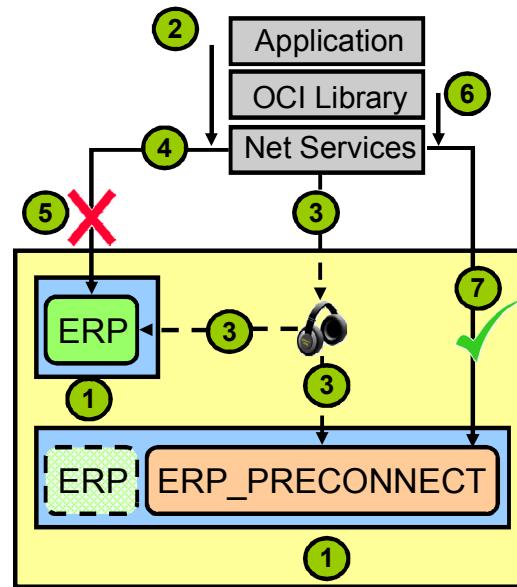
You can use the SQL query shown in the slide to monitor the Load Balancing Advisory FAN events for each of your services.

Transparent Application Failover: Overview

TAF Basic



① TAF Preconnect



ORACLE

Copyright © 2014, 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
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Before using TAF, it is recommended that you create and start a service that 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 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 © 2014, Oracle and/or its affiliates. All rights reserved.

To use client-side TAF, create and start your service using SRVCTL, then configure TAF by defining a TNS entry for it in your `tnsnames.ora` file as shown in the slide.

The `FAILOVER_MODE` parameter must be included in the `CONNECT_DATA` section of a connect descriptor.

In the example in the slide, if the instance fails after the connection, then the TAF application fails over to the other node's listener, reserving any `SELECT` statements in progress. If the failover connection fails, then Oracle Net waits 15 seconds before trying to reconnect again. Oracle Net attempts to reconnect up to 20 times.

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 © 2014, 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 addition 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 © 2014, 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 © 2014, Oracle and/or its affiliates. All rights reserved.

Because the 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.

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a, b, d

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
- Use the Load Balancing Advisory
- Describe the benefits of Fast Application Notification
- Configure server-side callouts
- Configure the server- and client-side ONS
- Configure Transparent Application Failover



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Part II

Application Continuity and Transaction Guard

A solid red horizontal bar spanning most of the page width.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the purpose of Transaction Guard and Application Continuity
- Describe the key concepts relating to Application Continuity
- Describe the side effects and restrictions relating to Application Continuity
- Describe the requirements for developing applications that leverage Application Continuity
- Configure Application Continuity



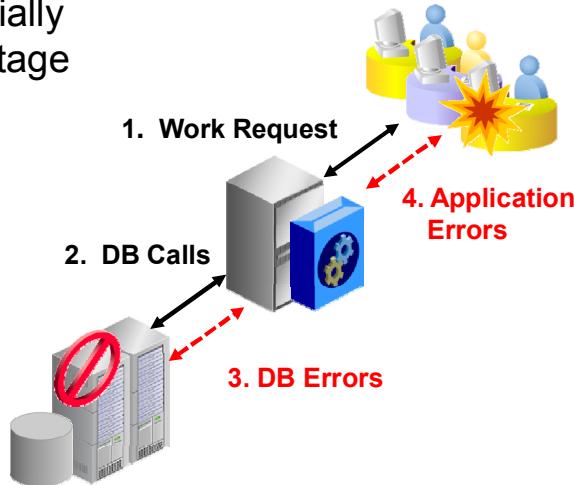
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Situation Prior to Application Continuity

- Database outages can cause:
 - In-flight work to be lost, leaving users in doubt
 - Users to restart applications and reenter data, leading to duplicate submission of requests
 - Additional failures, potentially prolonging the system outage
- Solving the problem inside the application is difficult and expensive.



Did the last transaction commit?



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

For end users, database session outages can lead to undesirable results:

- Confusion: Users do not know what happened to their application's transactions, such as funds transfers, orders, payments, and bookings.
- Duplication: Doubt over a failed transaction may lead users to reenter it. This may lead to undesirable duplication resulting in overpayments, over ordering or overbooking, for example.
- Loss of productivity: Even if duplication is not an issue, user productivity is adversely affected by the need to restart applications and reenter data.
- Prolonged system outages: A database failure may cause related applications or other infrastructure, like sensors and communication equipment, to stall or fail. This may require applications and other components to be rebooted or reinitialized resulting in a prolonged system outage.

Developing a solution for these problems inside the application has traditionally been difficult and expensive for the following reasons:

- Every possible exception must be considered and handled.
- If a failure occurs during a commit, it is difficult to determine whether the commit occurred.
- To legitimately replay the work associated with a failed session, the application must ensure that the database is in the correct state; otherwise, the replay may be invalid.

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 © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Database 12c introduces two fundamental capabilities for ensuring continuity of applications after database outages:

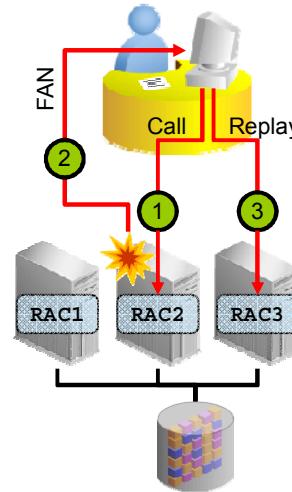
1. A foolproof way for applications to know the outcome of transactions
2. The ability to mask outages by reconnecting to the database and replaying the workload

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.

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 re-connect and replay
 - Request replay on another RAC instance



ORACLE

Copyright © 2014, 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.

Key Concepts for Application Continuity



Database Request: Unit of work submitted from the application, typically SQL, PL/SQL, RPC calls



Recoverable Error: Error that arises due to an external system failure, independent of the application



Commit Outcome: Outcome of last transaction; made durable by Transaction Guard



Mutable Functions: Functions that change their results each time they are executed



Session State Consistency: Describes how the application changes the nontransactional state during a database request



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The slide introduces some terms and concepts that are associated with Transaction Guard and Application Continuity. The following notes elaborate further:

Database Request

A database request is a unit of work submitted from the application. A database request typically consists of the SQL statements, PL/SQL blocks, local procedure calls, and database RPCs (Remote Procedure Calls), in a single web request on a single database connection. Database requests are generally demarcated by the calls made to check out and check in a database connection from a connection pool. For recoverable errors, Application Continuity reestablishes the database session and repeats the database request safely.

Recoverable Error

A recoverable error is an error that arises due to an external system failure, independent of the application session logic that is executing. Recoverable errors occur following planned and unplanned outages of foregrounds, networks, nodes, storage, and databases. The application receives an error code (such as ORA-03113 : end-of-file on communication channel) that can leave the application not knowing the status of the last operation submitted. Recoverable errors have been enhanced in Oracle Database 12c to include more errors, and now include a public API for OCI.

Application Continuity reestablishes database sessions and resubmits the pending work for recoverable errors. Application Continuity does not resubmit work following call failures due to nonrecoverable errors. An example of a nonrecoverable error is submission of an invalid data value, such as an invalid date or invalid numeric value.

Commit Outcome

In Oracle Database, a transaction is committed by updating its entry in the internal transaction table. Oracle Database generates a redo log record corresponding to this update. After the redo log record is written out to the redo log on disk, the transaction is deemed committed at the database. From a client perspective, the transaction is committed when a commit outcome message, generated after that redo is written, is received by the client. However, the commit outcome message is not durable. For example, the commit outcome can be lost if the client is disconnected from the database after the commit outcome is generated by the database but before it is received by the client. Transaction Guard provides a reliable commit outcome because it can be used to reliably obtain the commit outcome when it has been lost following a recoverable error that results in the failure of a database session.

Mutable Functions

Mutable functions are functions that can change their results each time that they are called. Mutable functions can cause replay to be rejected because the results visible to the application change at replay. Consider `sequence.NEXTVAL` that is often used in key values. If a primary key is built with a sequence value and it is later used in foreign keys or other binds, at replay the same function result must be returned if the application is using it.

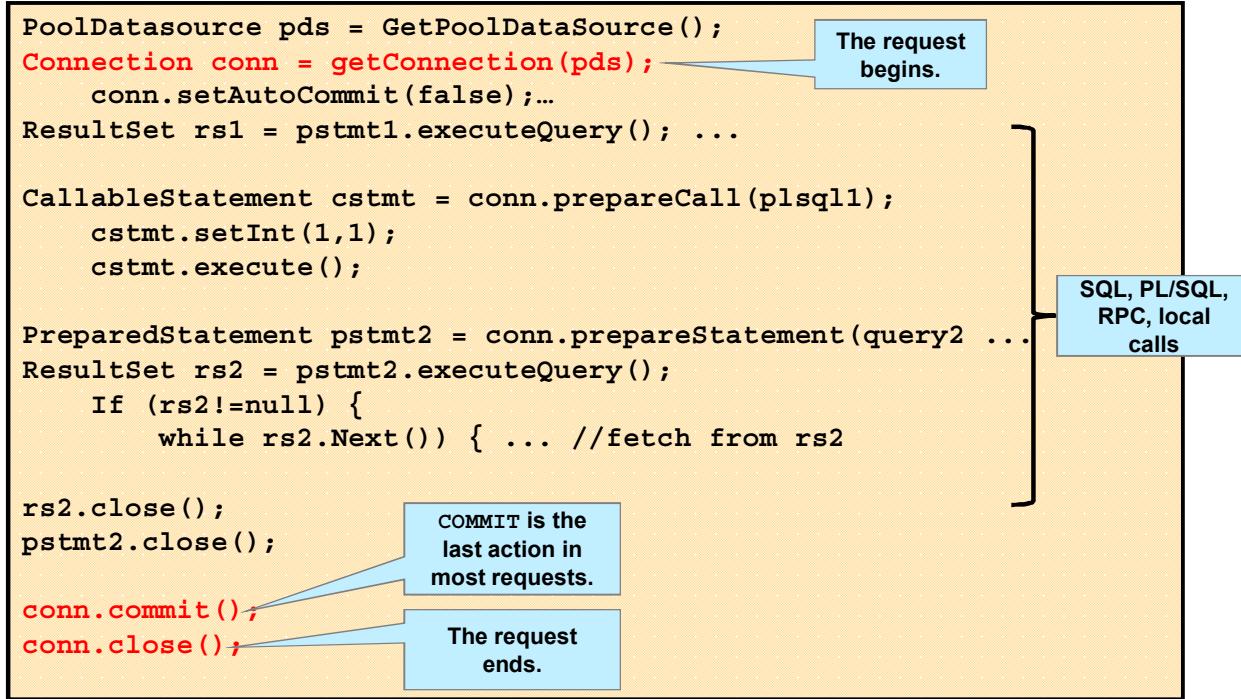
Application Continuity provides mutable object value replacement at replay for granted Oracle function calls to provide opaque bind-variable consistency. If the call uses database functions that are mutable, including `sequence.NEXTVAL`, `SYSDATE`, `SYSTIMESTAMP`, and `SYSGUID`, the original values returned from the function execution are saved and are reapplied at replay. If an application decides not to grant mutables, replay for these requests may be rejected.

Session State Consistency

Nontransactional state includes national language support (NLS) settings, cursors, events, and global PL/SQL package states. After a `COMMIT` statement has been executed, if the nontransactional state was changed in that transaction, 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 static or dynamic, and thus whether it is correct to continue past a `COMMIT` operation within a request.

While configuring Application Continuity, almost all applications should use the default `DYNAMIC` mode. In the `DYNAMIC` mode, replay is disabled from `COMMIT` until the end of the request. This is not a problem for most applications, because almost all requests have zero or one commit, and commit is most often the last statement in a database request.

Workflow of a Database Request



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A database request is a logical unit of work.

For example:

- Every action that the client driver receives from an Object Relational Manager (ORM)
- Every action that you submit at your automatic teller machine
- Every action that you submit at your browser while browsing, shopping, making bill payments, and so on

Typically, all database requests that use JDBC follow a standard pattern.

The slide contains a code segment that shows how typical JDBC applications are coded:

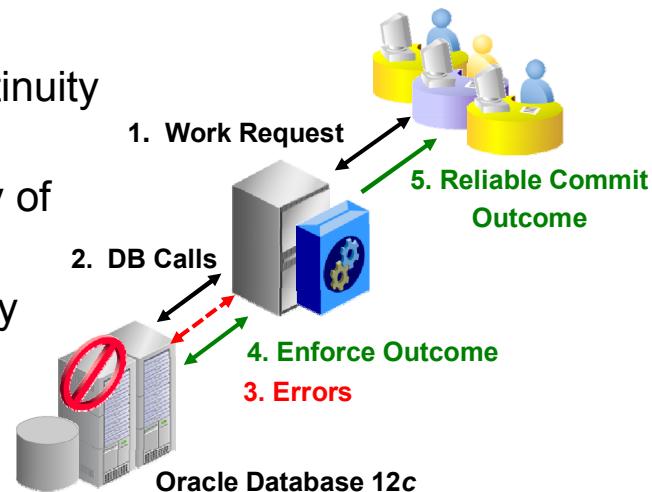
1. A database request begins with the call to `getConnection` to the `PoolDatasource`.
2. The application's logic is executed. This execution could include SQL, PL/SQL, RPC, or local procedure calls.
3. The transaction is committed.
4. The database request ends when the connection is returned to the connection pool.

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

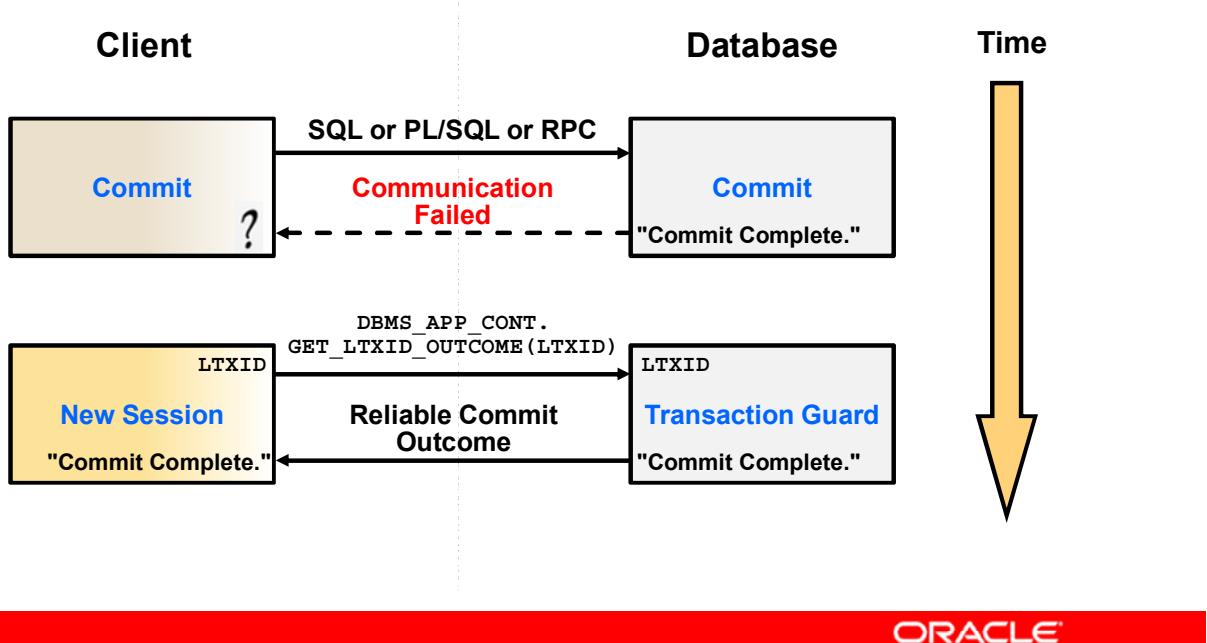
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Transaction Guard is a reliable protocol and API that applications use to provide a reliable commit outcome. The API is typically embedded in error handling and should be called following recoverable errors. The outcome indicates whether the last transaction was committed and completed. After the commit outcome is returned to the application, the outcome persists. Therefore, if Transaction Guard returns committed or uncommitted, the status stays this way. This enables the application or user to proceed with confidence.

Transaction Guard is used by Application Continuity and is automatically enabled by it, but it can also be enabled independently. Transaction Guard prevents the transaction being replayed by Application Continuity from being applied more than once. If the application has implemented application-level replay, integration with Transaction Guard can be used to ensure transaction idempotence; that is, executing the transaction multiple times has the same result as executing it only once.

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 © 2014, 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
- Not supported in release 12.1:
 - XA
 - 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 © 2014, Oracle and/or its affiliates. All rights reserved.

Transaction Guard supports all the transaction types listed in the slide. The primary exclusions in Oracle Database 12c release 12.1 are:

- Transaction Guard is not supported for applications developed by using Oracle XA.
- Transaction Guard is not supported if you are using 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).

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 © 2014, Oracle and/or its affiliates. All rights reserved.

Transaction Guard is a powerful feature. Some of the benefits are:

- For applications that integrate Transaction Guard, users can know what happened to their last submission and proceed with confidence and certainty. Without Transaction Guard, doubt following a failure can lead to resubmitting a database request, which can cause logical corruption.
- Because it is integrated into the database kernel, Transaction Guard provides better performance and reliability than home-built code for idempotence.

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



ORACLE

Copyright © 2014, 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.

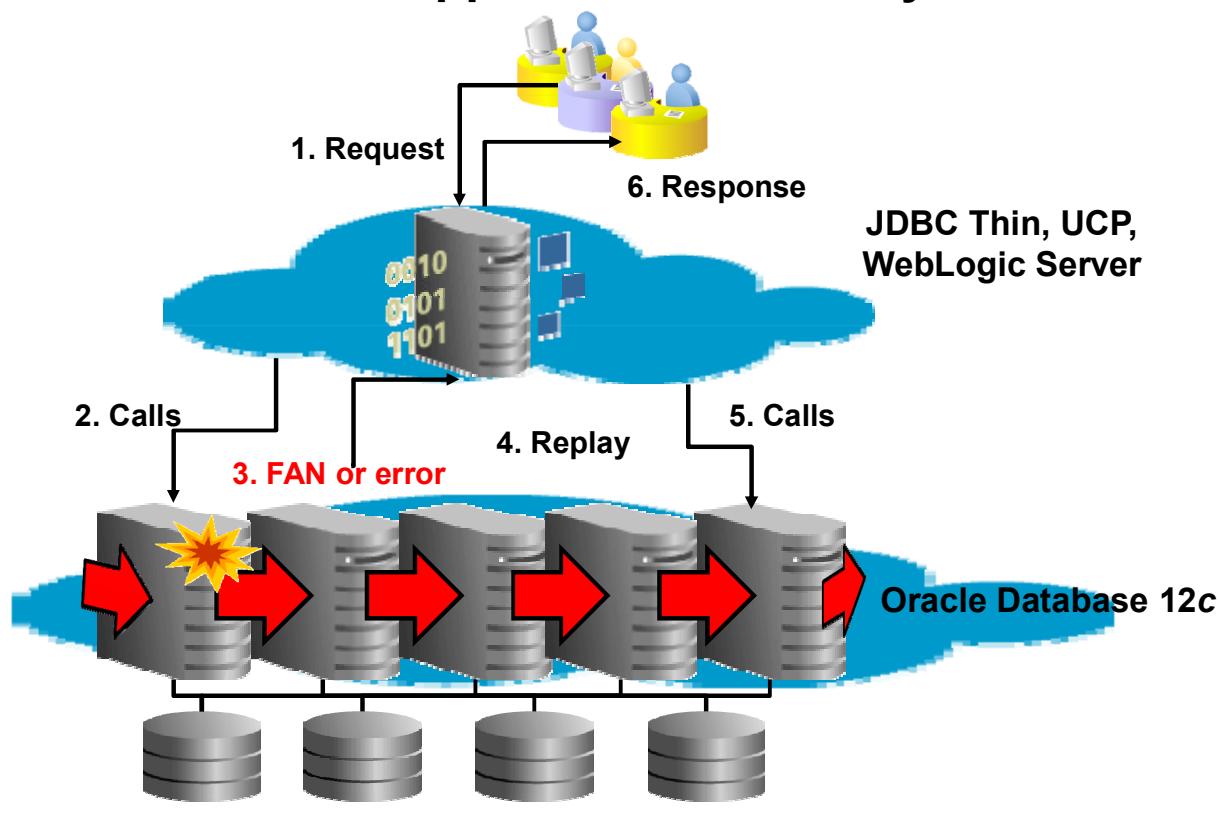
Application Continuity masks many recoverable database outages from applications and users. It achieves the masking by restoring the database session, the full session (including session state, cursors, and variables), and the last in-flight transaction (if there is one).

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.

How Does Application Continuity Work?



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The graphic in the slide illustrates how Application Continuity works. Following is a description of a typical workflow involving Application Continuity:

1. The client sends a work request to the application.
2. The application sends the calls that make up the request to the database using the JDBC replay driver.
3. The JDBC replay driver receives a Fast Application Notification (FAN) notification or a recoverable error.
4. The replay driver performs the following actions:
 - It checks that the request has replay enabled and that the replay initiation timeout has not expired. If all is in order, the driver obtains a new database session. If a callback is registered, the callback is executed to initialize the session.
 - It checks with the database to determine whether the last transaction completed.
 - If replay is required, the JDBC replay driver resubmits calls, receiving directions for each call from the database. Each call must result in the same client-visible state.
5. When the last call is replayed, the replay driver ends the replay and returns to normal runtime mode.
6. If the replay succeeds, the application responds normally to the user.

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, Universal Connection Pool, and WebLogic Server



Copyright © 2014, 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.

Application Continuity Processing Phases

Normal Run Time	Reconnect	Replay
<ul style="list-style-type: none"> • Demarcates the database request • Builds proxy objects • Holds original calls with validation • Manages queues 	<ul style="list-style-type: none"> • Ensures that the request has replay enabled • Handles timeouts • Creates a new connection • Validates the target database • Uses Transaction Guard to enforce last commit 	<ul style="list-style-type: none"> • Replays held calls • Continues replay, if user-visible results match, based on validations • Continues the request



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The operation of Application Continuity can be divided into three distinct phases:

- **Normal run time:** During normal run time, each new database request is tagged with a request beginning, either by checking out of the Universal Connection Pool or WebLogic Server Connection Pool, or by adding begin and end request markers to your own application or to your own Java connection pool at checkout and checkin. The JDBC replay driver and Oracle Database 12c collaborate so that calls in the database request are held in queues, together with validation received from the database. The JDBC Replay driver holds the calls until the end of the database request or until the replay is disabled. The JDBC replay driver is responsible for managing the queues and building proxy objects to maintain a record of nontransactional state, allowing objects to be replaced if replay is needed.

- **Reconnect:** The reconnect phase of Application Continuity is triggered when a recoverable outage occurs. In this phase, the request is checked to see whether replay is still enabled, and the replay initiation timeout is checked to ensure that it has not expired. If both checks are in order, a new connection to the database is obtained. Because the reconnection to the database can take some time, you may need to set FAILOVER_DELAY and FAILOVER_TIMEOUT to allow the service to be reestablished. After the driver has established a connection to the database, it checks whether the database is a valid target and whether the last transaction was committed successfully. Replay will not occur if the connection is to a logically different database or to the same database but transactions have been lost. For example, the database has been restored to a prior point in time. Application Continuity will not resubmit committed transactions. Idempotence is enforced using Transaction Guard.
- **Replay:** The replay phase starts when a new connection to the database is obtained. All calls held in the queues are replayed, and the request continues from the point where it failed. Replay is disabled if there are any user-visible changes in results for any request that is replayed.

Restrictions



Global	Request	Target Database
<ul style="list-style-type: none"> Do not use the default database service. Replay is not supported for an Oracle XA data source. Deprecated Java concrete classes are not supported. 	<ul style="list-style-type: none"> For Java streams, replay is on “best effort” basis. Restricted calls: <ul style="list-style-type: none"> ALTER SYSTEM ALTER DATABASE Replay is not supported for Active Data Guard with read/write database links 	<ul style="list-style-type: none"> Does not support: <ul style="list-style-type: none"> Logical Standby GoldenGate

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The slide lists some restrictions and other considerations that apply to Application Continuity. There are basically three levels of restrictions for Application Continuity:

- **Global:** The following restrictions prevent Application Continuity from being enabled or used on any request.
 - Do not use the default service corresponding to the DB_NAME or DB_UNIQUE_NAME. The use of the database service is not recommended for high availability, because this service cannot be enabled or disabled, and cannot be relocated on Oracle RAC or switched over to Oracle Data Guard.
 - Replay is not supported for applications developed by using Oracle XA.
 - For applications using JDBC, there is no support for oracle.sql deprecated concrete classes like BLOB, CLOB, BFILE, OPAQUE, ARRAY, STRUCT, or ORADATA.
- **Request:** The following restrictions disable Application Continuity for part of a request.
 - For JDBC stream arguments, replay is on a “best effort” basis. For example, if the application is using physical addresses, the address is no longer valid with the outage and cannot be repositioned.
 - Replay is disabled if the transaction executes the ALTER SYSTEM or ALTER DATABASE statement.
 - Replay is not supported if you are using Active Data Guard with read/write database links to another database.
- **Target:** Application Continuity is not supported for logically different databases (Oracle Logical Standby and Oracle GoldenGate).

Potential Side Effects

- Some applications may need to use the `disableReplay` API if there are any requests that should not be replayed.
- Examples of calls that create side effects are:
 - Autonomous transactions
 - `DBMS_ALERT` calls (email or other notifications)
 - `DBMS_FILE_TRANSFER` calls (copying files)
 - `DBMS_PIPE` and `RPC` calls (to external sources)
 - `UTL_FILE` calls (writing text files)
 - `UTL_HTTP` calls (making HTTP callouts)
 - `UTL_MAIL` calls (sending email)
 - `UTL_SMTP` calls (sending SMTP messages)
 - `UTL_TCP` calls (sending TCP messages)
 - `UTL_URL` calls (accessing URLs)



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Applications that use external actions should be reviewed to decide whether requests with side effects should be replayed or not.

Application Continuity replays SQL, PL/SQL, and RPCs. Application Continuity serves to rebuild the session as if the user submission were delayed. When a session is rebuilt, all states are rebuilt, including reexecuting statements that leave side effects. These side effects may be exactly what is required, such as writing a report, completing some auditing, or obtaining custom primary key ranges. However, the calls that are replayed might include some calls that should not be replayed. The application may want to take action to accommodate or mitigate the effects of the replay. Developers can elect to use the `disableReplay` API for requests that contain calls that they do not want to replay.

The slide shows some of the actions that create side effects, such as Autonomous transactions, email or other notifications using `DBMS_ALERT` calls, copying files using `DBMS_PIPE` and `RPC` calls, writing text files using `UTL_FILE` calls, making HTTP callouts using `UTL_HTTP` calls, sending email using `UTL_MAIL` calls, sending SMTP messages using `UTL_SMTP` calls, sending TCP messages using `UTL_TCP` calls, and accessing URLs using `UTL_URL` calls.

Actions That Disable Application Continuity



Normal Run Time	Reconnect	Replay
Any calls in same request after: <ul style="list-style-type: none"> • Successful commit in dynamic mode (the default) • A restricted call • disableReplay API call 	<ul style="list-style-type: none"> • Error is not recoverable • Reconnection failure <ul style="list-style-type: none"> – Replay initiation timeout – Max connection tries – Max retries per incident • Target database is not valid for replay • Last call committed, in dynamic mode 	<ul style="list-style-type: none"> • Validation detects different results

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

If it is disabled, Application Continuity is ineffective until the next request. The following events will disable replay for the request:

- **Normal run time:** Capture happens on all original calls until a successful COMMIT (in dynamic mode) or unless it is disabled for the request.
- **Reconnect:** Replay does not occur if the error is not recoverable or if:
 - Timeouts have been exceeded
 - The target database is not the same or an ancestor
 - The request has committed
- **Replay:** The calls must return the same user-visible results that the application has previously seen and potentially used to make a decision. If the validation for the replayed call does not pass, replay is aborted and the original error is returned.

When Is Application Continuity Transparent?

- Application Continuity is transparent for J2EE applications that:
 - Use standard JDBC and Oracle connection pools
 - Do not have external actions, or have external actions and correctness is preserved at replay
- For situations where Application Continuity is not transparent, the following changes are typically required:
 - Request boundaries are specified using the Application Continuity APIs.
 - The `disableReplay` API is used to selectively stop replay for calls that the application never wants to replay.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Application Continuity is transparent (performed automatically) for J2EE applications that use standard JDBC and that use Oracle connection pools (UCP or WLS Active GridLink). For applications with external actions (for example, autonomous transactions or using UTL_HTTP to issue a SOA call), Application Continuity is still transparent only if the application's correctness is preserved when these external actions are replayed after a failure.

For other scenarios in which Application Continuity is not transparent, the following infrastructure changes may be needed:

- If the connection pool or container does not use an Oracle connection pool, the application must use Application Continuity APIs to mark request boundaries. Request boundaries are needed to reclaim the memory used for holding calls, and to establish a point at which to resume recording following nonreplayable operations.
- If the application has requests that the application does not want repeated, the application can explicitly call an API to disable replay for those requests. Such calls are likely to be isolated to one or a few specialized pieces of application code.

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 © 2014, 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.

Application Assessment for Using Application Continuity

Decide	What You Should Do
Request Boundaries	Mark request boundaries if you are not using Oracle Pools.
JDBC Concrete Classes	Replace deprecated concrete classes with Java interfaces.
Side Effects	Use the <code>disable</code> API if a database request has an external call that should not be replayed.
Callbacks	Ensure that a callback is registered if the state changes outside a request (WLS/UCP labeling included by default).
Mutable Functions	Grant keeping mutable values if they are compatible with your application.



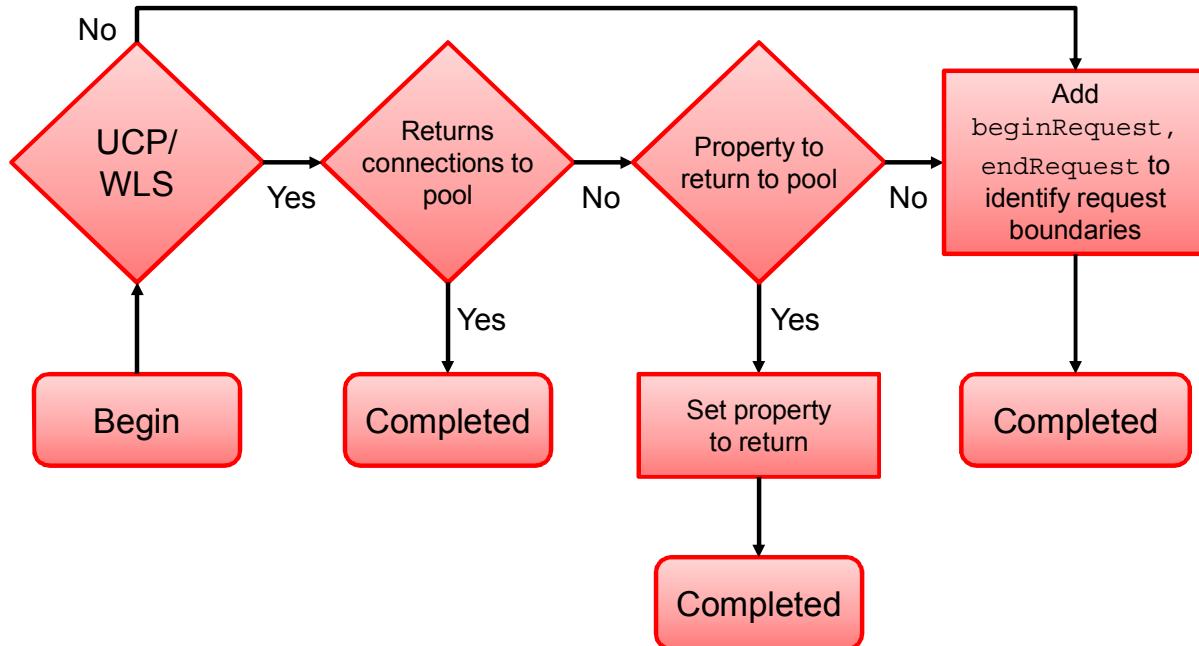
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Before using Application Continuity in conjunction with an application, the application owner must check the following:

1. Mark request boundaries if the application is not using Oracle Pools or is not releasing connections to Oracle Pools. Request boundaries tag the beginning and end of each request. Determine whether the application borrows and returns connections from the WebLogic Server Pool or Universal Connection Pool for each request, or whether `beginRequest` and `endRequest` APIs should be added to the application's own connection pool to identify request boundaries. If you are using Oracle Pools, but you are not releasing connections, consider your options to do so.
2. Determine whether the application uses Oracle JDBC concrete classes. To use Application Continuity, the deprecated concrete classes must be replaced. For information about the deprecation of concrete classes, including actions to take if an application uses them, see My Oracle Support Note 1364193.1.
3. Assess the impact of restrictions and side effects on your application. Use the `disableReplay` API for any request that should not be replayed. For example, if a request makes external calls by using one of the external PL/SQL messaging actions, decide whether the external call should be replayed or not.

4. Assess whether your application sets state outside the database request. If a state is set when a user starts a request and it is outside that request, replay needs to know about it in order to reexecute the calls.
When using Oracle WebLogic Server or the Universal Connection Pool, connection labeling is recommended. The labeling is used for both run time and replay.
If you use an application's own callback, register it at the WebLogic Admin Console, or at Universal Connection Pool, or JDBC replay driver levels.
5. Decide whether the application can keep mutables at replay. When a request is replayed, this function obtains a new value every time it is called. Keep mutable values when they are compatible with the application. Keep original values for seq.NEXTVAL, SYSDATE, SYSTIMESTAMP, and SYS_GUID during failover.

Handling Request Boundaries



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

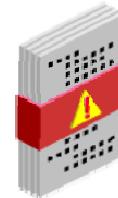
Request boundaries are used to tag the beginning and end of each request. They are required to resume recording following a nonreplayable operation, and also to reclaim the memory used for holding calls.

The flow chart shows how to decide whether request boundaries should be added:

1. Determine whether the application borrows and returns connections from the WebLogic Server Pool or Universal Connection Pool for each request, or whether beginRequest and endRequest APIs should be added to the application's own connection pool to identify request boundaries.
2. If you are using Oracle Pools, but you are not releasing connections to the pool, there is often a property to be set to release connections. Releasing connections scales much better and automatically embeds the request boundaries. This allows planned and unplanned failover support and better load balancing.

Disabling Replay by Using the disableReplay API

- Use the disableReplay API for requests that should not be replayed.
- Make a conscious decision to replay external actions.
 - Autonomous transactions, UTL_HTTP, UTL_URL, UTL_FILE, UTL_FILE_TRANSFER, UTL_SMTP, UTL_TCP, UTL_MAIL, DBMS_PIPE, and DBMS_ALERT
- In addition, consider disabling replay when the application:
 - Synchronizes independent sessions
 - Uses time at the middle-tier in a logic execution
 - Assumes that ROWID values do not change
 - Assumes that location values do not change



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

By default, the JDBC replay driver replays following a recoverable error. If the application has requests that the application does not want to be repeated, the application can explicitly call the disableReplay API to disable replay for those requests.

Application developers should make a conscious decision to allow replay for external actions, and especially for procedures that use the UTL_HTTP package. If they should not be replayed, use the disableReplay API.

Additionally, it is recommended that you consider disabling replay when the application:

- Synchronizes independent sessions
- Uses time at the middle-tier in a logic execution
- Assumes that ROWID values do not change
- Assumes that location values do not change

This is because the assumptions contained in these scenarios may not be valid during replay, potentially yielding incorrect results. For further discussion of these scenarios refer to the chapter entitled Ensuring Application Continuity in the *Oracle Database Development Guide 12c Release 1 (12.1)*.

Connection Initialization Options

- No callback
 - The application builds up its own state during each request.
- Connection labeling
 - Performance is improved by avoiding some initialization.
 - Available with UCP or WebLogic Server
 - The replay driver uses Connection Labeling when present.
- Connection initialization callback
 - The replay driver uses callback to set the initial state of the session at run time and replay.
 - Register with WebLogic, UCP, and JDBC Thin.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Nontransactional session state (NTSS) is state of a database session that exists outside transactions and is not protected by recovery. For applications that use stateful requests, the nontransactional state is reestablished as the session is rebuilt by Application Continuity.

For applications that set state only at the beginning of a request, or for stateful applications that gain performance benefits from using connections with a preset state, choose one of the following callback options:

- **No callback:** In this scenario, the application builds up its own state during each request.
- **Connection labeling:** Connection Labeling is a generic pool feature that is recommended for its excellent performance. When Connection Labeling is present, Application Continuity uses it. Connection Labeling is a feature of Universal Connection Pool (UCP) and Oracle WebLogic server. The application can be modified to take advantage of the preset state on connections. Connection Labeling APIs determine how well a connection matches, and use the labeling callback to populate the gap when a connection is borrowed. Not all applications can use Connection Labeling, because it requires some application recoding.

- **Connection initialization callback:** In this scenario, the replay driver uses an application callback to set the initial state of the session during run time and replay. The JDBC replay driver provides an optional connection initialization callback interface and methods to register and unregister connection initialization callbacks in the `oracle.jdbc.replay.OracleDataSource` interface.

When registered, the initialization callback is executed at each successful checkout at run time and each reconnection following a recoverable error. If the callback invocation fails, replay is disabled on that connection. Use the connection initialization callback only when the application has not implemented Connection Labeling, and needs state information that is not established at the request.

Callback registration is performed using the WebLogic Admin Console, in UCP, or in the JDBC Thin Replay driver.

Mutable Objects and Application Continuity

- Requests using mutable functions can fail to replay if the function result changes.
- When a mutable privilege is granted, replay applies the original function result.

Mutable Function	Application A	Application B	Application C
<code>SYSDATE,</code> <code>SYSTIMESTAMP</code>	Original	Current	Not Applicable
<code>SEQUENCE.NEXTVAL,</code> <code>SEQUENCE.CURRVAL</code>	Original	Original	Current
<code>SYS_GUID</code>	Original	Not Applicable	Not Applicable



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A mutable object is a function that obtains a new value every time it is called. An example of a mutable is a call to the `SYSTIMESTAMP` function. When a request is replayed, the default and desired treatment of mutable objects can vary. Applications using Application Continuity can determine whether to keep the original values from mutable functions if the request is replayed.

Support for keeping mutable object values is currently provided for `SYSDATE`, `SYSTIMESTAMP`, `SYS_GUID`, and `SEQUENCE.NEXTVAL`. If the original values are not kept and if different values for these mutable objects are returned to the client, replay will be rejected because the client will see different results.

The table in the slide shows examples of the different approaches that different applications might use for mutable objects. No approach is recommended over another. They are just presented to illustrate some of the possibilities that might reasonably exist.

Keeping Mutable Objects for Replay

- SQL commands:

```
SQL> GRANT [KEEP DATE TIME | KEEP SYSGUID] TO <user>;
SQL> REVOKE [KEEP DATE TIME | KEEP SYSGUID] FROM <user>;
SQL> GRANT KEEP SEQUENCE ON <sequence_name> TO <user>;
SQL> REVOKE KEEP SEQUENCE ON <sequence_name> FROM <user>;
SQL> CREATE SEQUENCE <sequence_name> [KEEP|NOKEEP];
SQL> ALTER SEQUENCE <sequence_name> [KEEP|NOKEEP];
```

- Examples:

```
SQL> CREATE SEQUENCE new_seq KEEP;
SQL> ALTER SEQUENCE existing_seq KEEP;
SQL> GRANT KEEP SEQUENCE on sales.seq1 TO user2;
SQL> GRANT KEEP DATE TIME TO user2;
SQL> GRANT KEEP SYSGUID TO user2;
```



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The slide shows the SQL commands that you can use to configure mutable objects for replay. The following notes elaborate further:

- The database user running the application may have the KEEP DATE TIME and KEEP SYSGUID privileges granted, and the KEEP SEQUENCE object privilege on each sequence whose value is to be kept. Note that these privileges cannot be granted using GRANT ALL ON <object>. In addition, it is not recommended to grant DBA privileges to database users running applications for which you want replay to be enabled. Grant only privileges that are necessary for such users.
- Sequences may be defined with the KEEP attribute, which keeps the original values of SEQUENCE.NEXTVAL for the sequence owner. Note that specifying KEEP or NOKEEP with CREATE SEQUENCE or ALTER SEQUENCE applies to the owner of the sequence.
- If granted privileges are revoked between run time and failover, the mutable values that are collected are not applied for replay.
- If new privileges are granted between run time and failover, mutable values are not collected and, therefore, cannot be applied for replay.

Configuring the JDBC Replay Data Source

Use the `oracle.jdbc.replay.OracleDataSourceImpl` data source to obtain JDBC connections.

- Configure the data source in the property file for UCP or WebLogic Server.
- Reference the data source in JDBC Thin applications.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To use Application Continuity for Java, you must use the `oracle.jdbc.replay.OracleDataSourceImpl` data source to obtain JDBC connections. This data source supports all the properties and configuration parameters of all the Oracle JDBC data sources (for example, `oracle.jdbc.pool.OracleDataSource`).

You can configure the replay data source by changing the data source property for Universal Connection Pool or by setting it using the WebLogic Console. You can also set the replay data source for JDBC Thin applications in the property file or directly in the application.

Creating Services for Application Continuity

- To create a service for Application Continuity for a policy-managed RAC database:

```
$ srvctl add service -db racdb -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 racdb -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 © 2014, 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.

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;
```

ORACLE

Copyright © 2014, 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;
```

Resource Requirements for Application Continuity

- Application Continuity
 - Java client
 - Increase memory to maintain replay queues.
 - Additional CPU is consumed for garbage collection and to build proxies.
 - Database server
 - Additional CPU is required for validation.
 - CPU overhead is minimal on current Intel and SPARC CPUs where validation is hardware-assisted.
- Transaction Guard
 - Built into the Database kernel
 - Minimal overhead
 - Excellent scalability



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Application Continuity has the following resource requirements:

- **CPU:** Application Continuity operates on both the database server and application client and needs additional CPU resources to operate. Application Continuity consumes CPU resources on the database server to perform validation during replay. Note that CPU overhead is reduced on platforms with current Intel and SPARC CPUs where validation is assisted in hardware (using Cyclic Redundancy Checks [CRCs]). On the client side, Application Continuity uses some CPU resources above the base driver. Additional CPU costs are incurred for building proxy objects and garbage collection.
- **Memory:** The JDBC replay driver requires more memory than the base driver because the calls are retained until the end of a request. At the end of the request, the calls are released to the garbage collector. This action differs from the base driver, which releases closed calls.

The memory consumption of the replay driver depends on the number of calls per request. If this number is small, the additional memory consumption of the replay driver is less and is comparable to the base driver.

To obtain the best performance, set the same value for both the `-Xmx` and `-Xms` parameters on the client. This sets the initial heap size and maximum heap size to the same size ensuring that JVM performance is not impacted by attempts to use lower settings.

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Quiz

Which of these is not one of the phases in Application Continuity?

- a. Run time
- b. Replay
- c. Reconnect
- d. Restore



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: d

Quiz

Select the statements that apply to Application Continuity.

- a. Application Continuity is supported with thin JDBC clients.
- b. The replay target database must have the same database ID, ancestors, and descendants as the source database.
- c. Replay is not supported if the transaction executes the ALTER SYSTEM or ALTER DATABASE statement.
- d. Replay is supported for applications developed using Oracle XA.
- e. Replay is supported if you are using Active Data Guard with read/write database links to another database.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a, b, c

Summary

In this lesson, you should have learned how to:

- Describe the purpose of Transaction Guard and Application Continuity
- Describe the key concepts relating to Application Continuity
- Describe the side effects and restrictions relating to Application Continuity
- Describe the requirements for developing applications that leverage Application Continuity
- Configure Application Continuity



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 9 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 © 2014, Oracle and/or its affiliates. All rights reserved.

10

Upgrading and Patching Oracle RAC

ORACLE

Copyright © 2014, 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 patchset with the Oracle Universal Installer (OUI) utility
- Install a patch with the `opatch` utility



Copyright © 2014, 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 © 2014, 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



Copyright © 2014, 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 patchset to be installed before you can apply them. These patches are not versioned and are generally available in a future patchset 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 OUI to install patch sets. PSUs are rolling installable but patch sets are not.

Configuring the Software Library

Use the Provisioning page of Enterprise Manager.

- Add a software library location:

The screenshot shows the Oracle Enterprise Manager Software Library Administration interface. On the left, a sidebar menu has 'Administration' selected. A red arrow points from this selection to a modal dialog box titled 'Add OMS Shared Filesystem Location'. This dialog box contains fields for 'Name' (set to '12c Patches') and 'Location' (set to '/stage/patches/12.1/db'). The 'OK' button is visible at the bottom right of the dialog. The main pane shows a table with columns 'Name', 'Status', and 'Last Refreshed', which is currently empty.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To use the patching features of Enterprise Manager, you configure a software library location. You can have one or more library locations. This will be the directory that patches will be stored in when they are transferred to the local cluster.

To navigate to the Provisioning page from the Cluster Database home page:

1. Select Patching and Provisioning, then Software Library from the Enterprise pull-down menu.
2. Select Administration from the Actions pull down menu.
3. Click Add from either the Software Library: Administration page or select Add from the Actions pull-down menu.

Provide a sensible name for the location as well as the full directory path (Location). The software library directory location must reference an existing directory.

Obtaining Oracle RAC Patches

The screenshot shows the Oracle My Oracle Support interface. The top navigation bar includes links for Dashboard, Knowledge, Service Requests, Patches & Updates (which is highlighted with a red box), Community, Certifications, Systems, and More... The right side of the header shows Welcome, James | Contact Us | Sign Out | Help. Below the header is a search bar with options for Favorites, Search Knowledge Base, and Advanced search. A message center indicates last refresh was 7 hours, 59 minutes ago.

The main content area is titled "Patch Search". It features a "Patch Quick Links" sidebar on the left with sections for Software and Patch Search Sites (Oracle Software Delivery Cloud, JD Edwards, PeopleSoft, Sun), Oracle E-Business Suite (Latest R12 Patches, Recommended R12 Patches, Latest 11i Patches, Recommended 11i Patches), and Oracle Server and Tools (Latest Patchsets). The "Latest Patchsets" link is also highlighted with a red box. A note below the sidebar states "All Quick Links open in a new window" and "Latest Patchsets will open in a new browser window".

The main search interface includes tabs for Search, Saved, and Recent. It has fields for "Number/Name or Sun CR ID (Oracle)" and "Product or Family (Advanced)". A "Patch Type" dropdown is set to "Patch". Below these are search criteria fields for "and" and "and", with a "Patch Type" dropdown currently set to "Patch". There are "Search", "Advanced Search", and "Saved Searches" buttons.

The main content area displays two tables of patch sets:

Patch Sets for Product Bundles	Product
Beehive	Oracle Collaboration Suite
Oracle Database	Oracle Developer
Oracle Developer Suite	Oracle Fusion Middleware
Oracle Gateways	Oracle ODBC COREd
Oracle RDB	

Patch Bundles for Individual Products or Components	Product
AMP for PeopleSoft Enterprise	Access Manager for AS/400
Advanced/Secure Networking	Appliance
BEA Tuxedo Tux	BEA WebLogic Network Gatekeeper Services Access Gateway
BEA WebLogic Platform	CDD/Repository
CODASYL DBMS	COM Automation Feature
Change Management Pack	Clickstream Intelligence
Database Certified Configuration	Demand Planning

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The latest patchsets and recommended BPs can be downloaded from the My Oracle Support Web site at the following URL:

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

After signing in to the Web site, 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 patchset 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.

Obtaining Oracle RAC Patches

The screenshot shows the Oracle Patch Search interface. At the top, there is a search bar with filters: Product is Oracle Database, Release is 11.2.0.3.0, and Platform is Linux x86-64. A red box highlights the 'Search' button. Below the search bar, a message says 'For JD Edwards & PeopleSoft, see the Patching Quick Links region... Learn More...' with a red arrow pointing to it.

Patch Search Results

Filters: Product is Oracle Database; Release is Oracle 11.2.0.3.0; Platform is Linux x86-64

Patch Name	Description	Release	Platform (Language)	Classification	Product/Family	Updated	Size
10032375	FAST REFRESH FAILS BY ORA-1008/ORA-12801 (Patch)	11.2.0.3.0	Linux x86-64 (American En...)	General	Oracle Database Fa...	3 days ago	285.9 KB
10102373	UNABLE TO USE ORA11GR2 (11.2.0.1.0) AQAPIJAR WHEN DOING XA FROM JCAPS 6.3 (Patch)	11.2.0.3.0	Generic Platform (Americ...	General	Oracle Database Fa...	6+ weeks ago	21.9 KB
10109915	ASH HANGS IN HIGH REDUNDANCY CONFIG IF 1 OF 5 DISKS GOES OFFLINE (Patch)	11.2.0.3.0	Linux x86-64 (American En...)	General	Oracle Database Fa...	4 days ago	157.3 KB
10145667	ERRORS TRYING TO REWRITE QUERY WITH EXACT TEXT MATCH TO MVIEW (Patch)	11.2.0.3.0	Linux x86-64 (American En...)	General	Oracle Database Fa...	5+ weeks ago	699.6 KB
10182005	SR12.1MAZD - TRC - KRBRRD_COMP (Patch)	11.2.0.3.0	Linux x86-64 (American En...)	General	Oracle Database Fa...	3+ weeks ago	263.2 KB
10263668	ORA-06060 [KOKEGPINLOB1] PRINTING RESULTS OF SQL QUERY (Patch)	11.2.0.3.0	Linux x86-64 (American En...)	General	Oracle Database Fa...	3+ weeks ago	59.2 KB
10359307	KOKC LATCH RECOVERY AREA NEEDS OPTIMIZATION (Patch)	11.2.0.3.0	Linux x86-64 (American En...)	General	Oracle Database Fa...	3+ weeks ago	39.3 KB
11703330	PLACEHOLDER BUG FOR METADATA XML IN 11203 (Patch)	11.2.0.3.0	Generic Platform (Americ...	General	Oracle Database Fa...	4+ weeks ago	786 Byte(s)
11896575	WHEN CREATING SYNONYM FOR SYS.XMLTYPE ON DATABASE 11.2.0.2 GET ORA-22933 (Patch)	11.2.0.3.0	Linux x86-64 (American En...)	General	Oracle Database Fa...	5+ weeks ago	705.9 KB
11901407	REMOVE GATHERING STATISTICS FROM DATABASE UPGRADE PROCESS (Patch)	11.2.0.3.0	Generic Platform (Americ...	General	Oracle Database Fa...	1 week ago	11.3 KB
12312133	STANDBY DB CRASHES WITH ORA-600 [KRCCCB_BUSY] (Patch)	11.2.0.3.0	Linux x86-64 (American En...)	General	Oracle Database Fa...	1 week ago	636.7 KB
12358753	INDEX_STATS TABLE RETURNS WRONG VALUE (Patch)	11.2.0.3.0	Linux x86-64 (American En...)	General	Oracle Database Fa...	4+ weeks ago	122.5 KB
12378705	PLS-001 [54404] REFERENCED ON ACTIVE DATA GUARD DATABASE WHEN BUFFER SPLIT HAPPENED ON RI TARGET, XDTC PARSER GOT SEGMENTATION FAULT (Patch)	11.2.0.3.0	Linux x86-64 (American En...)	General	Oracle Database Fa...	1 week ago	506.2 KB
12424121	ORA-01790: EXPRESSION MUST HAVE SAME DATATYPE AS CORRESPONDING EXPRESSION (Patch)	11.2.0.3.0	Linux x86-64 (American En...)	General	Oracle Database Fa...	6+ weeks ago	143.2 KB
12552578		11.2.0.3.0	Linux x86-64 (American En...)	General	Oracle Database Fa...	1 week ago	1.3 MB

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

For a complete list of patchsets available for your version and platform, click the Product or Family link.

After specifying Product, Release, and Platform, click the Search button. The patch search results are displayed.

Obtaining Oracle RAC Patches

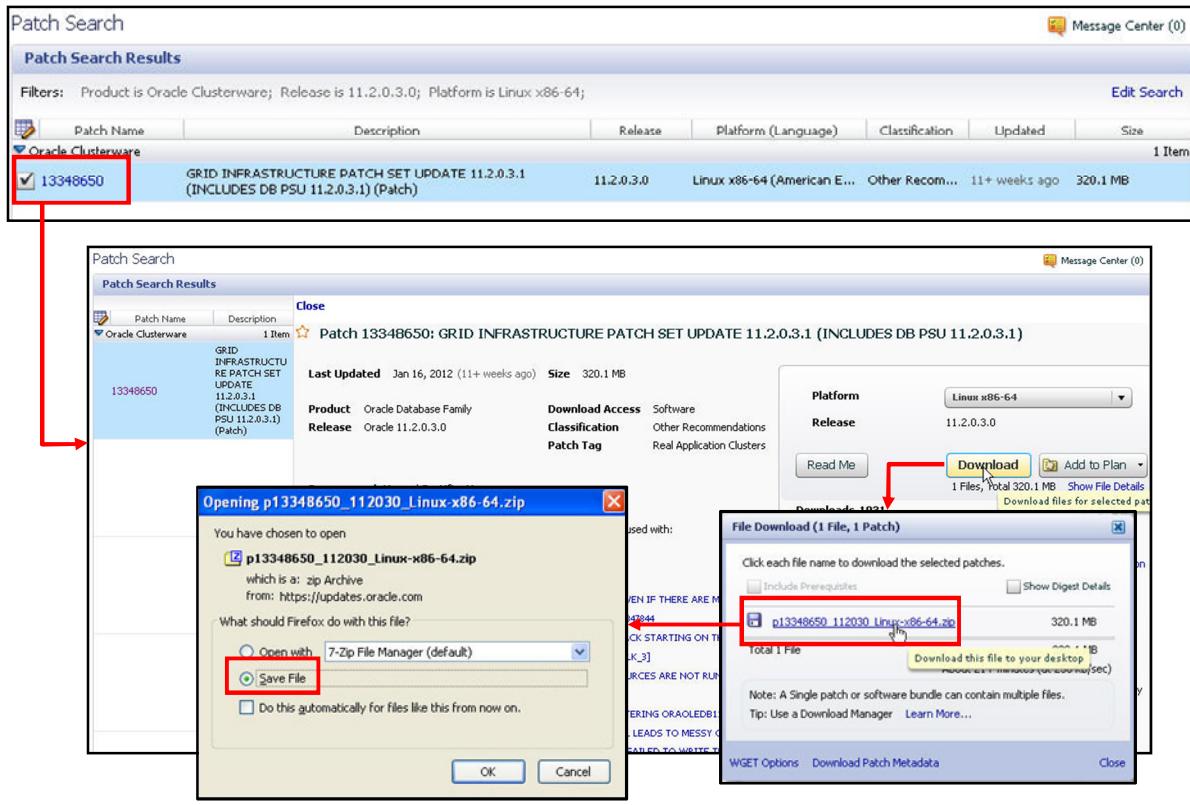
The screenshot shows two windows related to Oracle Patch Search. The top window is titled "Patch Search" and displays search filters: Product (Oracle Real Application Clusters), Release (11.2.0.3.0), and Platform (Linux x86-64). A red arrow points from the "Search" button in this window down to the "Patch Search Results" window below. The "Patch Search Results" window shows a single patch entry:

Patch Name	Description	Release	Platform (Language)	Classification	Updated	Size
13696251	GRID INFRASTRUCTURE PATCH SET UPDATE 11.2.0.3.2 (INCLUDES DB PSU 11.2.0.3.2) (Patch)	11.2.0.3.0	Linux x86-64 (American English)	Other Recomme...	3+ weeks ago	344 MB

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To locate recommended patch sets, 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 patchsets.

Downloading Patches



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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.

Reduced Down-Time Patching for Cluster Environments

Patching Oracle RAC can be completed without taking the entire cluster down.

- OPatch can apply patches in multinode, multipatch fashion.
- OPatch detects whether the database schema is at an earlier patch level than the new patch, and runs SQL commands to bring the schema up to the new patch level.
- OUI installs patchsets as out-of-place upgrades, reducing the down time required for patching.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Patching Oracle RAC can be completed without taking the entire cluster down. In many cases, patching can be performed with zero down time. This also allows for out-of-place upgrades to the cluster software and Oracle Database, reducing the planned maintenance down time required in an Oracle RAC environment.

OPatch can now apply patches in multinode, multipatch fashion. OPatch will not start up instances that have a nonrolling patch applied to it if other instances of the database do not have that patch. OPatch also detects whether the database schema is at an earlier patch level than the new patch, and it runs SQL commands to bring the schema up to the new patch level.

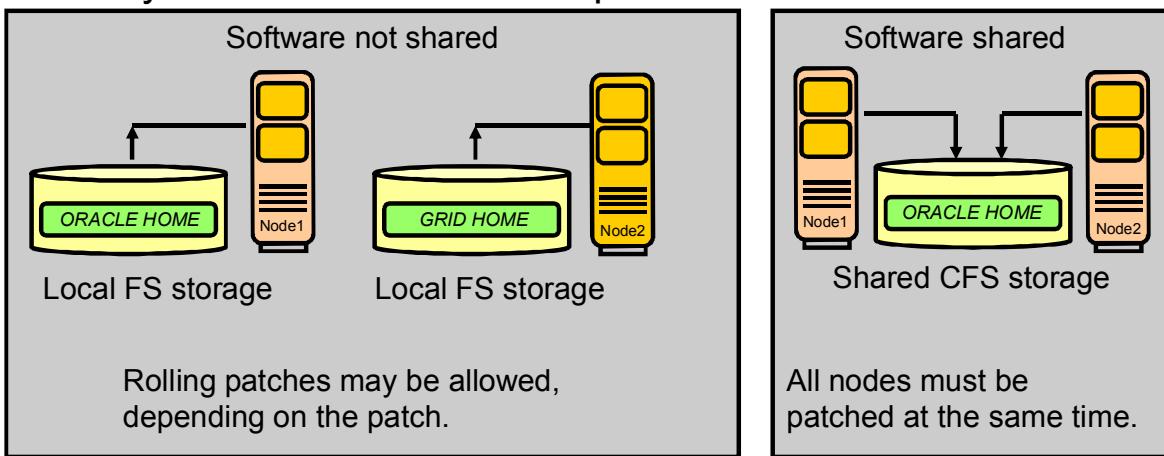
You can use `srvctl` to shut down the Oracle software running within an Oracle home, in preparation for patching. Oracle Grid Infrastructure patching is automated across all nodes, and patches can be applied in a multinode, multipatch fashion.

Patchsets 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.

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 © 2014, Oracle and/or its affiliates. All rights reserved.

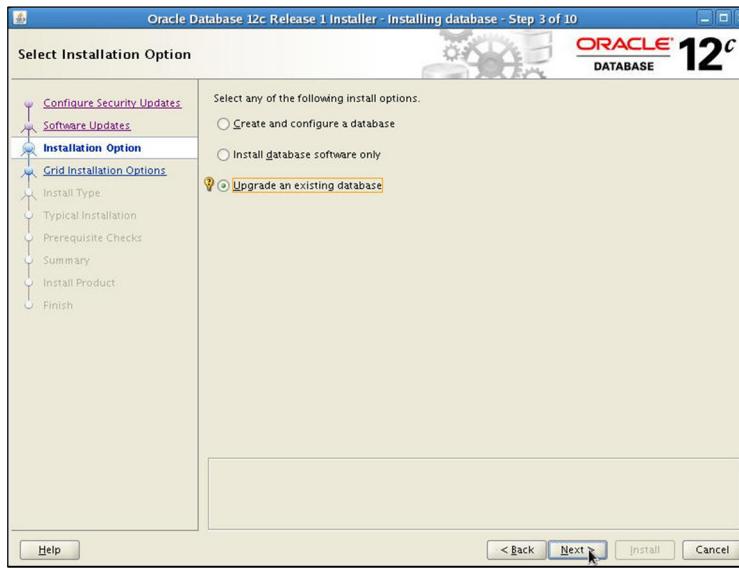
A rolling patch allows one node to be patched to the latest version, while other nodes continue to use the older version and provide business service. This methodology is best suited when you are applying one-off patches that support rolling methodology, maintaining high availability of your targets,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

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

Out-of-Place Database Upgrades

- 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.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

Installing a Rolling Patchset 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 using the Oracle Universal Installer. The upgrade process is completed by the Oracle Database Upgrade Assistant.

Out-of-Place Database Upgrade 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.

```
SQL> @NEW_ORACLE_HOME/rdbms/admin/preupgrd.sql
```

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



Copyright © 2014, 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 Oracle Universal Installer. 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. To use the tool, start SQL*Plus, connect to the database instance as a user with SYSDBA privileges, and execute the `preupgrd.sql` script.

```
SQL> @NEW_ORACLE_HOME/rdbms/admin/preupgrd.sql
```

If `Oracle_Base` is defined, then the generated scripts and log files are created in `Oracle_Base/cfgtoollogs/`. If `Oracle_Base` is not defined, then the generated scripts and log files are created in `ORACLE_HOME/cfgtoollogs/`.

5. Run the `root.sh` script as directed by the OUI. If you are ready, use DBUA to complete the upgrade process.

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 © 2014, 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/12.1.0/grid  
$ opatch command [options]
```

or

```
$ opatch command -oh /u01/app/12.1.0/grid [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 -all | grep -i Rolling
```



Copyright © 2014, 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 Clusterware, ORACLE_HOME refers to the installation directory for Oracle Clusterware, 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 *Grid_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, CRS BPs and CRS MLR 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 © 2014, 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
```

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 © 2014, Oracle and/or its affiliates. All rights reserved.

The OPatch utility has automated the patch application for the Oracle Grid Infrastructure home and the Oracle RAC database homes. It operates by querying existing configurations and automating the steps required for patching each Oracle RAC database home of the same version and the GI home.

The utility must be executed by an operating system user with root privileges (usually the user `root`), and it must be executed on each node in the cluster if the Grid Infrastructure home or Oracle RAC database home is in non-shared storage. The utility should not be run in parallel on the cluster nodes.

Depending on the command-line options specified, one invocation of OPatch can patch the GI home, one or more Oracle RAC database homes, or both GI and Oracle RAC database homes of the same Oracle release version. You can also roll back the patch with the same selectivity.

OPatch Automation Examples

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

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

- To patch only the GI home:

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

- To patch one or more Oracle RAC database homes:

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



Copyright © 2014, 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> -ocmrft
<ocm_response_file>
```

To patch only the GI home:

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

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

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

To roll back the patch from the GI home and each Oracle RAC database home:

```
# opatchauto rollback <UNZIPPED_PATCH_LOCATION> -ocmrf  
<ocm_response_file>
```

To roll back the patch from the GI home:

```
# opatchauto rollback <UNZIPPED_PATCH_LOCATION> -oh <Grid_home>  
-ocmrf <ocm_response_file>
```

To roll back the patch from one or more Oracle RAC database homes:

```
# opatchauto rollback <UNZIPPED_PATCH_LOCATION> -database db1, db2  
-ocmrf <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 © 2014, 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, 2013 at 11:55 PM, then it is named as follows:

opatch_05-17-2013_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 : Tue Apr 26 23:00:55 PDT 2013
Oracle Home : /u01/app/oracle/product/12.1.0/dbhome_1/
OPatch Ver. : 12.1.0.0.0
Current Dir : /scratch/oui/OPatch
Command : lsinventory
Log File :
/u01/app/oracle/product/12.1.0/dbhome_1/cfgtoollogs/opatch/opatch-
2013_Apr_26_23-00-55-PDT_Tue.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 © 2014, 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 Oracle Universal Installer (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

Quiz

Which tools can be used to install a patchset?

- a. Oracle Universal Installer
- b. OPatch
- c. Enterprise Manager Database Console
- d. Database Configuration Assistant



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a, c

In Oracle 12c, the Oracle Universal Installer or Enterprise Manager can be used.

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 patchset with the Oracle Universal Installer (OUI) utility
- Install a patch with the `opatch` utility



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

11

Oracle RAC One Node

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Perform an online database migration
- 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 © 2014, 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 © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle RAC One Node is an option to Oracle Database Enterprise Edition available since Oracle Database 11g release 2. 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 © 2014, 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 only perform 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/12.1.0/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 © 2014, 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 Migration

- Oracle RAC One Node allows the online relocation of the database from one server to another.
- The migration 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 © 2014, 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 migration 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 Migration 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 © 2014, 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 Migration

```
[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 © 2014, 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 migration 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
```

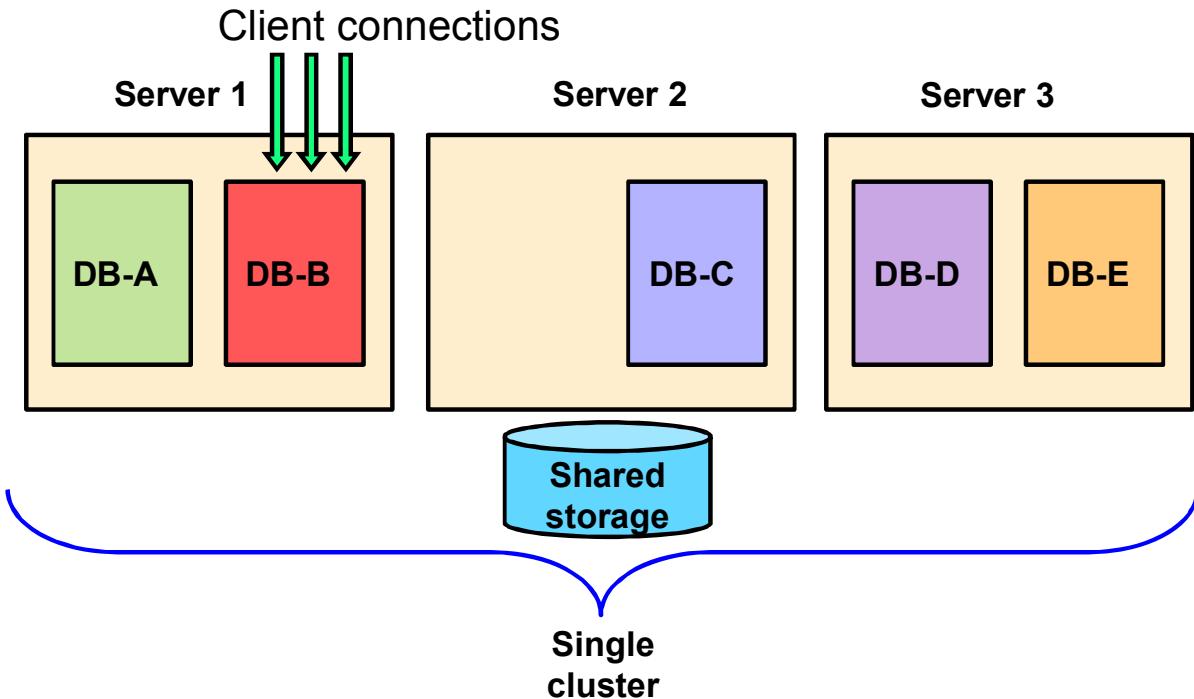
During the relocation:

```
[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 Migration Illustration



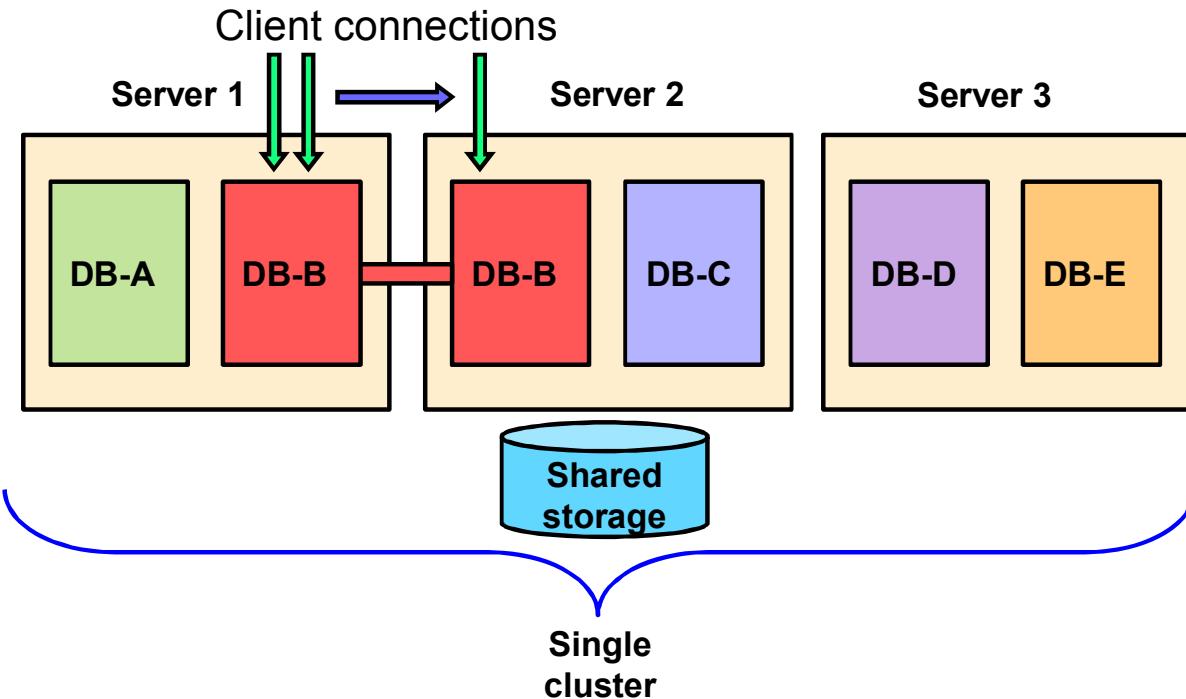
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the example configuration in the slide, you have five single-instance Oracle RAC One Node databases running in a cluster of three servers. Server 1 is hosting Oracle RAC One Node databases DB-A (database A) and DB-B (database B), server 2 is hosting database DB-C (database C) and server 3 is hosting databases DB-D (database D) and DB-E (database E).

Each server runs one OS. In servers 1 and 3, multiple databases are consolidated onto a single OS. This deployment itself provides many consolidation benefits. However, online database relocation, a unique feature of Oracle RAC One Node that provides live migration of databases across nodes in the cluster, enables many additional benefits.

Online Migration Illustration



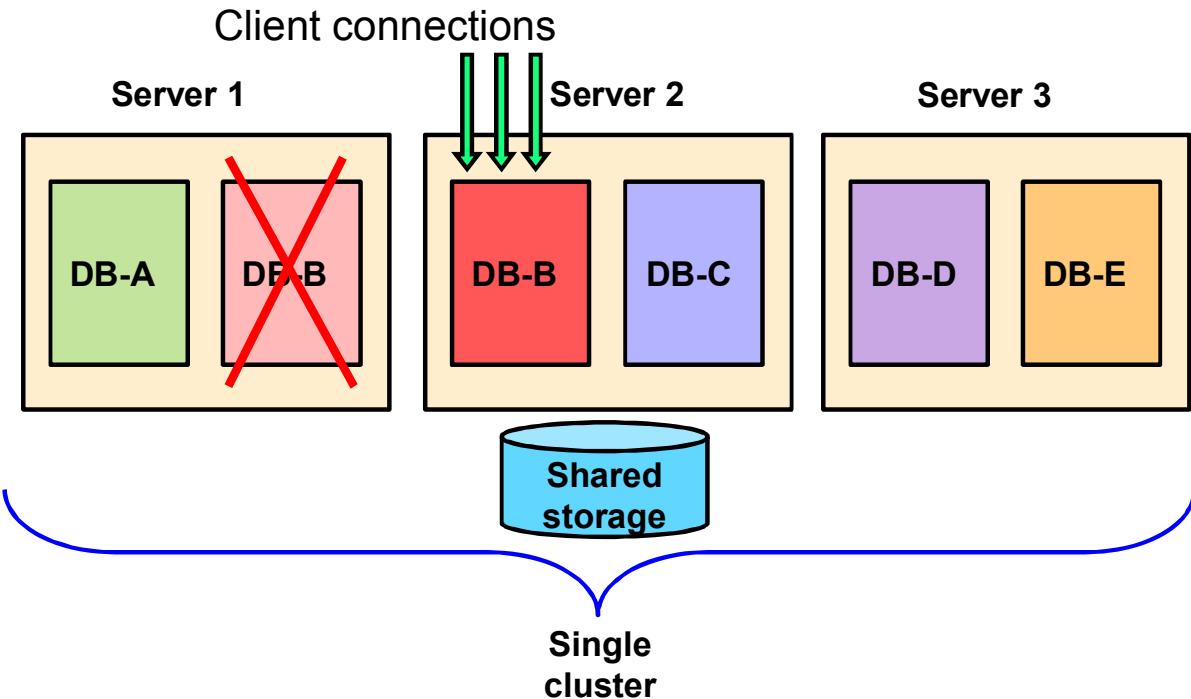
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Online database relocation allows an online migration of a database from one server to another server. Online database relocation leverages the ability of Oracle Real Application Clusters to simultaneously run multiple instances servicing a single database. In the figure in the slide, the database B RAC One Node database on server 1 is migrated to server 2. Oracle RAC One Node starts up a second DB-B instance on server 2, and for a short period of time runs in an active-active configuration.

As connections complete their transactions on server 1, they are migrated to the instance on server 2.

Online Migration Illustration

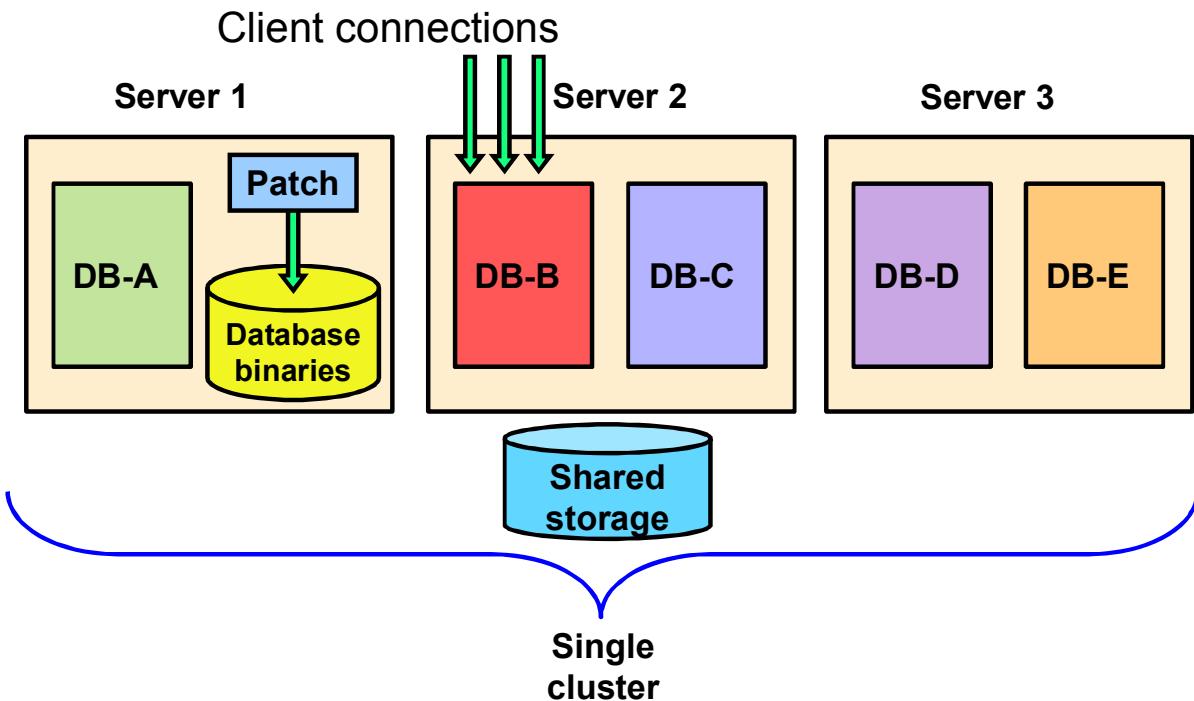


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Once all the connections have migrated to the database B instance on server 2, the instance on server 1 is shut down and the migration is complete. To sum up, you have migrated database B from server 1 to server 2 while the database was online and performing work. To extend the example, you could initiate an operating system upgrade or patch including a reboot of server 1 by simply migrating database A (DB-A).

Online Maintenance: Rolling Patches



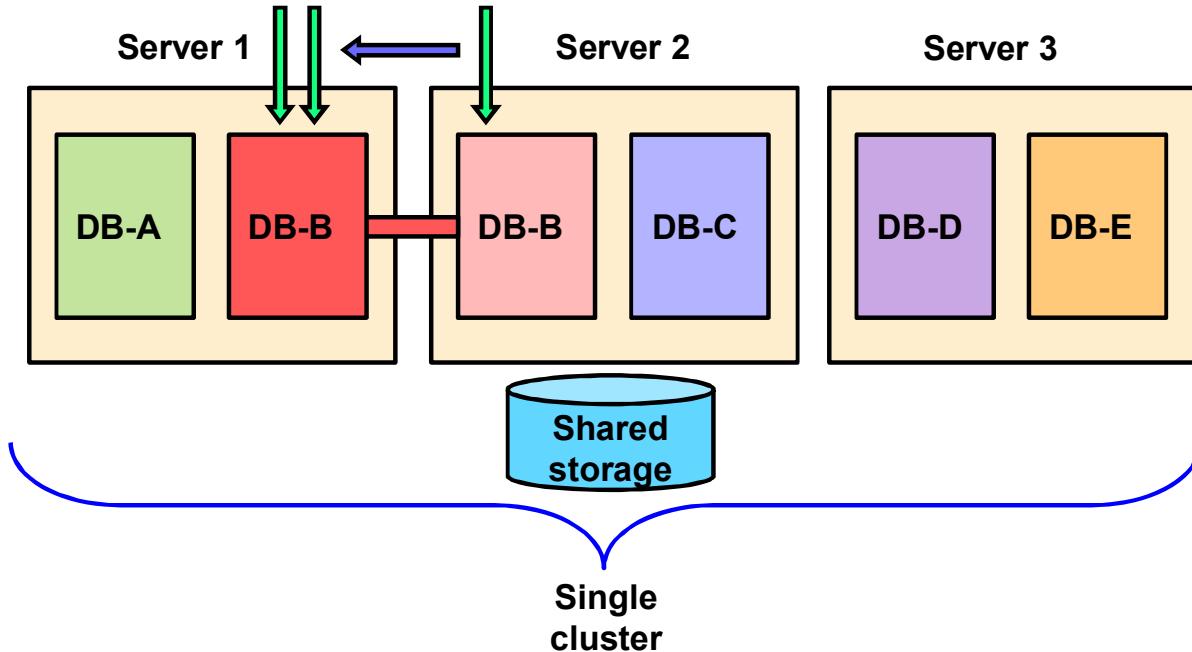
ORACLE

Copyright © 2014, 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 migration, the database binaries that had hosted the instance formerly running on server 1 remain available for patching.

Online Maintenance: Rolling Patches



ORACLE

Copyright © 2014, 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 migration between instances at different patch levels, the operation is completely online and requires no disruption to end users. In this case, the online database migration migrates the connections back to server 1, then the instance on server 2 is shut down, completing the migration. 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 © 2014, 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 © 2014, 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
Database unique name: orcl
Database name: orcl
Oracle home: /u01/app/oracle/product/12.1.0/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 © 2014, 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.
 - It has shared storage: either Oracle Cluster File System or Oracle ASM is available and accessible from all nodes.
 - Your applications have no design characteristics that preclude their use in a clustered environment.



Copyright © 2014, 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.
- It has shared storage: either Oracle Cluster File System or Oracle ASM 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 © 2014, 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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

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.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a, b, d

Summary

In this lesson, you should have learned how to:

- Perform an online database migration
- 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 © 2014, 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 using DBCA
- RAC One Node online migration
- Convert a RAC One Node database to a RAC Database



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

12

Quality of Service Management

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

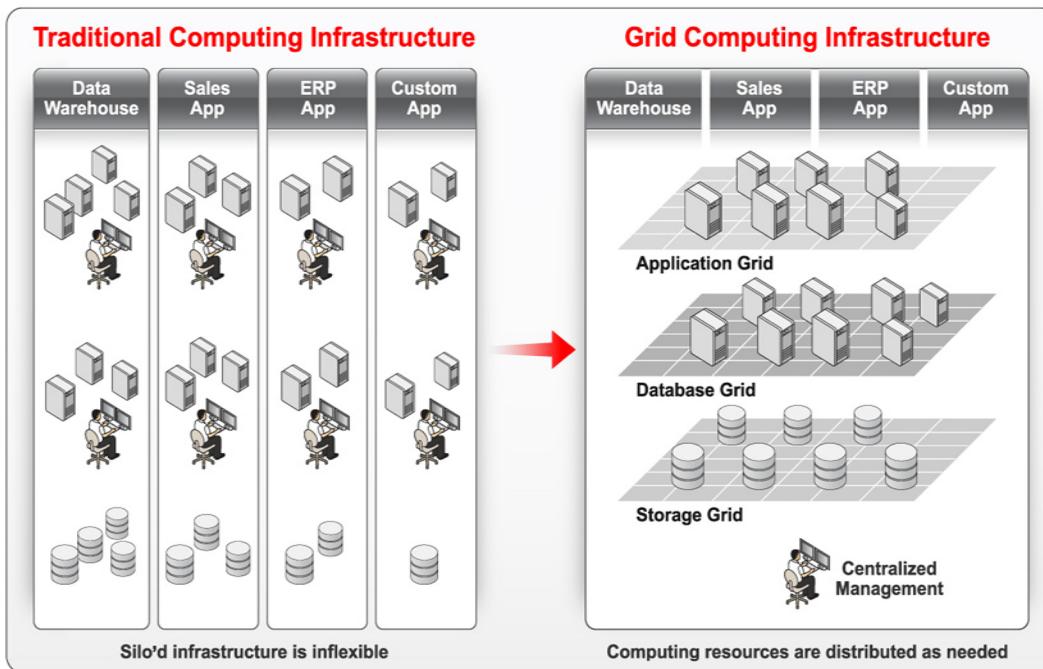
After completing this lesson, you should be able to describe:

- 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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

QoS Management Background



ORACLE

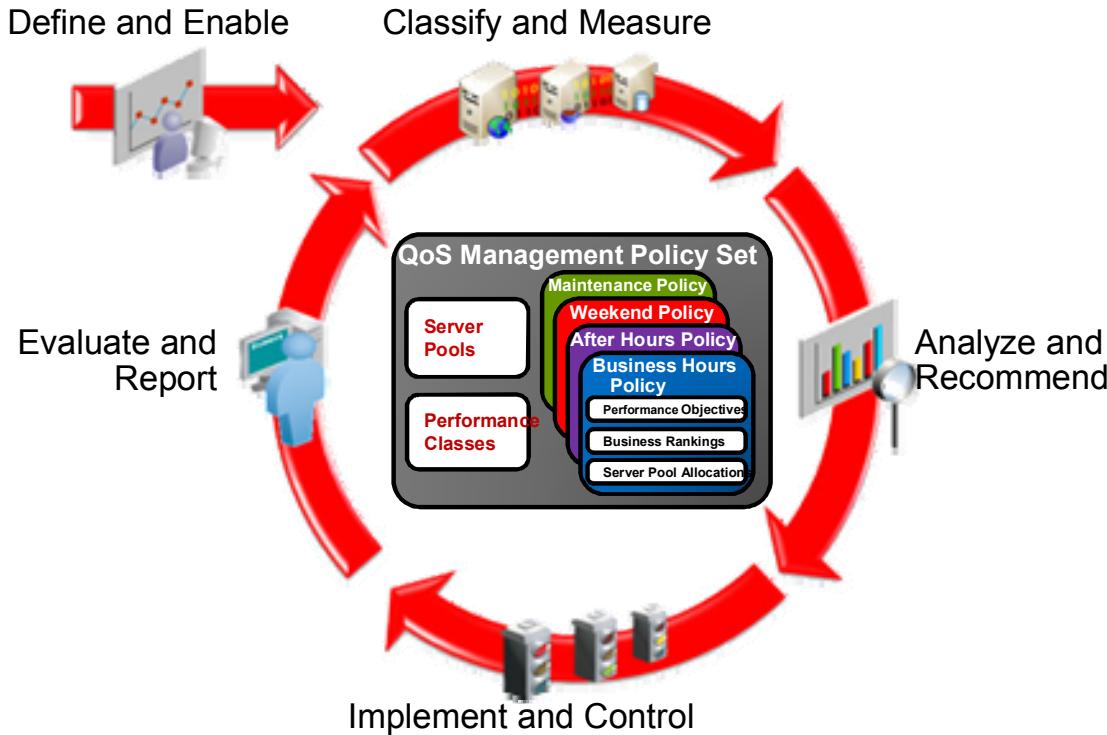
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 © 2014, 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 © 2014, Oracle and/or its affiliates. All rights reserved.

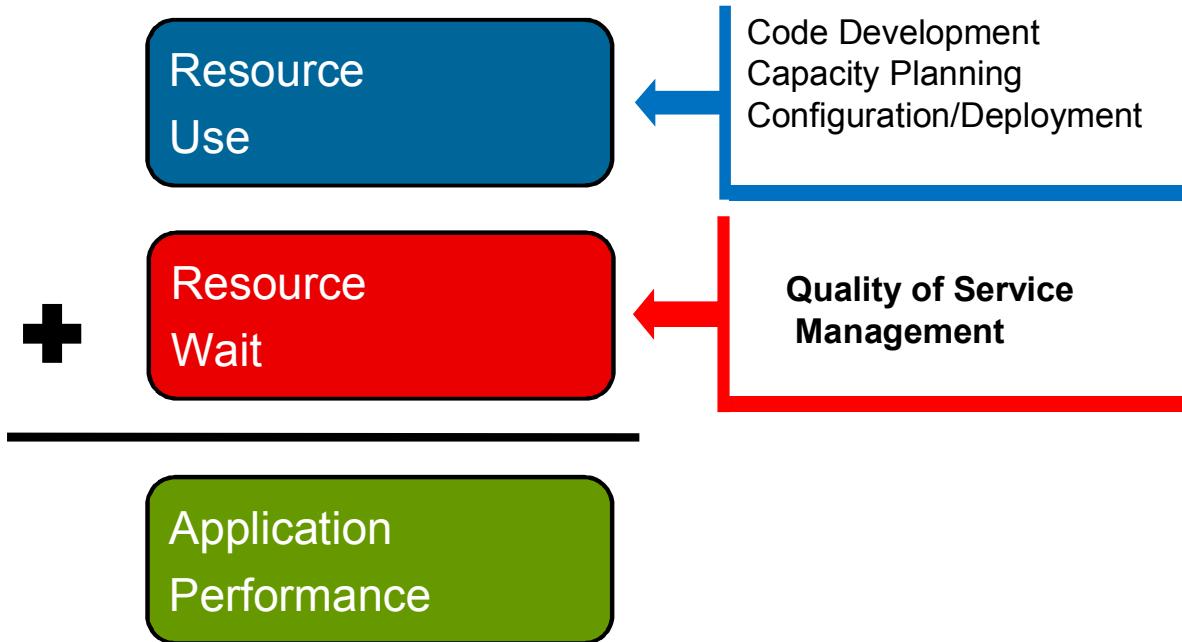
The initial incarnation of QoS Management is as a feature of the Oracle Database product family in association with Oracle Real Application Clusters (RAC) software. It was first introduced in Oracle Database 11g release 2.

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 towards a broader workload management solution.

QoS Management Focus



ORACLE

Copyright © 2014, 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 for example. 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 © 2014, 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 © 2014, 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.

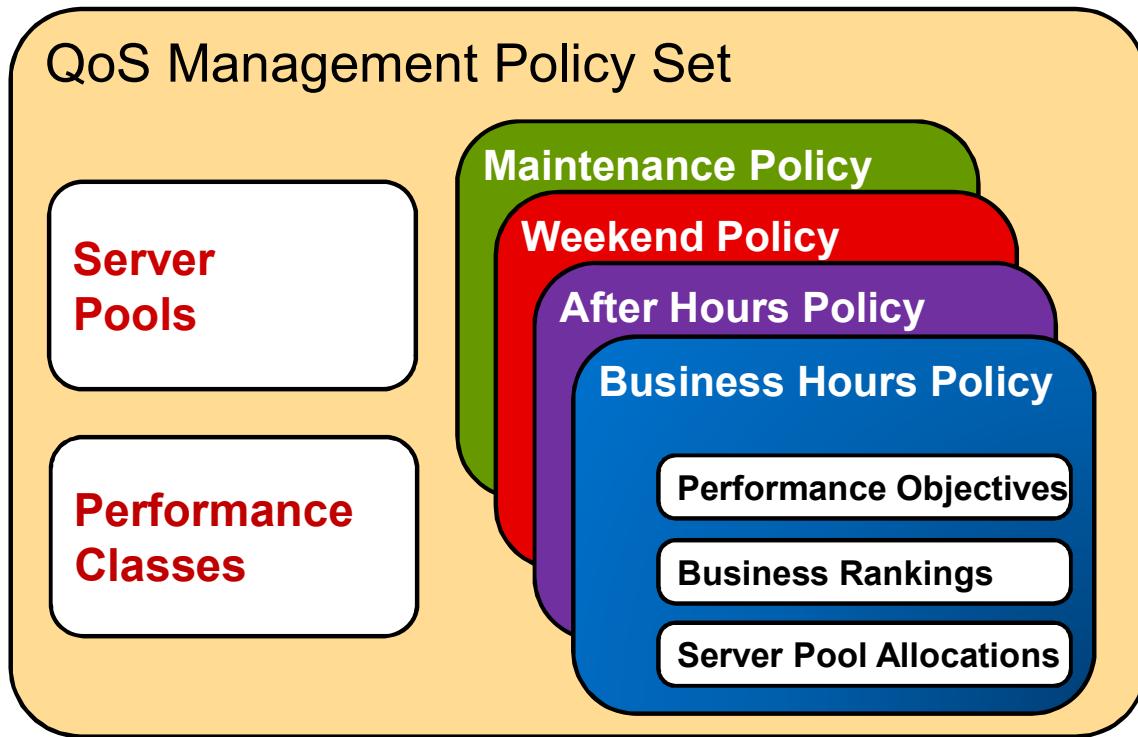
QoS Management periodically evaluates database server CPU wait times to identify workloads that are not meeting performance objectives. If needed, QoS Management provides recommendations for adjusting the size of the server pools or alterations to Database Resource Manager (DBRM) consumer group mappings. 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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

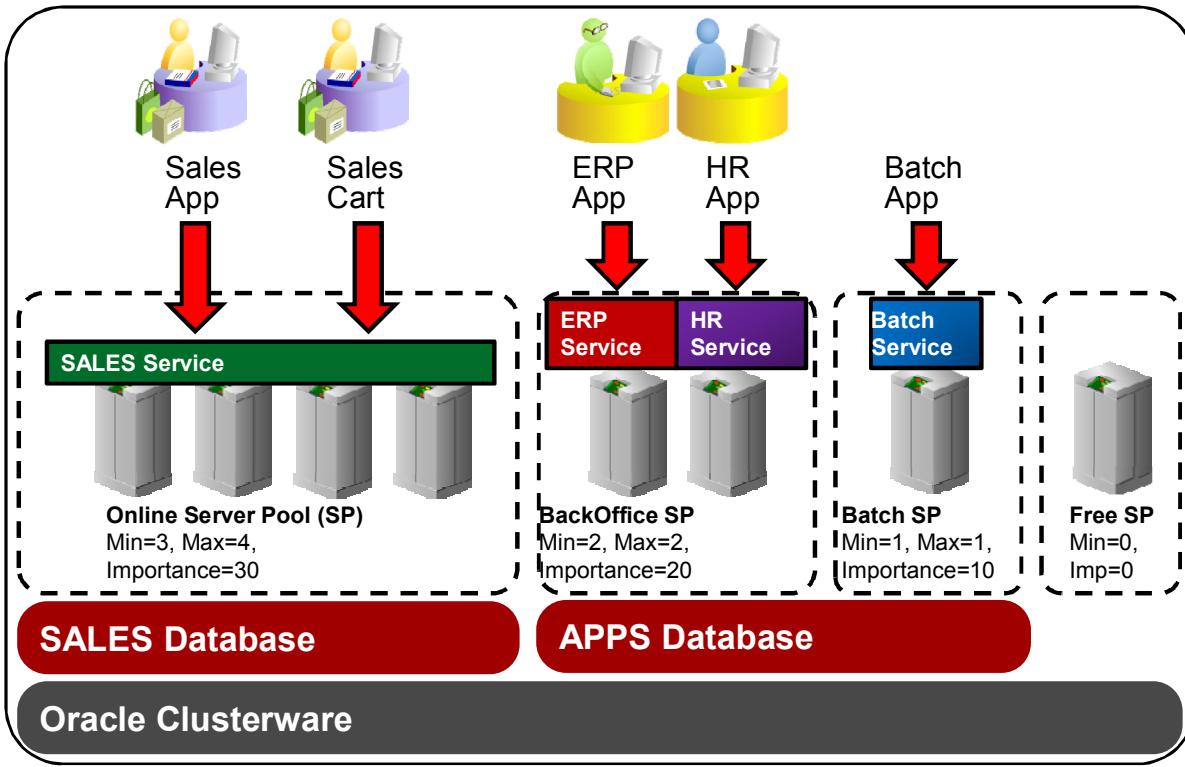
A central concept in QoS Management is the policy set. A policy set allows you to specify your resources, performance classes (workloads), and a collection of performance policies that specify the performance objective for each performance class and sets constraints for resource availability. QoS Management 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 © 2014, 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 of the workloads 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 © 2014, 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 part of the application, you can create additional performance classes and use classifiers that include MODULE, ACTION, or PROGRAM in addition to the 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 © 2014, 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 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 © 2014, 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 © 2014, Oracle and/or its affiliates. All rights reserved.

Within a performance policy, you can also rank each performance class. This rank assigns a relative level of business criticality to each performance objective. When there are not enough resources available to meet all the performance objectives for all performance classes, the performance objectives for the more critical performance classes must be met at the expense of the less critical ones. The available rank settings are Highest, High, Medium, Low, or Lowest. Note that if more than one class is assigned a particular rank (for example, Medium), classes are then ordered within that ranking alphabetically.

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 © 2014, Oracle and/or its affiliates. All rights reserved.

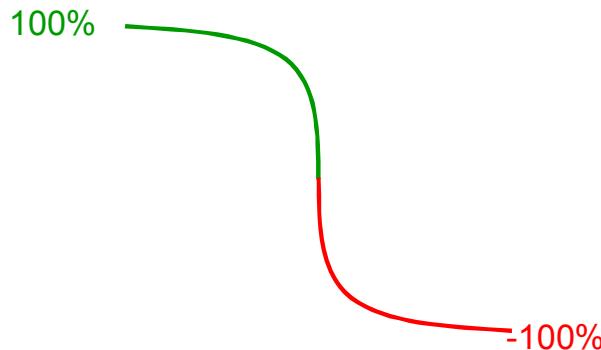
You create a performance objective for each performance class to specify the desired performance level for that performance class. A performance objective specifies both a business requirement, and the work to which it applies (the performance class). For example, a performance objective might say that 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.



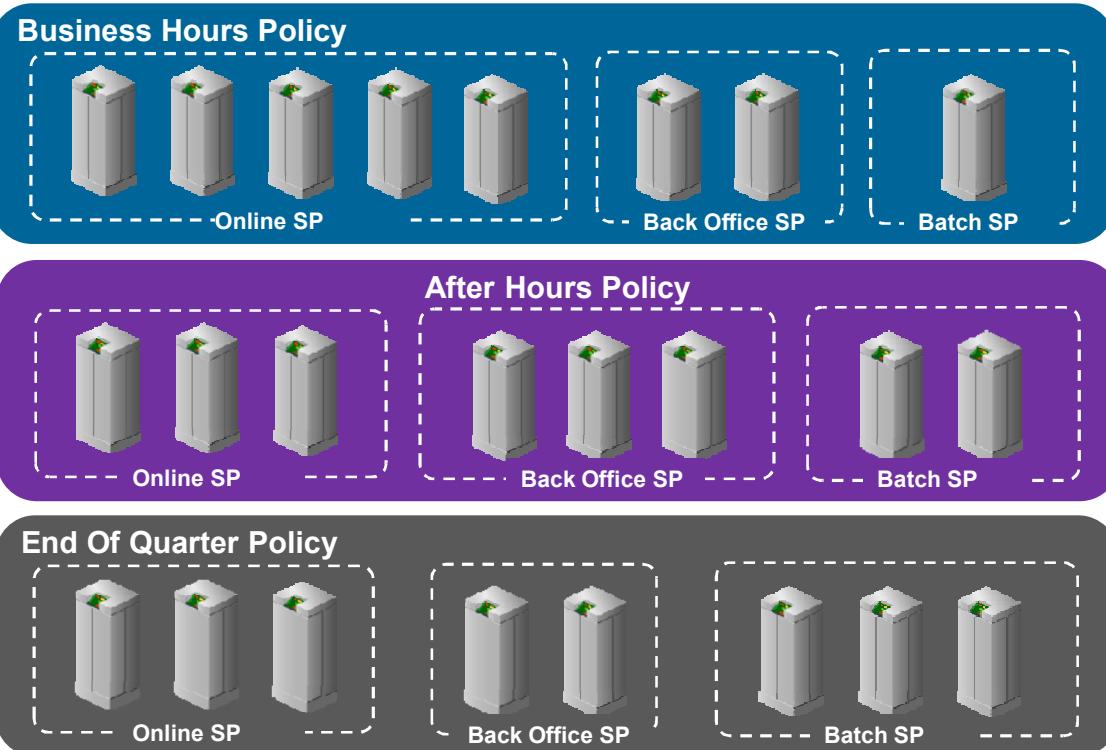
ORACLE

Copyright © 2014, 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 © 2014, Oracle and/or its affiliates. All rights reserved.

A performance policy can also include a set of server pool directive overrides. A server pool directive override sets the minimum server count, maximum server count, and importance attributes for a server pool when the performance policy is in effect. Server pool directive overrides serve as constraints on the recommendations proposed by QoS Management, because the server pool directive overrides are honored while the performance policy is active. For example, QoS Management will never recommend moving a server out of a server pool if doing so will leave the server pool below its minimum server count value.

Server pool directive overrides can be used to define the normal state of server pools at different points in time. The slide illustrates an example. Under normal conditions, these server pool settings would be expected to handle the prevailing workload. If there is a sudden increase in the workload requests for a performance class, then the associated server pool might require additional resources beyond what is specified in the performance policy.

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 © 2014, Oracle and/or its affiliates. All rights reserved.

QoS Management uses a standardized set of metrics, which are 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.

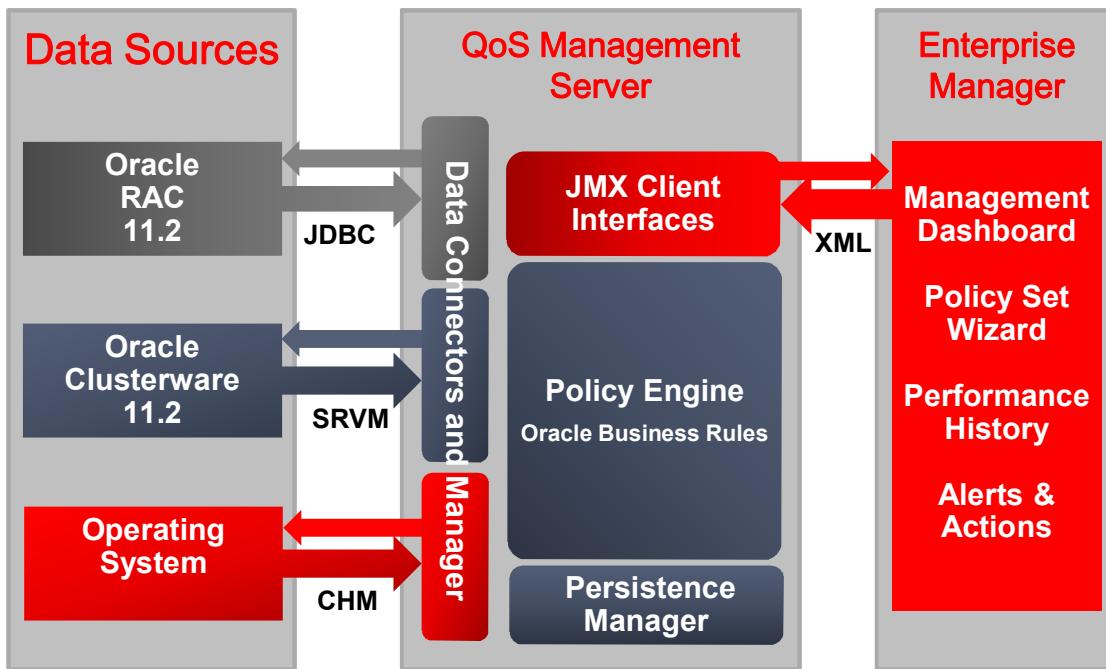
A key performance metric is response time, or the difference between the time a request comes in and the time a response is sent out. The response time for all database calls in a performance class is averaged and presented as the average response time. Another important performance metric is the arrival 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 © 2014, 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 © 2014, 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

The screenshot shows the Oracle Enterprise Manager 11g Database Control interface. The top navigation bar includes links for Setup, Preferences, Help, Logout, Cluster, and Database. The main content area is titled "QoS Management Dashboard". A section titled "Recommended Actions" lists an action: "Rank 1: Promote sales cart from Consumer Group 2 to Consumer Group 0" with a checked checkbox. Below this, it says "Action Promote sales cart from Consumer Group 2 to Consumer Group 0.", "Estimated Time 2 minutes", "Rationale All potential single mapping changes have been analyzed. Changes evaluated and rejected are listed below.", and "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." A table titled "Projected Results" shows performance metrics for various application classes:

Performance Class	Performance Satisfaction Metric (Last 5 min)			Average Response Time	
	Projected (%)	Projected Change (%)	Objective Value (sec)		Current Value (sec)
Default	100	0.0	0.0000	0.01155	0.01706
hr app	71	0.0	0.0100	0.00293	0.00293
sales app	84	-16.1	0.00800	0.02504	0.05039
erp app	42	0.0	0.00500	0.00291	0.00291
sales cart	84	3.8	0.00500	0.04141	0.03157

A button labeled "Implement" is located at the bottom of this section. Below the table, there are two sections: "Donor Performance Classes" and "Donor Server Pools", each with detailed descriptions. At the bottom of the page, there is a copyright notice: "Copyright © 1996, 2009, Oracle. All rights reserved. Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners. About Oracle Enterprise Manager".

ORACLE®

Copyright © 2014, 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.

Quiz

Oracle Database Quality of Service Management helps to meet performance objectives by reducing resource usage.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Oracle Database of Service Management helps to meet performance objectives by managing and reducing resource wait times, not resource usage.

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a, c

Oracle Database Quality of Service Management can identify I/O performance issues or global cache bottlenecks, but cannot address them in this release.

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Lesson 12 Demonstrations

- Configuring Quality of Service Management
- Using Quality of Service Management



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

13

Multitenant Architecture and RAC

ORACLE®

Copyright © 2014, 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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

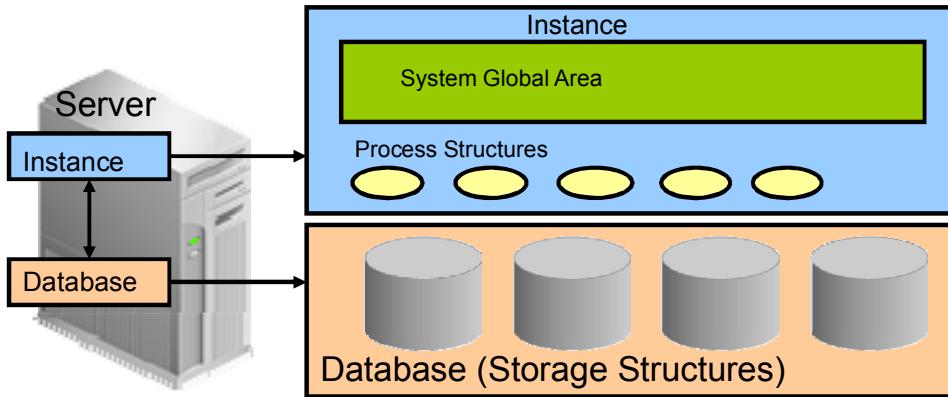
Note: For a complete understanding of Oracle Multitenant new option and usage, refer to the following guides in the Oracle documentation:

- *Oracle Database Administrator's Guide 12c Release 1 (12.1)*
- *Oracle Database Concepts 12c Release 1 (12.1)*

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 © 2014, 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, memory allocation for the SGA, and it 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 © 2014, 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.

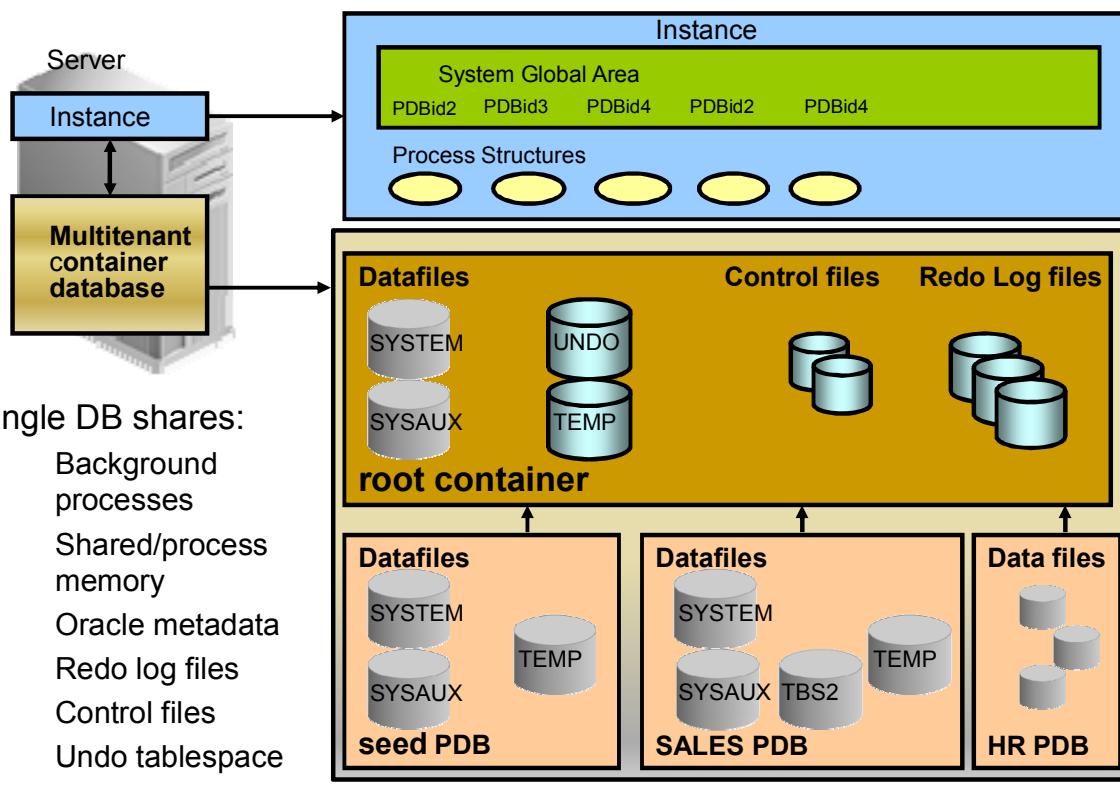
DBA resource costs are reduced with:

- *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



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The graphic in the slide shows a CDB with four containers: the root, the seed, and two PDBs. The two applications use a single instance and are maintained separately.

At the physical level, the CDB has a database instance and database files, just as a non-CDB does.

- The redo log files are common for the whole CDB. The information it contains is annotated with the identity of the PDB where a change occurs. Oracle GoldenGate is enhanced to understand the format of the redo log for a CDB. All PDBs in a CDB share the ARCHIVELOG mode of the CDB.
- The control files are common for the whole CDB. The control files are updated to reflect any additional tablespace and data files of plugged PDBs.
- The UNDO tablespace is common for all containers in the database instance.
- A temporary tablespace common to all containers is required. But each PDB can hold its own temporary tablespace for its own local users.
- Each container has its own data dictionary stored in its proper SYSTEM tablespace, containing its own metadata, and a SYSAUX tablespace.
- The PDBs can create tablespaces within the PDB according to application needs.
- Each datafile is associated with a specific container, named `CON_ID`.

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 253 PDBs in a CDB including the seed
 - Limit of 1024 services in a CDB



Copyright © 2014, 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 appears 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 253 PDBs, including the seed, and 252 user-defined PDBs can therefore be created, with IDs ranging from 3 to 254.

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

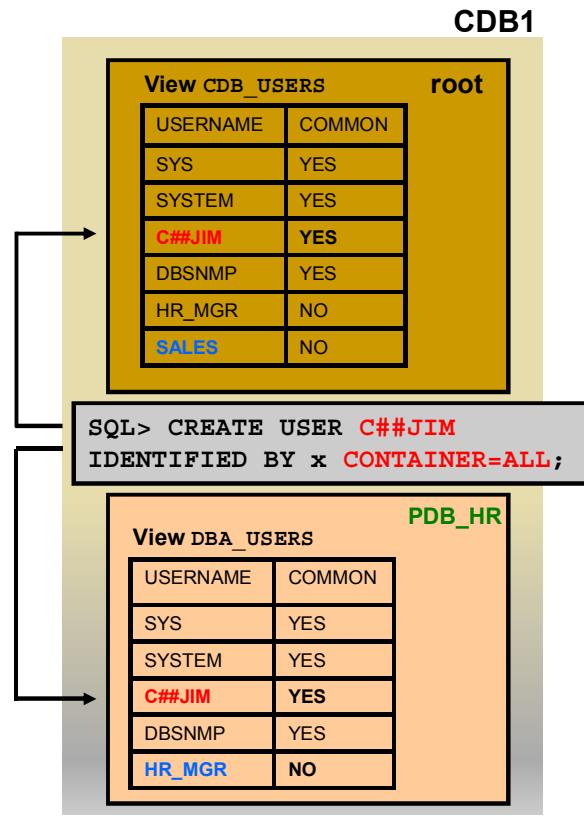
- A **common role** can only be created in the root container.

```
SQL> CREATE ROLE C##manager
CONTAINER=ALL;
```

- A **local role** can only be created and known in a PDB.

- Privileges

- CDB vs PDB level



ORACLE

Copyright © 2014, 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 one and 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

CDB_xxx All objects in the multitenant container database across all PDBs

DBA_xxx All of the objects in a container or pluggable database

ALL_xxx Objects accessible by the current user

USER_xxx Objects owned by the current user

```
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%';
```

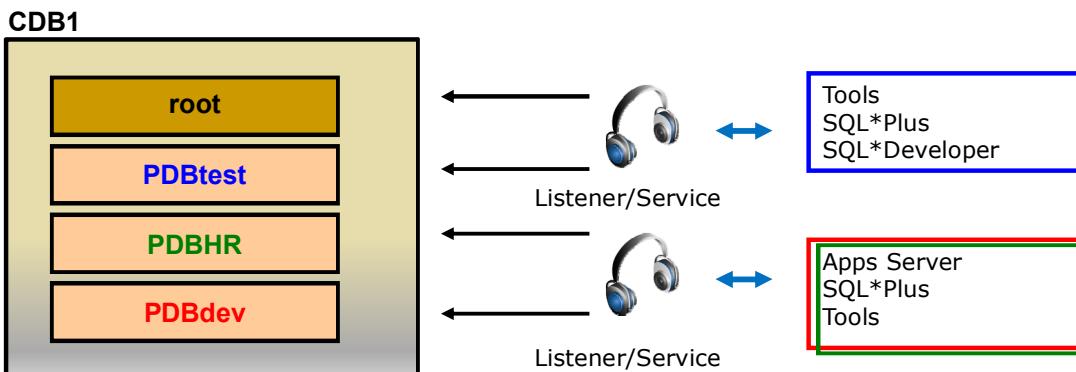
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

For backwards-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 shows, 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*, that 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 backwards-compatibility principle implies 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 © 2014, 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 domain name.
- Each new PDB is assigned a service name: the service name is the PDB name given at PDB creation concatenated with domain name. If you create or plug a PDBtest PDB, its service name is PDBtest concatenated with 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 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 under common SYS user granted 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:

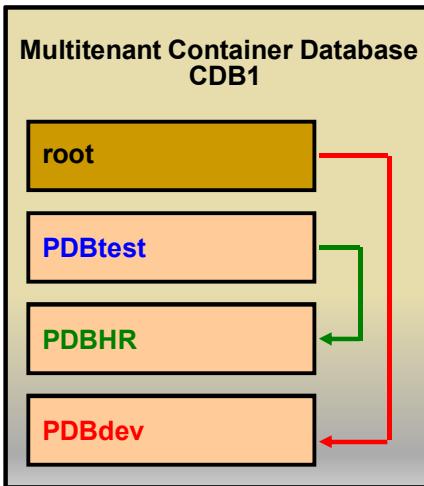
```
SQL> CONNECT username@"hostname:portnumber/service_name"  
SQL> CONNECT username@"localhost:portnumber/service_name"
```

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 common or local user.
- Using ALTER SESSION SET CONTAINER allows connection under common user only who is granted new system privilege SET CONTAINER.
 - AFTER LOGON triggers do not fire.
 - Transactions are still pending after switching containers.

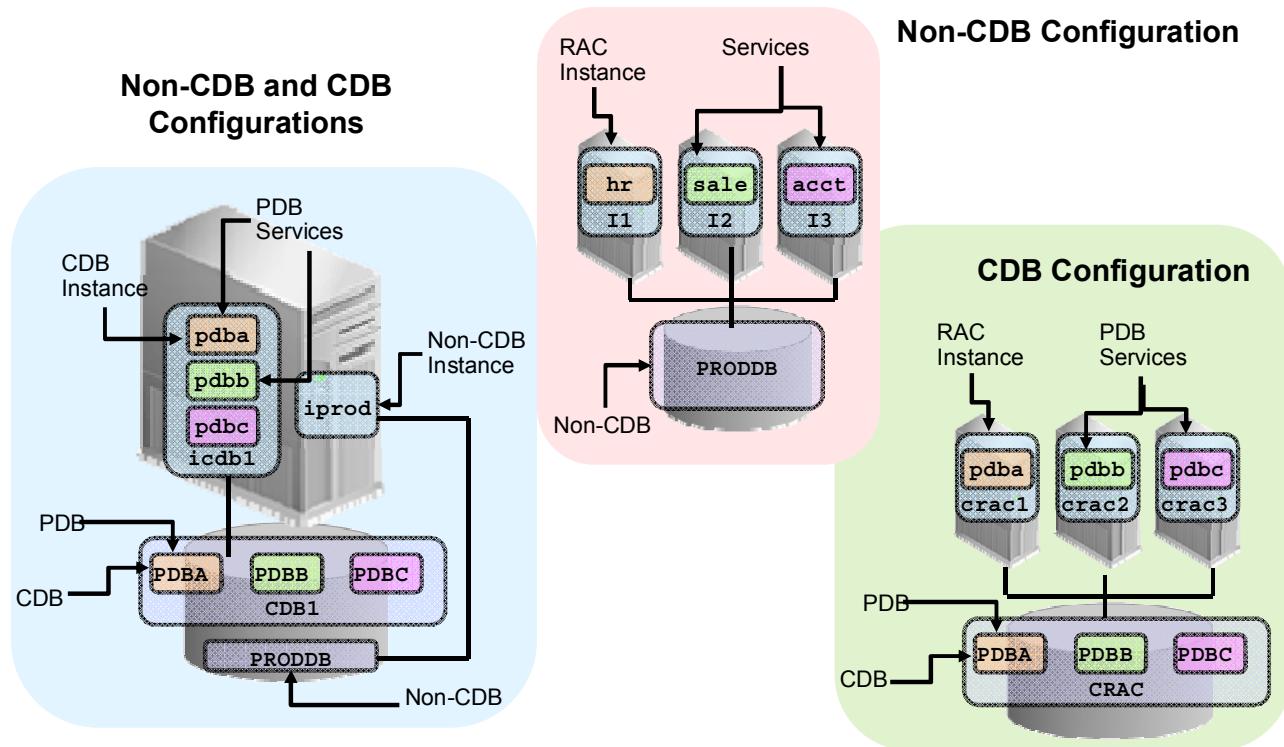
ORACLE

Copyright © 2014, 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 under a common user only who is granted the new system privilege SET CONTAINER.
 - Be aware that AFTER LOGON trigger do not fire.
 - Transactions that are not 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 © 2014, Oracle and/or its affiliates. All rights reserved.

In Oracle Database 12c, there are now three possible configuration options:

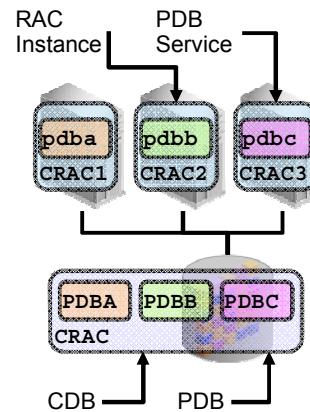
- **Non-CDB:** The old Oracle Database 11g architecture
- **Single-tenant configuration:** The special case of the new architecture, which does not require the licensed option (Oracle Multitenant option)
- **Multitenant configuration:** Typically more than one pluggable database (PDB) per multitenant 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 one and only one non-CDB or CDB.
- In a RAC environment, several instances can be associated to 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
 - One UNDO tablespace per CDB instance
 - 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 © 2014, Oracle and/or its affiliates. All rights reserved.

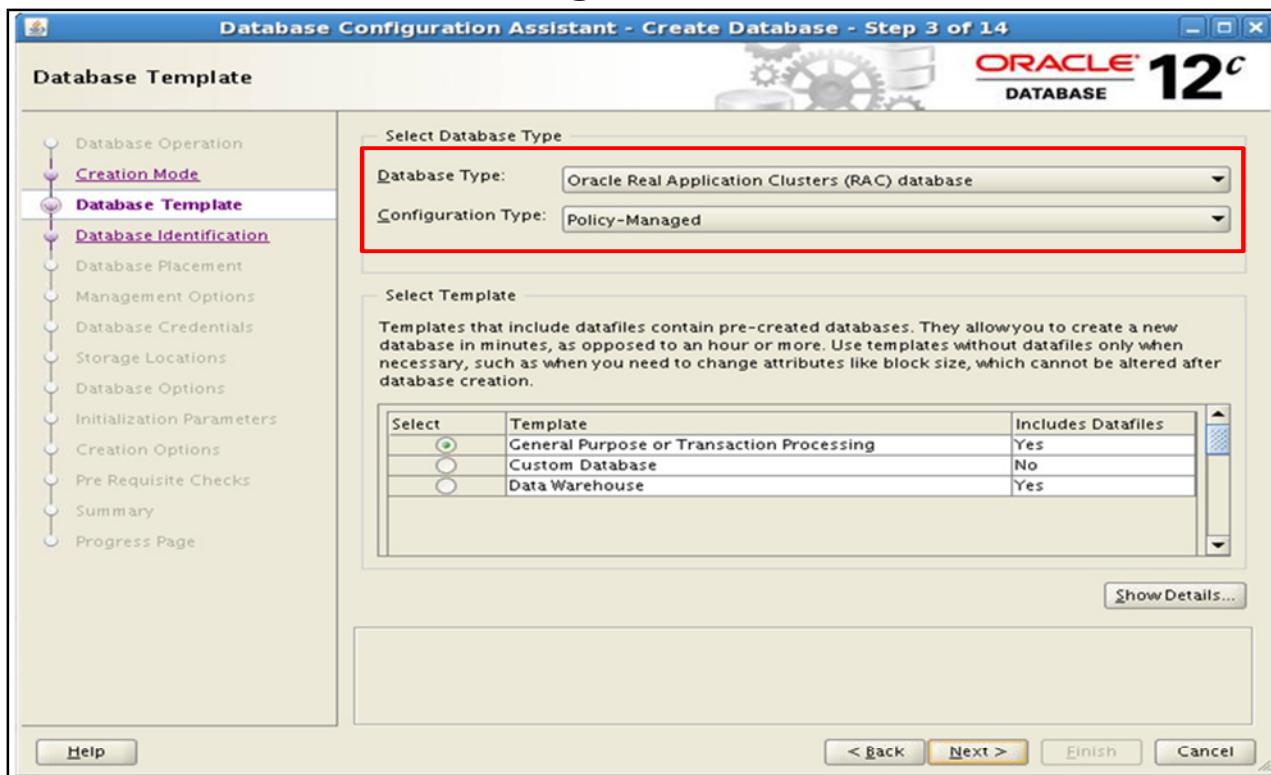
A RAC CDB, exactly as a RAC non-CDB, contains:

- One UNDO tablespaces per instance
- At least two groups of redo log files per instance
- Internal services at CDB level: Like in a RAC non-CDB, `SYS$BACKGROUND` is used by the background processes only. `SYS$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. A special Oracle CDB service is created by default for the Oracle RAC CDB whose name is the CDB name.

With the multitenant architecture, each container is exposed as a service. The root and each PDB is assigned a database service when the container is created.

With clusterware policy-managed databases, each PDB service can be exposed across all the RAC instances or across a subset of instances or on a single instance (as shown in the diagram on 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 neither PDB service-to-node-name mappings.

Creating a RAC CDB



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

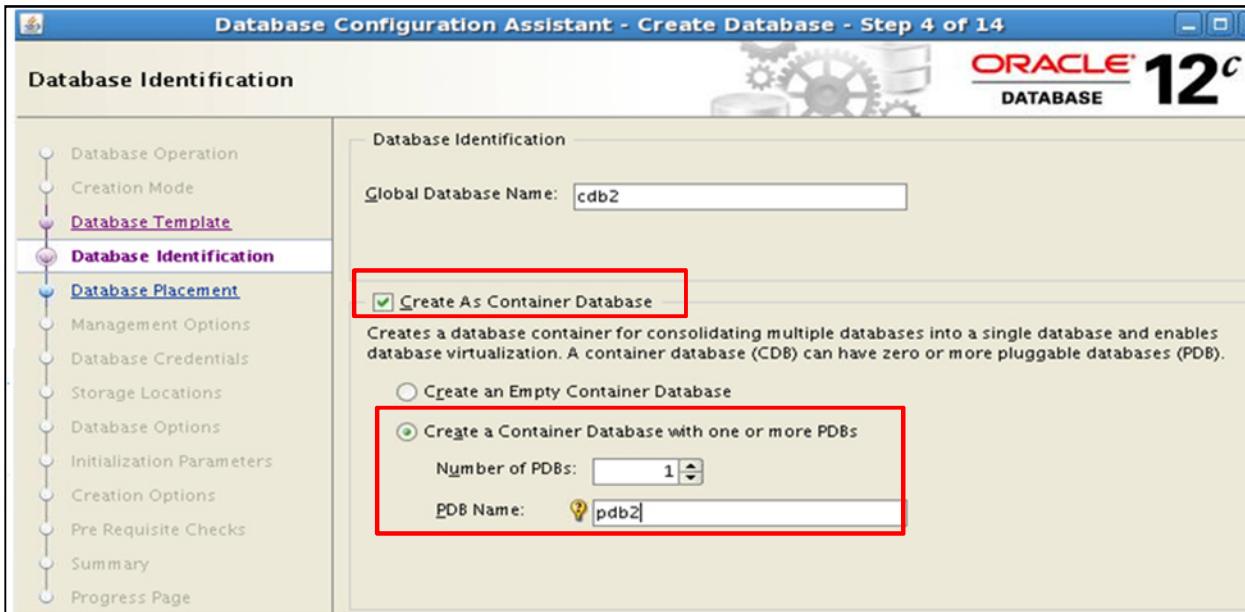
When creating a RAC CDB with DBCA, define the type of database, like you do when you create a RAC non-CDB:

- 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 style:

- Policy-Managed
- Admin-Managed

Creating a RAC CDB Including PDBs



ORACLE

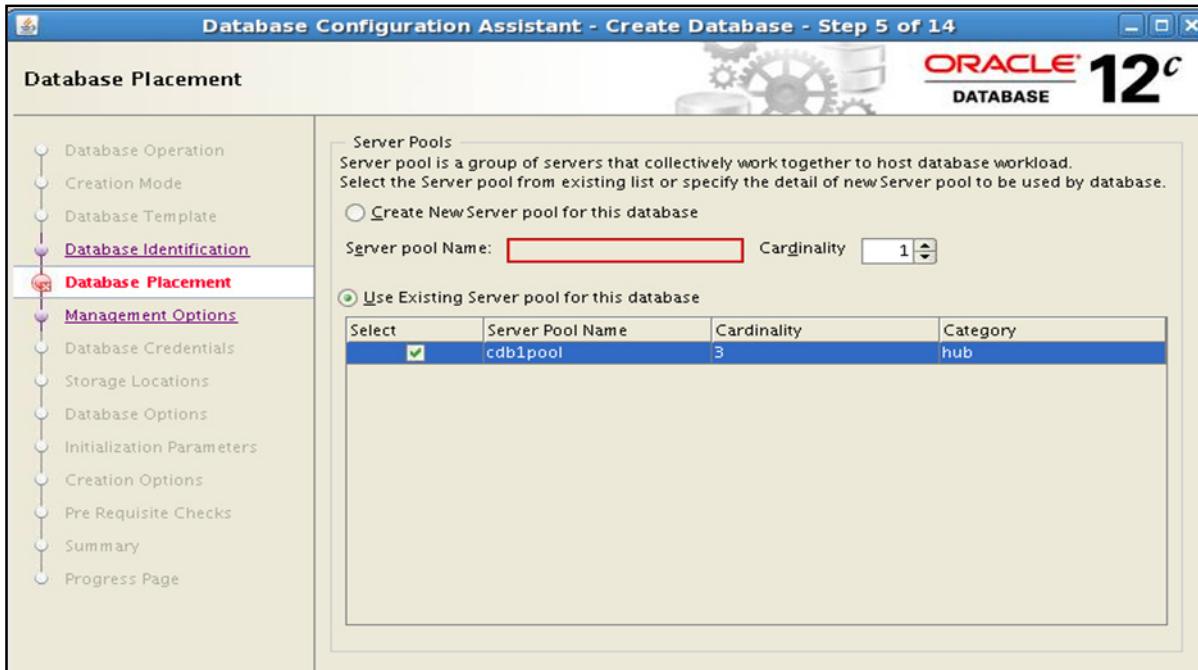
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Then select the “Create As Container Database” check box to create the database as a CDB, else the database is created as a non-CDB.

Provide a pluggable database (PDB) name when using the “Create a database with default configuration” option. If the “Advanced Mode” option is selected as this is the case in the example of the slide, an empty CDB can be created with only the root and seed containers or the CDB can be created with one or several PDBs. If the CDB contains only one PDB, be aware not to 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 and or configured for Enterprise Manager Database Express. The passwords for SYS and SYSTEM users can also be set.

Hosting a RAC CDB in Server Pools



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Because the 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.

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.

```
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 © 2014, 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
```

ORACLE

Copyright © 2014, 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:

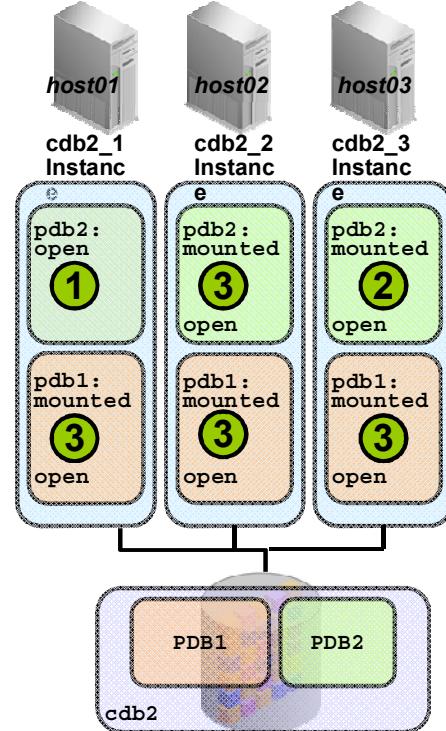
```
SQL> CONNECT sys@host01/cdb2 AS SYSDBA
SQL> ALTER PLUGGABLE DATABASE pdb2 OPEN;
```

- 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;
```



ORACLE

Copyright © 2014, 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 on any of the 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 this is 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 parenthesis to modify the state of the PDB in those instances as this is 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 all 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 © 2014, 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, then all the control files are closed and in the last step the instances are shut down.

A PDB can be closed with the `INSTANCES` clause as this is shown in the examples of 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 the CDB instances
 - PDB (application) services: Limit of 1024 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 © 2014, 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 © 2014, Oracle and/or its affiliates. All rights reserved.

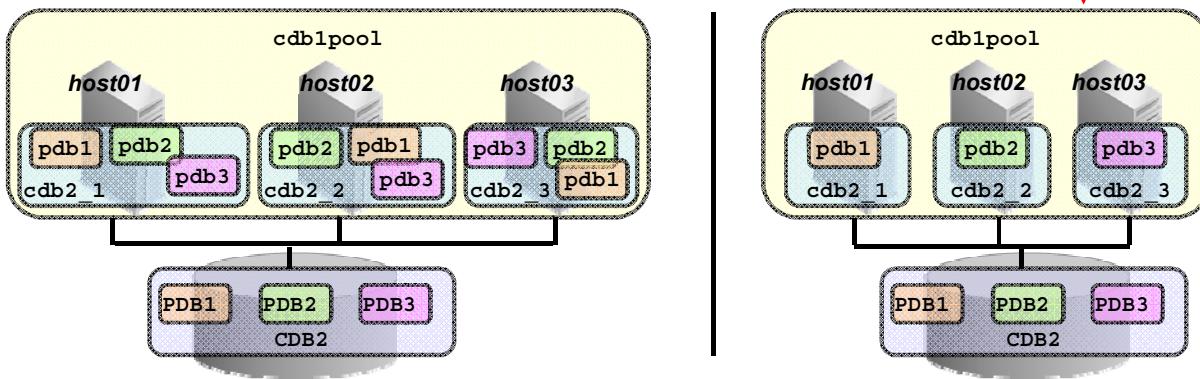
There are two categories of services:

- **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 openn 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 © 2014, 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');
```

```
SQL> CREATE PLUGGABLE DATABASE pdb3 FROM pdb1;
```

- Open the new PDB in the CDB instances

The default PDB service is automatically started in the CDB instances.



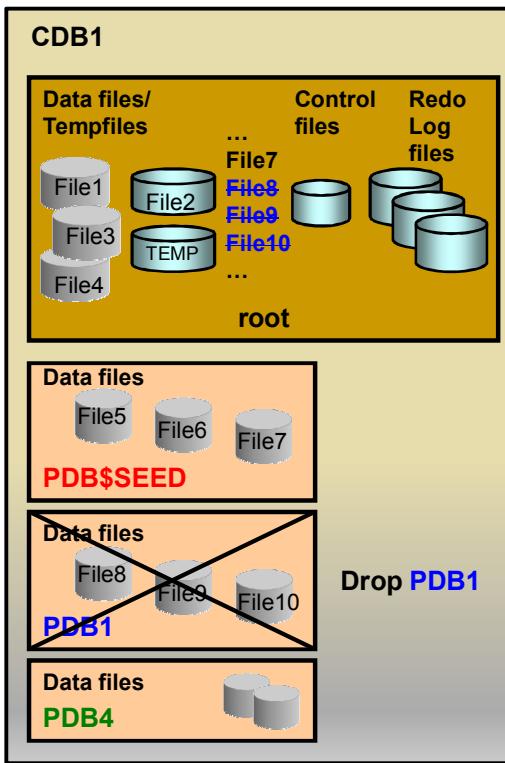
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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



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];
```

ORACLE

Copyright © 2014, 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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: d, e

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



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 13: Overview

This practice covers the following topics:

- Creating a RAC CDB with DBCA
- Cloning a PDB in a RAC CDB
- Affinitizing PDB services to CDB instances
- Dropping a PDB from a RAC CDB



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.