Oracle Database 11*g*: RAC Administration

Volume I • Student Guide

D50311GC11
Edition 1.1
September 2008
D56241

ORACLE

Author

James Womack

Technical Contributors and Reviewers

Christopher Andrews
Troy Anthony
Lothar Auert
Bruce Carter
Michael Cebulla
Carol Colrain
Jonathan Creighton
Joel Goodman
Arturo Gutierrez
Lutz Hartmann
Pete Jones
David Kirby
Roland Knapp
Miroslav Lorenc
Barb Lundhild
Roderick Manalac
Sabiha Miri
Philip Newlan
Roman Niehoff
Erik Peterson
Stefan Pommerenk
Marshall Presser
Rick Pulliam
Srinivas Putrevu
Roy Rossebo
Ira Singer
Linda Smalley
Ranbir Singh
Harald van Breederode
Michael Zoll

Editors

Raj Kumar
Richard Wallis
Nita Pavitran
Amitha Narayan

Graphic Designer

Satish Bettegowda

Publishers

Nita Brozowski
Veena Narasimhan

Contents

7 Services

Introduction

Overview

- This course is designed for anyone interested in implementing a Real Application Clusters (RAC) database.
- Although coverage is general, most of the examples and labs in this course are Linux based.
- Knowledge of and experience with Oracle Database 11*g* architecture are assumed.
- Lecture material is supplemented with hands-on practices.

Overview

The material in this course is designed to provide basic information that is needed to plan or manage Oracle Database 11*g* for Real Application Clusters.

The lessons and practices are designed to build on your knowledge of Oracle used in a nonclustered environment. The material does not cover basic architecture and database management: These topics are addressed by the Oracle Database 11*g* administration courses offered by Oracle University. If your background does not include working with a current release of the Oracle database, you should consider taking such training before attempting this course.

The practices provide an opportunity for you to work with the features of the database that are unique to Real Application Clusters.

Course Objectives

In this course, you:

- Learn the principal concepts of RAC
- Install the RAC components
- Administer database instances in a RAC and ASM environment
- Manage services
- Back up and recover RAC databases
- Monitor and tune performance of a RAC database
- Administer Oracle Clusterware

Course Objectives

This course is designed to give you the necessary information to successfully administer Real Application Clusters and Oracle Clusterware.

You install Oracle Database 11*g* with the Oracle Universal Installer (OUI) and create your database with the Database Configuration Assistant (DBCA). This ensures that your RAC environment has the optimal network configuration, database structure, and parameter settings for the environment that you selected. As a DBA, after installation, your tasks are to administer your RAC environment at three levels:

- Instance administration
- Database administration
- Cluster administration

Throughout this course, you use various tools to administer each level of RAC:

- Oracle Enterprise Manager 10*g* Grid Control to perform administrative tasks whenever feasible
- Task-specific GUIs such as the Database Configuration Assistant (DBCA)
- Command-line tools such as SQL*Plus, Recovery Manager, Server Control (`SRVCTL`), CLUVFY, CRSCTL, and OCRCONFIG

Typical Schedule

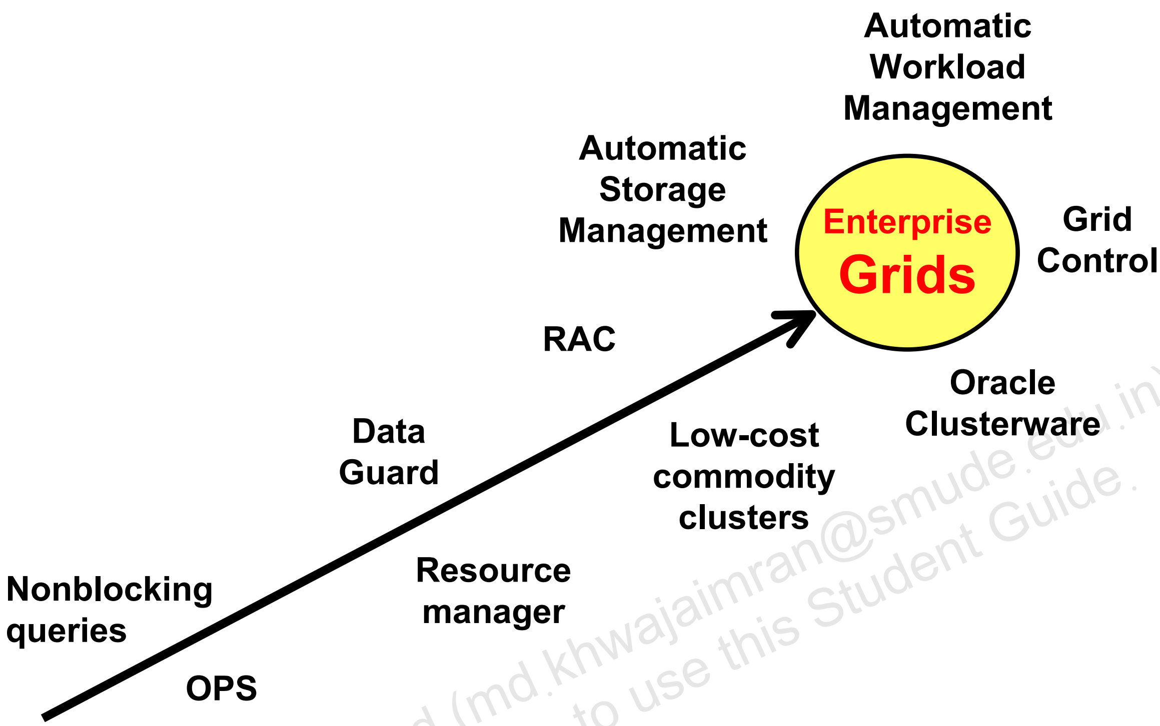
Topics	Lessons	Day
Introduction and installation	I, 1, 2, 3	1
RAC administration and tuning	4, 5, 6	2
	7, 8	3
Advanced topics	9, 10	4
	11, 12	5
Workshop: Cloning		5

Typical Schedule

The lessons in this guide are arranged in the order that you will probably study them in class, and are grouped into the topic areas that are shown in the slide. The individual lessons are ordered so that they lead from more familiar to less familiar areas. The related practices are designed to let you explore increasingly powerful features of a Real Application Clusters database.

In some cases, the goals for the lessons and goals for the practices are not completely compatible. Your instructor may, therefore, choose to teach some material in a different order than found in this guide. However, if your instructor teaches the class in the order in which the lessons are printed in this guide, then the class should run approximately as shown in this schedule.

A History of Innovation



A History of Innovation

Oracle Database 11*g* and the specific new manageability enhancements provided by Oracle RAC 11*g* enable RAC for everyone—all types of applications and enterprise grids (the basis for fourth-generation computing). Enterprise grids are built from large configurations of standardized, commodity-priced components: processors, network, and storage. With Oracle RAC's cache fusion technology, the Oracle database adds to this the highest levels of availability and scalability.

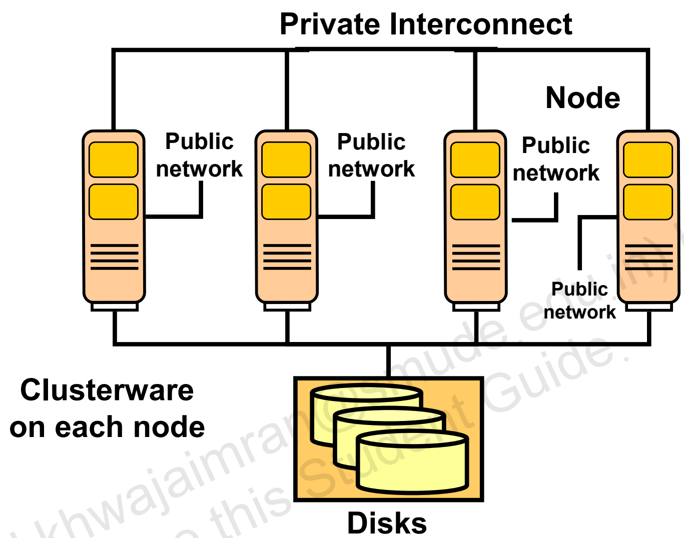
Also, with Oracle RAC 11*g*, it becomes possible to perform dynamic provisioning of nodes, storage, CPUs, and memory to maintain service levels more easily and efficiently.

Enterprise grids are the data centers of the future and enable business to be adaptive, proactive, and agile for the fourth generation.

The next major transition in computing infrastructure is going from the era of big symmetric multiprocessing (SMP) models to the era of grids.

What Is a Cluster?

- Interconnected nodes act as a single server.
- Cluster software hides the structure.
- Disks are available for read and write by all nodes.
- Operating system is the same on each machine.



What Is a Cluster?

A cluster consists of two or more independent, but interconnected, servers. Several hardware vendors have provided cluster capability over the years to meet a variety of needs. Some clusters were intended only to provide high availability by allowing work to be transferred to a secondary node if the active node fails. Others were designed to provide scalability by allowing user connections or work to be distributed across the nodes.

Another common feature of a cluster is that it should appear to an application as if it were a single server. Similarly, management of several servers should be as similar to the management of a single server as possible. The cluster management software provides this transparency.

For the nodes to act as if they were a single server, files must be stored in such a way that they can be found by the specific node that needs them. There are several different cluster topologies that address the data access issue, each dependent on the primary goals of the cluster designer.

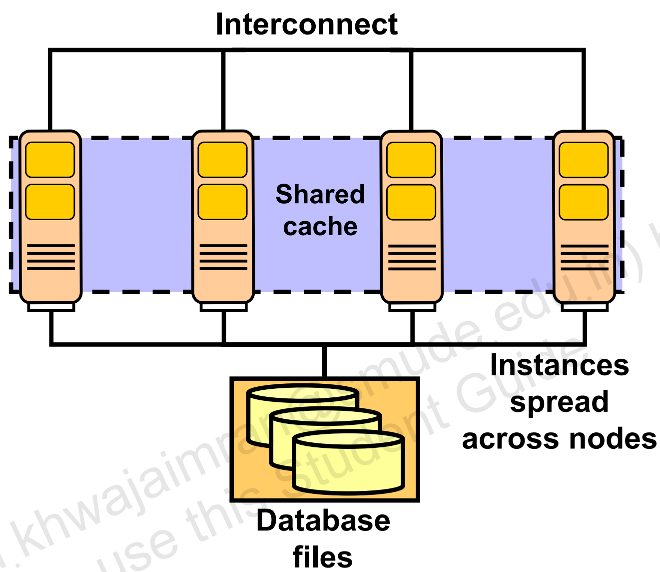
The interconnect is a physical network used as a means of communication between each node of the cluster.

In short, a cluster is a group of independent servers that cooperate as a single system.

Note: The clusters you are going to manipulate in this course all have the same operating system. This is a requirement for RAC clusters.

Oracle Real Application Clusters

- Multiple instances accessing the same database
- One instance per node
- Physical or logical access to each database file
- Software-controlled data access



ORACLE

Oracle Real Application Clusters

Real Application Clusters is a software that enables you to use clustered hardware by running multiple instances against the same database. The database files are stored on disks that are either physically or logically connected to each node, so that every active instance can read from or write to them.

The Real Application Clusters software manages data access, so that changes are coordinated between the instances and each instance sees a consistent image of the database. The cluster interconnect enables instances to pass coordination information and data images between each other.

This architecture enables users and applications to benefit from the processing power of multiple machines. RAC architecture also achieves redundancy in the case of, for example, a system crashing or becoming unavailable; the application can still access the database on any surviving instances.

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

Benefits of Using RAC

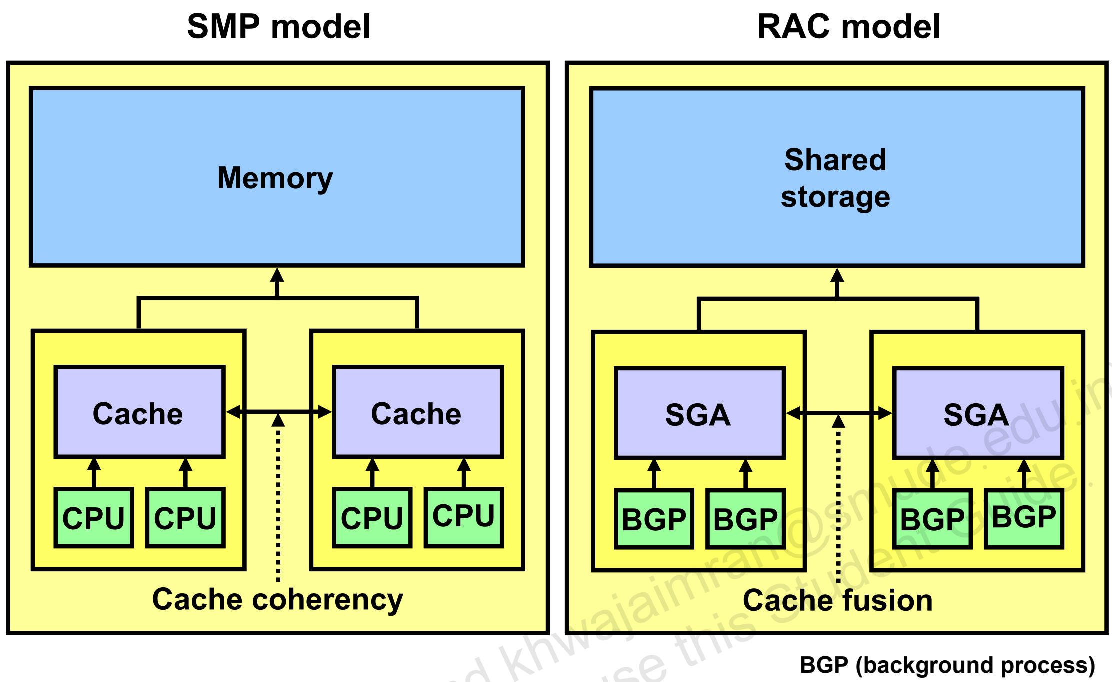
Oracle Real Application Clusters (RAC) enables high utilization of a cluster of standard, low-cost modular servers such as blades.

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, the Oracle server 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



Clusters and Scalability

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.

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.

Note: What is true for SMP is also true for Non-Uniform Memory Architecture (NUMA) architectures. NUMA architectures are the logical next step in scaling from SMP architectures.

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

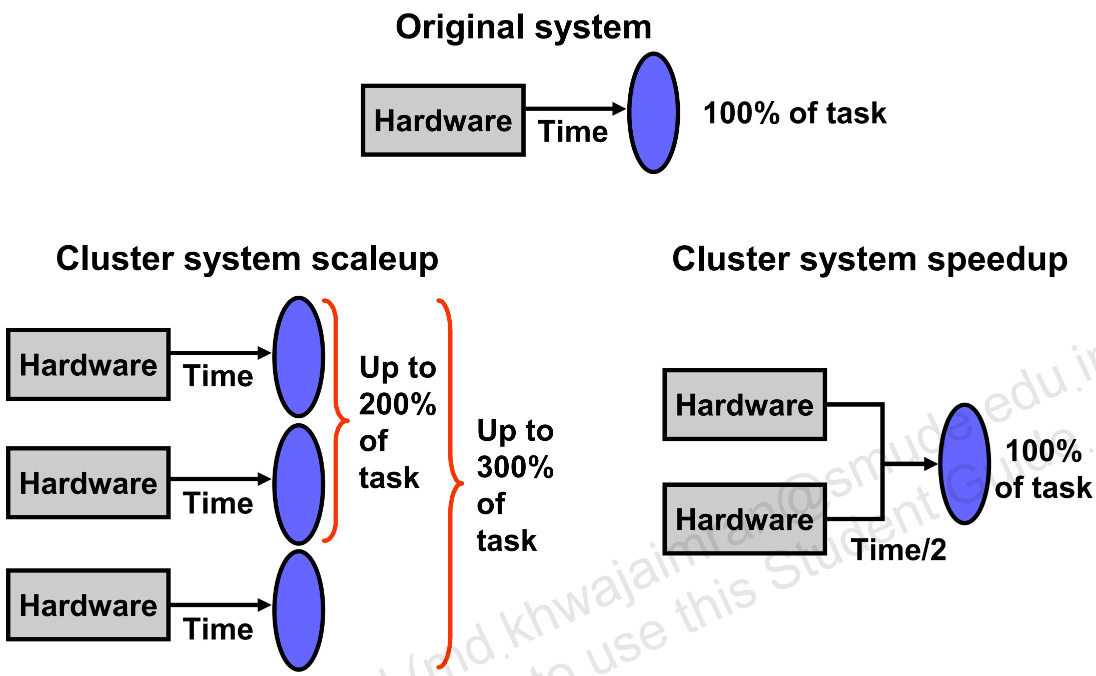
Levels of Scalability

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 system too.

It is important to remember that if any of the areas above 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. A 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



Scaleup and Speedup

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

Scaleup = (volume parallel) / (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:

Speedup = (time original) / (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 were they 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

ORACLE

Speedup/Scaleup and Workloads

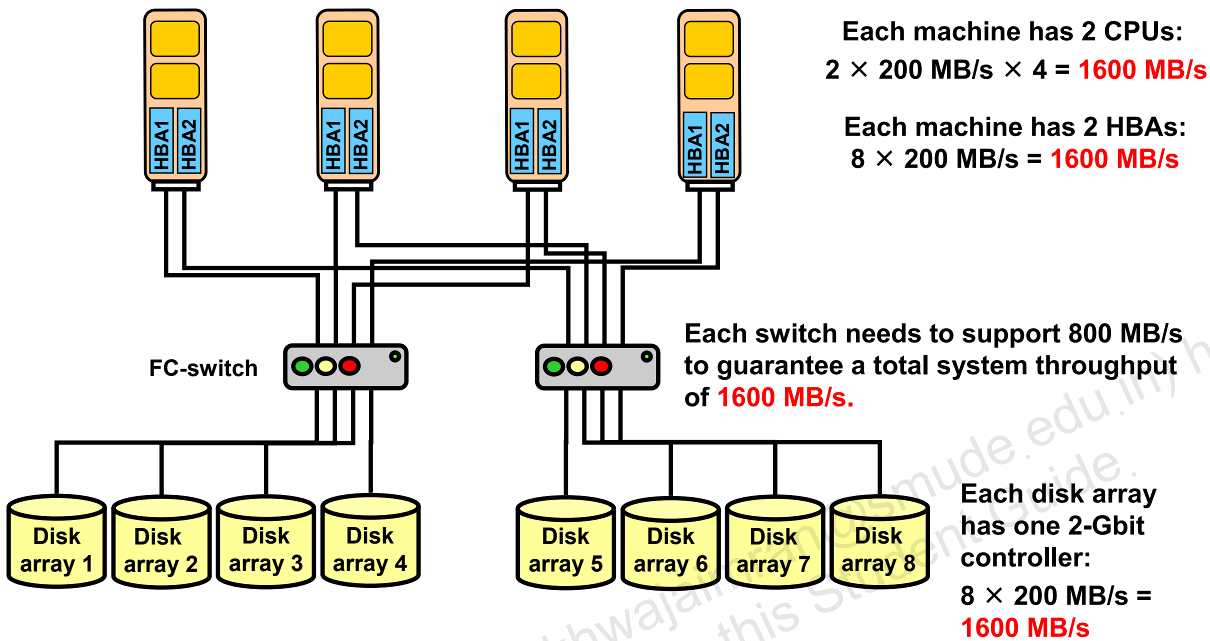
The type of workload determines whether scaleup or speedup capabilities can be achieved using parallel processing.

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



I/O Throughput Balanced: Example

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

The weakest link determines the IO 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 IO 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-GBit HBA, the nodes can read about 8 × 200 MB/s = 1.6 GBytes/s. However, assuming that each disk array has one controller, all 8 arrays can also do 8 × 200 MB/s = 1.6 GBytes/s. Therefore, each of the fibre channel switches also need to deliver at least 2 GBit/s per port, to a total of 800 MB/s throughput. The two switches will then deliver the needed 1.6 GBytes/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.

Performance of Typical Components

Component	Throughput Performance Theory (Bit/s)	Throughput Performance Maximal Byte/s
HBA	½ Gbit/s	100/200 Mbytes/s
16 Port Switch	8 × 2 Gbit/s	1600 Mbytes/s
Fibre Channel	2 Gbit/s	200 Mbytes/s
Disk Controller	2 Gbit/s	200 Mbytes/s
GigE NIC	1 Gbit/s	80 Mbytes/s
Infiniband	10 Gbit/s	890 Mbytes/s
CPU		200–250 MB/s

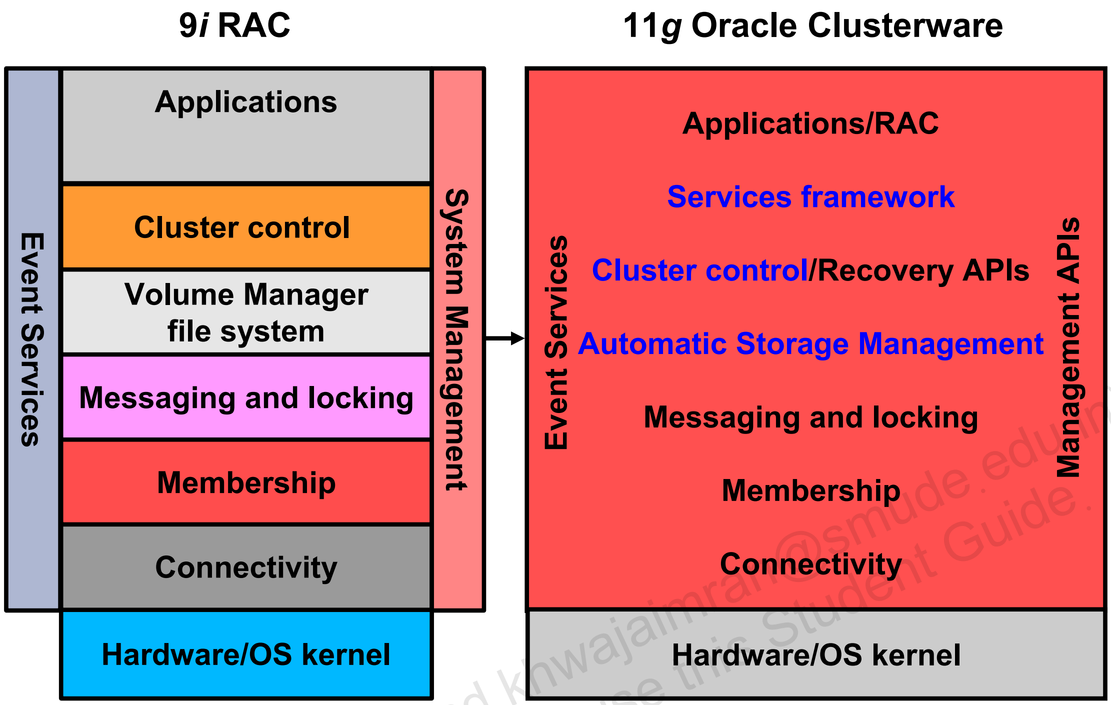
Performance of Typical Components

While discussing, people often confuse bits with bytes. This confusion originates mainly from the fact that hardware vendors tend to describe component's performance in bits/s whereas database vendors and customers describe their performance requirements in bytes/s.

The following is a list of common hardware components with their theoretical performance in bits/second and typical performance in bytes/second:

- HBAs come in 1 or 2 GBit per second with a typical throughput of 100 or 200 MB/s.
- A 16 Port Switch comes with sixteen 2-GBit ports. However, the total throughput is 8 times 2 Gbit, which results in 1600 Mbytes/s.
- Fibre Channel cables have a 2-GBit/s throughput, which translates into 200 MB/s.
- Disk Controllers come in 2-GBit/s throughput, which translates into about 200 MB/s.
- GigE has a typical performance of about 80 MB/s whereas Infiniband delivers about 160 MB/s.

Complete Integrated Clusterware



Complete Integrated Clusterware

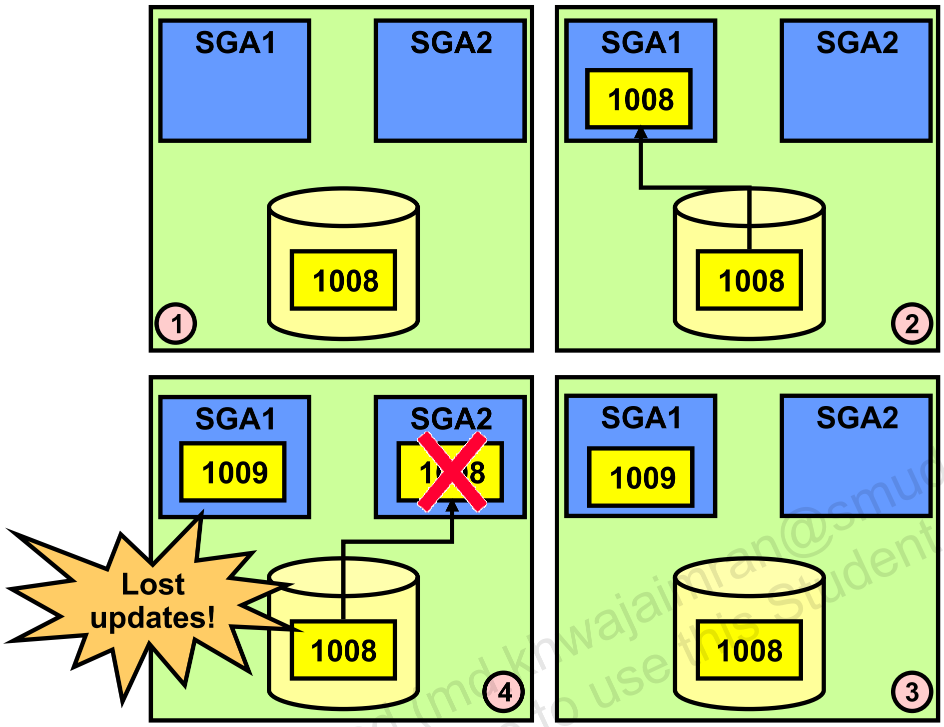
Oracle introduced Real Application Clusters in Oracle9*i* Database. For the first time, you were able to run online transaction processing (OLTP) and decision support system (DSS) applications against a database cluster without having to make expensive code changes or spend large amounts of valuable administrator time partitioning and repartitioning the database to achieve good performance.

Although Oracle9*i* Real Application Clusters did much to ease the task of allowing applications to work in clusters, there are still support challenges and limitations. Among these cluster challenges are complex software environments, support, inconsistent features across platforms, and awkward management interaction across the software stack. Most clustering solutions today were designed with failover in mind. Failover clustering has additional systems standing by in case of a failure. During normal operations, these failover resources may sit idle.

With the release of Oracle Database 11*g*, Oracle provides you with an integrated software solution that addresses cluster management, event management, application management, connection management, storage management, load balancing, and availability. These capabilities are addressed while hiding the complexity through simple-to-use management tools and automation.

Oracle Clusterware provides an integrated clusterware layer that delivers a complete environment for applications.

Necessity of Global Resources



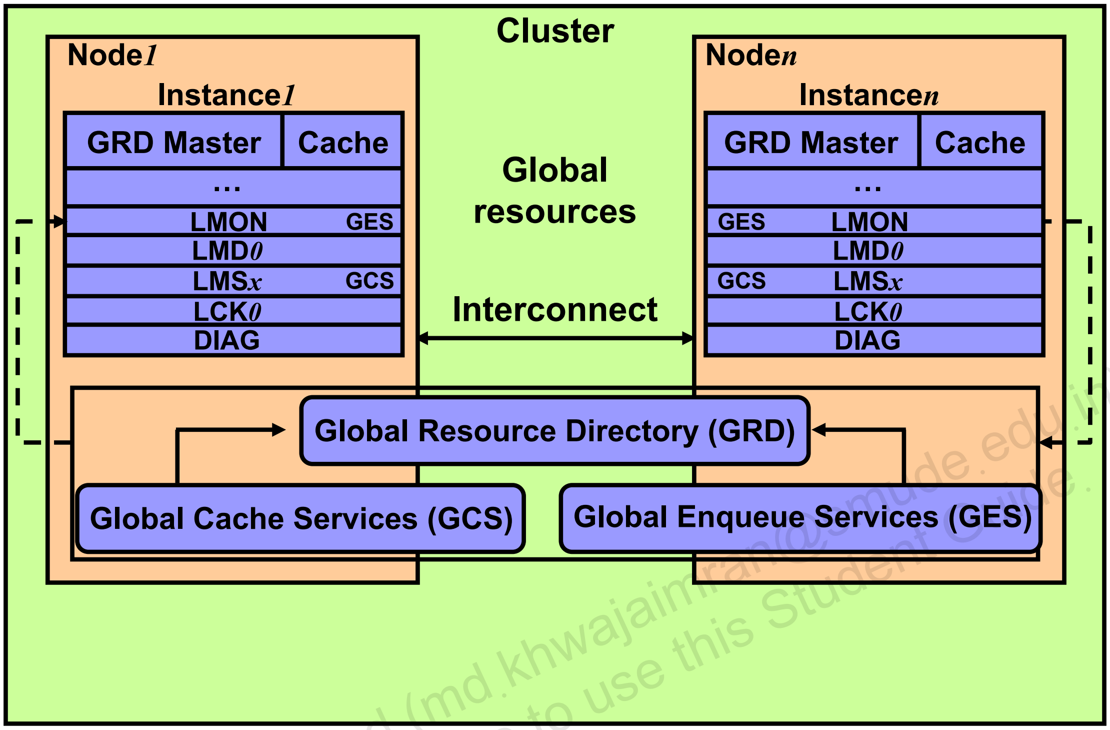
Necessity of Global Resources

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.

Note: The slide shows you what would happen in the absence of cache coordination. RAC prohibits this problem.

Global Resources Coordination



Global Resources Coordination

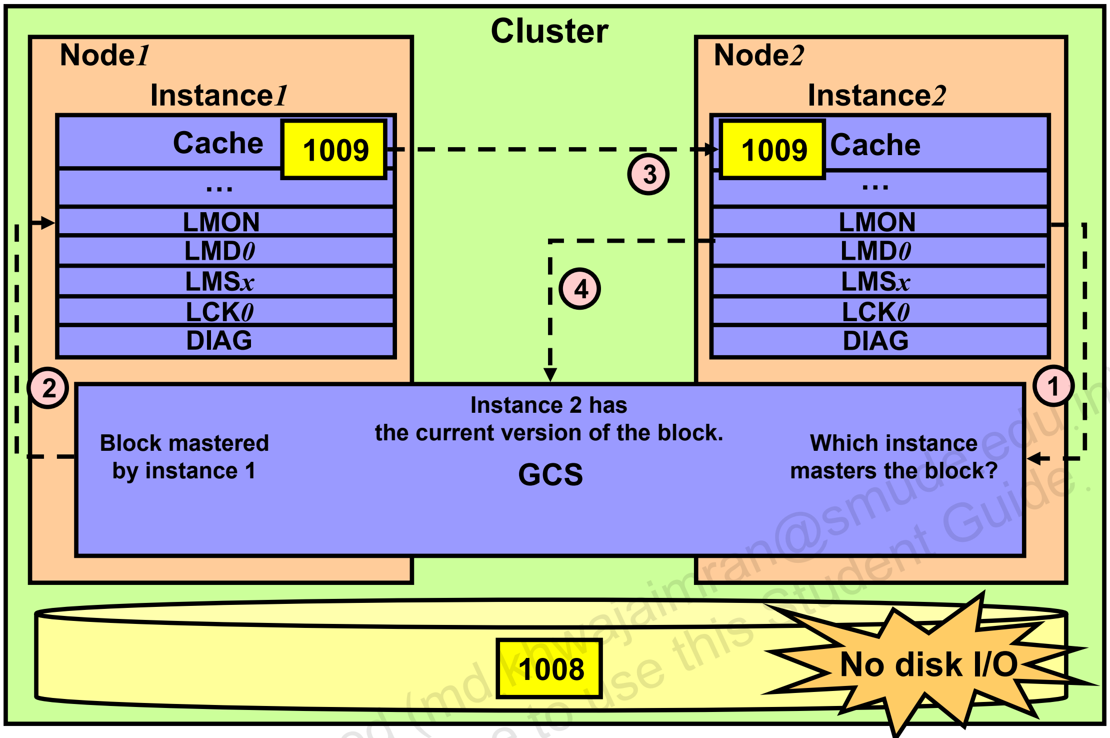
Cluster operations require synchronization among all instances to control shared access to resources. RAC uses the Global Resource Directory (GRD) to record information about how resources are used within a cluster database. The Global Cache Services (GCS) and Global Enqueue Services (GES) manage the information in the GRD.

Each instance maintains a part of the GRD in its System Global Area (SGA). The GCS and GES nominate one instance to manage all information about a particular resource. This instance is called the resource master. Also, each instance knows which instance masters which resource.

Maintaining cache coherency is an important part of a RAC activity. Cache coherency is the technique of keeping multiple copies of a block consistent between different Oracle instances. GCS implements cache coherency by using what is called the Cache Fusion algorithm.

The GES manages all non–Cache Fusion interinstance resource operations and tracks the status of all Oracle enqueuing mechanisms. The primary resources of the GES controls are dictionary cache locks and library cache locks. The GES also performs deadlock detection to all deadlock-sensitive enqueues and resources.

Global Cache Coordination: Example



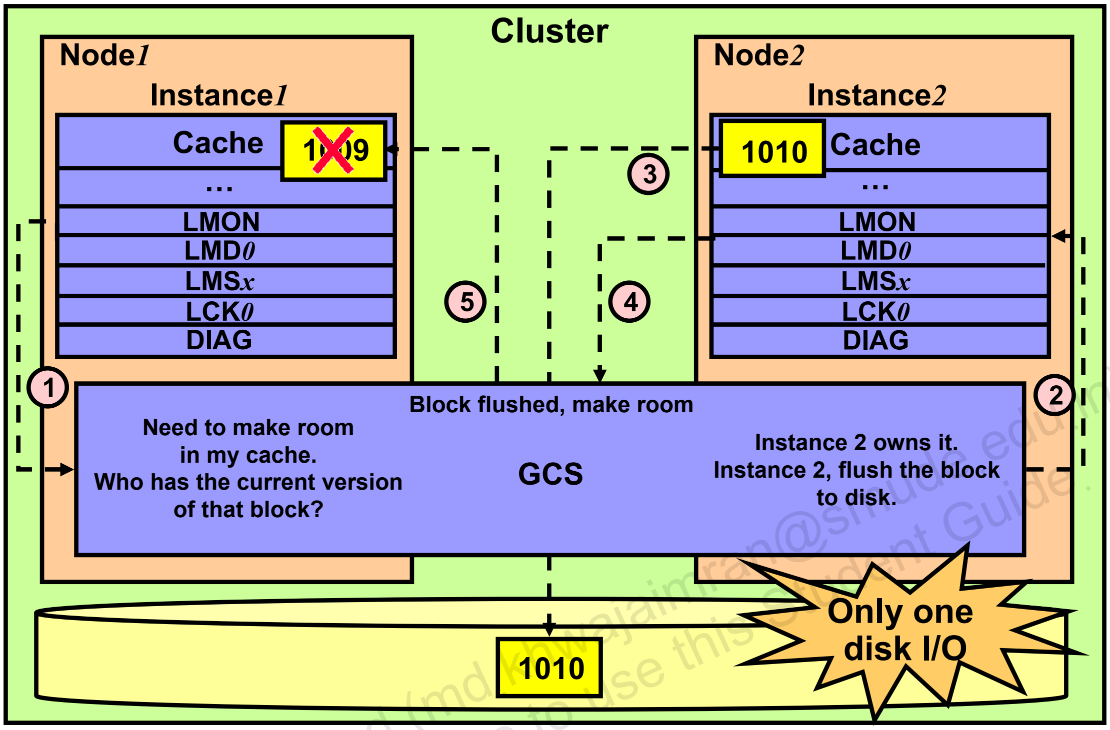
Global Cache Coordination: Example

The scenario described in the slide assumes that the data block has been changed, or dirtied, by the first instance. Furthermore, only one copy of the block exists clusterwide, and the content of the block is represented by its SCN.

1. The second instance attempting to modify the block submits a request to the GCS.
2. The GCS transmits the request to the holder. In this case, the first instance is the holder.
3. The first instance receives the message and sends the block to the second instance. The first instance retains the dirty buffer for recovery purposes. This dirty image of the block is also called a past image of the block. A past image block cannot be modified further.
4. On receipt of the block, the second instance informs the GCS that it holds the block.

Note: The data block is not written to disk before the resource is granted to the second instance.

Write to Disk Coordination: Example



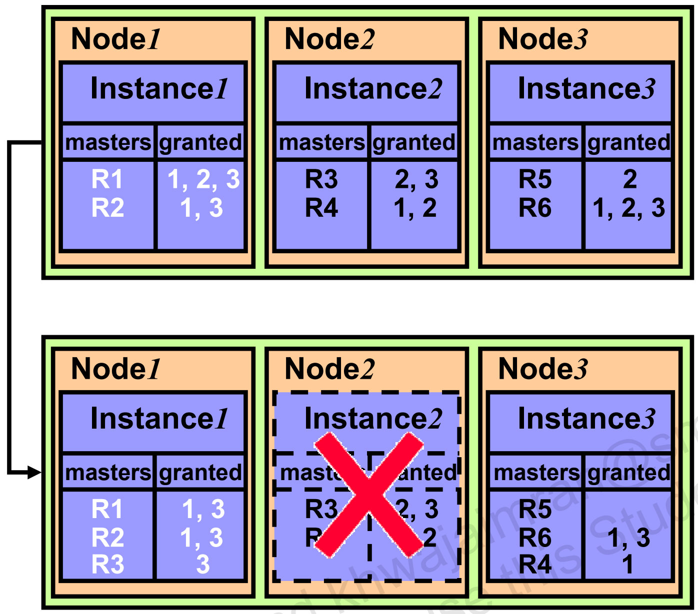
Write to Disk Coordination: Example

The scenario described in the slide illustrates how an instance can perform a checkpoint at any time or replace buffers in the cache as a response to free buffer requests. Because multiple versions of the same data block with different changes can exist in the caches of instances in the cluster, a write protocol managed by the GCS ensures that only the most current version of the data is written to disk. It must also ensure that all previous versions are purged from the other caches. A write request for a data block can originate in any instance that has the current or past image of the block. In this scenario, assume that the first instance holding a past image buffer requests that the Oracle server writes the buffer to disk:

1. The first instance sends a write request to the GCS.
2. The GCS forwards the request to the second instance, which is the holder of the current version of the block.
3. The second instance receives the write request and writes the block to disk.
4. The second instance records the completion of the write operation with the GCS.
5. After receipt of the notification, the GCS orders all past image holders to discard their past images. These past images are no longer needed for recovery.

Note: In this case, only one I/O is performed to write the most current version of the block to disk.

Dynamic Reconfiguration

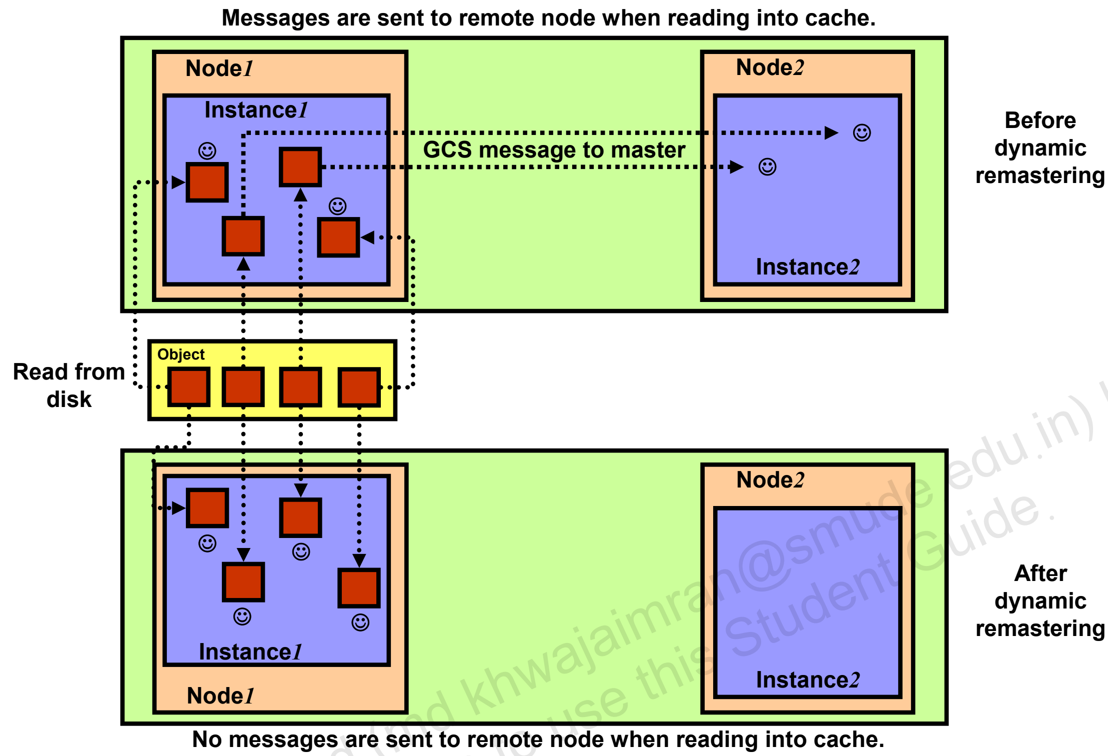


Dynamic Reconfiguration

When one instance departs the cluster, the GRD portion of that instance needs to be redistributed to the surviving nodes. Similarly, when a new instance enters the cluster, the GRD portions of the existing instances must be redistributed to create the GRD portion of the new instance.

Instead of remastering all resources across all nodes, RAC uses an algorithm called lazy remastering to remaster only a minimal number of resources during a reconfiguration. This is illustrated on the slide. For each instance, a subset of the GRD being mastered is shown along with the names of the instances to which the resources are currently granted. When the second instance fails, its resources are remastered on the surviving instances. As the resources are remastered, they are cleared of any reference to the failed instance.

Object Affinity and Dynamic Remastering



ORACLE

Object Affinity and Dynamic Remastering

In addition to dynamic resource reconfiguration, the GCS, which is tightly integrated with the buffer cache, enables the database to automatically adapt and migrate resources in the GRD. This is called dynamic remastering. The basic idea is to master a buffer cache resource on the instance where it is mostly accessed. In order to determine whether dynamic remastering is necessary, the GCS essentially keeps track of the number of GCS requests on a per-instance and per-object basis. This means that if an instance, compared to another, is heavily accessing blocks from the same object, the GCS can take the decision to dynamically migrate all of that object's resources to the instance that is accessing the object most.

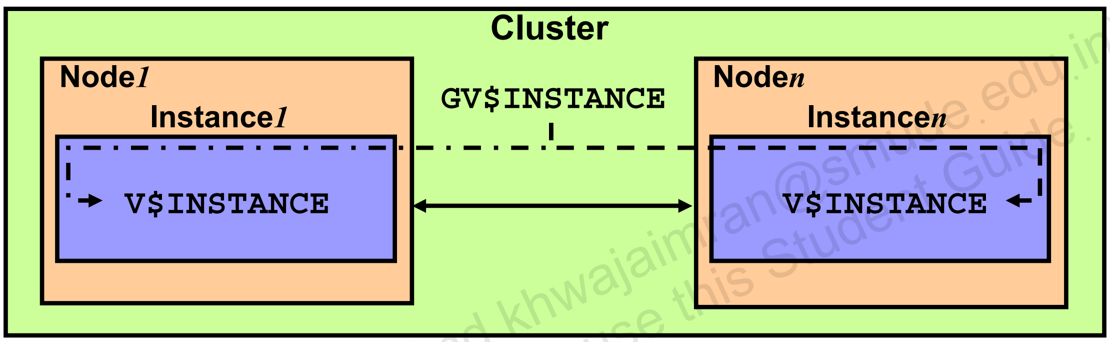
The upper part of the graphic shows you the situation where the same object has master resources spread over different instances. In that case, each time an instance needs to read a block from that object whose master is on the other instance, the reading instance must send a message to the resource's master to ask permission to use the block.

The lower part of the graphic shows you the situation after dynamic remastering occurred. In this case, blocks from the object have affinity to the reading instance which no longer needs to send GCS messages across the interconnect to ask for access permissions.

Note: The system automatically moves mastership of undo segment objects to the instance that owns the undo segments.

Global Dynamic Performance Views

- Retrieve information about all started instances
- Have one global view for each local view
- Use one parallel slave on each instance



Global Dynamic Performance Views

Global dynamic performance views retrieve information about all started instances accessing one RAC database. In contrast, standard dynamic performance views retrieve information about the local instance only.

For each of the `V$` views available, there is a corresponding `GV$` view except for a few exceptions. In addition to the `V$` information, each `GV$` view possesses an additional column named `INST_ID`. The `INST_ID` column displays the instance number from which the associated `V$` view information is obtained. You can query `GV$` views from any started instance.

`GV$` views use a special form of parallel execution. The parallel execution coordinator runs on the instance that the client connects to, and one slave is allocated in each instance to query the underlying `V$` view for that instance.

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%';
```


Additional Memory Requirement for RAC

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.

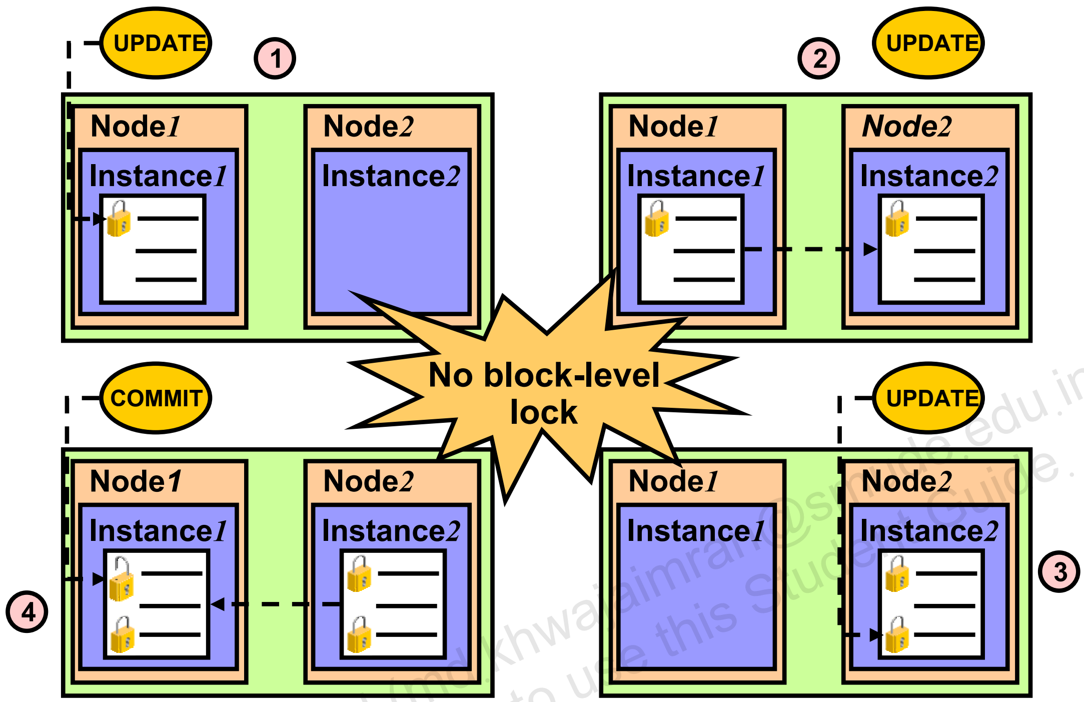
Therefore, when migrating your Oracle database from single instance to RAC, keeping the workload requirements per instance the same as with the single-instance case, about 10% more buffer cache and 15% more shared pool are needed to run on RAC. These values are heuristics, based on RAC sizing experience.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

Efficient Internode Row-Level Locking



Efficient Internode Row-Level Locking

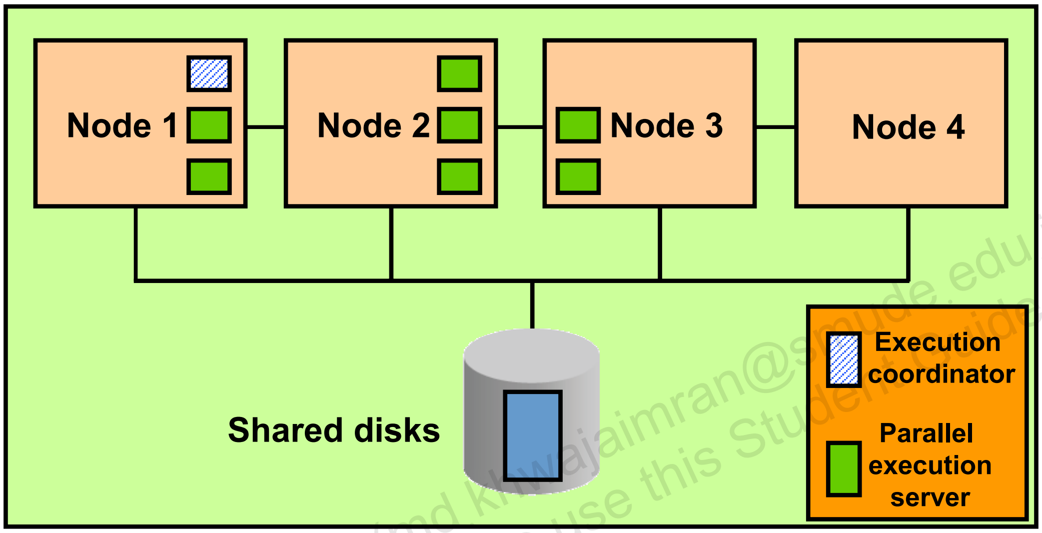
Oracle supports efficient row-level locks. These row-level locks are created when data manipulation language (DML) operations, such as `UPDATE`, are executed by an application. These locks are held until the application commits or rolls back the transaction. Any other application process will be blocked if it requests a lock on the same row.

Cache Fusion block transfers operate independently of these user-visible row-level locks. The transfer of data blocks by the GCS is a low-level process that can occur without waiting for row-level locks to be released. Blocks may be transferred from one instance to another while row-level locks are held.

GCS provides access to data blocks allowing multiple transactions to proceed in parallel.

Parallel Execution with RAC

Execution slaves have node affinity with the execution coordinator but will expand if needed.



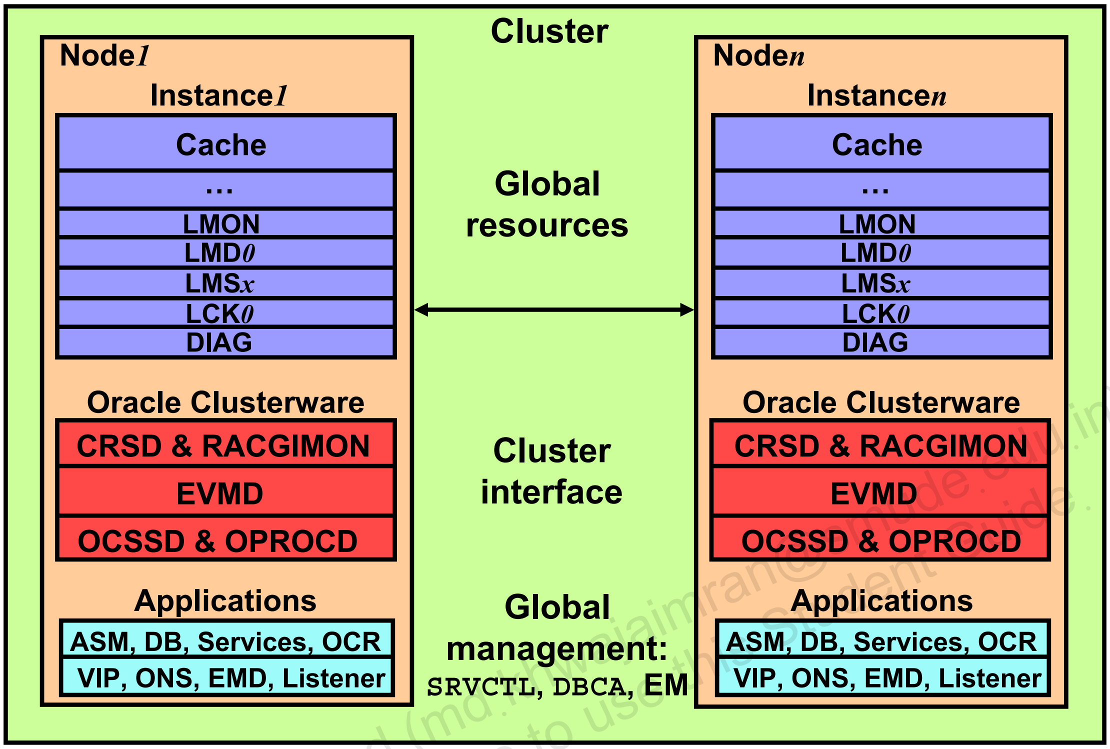
Parallel Execution with RAC

Oracle's cost-based optimizer incorporates parallel execution considerations as a fundamental component in arriving at optimal execution plans.

In a RAC environment, intelligent decisions are made with regard to intranode and internode parallelism. For example, if a particular query requires six query processes to complete the work and six parallel execution slaves are idle on the local node (the node that the user connected to), then the query is processed by using only local resources. This demonstrates efficient intranode parallelism and eliminates the query coordination overhead across multiple nodes. However, if there are only two parallel execution servers available on the local node, then those two and four of another node are used to process the query. In this manner, both internode and intranode parallelism are used to speed up query operations.

In real-world decision support applications, queries are not perfectly partitioned across the various query servers. Therefore, some parallel execution servers complete their processing and become idle sooner than others. The Oracle parallel execution technology dynamically detects idle processes and assigns work to these idle processes from the queue tables of the overloaded processes. In this way, the Oracle server efficiently redistributes the query workload across all processes. Real Application Clusters further extends these efficiencies to clusters by enabling the redistribution of work across all the parallel execution slaves of a cluster.

RAC Software Principles



RAC Software Principles

You may see a few additional background processes associated with a RAC instance than you would with a single-instance database. These processes are primarily used to maintain database coherency among each instance. They manage what is called the global resources:

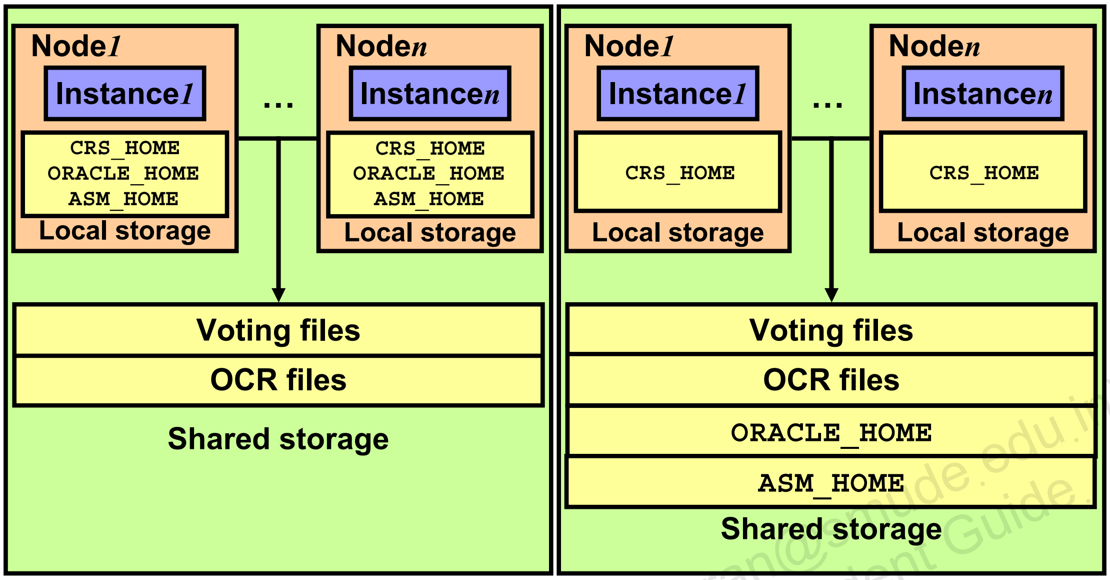
- **`LMON`:** Global Enqueue Service Monitor
- **`LMD`*0*:** Global Enqueue Service Daemon
- **`LMS`*x*:** Global Cache Service Processes, where *x* can range from *0* to *j*
- **`LCK`*0*:** Lock process
- **`DIAG`:** Diagnosability process

At the cluster level, you find the main processes of Oracle Clusterware. They provide a standard cluster interface on all platforms and perform high-availability operations. You find these processes on each node of the cluster:

- **`CRSD`** and **`RACGIMON`:** Are engines for high-availability operations
- **`OCSSD`:** Provides access to node membership and group services
- **`EVMD`:** Scans callout directory and invokes callouts in reactions to detected events
- **`OPROCD`:** Is a process monitor for the cluster

There are also several tools that are used to manage the various resources available on the cluster at a global level. These resources are the Automatic Storage Management (ASM) instances, the RAC databases, the services, and node applications. Some of the tools that you use throughout this course are Server Control (`SRVCTL`), DBCA, and Enterprise Manager.

RAC Software Storage Principles



Permits rolling patch upgrades

Software is not a single point of failure.

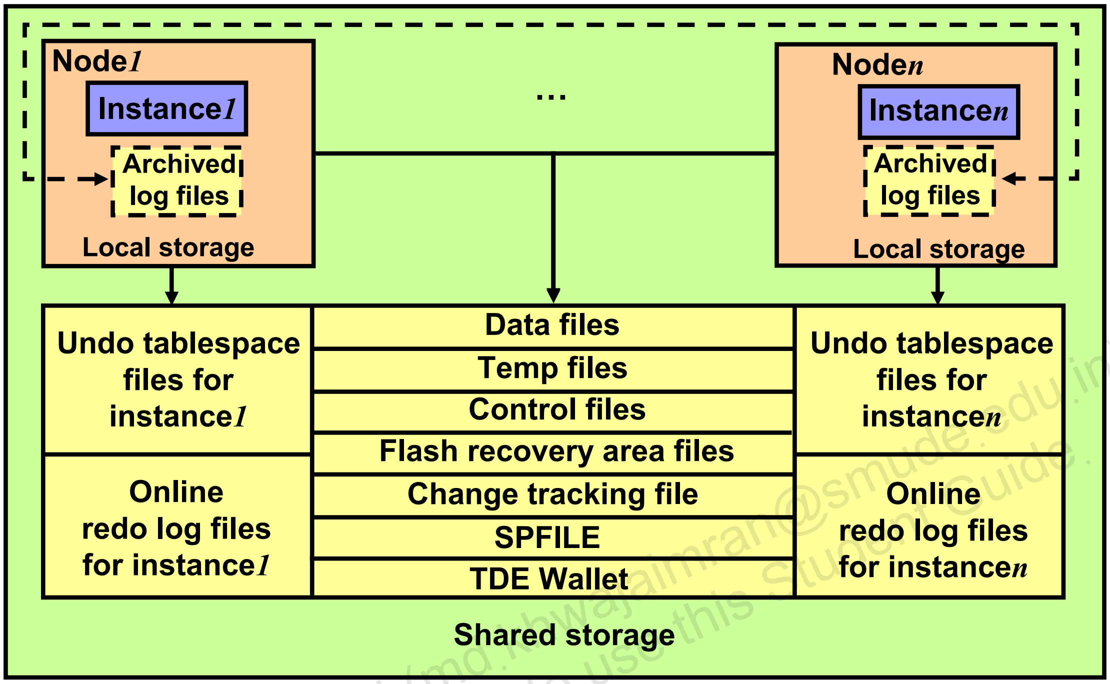
RAC Software Storage Principles

The Oracle Database 11*g* Real Application Clusters installation is a two-phase installation. In the first phase, you install Oracle Clusterware. In the second phase, you install the Oracle database software with RAC components and create a cluster database. The Oracle Home that you use for Oracle Clusterware must be different from the one that is used for the RAC software. Although it is possible to install the RAC software on your cluster shared storage when using certain cluster file systems, software is usually installed on a regular file system that is local to each node. This permits rolling patch upgrades and eliminates the software as a single point of failure. In addition, at least two files must be stored on your shared storage:

- The monitoring information across the cluster. Its size is set to around 20 MB.
- The Oracle Cluster Registry (OCR) file is also a key component of Oracle Clusterware. It maintains information about the high-availability components in your cluster, such as the voting file is essentially used by the Cluster Synchronization Services daemon for node-cluster node list, cluster database instance to node mapping, and CRS application resource profiles (such as services, Virtual Interconnect Protocol addresses, and so on). This file is maintained by administrative tools such as `SRVCTL`. Its size is around 100 MB.

The voting and OCR files cannot be stored in ASM because they must be accessible before starting any Oracle instance. OCR and voting files can be on redundant, reliable storage such as RAID, or mirrored on different disks. The recommended best practice location for those files is raw devices on the fastest I/O disks.

RAC Database Storage Principles



RAC Database Storage Principles

The primary difference between RAC storage and storage for single-instance Oracle databases is that all data files in RAC must reside on shared devices (either raw devices or cluster file systems) in order to be shared by all the instances that access the same database. You must also create at least two redo log groups for each instance, and all the redo log groups must also be stored on shared devices for instance or crash recovery purposes. Each instance's online redo log groups are called an instance's thread of online redo.

In addition, you must create one shared undo tablespace for each instance for using the recommended automatic undo management feature. Each instance's undo tablespace must be shared by all other instances for recovery purposes.

Archive logs cannot be placed on raw devices because their names are automatically generated and are different for each archive log. That is why they must be stored on a file system. If you use a cluster file system (CFS), it enables you to access these archive files from any node at any time. If you do not use a CFS, you are always forced to make the archives available to the other cluster members at the time of recovery—for example, by using a network file system (NFS) across nodes. If you are using the recommended flash recovery area feature, then it must be stored in a shared directory so that all instances can access it.

Note: A shared directory can be an ASM disk group, or a cluster file system.

RAC and Shared Storage Technologies

- **Storage is a critical component of grids:**
  - **Sharing storage is fundamental.**
  - **New technology trends**
- **Supported shared storage for Oracle grids:**
  - **Network Attached Storage**
  - **Storage Area Network**
- **Supported file storage for Oracle grids:**
  - **Raw volumes**
  - **Cluster file system**
  - **ASM**

RAC and Shared Storage Technologies

Storage is a critical component of any grid solution. Traditionally, storage has been directly attached to each individual server (Directly Attached Storage, or DAS). Over the past few years, more flexible storage, which is accessible over storage area networks or regular Ethernet networks, has become popular. These new storage options enable multiple servers to access the same set of disks, simplifying the provisioning of storage in any distributed environment.

Storage Area Network (SAN) represents the evolution of data storage technology to this point. Traditionally, on client server systems, data was stored on devices either inside or directly attached to the server. Next in the evolutionary scale came Network Attached Storage (NAS) that took the storage devices away from the server and connected them directly to the network. SANs take the principle a step further by allowing storage devices to exist on their own separate networks and communicate directly with each other over very fast media. Users can gain access to these storage devices through server systems that are connected to both the local area network (LAN) and SAN.

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

RAC and Shared Storage Technologies (continued)

Thus, three major approaches exist for providing the shared storage needed by RAC:

- **Raw volumes:** These are directly attached raw devices that require storage that operates in block mode such as fiber channel or *i*SCSI.
- **Cluster file system:** One or more cluster file systems can be used to hold all RAC files. Cluster file systems require block mode storage such as fiber channel or *i*SCSI.
- **Automatic Storage Management (ASM):** It is a portable, dedicated, and optimized cluster file system for Oracle database files.

Note: *i*SCSI is important to SAN technology because it enables a SAN to be deployed in a local area network (LAN), wide area network (WAN), or Metropolitan Area Network (MAN).

Oracle Cluster File System

- Is a shared disk cluster file system for Linux and Windows
- Improves management of data for RAC by eliminating the need to manage raw devices
- Provides open solution on the operating system side
- Can be downloaded from OTN:
  - http://oss.oracle.com/projects/ocfs2/ (Linux)

Oracle Cluster File System

Oracle Cluster File System (OCFS) is a shared file system designed specifically for Oracle Real Application Clusters. OCFS eliminates the requirement that Oracle database files be linked to logical drives and enables all nodes to share a single Oracle Home (on Windows 2000 and 2003 only), instead of requiring each node to have its own local copy. OCFS volumes can span one shared disk or multiple shared disks for redundancy and performance enhancements. The following is a list of files that can be placed on Oracle Cluster File System version 1:

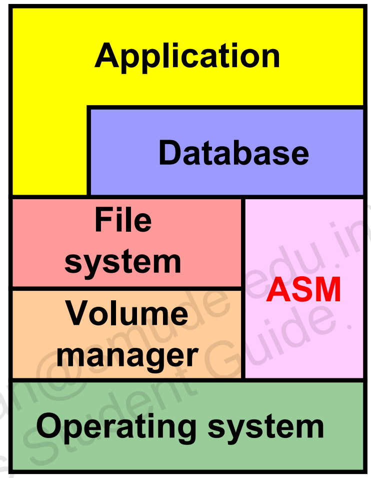
- Oracle software installation: Currently, this configuration is supported only on Windows 2000 and 2003. Oracle Cluster File System 2 1.2.1 provides support for Oracle Home on Linux as well.
- Oracle files (control files, data files, redo logs, bfiles, and so on)
- Shared configuration files (spfile)
- Files created by Oracle during run time
- Voting and OCR files

Oracle Cluster File System is free for developers and customers. The source code is provided under the General Public License (GPL) on Linux. It can be downloaded from the Oracle Technology Network Web site.

Note: From OTN, you can specifically download OCFS for Linux. However, when you download the database software for Windows, OCFS is already included.

Automatic Storage Management

- Provides the first portable and high-performance database file system
- Manages Oracle database files
- Contains data spread across disks to balance load
- Provides integrated mirroring across disks
- Solves many storage management challenges



Automatic Storage Management

Automatic Storage Management (ASM) provides a vertical integration of the file system and the volume manager that is specifically built for Oracle database files. ASM can provide management for single SMP machines or across multiple nodes of a cluster for Oracle Real Application Clusters support.

ASM distributes I/O load across all available resources to optimize performance while removing the need for manual I/O tuning. It helps DBAs manage a dynamic database environment by allowing them to increase the database size without having to shut down the database to adjust the storage allocation.

ASM can maintain redundant copies of data to provide fault tolerance, or it can be built on top of vendor-supplied, reliable storage mechanisms. Data management is done by selecting the desired reliability and performance characteristics for classes of data rather than with human interaction on a per-file basis.

The ASM capabilities save DBAs time by automating manual storage and thereby increasing their ability to manage larger databases (and more of them) with increased efficiency.

Note: ASM is the strategic and stated direction as to where Oracle database files should be stored. However, OCFS will continue to be developed and supported for those who are using it.

CFS or Raw?

- **Using CFS:**
  - Simpler management
  - Use of OMF with RAC
  - Single Oracle software installation
  - Autoextend
- **Using raw:**
  - Performance
  - Use when CFS not available
  - Cannot be used for archivelog files
  - ASM eases work

CFS or Raw?

As already explained, you can either use a cluster file system or place files on raw devices.

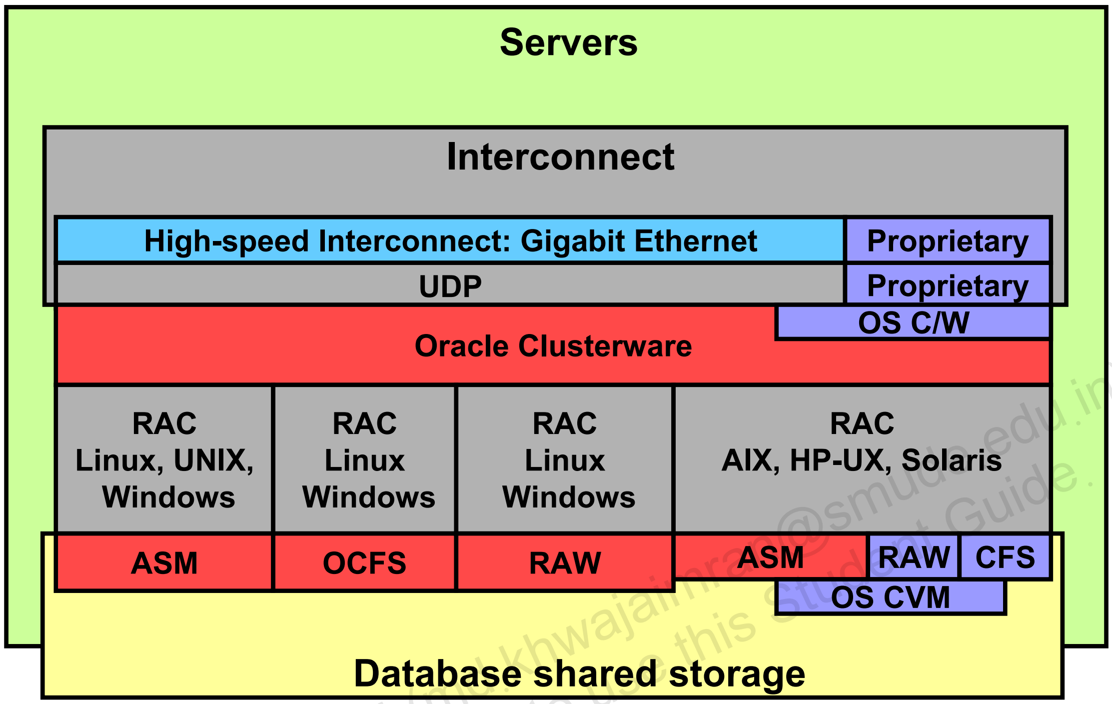
Cluster file systems provide the following advantages:

- Greatly simplified installation and administration of RAC
- Use of Oracle Managed Files with RAC
- Single Oracle software installation
- Autoextend enabled on Oracle data files
- Uniform accessibility to archive logs in case of physical node failure

Raw devices implications:

- Raw devices are always used when CFS is not available or not supported by Oracle.
- Raw devices offer best performance without any intermediate layer between Oracle and the disk.
- Autoextend fails on raw devices if the space is exhausted.
- ASM, Logical Storage Managers, or Logical Volume Managers can ease the work with raw devices. Also, they can enable you to add space to a raw device online, or you may be able to create raw device names that make the usage of this device clear to the system administrators.

Typical Cluster Stack with RAC



Typical Cluster Stack with RAC

Each node in a cluster requires a supported interconnect software protocol to support interinstance communication, and Transmission Control Protocol/Internet Protocol (TCP/IP) to support Oracle Clusterware polling.

All UNIX and Linux platforms use User Datagram Protocol (UDP) on Gigabit Ethernet (GbE) as one of the primary protocols and interconnect for RAC interinstance IPC communication. Other vendor-specific interconnect protocols include Remote Shared Memory for SCI and SunFire Link interconnects, and Hyper Messaging Protocol for Hyperfabric interconnects. In any case, your interconnect must be certified by Oracle for your platform.

Using Oracle Clusterware, you can reduce installation and support complications. However, vendor clusterware may be needed if customers use non-Ethernet interconnect or if you have deployed clusterware-dependent applications on the same cluster where you deploy RAC.

Similar to the interconnect, the shared storage solution you choose must be certified by Oracle for your platform. If a cluster file system (CFS) is available on the target platform, then both the database area and flash recovery area can be created on either CFS or ASM. If a CFS is unavailable on the target platform, then the database area can be created either on ASM or on raw devices (with the required volume manager), and the flash recovery area must be created on the ASM.

Note: It is strongly recommended that you use UDP over GbE.

RAC Certification Matrix

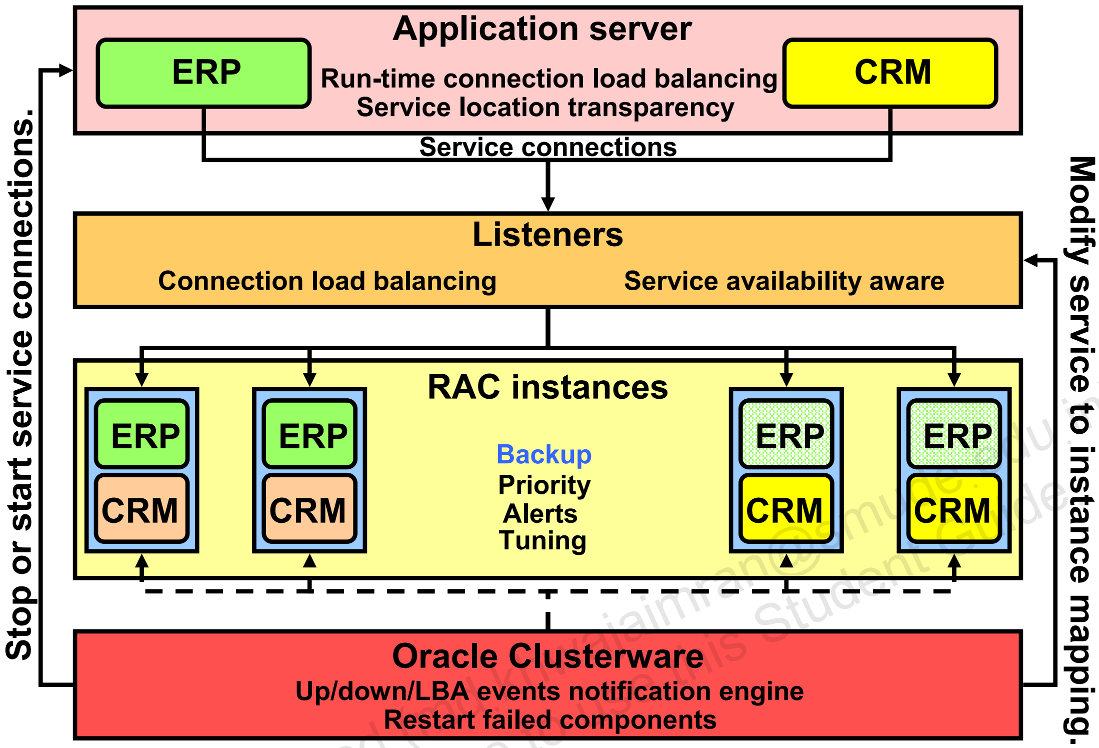
1. Connect and log in to http://metalink.oracle.com.
2. Click the Certify tab on the menu frame.
3. Click the “View Certifications by Product” link.
4. Select Real Application Clusters. Then click Submit.
5. Select the correct platform and click Submit.

Technology Category	Technology	Exclusions/Limitations/Notes
Storage	**Fibre Channel** o Fibre Channel Switched Fabric (FC-SW) that adhere to the ANSI Fibre Channel FC-FS standards o Fibre Channel Arbitrated Loop (FC-AL) that adhere to the ANSI Fibre Channel FC-AL standards **SCSI** o Direct attach for two nodes	**Fibre Channel** o N/A **SCSI** o Greater than two nodes requires SCSI-3 Persistent Group Reservations
Network Interconnect	**Network Interconnect** o 100Mbps and Gigabit NICs and switches using the UDP protocol o Several proprietary interconnects (see vendor entries for specific information)	**Crossover Cable:** o Crossover Cable is not supported as an Interconnect with 9iRAC/10gRAC on any platform.

RAC Certification Matrix

Real Application Clusters Certification Matrix is designed to address any certification inquiries. Use this matrix to answer any certification questions that are related to RAC. To navigate to Real Application Clusters Certification Matrix, perform the steps shown in the slide.

RAC and Services



RAC and Services

Services are a logical abstraction for managing workloads. Services divide the universe of work executing in the Oracle database into mutually disjoint classes. Each service represents a workload with common attributes, service-level thresholds, and priorities.

Services are built into the Oracle database providing a single-system image for workloads, prioritization for workloads, performance measures for real transactions, and alerts and actions when performance goals are violated. These attributes are handled by each instance in the cluster by using metrics, alerts, scheduler job classes, and the resource manager.

With RAC, services facilitate load balancing, allow for end-to-end lights-out recovery, and provide full location transparency.

A service can span one or more instances of an Oracle database in a cluster, and a single instance can support multiple services. The number of instances offering the service is transparent to the application. Services enable the automatic recovery of work. Following outages, the service is recovered automatically at the surviving instances. When instances are later repaired, services that are not running are restored automatically by Oracle Clusterware. Immediately the service changes state, up, down, or too busy; a notification is available for applications using the service to trigger immediate recovery and load-balancing actions. Listeners are also aware of services availability, and are responsible for distributing the workload on surviving instances when new connections are made. This architecture forms an end-to-end continuous service for applications.

Available Demonstrations

- RAC scalability and transaction throughput
- RAC speedup and parallel queries
- Use TAF with `SELECT` statements

http://www.oracle.com/technology/obe/demos/admin/demos.html

ORACLE

Available Demonstrations

To illustrate the major concepts that were briefly introduced in this lesson, online demonstrations are available at http://www.oracle.com/technology/obe/demos/admin/demos.html.

1

Oracle Clusterware Installation and Configuration

Objectives

After completing this lesson, you should be able to:

- Describe the installation of Oracle RAC 11*g*
- Perform RAC preinstallation tasks
- Perform cluster setup tasks
- Install Oracle Clusterware

Oracle RAC 11*g* Installation

Oracle RAC 11*g* incorporates a two-phase installation process:

- **Phase one installs Oracle Clusterware.**
- **Phase two installs the Oracle Database 11*g* software with RAC.**

ORACLE

Oracle RAC 11*g* Installation

The installation of Oracle Database 11*g* requires that you perform a two-phase process in which you run the Oracle Universal Installer (OUI) twice. The first phase installs Oracle Clusterware. Oracle Clusterware provides high-availability components and can also interact with vendor clusterware, if present, to coordinate cluster membership information.

The second phase installs the Oracle Database 11*g* software with RAC. The installation also enables you to configure services for your RAC environment. If you have a previous Oracle cluster database version, OUI activates the Database Upgrade Assistant (DBUA) to automatically upgrade your preexisting cluster database. The Oracle Database 11*g* installation process provides a single-system image, ease of use, and accuracy for RAC installations and patches.

Installation Utilities: New Features

There are new features and changed screens for OUI, Database Configuration Assistant (DBCA), Network Configuration Assistant (NetCA), and DBUA. The enhancements include:

DBCA

- Provides a command-line feature (`deleteASM`) that removes ASM software
- Provides the option to switch from a database configured for Oracle Enterprise Manager Database Control to Oracle Enterprise Manager Grid Control

Installation Utilities: New Features (continued)

DBUA

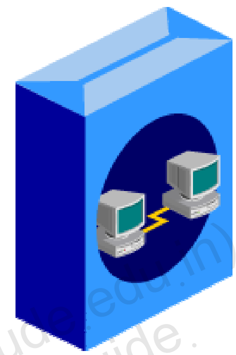
- On request for a backup during the upgrade, DBUA creates a restore script to restore the database.
- Includes an improved pre-upgrade script to provide space estimation, initialization parameters, statistics gathering, and new warnings. DBUA also provides upgrades from Oracle Database releases 9.0, 9.2, 10.1, and 10.2.
- Supports in-place patch set upgrades
- Starts any services running prior to upgrades

NetCA

- NetCA deinstallation removes listener CRS resources that are defined in the same Oracle home and do not have end points in the `listener.ora` file.
- NetCA converts existing noncluster listeners to cluster listeners.

Oracle RAC 11*g* Installation: Outline

1. **Complete preinstallation tasks:**
   – Hardware requirements
   – Software requirements
   – Environment configuration, kernel parameters, and so on
2. **Perform Oracle Clusterware installation.**
3. **Perform ASM installation.**
4. **Perform Oracle Database 11*g* software installation.**
5. **Install EM agent on cluster nodes if using Grid Control.**
6. **Perform cluster database creation.**
7. **Complete postinstallation tasks.**



Oracle RAC 11*g* Installation: Outline

To successfully install Oracle RAC 11*g*, it is important that you have an understanding of the tasks that must be completed and the order in which they must occur. Before the installation can begin in earnest, each node that is going to be part of your RAC installation must meet the hardware and software requirements that are covered in this lesson. You must perform step-by-step tasks for hardware and software verification, as well as for the platform-specific preinstallation procedures. You must install the operating system patches required by the cluster database, and you must verify that the kernel parameters are correct for your needs.

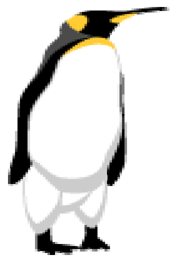
Oracle Clusterware must be installed using OUI. Make sure that your cluster hardware is functioning normally before you begin this step. Failure to do so results in an aborted or nonoperative installation.

After Oracle Clusterware has been successfully installed and tested, again use OUI to install Automatic Storage Management (ASM) and the Oracle Database 11*g* software, including software options required for a RAC configuration. If you intend to use Enterprise Manager Grid Control to manage your RAC deployments, you must next install the Enterprise Manager (EM) agent on each cluster node. After the database has been created, there are a few post-installation tasks that must be completed before your RAC database is fully functional. The remainder of this lesson provides the necessary knowledge to complete these tasks successfully.

Note: It is not mandatory to install ASM to a separate ORACLE_HOME. This is considered a best practice.

Windows and UNIX Installation Differences

- Startup and shutdown services
- Environment variables
- DBA account for database administrators
- Account for running OUI



Installation Differences Between Windows and UNIX

If you are experienced with installing Oracle components in UNIX environments, note that many manual setup tasks required on UNIX are not required on Windows. The key differences between UNIX and Windows installations are discussed below:

- **Startup and Shutdown Services**
  In Windows, OUI creates and sets startup and shutdown services at installation time. In UNIX systems, administrators are responsible for creating these services.
- **Environment Variables**
  In Windows, OUI sets environment variables such as `PATH`, `ORACLE_BASE`, `ORACLE_HOME`, and `ORACLE_SID` in the registry. In UNIX systems, you must manually set these environment variables.
- **DBA Account for Database Administrators**
  In Windows, OUI creates the `ORA_DBA` group. In UNIX systems, you must create the DBA account manually.
- **Account for Running OUI**
  In Windows, you log in with Administrator privileges. You do not need a separate account. In UNIX systems, you must create this account manually. On Oracle RAC systems, each member node of the cluster must have user equivalency for the account that installs the database. This means that the administrative privileges user account and password must be the same on all nodes.

Preinstallation Tasks

- Check system requirements.
- Check software requirements.
- Check kernel parameters.
- Create groups and users.
- Perform cluster setup.

ORACLE

Preinstallation Tasks

Several tasks must be completed before the Oracle Clusterware and Oracle Database 11*g* software can be installed. Some of these tasks are common to all Oracle database installations and should be familiar to you. Others are specific to Oracle RAC 11*g*.

Attention to details here simplifies the rest of the installation process. Failure to complete these tasks can certainly affect your installation and possibly force you to restart the process from the beginning.

Note: It is strongly recommended that Network Time Protocol (NTP) be configured on all cluster nodes before you install RAC.

Hardware Requirements

- At least 1 GB of physical memory is needed.

```
# grep MemTotal /proc/meminfo
MemTotal:        1126400 kB
```

- A minimum of 1 GB of swap space is required.

```
# grep SwapTotal /proc/meminfo
SwapTotal:      1566328 kB
```

- The `/tmp` directory should be at least 400 MB.

```
# df -k /tmp
Filesystem   1K-blocks     Used Available Use%
/dev/sda6      6198556  3137920   2745756  54%
```

- The Oracle Database 11*g* software requires up to 4 GB of disk space.

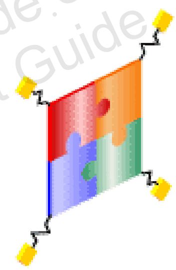
Hardware Requirements

The system must meet the following minimum hardware requirements:

- At least 1 gigabyte (GB) of physical memory is needed. To determine the amount of physical memory, enter the following command: `grep MemTotal /proc/meminfo`
- A minimum of 1 GB of swap space or twice the amount of physical memory is needed. On systems with 2 GB or more of memory, the swap space can be between one and two times the amount of physical memory. To determine the size of the configured swap space, enter the following command: `grep SwapTotal /proc/meminfo`
- At least 400 megabytes of disk space must be available in the `/tmp` directory. To determine the amount of disk space available in the `/tmp` directory, enter the following command: `df -k /tmp`. Alternatively, to list disk space in megabytes or gigabytes, enter: `df -h`.
- Up to 4 GB of disk space is required for the Oracle Database 11*g* software, depending on the installation type. The `df` command can be used to check for the availability of the required disk space.

Network Requirements

- 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).
- The host name and IP address associated with the public interface must be registered in the domain name service (DNS) or the `/etc/hosts` file.



Network Requirements

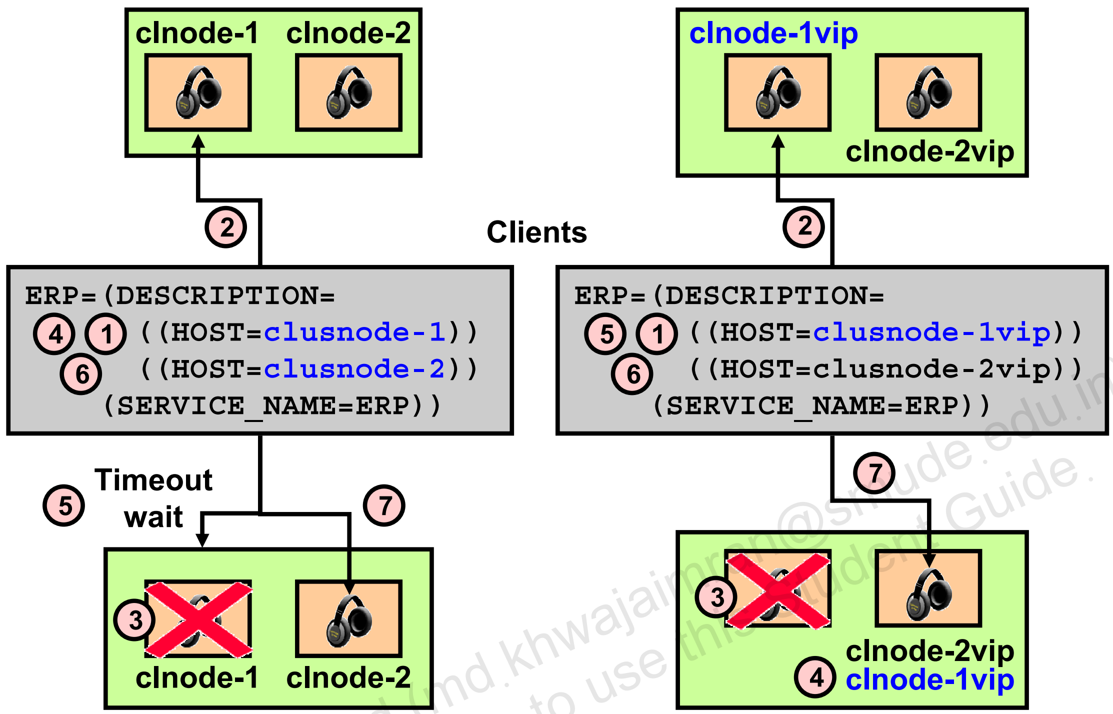
Each node must have at least two network adapters: one for the public network interface and the other for the private network interface or interconnect. In addition, the interface names associated with the network adapters for each network must be the same on all nodes.

For the public network, each network adapter must support TCP/IP. For the private network, the interconnect must support UDP (TCP for Windows) using high-speed network adapters and switches that support TCP/IP. Gigabit Ethernet or an equivalent is recommended.

Note: For a more complete list of supported protocols, see MetaLink Note: 278132.1.

Before starting the installation, each node requires an IP address and an associated host name registered in the DNS or the `/etc/hosts` file for each public network interface. One unused virtual IP address and an associated VIP name registered in the DNS or the `/etc/hosts` file that you configure for the primary public network interface are needed for each node. The virtual IP address must be in the same subnet as the associated public interface. After installation, you can configure clients to use the VIP name or IP address. If a node fails, its virtual IP address fails over to another node. For the private IP address and optional host name for each private interface, Oracle recommends that you use private network IP addresses for these interfaces, for example, `10.*.*.*` or `192.168.*.*`. You can use the `/etc/hosts` file on each node to associate private host names with private IP addresses.

Virtual IP Addresses and RAC



Virtual IP Addresses and RAC

Virtual IP (VIP) addresses are all about availability of applications when an entire node fails.

When a node fails, the VIP address associated with it automatically fails over to some other node in the cluster. When this occurs:

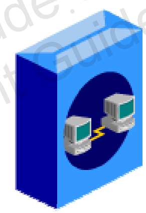
- The new node indicates the new Media Access Control (MAC) address for the VIP. For directly connected clients, this usually causes them to see errors on their connections to the old address.
- Subsequent packets sent to the VIP address go to the new node, which will send error reset (RST) packets back to the clients. This results in the clients getting errors immediately.

This means that when the client issues SQL to the node that is now down (3), or traverses the address list while connecting (1), rather than waiting on a very long TCP/IP timeout (5), which could be as long as ten minutes, the client receives a TCP reset. In the case of SQL, this results in an `ORA-3113` error. In the case of connect, the next address in tnsnames is used (6). The slide shows you the connect case with and without VIP. Without using VIPs, clients connected to a node that died will often wait a 10-minute TCP timeout period before getting an error. As a result, you do not really have a good High Availability solution without using VIPs.

Note: After you are in the SQL stack and blocked on read/write requests, you need to use Fast Application Notification (FAN) to receive an interrupt. FAN is discussed in more detail in the lesson titled “High Availability of Connections.”

RAC Network Software Requirements

- **Supported interconnect software protocols are required:**
  - **TCP/IP**
  - **UDP**
  - **Reliable Data Gram**
- **Token Ring is *not* supported on AIX platforms.**



RAC Network Software Requirements

Each node in a cluster requires a supported interconnect software protocol to support Cache Fusion, and TCP/IP to support Oracle Clusterware polling. In addition to UDP, other supported vendor-specific interconnect protocols include Remote Shared Memory, Hyper Messaging protocol, and Reliable Data Gram. Note that Token Ring is not supported for cluster interconnects on AIX. Your interconnect must be certified by Oracle for your platform. You should also have a Web browser to view online documentation.

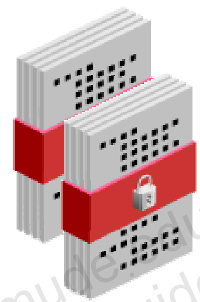
Oracle Corporation has done extensive testing on the Oracle-provided UDP libraries (and TCP for Windows). On the basis of this testing and extensive experience with production customer deployments, Oracle Support strongly recommends the use of UDP (or TCP on Windows) for RAC environments related to Oracle Database 11*g*. Best practices for UDP include:

- Use at least a gigabit Ethernet for optimal performance.
- Cross-over cables are not supported (use a high-speed switch).
- Increase the UDP buffer sizes to the OS maximum.
- Turn on UDP checksumming.

Oracle’s clusterware provides the same functionality as vendor-specific clusterware as required by Oracle RAC. Also, using Oracle Clusterware reduces installation and support complications. However, vendor clusterware may be needed if customers use non-Ethernet interconnect or if you have deployed clusterware-dependent applications.

Package Requirements

- **Package versions are checked by the `cluvfy` utility.**
- **For example, required packages and versions for Red Hat 4.0 and Oracle Enterprise Linux 4 include:**
  - `glibc-2.3.4-2.25`
  - `glibc-common-2.3.4.2-25`
  - `glibc-devel-2.3.4.2-25`
  - `gcc-3.4.6-3`
  - `gcc-c++-3.4.6-3`
  - `libaio-0.3.105-2`
  - `libaio-devel-0.3.105-2`
  - `libstdc++-3.4.6-3.1`
  - `make-3.80-6`
  - `sysstat-5.0.5-11`



Package Requirements

Depending on the products that you intend to install, verify that the packages listed in the slide above are installed on the system. Oracle Universal Installer (OUI) performs checks on your system to verify that it meets the Linux package requirements of the cluster database and related services. To ensure that these checks succeed, verify the requirements before you start OUI.

To determine whether the required packages are installed, enter the following commands:

```
# rpm -q package_name
# rpm -qa |grep package_name_segment
```

For example, to check the `gcc` compatibility packages, run the following command:

```
# rpm -qa |grep compat
compat-libstdc++-33-3.2.3-47.3
compat-gcc-32-3.2.3-47.3
compat-libgcc-296-2.96-132.7.2
compat-gcc-32-c++-3.2.3-47.3
```

If a package is not installed, install it from your Linux distribution media as the `root` user by using the `rpm -i` command. For example, to install the `compat-db` package, use the following command:

```
# rpm -i compat-db-4.1.25-9.i386.rpm
```

Required UNIX Groups and Users

- **Create an `oracle` user, a `dba`, and an `oinstall` group on each node:**

```
# groupadd -g 500 oinstall
# groupadd -g 501 dba
# useradd -u 500 -d /home/oracle -g "oinstall" \
  -G "dba" -m  -s /bin/bash oracle
```

- **Verify the existence of the `nobody` nonprivileged user:**

```
# grep nobody /etc/passwd
Nobody:x:99:99:Nobody:/:/sbin/nobody
```

ORACLE

Required UNIX Groups and Users

You must create the `oinstall` group the first time you install the Oracle database software on the system. This group owns the Oracle inventory, which is a catalog of all the Oracle database software installed on the system.

You must create the `dba` group the first time you install the Oracle database software on the system. It identifies the UNIX users that have database administrative privileges. If you want to specify a group name other than the default `dba` group, you must choose the custom installation type to install the software, or start OUI as a user that is not a member of this group. In this case, OUI prompts you to specify the name of this group.

You must create the `oracle` user the first time you install the Oracle database software on the system. This user owns all the software installed during the installation. The usual name chosen for this user is `oracle`. This user must have the Oracle Inventory group as its primary group. It must also have the `OSDBA` (`dba`) group as the secondary group.

You must verify that the unprivileged user named `nobody` exists on the system. The `nobody` user must own the external jobs (`extjob`) executable after the installation.

Note: The GIDs (group ID) for the `oinstall` and `dba` groups and the UID (user ID)for the `oracle` user must be identical on both nodes.

`oracle` User Environment

- **Set `umask` to `022`.**
- **Set the `DISPLAY` environment variable.**
- **Set the `ORACLE_BASE` environment variable.**
- **Set the `TMP` and `TMPDIR` variables, if needed.**

```
$ cd
$ vi .bash_profile
umask 022
ORACLE_BASE=/u01/app/oracle; export ORACLE_BASE
TMP=/u01/mytmp; export TMP
TMPDIR=$TMP; export TMPDIR
```


`oracle` User Environment

You must run OUI as the `oracle` user. However, before you start OUI, you must configure the environment of the `oracle` user. To configure the environment, you must:

- Set the default file mode creation mask (`umask`) to `022` in the shell startup file
- Set the `DISPLAY` and `ORACLE_BASE` environment variables
- Secure enough temporary disk space for OUI

If the `/tmp` directory has less than 400 megabytes of free disk space, identify a file system that is large enough and set the `TMP` and `TMPDIR` environment variables to specify a temporary directory on this file system. Use the `df -k` command to identify a suitable file system with sufficient free space. Make sure that the `oracle` user and the `oinstall` group can write to the directory.

```
# df -k
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hdb1              3020140   2471980    394744  87% /
/dev/hdb2              3826584     33020   3599180   1% /home
/dev/dha1               386008    200000    186008   0% /dev/shm
/dev/hdb5             11472060   2999244   7890060  28% /u01
# /mkdir /u01/mytmp
# chmod 777 /u01/mytmp
```

User Shell Limits

- **Add the following lines to the `/etc/security/limits.conf` file:**

```
*   soft  nproc    2047
*   hard  nproc    16384
*   soft  nofile  1024
*   hard  nofile  65536
```

- **Add the following line to the `/etc/pam.d/login` file:**

```
session  required   /lib/security/pam_limits.so
```


User Shell Limits

To improve the performance of the software, you must increase the following shell limits for the `oracle` user:

- **`nofile`:** The maximum number of open file descriptors should be 65536.
- **`nproc`:** The maximum number of processes available to a single user must not be less than 16384.

The hard values, or upper limits, for these parameters can be set in the `/etc/security/limits.conf` file as shown in the slide above. The entry configures Pluggable Authentication Modules (PAM) to control session security. PAM is a system of libraries that handle the authentication tasks of applications (services) on the system. The principal feature of the PAM approach is that the nature of the authentication is dynamically configurable.

Configuring for Remote Installation

To configure Secure Shell:

1. **Create the public and private keys on all nodes:**

```
[vx0044]$ /usr/bin/ssh-keygen -t dsa
[vx0045]$ /usr/bin/ssh-keygen -t dsa
```

2. **Concatenate `id_dsa.pub` from all nodes into the `authorized_keys` file on the first node:**

```
[vx0044]$ ssh vx0044 "cat ~/.ssh/id_dsa.pub" >> \
~/.ssh/authorized_keys
[vx0044]$ ssh vx0045 "cat ~/.ssh/id_dsa.pub" >> \
~/.ssh/authorized_keys
```

3. **Copy the `authorized_keys` file to the other nodes:**

```
[vx0044]$ scp ~/.ssh/authorized_keys vx0045:/home/oracle/.ssh/
```


Configuring for Remote Installation

```
$ rpm -qa|grep -i openssh
openssh-3.9p1-8.RHEL4.20
openssh-askpass-3.9p1-8.RHEL4.20
openssh-clients-3.9p1-8.RHEL4.20
openssh-server-3.9p1-8.RHEL4.20
```

Assume that your cluster comprises two nodes, vx0044 and vx0045. You can perform the following steps to configure SSH using DSA on that cluster. Note that SSH using SSA is also supported.

1. As the `oracle` user, create the public and private keys on both nodes:

```
[vx0044]$ /usr/bin/ssh-keygen -t dsa
[vx0045]$ /usr/bin/ssh-keygen -t dsa
```

   Accept the default location for the key file. When prompted for the pass phrase, just press the Enter key.

2. Concatenate the contents of the `id_dsa.pub` file from each node into the `authorized_keys` file on the first node.

```
[vx0044]$ ssh vx0044 "cat ~/.ssh/id_dsa.pub" >> \
~/.ssh/authorized_keys
[vx0044]$ ssh vx0045 "cat ~/.ssh/id_dsa.pub" >> \
~/.ssh/authorized_keys
```

Configuring for Remote Installation (continued)

3. Copy the `authorized_keys` file to the same location on the second node.

```
[vx0044]$ scp ~/.ssh/authorized_keys vx0045:/home/oracle/.ssh/
```

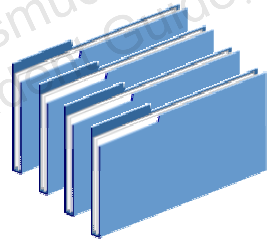
4. Test the configuration.

```
[vx0044]$ ssh vx0045 hostname
$ ssh vx0045 uptime
vx0045.us.oracle.com
```

Required Directories for the Oracle Database Software

You must identify five directories for the Oracle database software:

- **Oracle base directory**
- **Oracle inventory directory**
- **Oracle Clusterware home directory**
- **Oracle home directory for the database**
- **Oracle home directory for ASM**



Required Directories for the Oracle Database Software

The Oracle base (`ORACLE_BASE`) directory acts as a top-level directory for the Oracle database software installations. On UNIX systems, the Optimal Flexible Architecture (OFA) guidelines recommend that you must use a path similar to the following for the Oracle base directory:

`/mount_point/app/oracle_sw_owner`

where *`mount_point`* is the mount-point directory for the file system that contains the Oracle database software and *`oracle_sw_owner`* is the UNIX username of the Oracle database software owner, which is usually `oracle`.

The Oracle inventory directory (`oraInventory`) stores the inventory of all software installed on the system. It is required by, and shared by, all the Oracle database software installations on a single system. The first time you install the Oracle database software on a system, OUI prompts you to specify the path to this directory. If you are installing the software on a local file system, it is recommended that you choose the following path: `ORACLE_BASE/oraInventory`

OUI creates the directory that you specify and sets the correct owner, group, and permissions on it.

Required Directories for the Oracle Database Software (continued)

The Oracle Clusterware home directory is the directory where you choose to install the software for Oracle Clusterware. You must install Oracle Clusterware in a separate home directory. Because the clusterware parent directory should be owned by `root`, it requires a separate base directory from the one used by the database files. When you run OUI, it prompts you to specify the path to this directory, as well as a name that identifies it. It is recommended that you specify a path similar to the following for the Oracle Clusterware home directory:

```
/u01/crs11g
```

Note that in the example above, `/u01` should be owned by the `root` user and writable by group `oinstall`.

The Oracle home directory is the directory where you choose to install the software for a particular Oracle product. You must install different Oracle products, or different releases of the same Oracle product, in separate Oracle home directories. When you run OUI, it prompts you to specify the path to this directory, as well as a name that identifies it. The directory that you specify must be a subdirectory of the Oracle base directory. It is recommended that you specify a path similar to the following for the Oracle home directory:

```
ORACLE_BASE/product/11.1.0/db_1
```

Consider creating a separate home directory for ASM if you will be using it to manage your shared storage. Specify a path similar to the following directory for ASM:

```
ORACLE_BASE/product/11.1.0/asm_1
```

Linux Operating System Parameters

Parameter	Value	File
semmsl	250	/proc/sys/kernel/sem
semmns	32000	/proc/sys/kernel/sem
semopm	100	/proc/sys/kernel/sem
semmni	128	/proc/sys/kernel/sem
shmall	2097152	/proc/sys/kernel/shmall
shmmax	½ physical memory	/proc/sys/kernel/shmmax
shmmni	4096	/proc/sys/kernel/shmmni
file-max	65536	/proc/sys/fs/file-max
rmem_max	4194304	/proc/sys/net/core/rmem_max
rmem_default	4194304	/proc/sys/net/core/rmem_default
wmem_max	262144	/proc/sys/net/core/wmem_max
wmem_default	262144	/proc/sys/net/core/wmem_default

Linux Operating System Parameters

Verify that the parameters shown in the table above are set to values greater than or equal to the recommended value shown. Use the `sysctl` command to view the default values of the various parameters. For example, to view the semaphore parameters, run the following command:

```
# sysctl -a|grep sem
kernel.sem = 250   32000   32   128
```

The values shown represent `semmsl`, `semmns`, `semopm`, and `semmni` in that order. Kernel parameters that can be manually set include:

- **`SEMMNS:`** The number of semaphores in the system
- **`SEMMNI:`** The number of semaphore set identifiers that control the number of semaphore sets that can be created at any one time
- **`SEMMSL:`** Semaphores are grouped into semaphore sets, and `SEMMSL` controls the array size, or the number of semaphores that are contained per semaphore set. It should be about ten more than the maximum number of the Oracle processes.
- **`SEMOPM:`** The maximum number of operations per semaphore operation call
- **`SHMMAX:`** The maximum size of a shared-memory segment. This must be slightly larger than the largest anticipated size of the System Global Area (SGA), if possible.
- **`SHMMNI:`** The number of shared memory identifiers

Linux Operating System Parameters (continued)

- **`RMEM_MAX`:** The maximum TCP receive window (buffer) size
- **`RMEM_DEFAULT`:** The default TCP receive window size
- **`WMEM_MAX`:** The maximum TCP send window size
- **`WMEM_DEFAULT`:** The default TCP send window size

You can adjust these semaphore parameters manually by writing the contents of the `/proc/sys/kernel/sem` file:

```
# echo SEMMSL_value SEMMNS_value SEMOPM_value \
  SEMMNI_value > /proc/sys/kernel/sem
```

To change these parameter values and make them persistent, edit the `/etc/sysctl.conf` file as follows:

```
# vi /etc/sysctl.conf
...
kernel.sem = 250 32000 100 128
kernel.shmall = 2097152
kernel.shmmax = 2147483648
kernel.shmmni = 4096
fs.file-max = 65536
rmem_max = 4194304
rmem_default = 4194304
wmem_default = 262144
wmem_max = 262144
net.ipv4.ip_local_port_range = 1024 65000
```

The kernel parameters shown above are recommended values only. For production database systems, it is recommended that you tune these values to optimize the performance of the system.

Note: Because they are a lot of parameters to check, you can use the Cluster Verification Utility to automatically do the verification.

Cluster Setup Tasks

1. View the Certifications by Product section at http://metalink.oracle.com/.
2. Verify your high-speed interconnects.
3. Determine the shared storage (disk) option for your system:
   – OCFS or other shared file system solution
   – Raw devices
   – ASM

   ASM cannot be used for the OCR and Voting Disk files!
4. Install the necessary operating system patches.

ORACLE

Cluster Setup Tasks

Ensure that you have a certified combination of the operating system and the Oracle database software version by referring to the certification information on Oracle MetaLink in the Availability & Certification section. See the Certifications by Product section at http://metalink.oracle.com.

Verify that your cluster interconnects are functioning properly. If you are using vendor-specific clusterware, follow the vendor's instructions to ensure that it is functioning properly.

Determine the storage option for your system, and configure the shared disk. Oracle recommends that you use Automatic Storage Management (ASM) and Oracle Managed Files (OMF), or a cluster file system such as Oracle Cluster File System (OCFS). If you use ASM or a cluster file system, you can also utilize OMF and other Oracle Database 11*g* storage features.

Oracle Clusterware requires that the OCR files and voting disk files be shared. Note that ASM cannot be used to store these files as the clusterware components are started before the ASM or RAC instances. These files could map to shared block or raw devices or exist on an OCFS volume.

Verifying Cluster Setup with `cluvfy`

- **Install the `cvuqdisk` rpm required for `cluvfy`:**

```
# su root
# cd /stage/db/rpm
# export CVUQDISK_GRP=dba
# rpm -iv cvuqdisk-1.0.1-1.rpm
```

- **Run the `cluvfy` utility as `oracle` as shown below:**

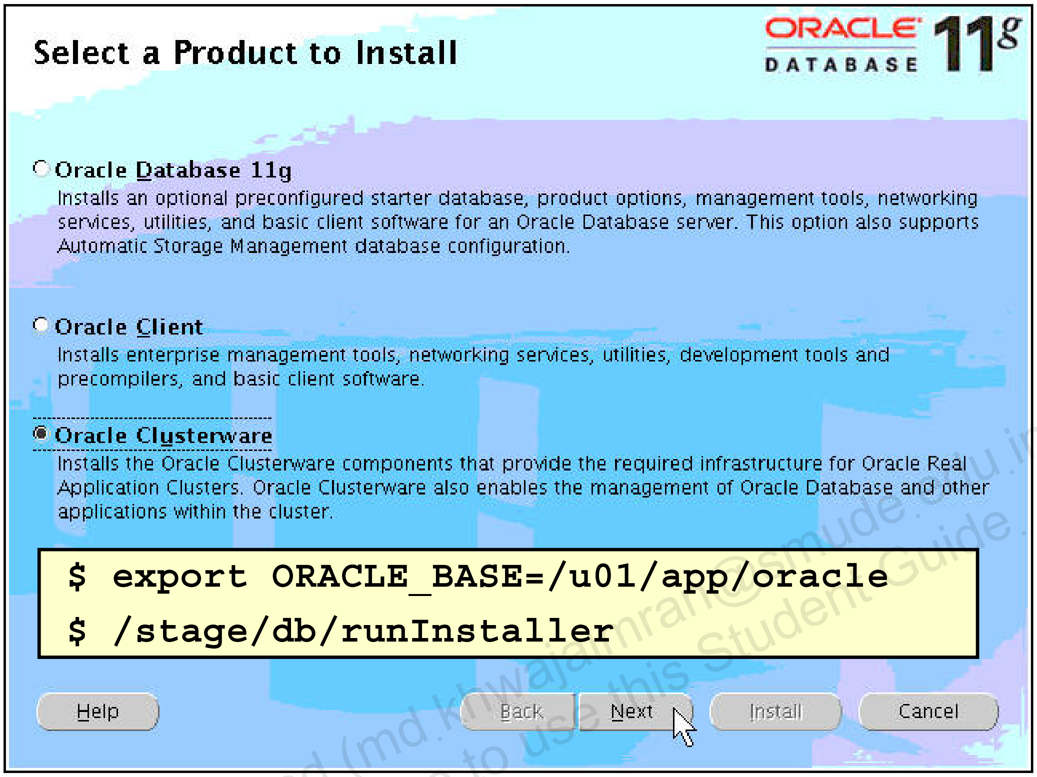
```
# cd /stage/db
./runcluvfy.sh stage -post hwos -n all -verbose
```


Verifying Cluster Setup with `cluvfy`

The Cluster Verification Utility (`cluvfy`) enables you to perform many preinstallation and postinstallation checks at various stages of your RAC database installation. The `cluvfy` utility is available in Oracle Database 11*g* Release 1. To check the readiness of your cluster for an Oracle Clusterware installation, run `cluvfy` as shown below:

```
$ runcluvfy.sh stage -post hwos -n all -verbose
Performing post-checks for hardware and operating system setup
Checking node reachability...
...
Result: Node reachability check passed from node "vx0044".
Checking user equivalence...
...
Result: User equivalence check passed for user "oracle".
Checking node connectivity...
...
Result: Node connectivity check passed.
Checking shared storage accessibility...
...
Shared storage check passed on nodes "vx0045,vx0044".
...
Post-check for hardware and operating system setup was successful on all
the nodes.
```

Installing Oracle Clusterware

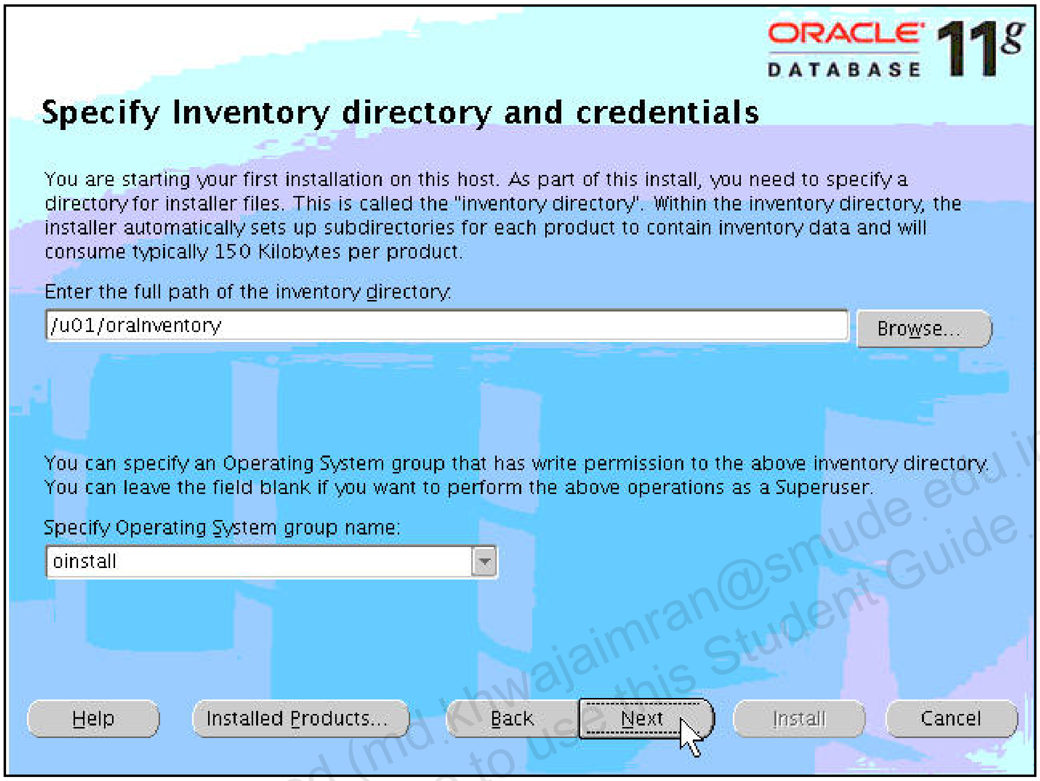


Installing Oracle Clusterware

Set the ORACLE_BASE environment variable in accordance with your installation location scheme. Run OUI by executing the `runInstaller` command from the installation CD or the staged software directory. The “Select a Product to Install” screen allows you to install and create a database, install Oracle client software, or install Oracle Clusterware. Click the **Oracle Clusterware** button and then click **Next**.

If you are performing this installation in an environment in which you have never installed the Oracle database software (that is, if the environment does not have an OUI inventory), OUI displays the “Specify Inventory directory and credentials” screen. If you are performing this installation in an environment where the OUI inventory is already set up, OUI displays the Specify File Locations screen instead of the “Specify Inventory directory and credentials” screen.

Specifying the Inventory Directory

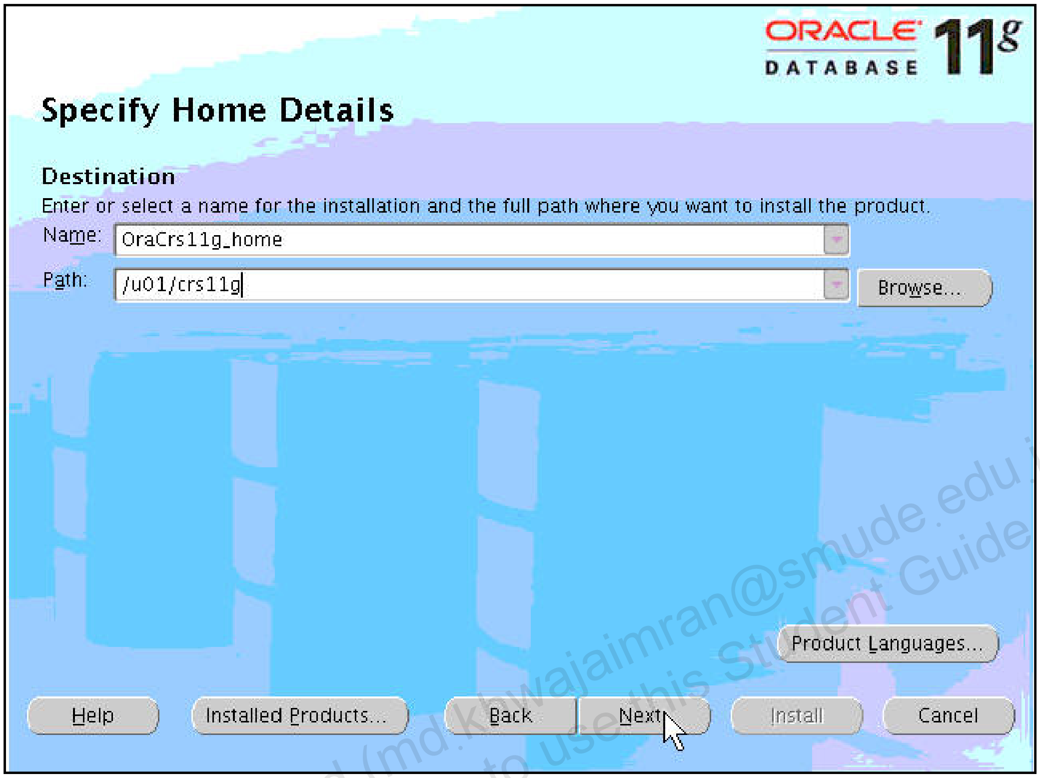


ORACLE

Specifying the Inventory Directory

On the "Specify Inventory directory and credentials" screen, enter the inventory location. If `ORACLE_BASE` has been properly set, OUI suggests the proper directory location for the inventory location as per OFA guidelines. If `ORACLE_BASE` has not been set, enter the proper inventory location according to your requirements. Enter the UNIX group name information `oinstall` in the "Specify Operating System group name" field, and then click Next.

Specify Home Details

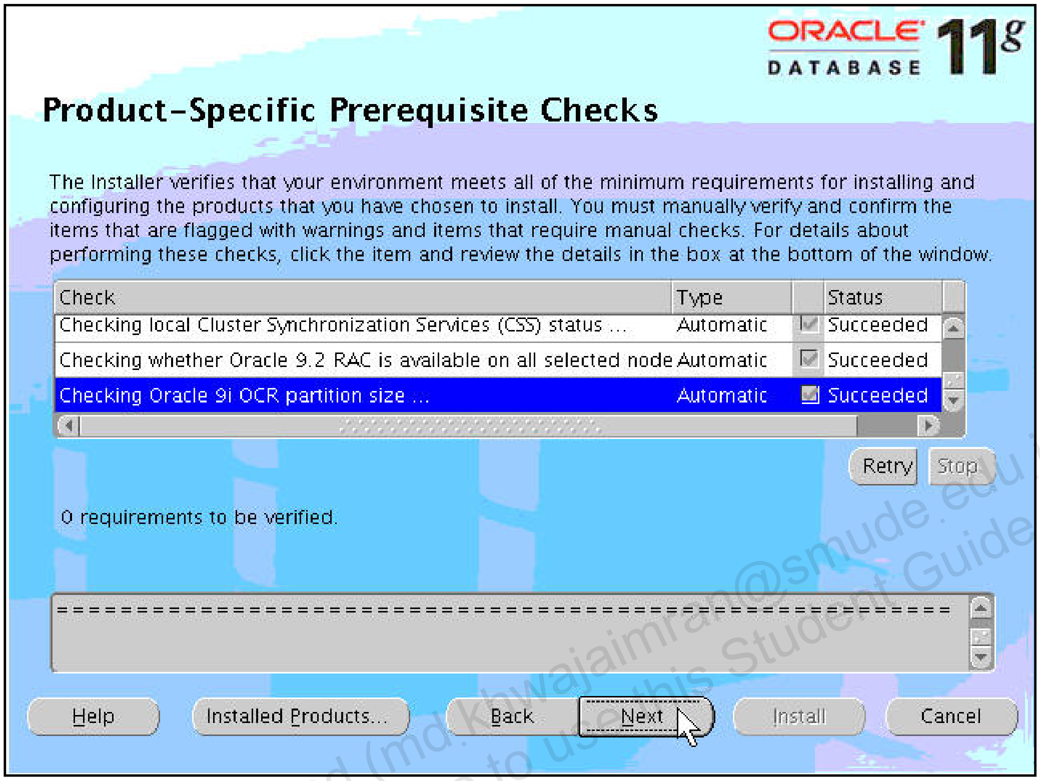


Specify Home Details

Next, OUI displays the Specify Home Details screen. The "Specify Home Details" screen contains predetermined information for the source of the installation files and the target destination information. OUI provides an Oracle Clusterware Home name in the Name field located in the Destination section of the screen. You may accept the name or enter a new name at this time. If `ORACLE_BASE` has been set, an OFA-compliant directory path appears in the Path field located below the Destination section. If not, enter the location in the target destination, and click **Next** to continue.

If `ORACLE_HOME` is set in the environment, this appears in the OUI location window. `ORACLE_HOME` typically refers to the DB home, and there is no corresponding environment variable for the Clusterware installation. You should be aware of this, and not just click through because there is a value. The parent directory of the Clusterware Home should be owned by root and writable by the install group, oinstall.

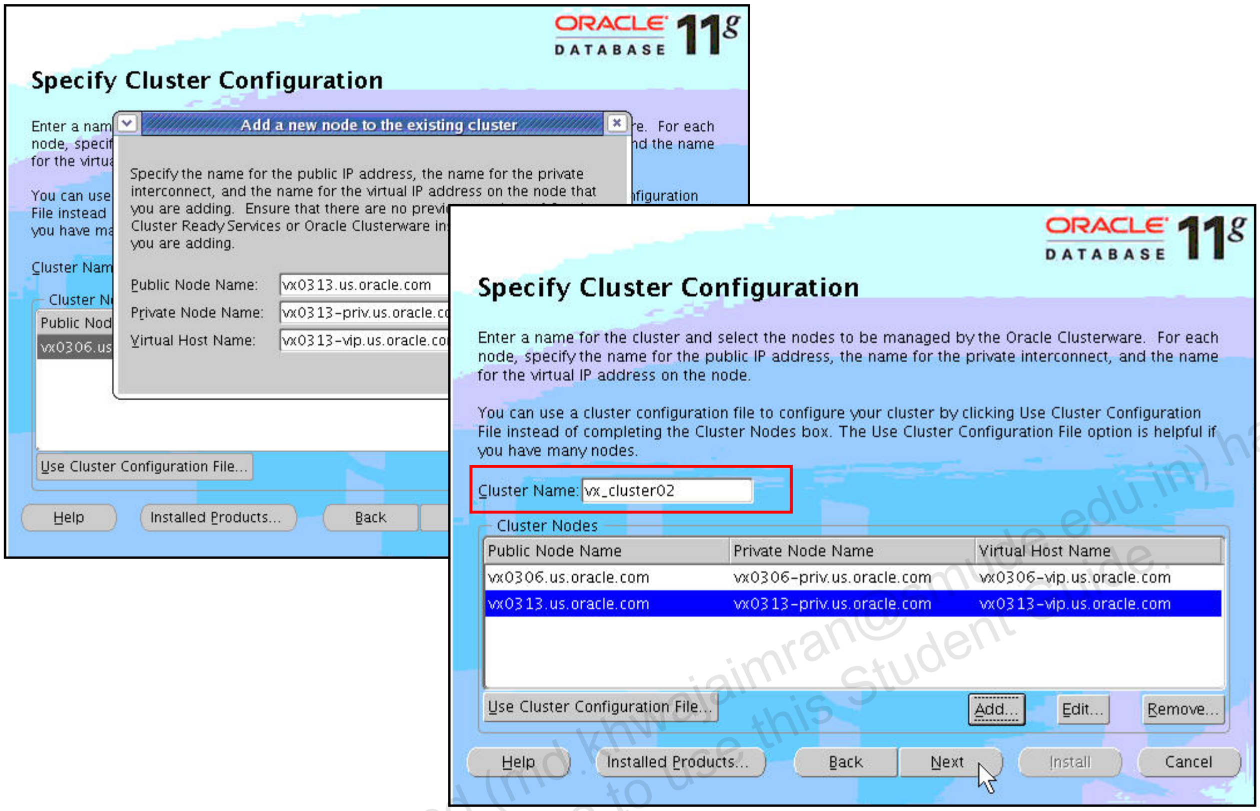
Product-Specific Prerequisite Checks



Product-Specific Prerequisite Checks

The installer then checks your environment to ensure that it meets the minimum requirements for an Oracle Clusterware installation. The installer checks for the existence of critical packages and release levels, proper kernel parameter settings, network settings, and so on. If discrepancies are found, they are flagged and you are given an opportunity to correct them. If you are sure that the flagged items will not cause a problem, it is possible to click the item and change the status to self-checked, and continue with the installation. Only do this if you are absolutely sure that no problems actually exist, otherwise correct the condition before proceeding. When all checks complete successfully, click the **Next** button to proceed.

Cluster Configuration



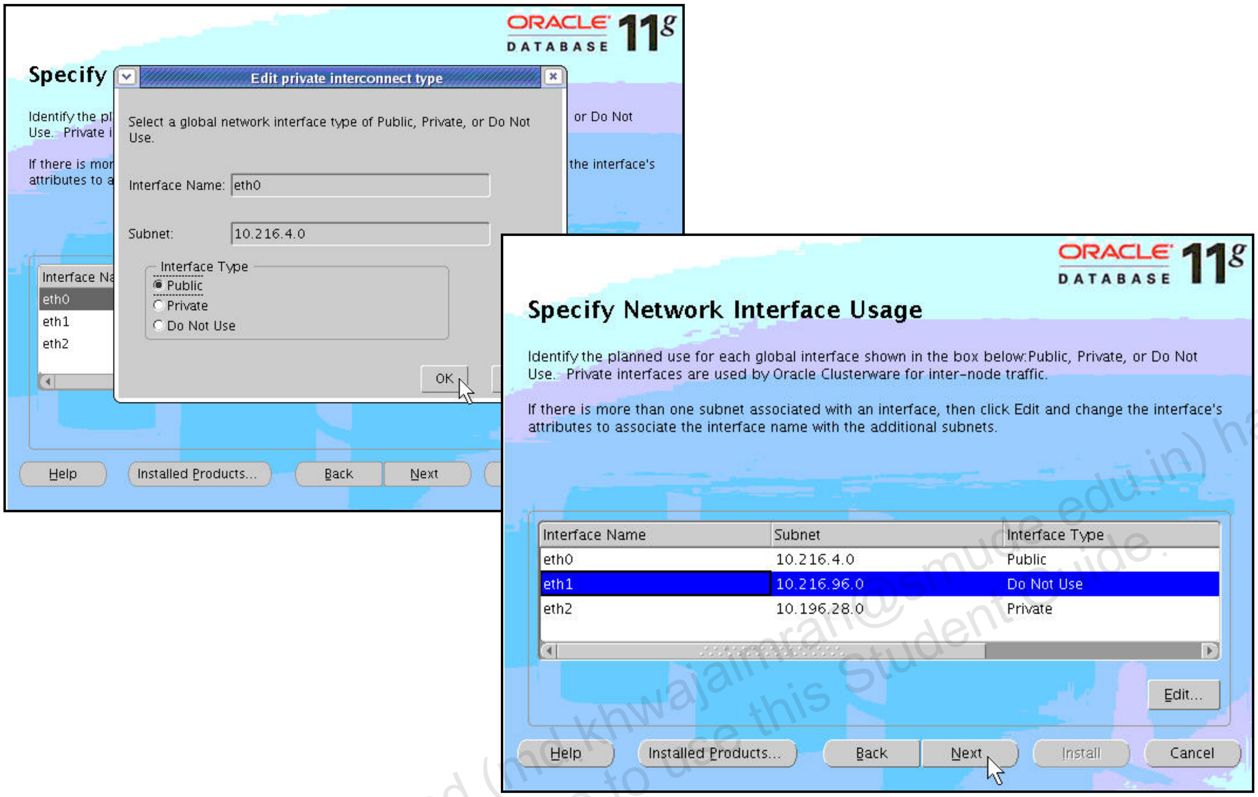
ORACLE

Cluster Configuration

The Specify Cluster Configuration screen displays predefined node information if OUI detects that your system has vendor clusterware. Otherwise, OUI displays the Cluster Configuration screen without the predefined node information. If all your nodes do not appear in the cluster nodes window, click the Add button. You must supply the public node names, private node names, and virtual host names for each node that you add. All of these names must be resolvable on every node by using either DNS or the `/etc/hosts` file.

In the Cluster Name field, enter a name for your cluster. Ensure that the cluster name is unique in your network.

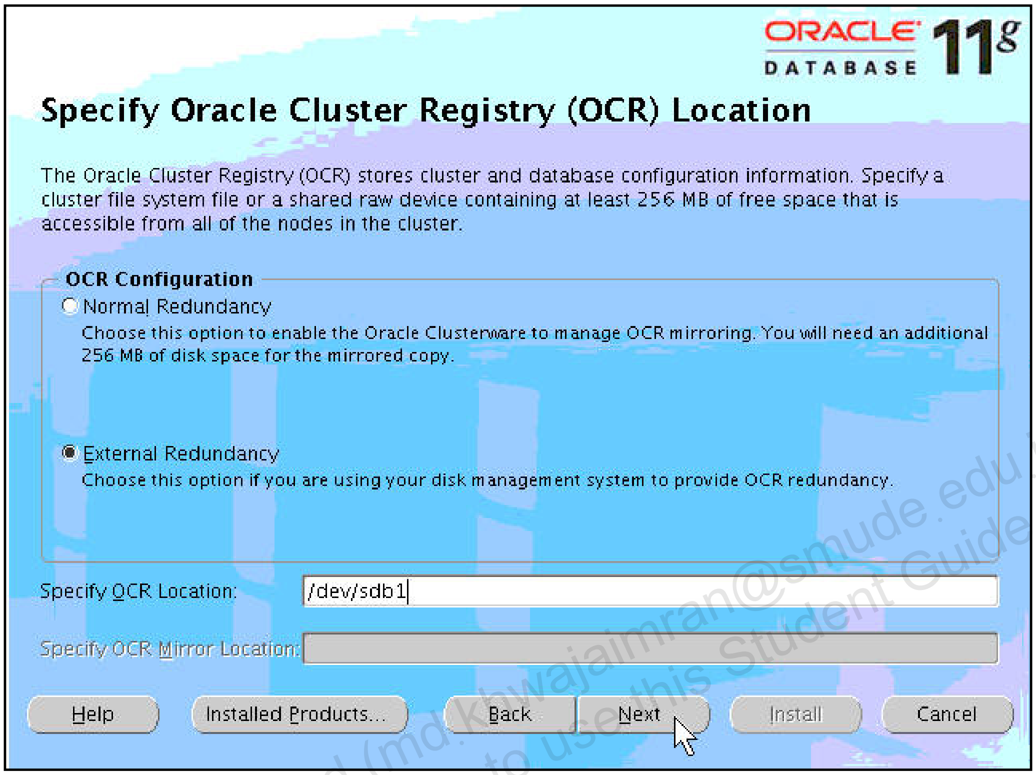
Private Interconnect Enforcement



Private Interconnect Enforcement

The Specify Network Interface Usage screen enables you to select the network interfaces on your cluster nodes to use for internode communication. Ensure that the network interfaces that you choose for the interconnect have enough bandwidth to support the cluster and RAC-related network traffic. A gigabit Ethernet interface is highly recommended for the private interconnect. To configure the interface for private use, click the interface name, and then click Edit. A pop-up window appears and allows you to indicate the usage for the network interfaces. In the example shown in the slide, there are three interfaces: `eth0`, `eth1`, and `eth2`. The `eth0` interface is the hosts' primary network interface and should be marked Public. The `eth1` interface is dedicated for the storage network, which supports this cluster's shared storage. It should be marked Do Not Use. The `eth2` interface is configured for the private interconnect and should be marked Private. When you finish, click the **Next** button to continue.

Oracle Cluster Registry File

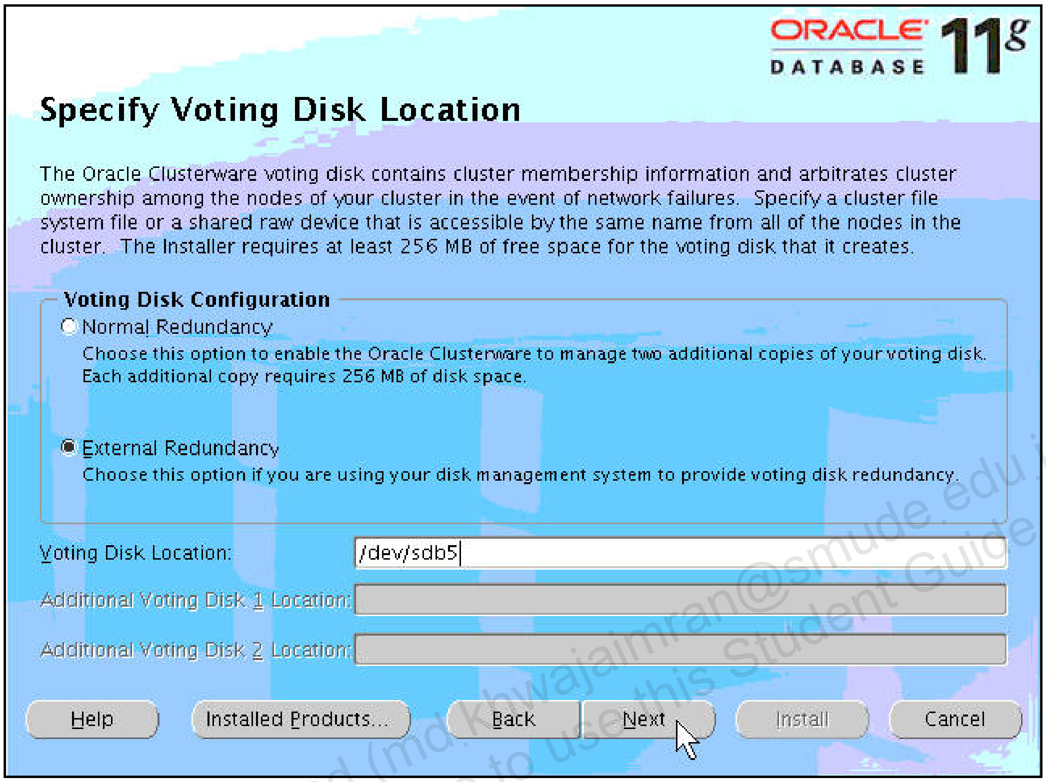


ORACLE

Oracle Cluster Registry File

The Specify Oracle Cluster Registry Location screen appears next. Enter a fully qualified file name for the shared block or raw device or a *shared* file system file for the OCR file. If you are using an external disk mirroring scheme, click the **External Redundancy** option button. You will be prompted for a single OCR file location. If no mirroring scheme is employed, click the **Normal Redundancy** option button. You will be prompted for two file locations. For highest availability, provide locations that exist on different disks or volumes. Click **Next** to continue.

Voting Disk File



ORACLE

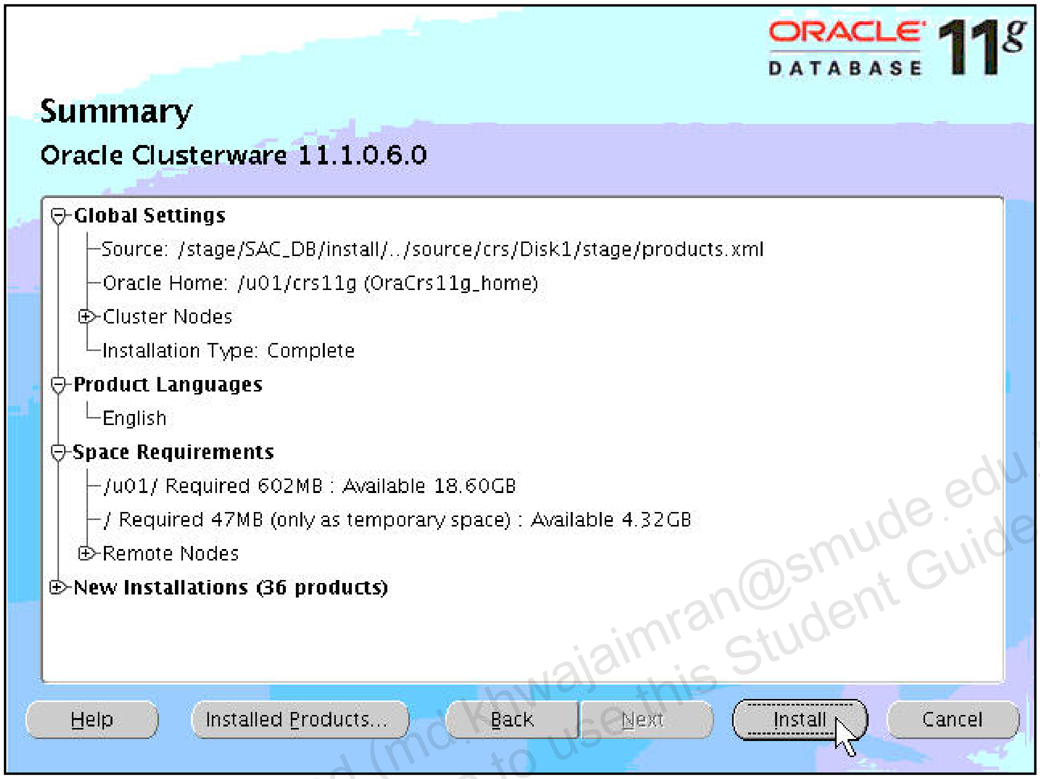
Voting Disk File

The primary purpose of the voting disk is to help in situations where the private network communication fails. When the private network fails, the clusters are unable to have all nodes remain available because they cannot synchronize I/O to the shared disk. Therefore, some of the nodes must go offline. The voting disk is used to communicate the node state information used to determine which nodes will go offline.

Because the voting disk must be accessible to all nodes to accurately assess membership, the file must be stored on a shared disk location. The voting disk can reside on a block or raw device or a cluster file system.

In Oracle Database 11*g* Release 1, voting disk availability is improved by the configuration of multiple voting disks. If the voting disk is not mirrored, then there should be at least three voting disks configured.

Summary and Install



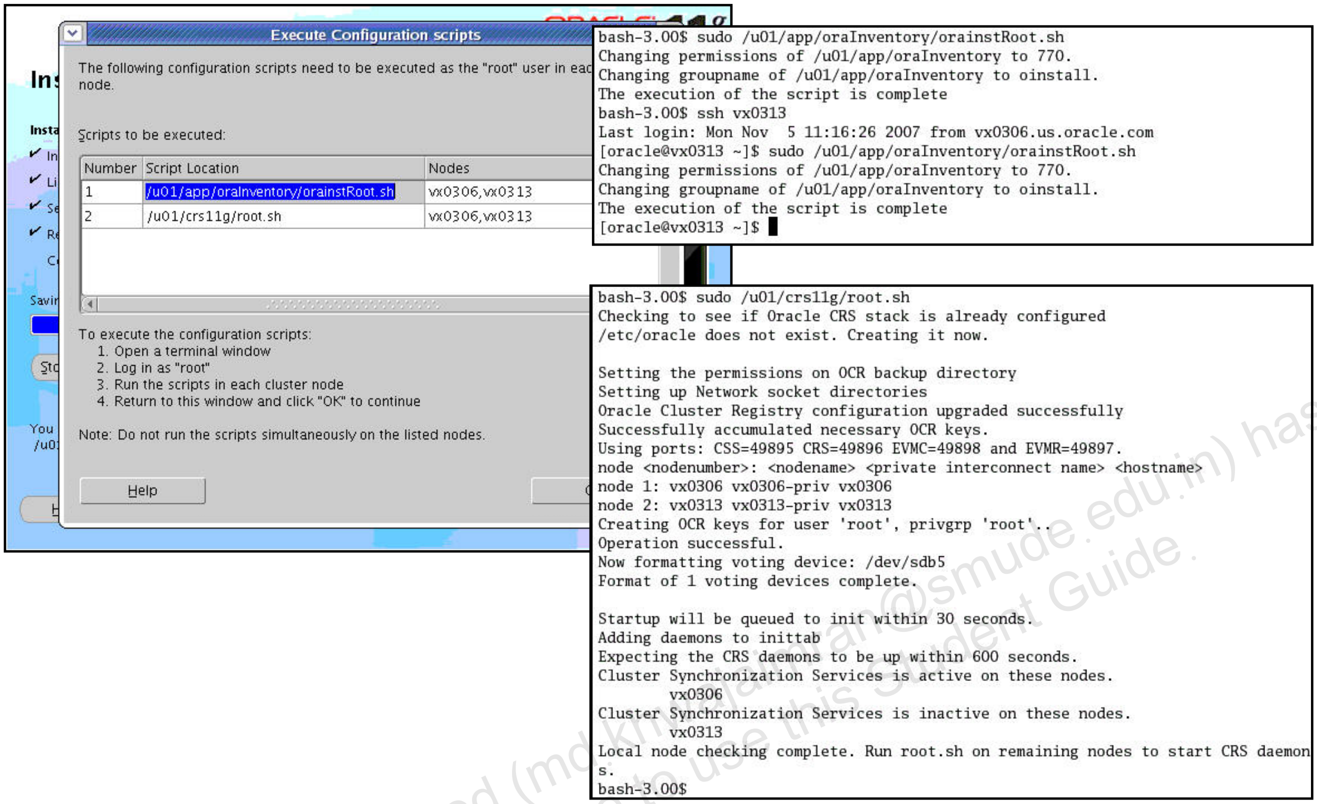
ORACLE

Summary and Install

OUI displays the Summary screen. Note that OUI must install the components shown in the summary window. Click the Install button. The Install screen is then displayed, informing you about the progress of the installation.

During the installation, OUI first copies the software to the local node and then copies the software to the remote nodes.

Run Configuration Scripts on All Nodes



ORACLE

Run Configuration Scripts on All Nodes

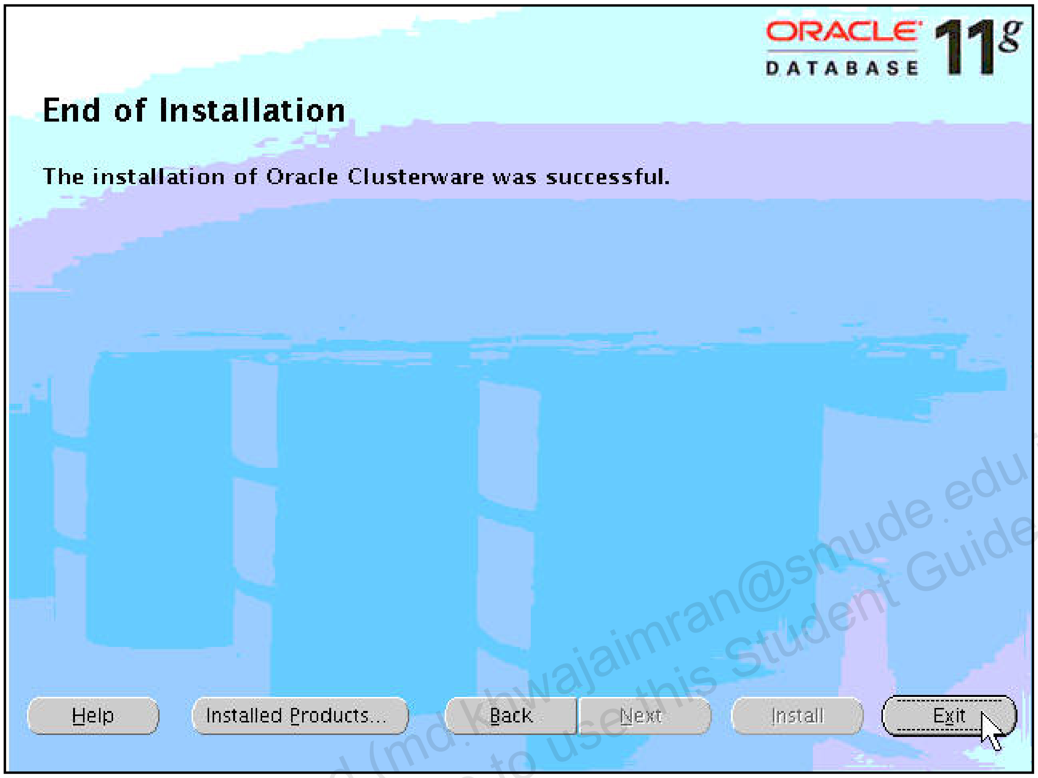
Next, OUI displays a dialog box indicating that you must run the `orainstRoot.sh` and `root.sh` script on all the nodes that are part of this installation. The `root.sh` script runs the following assistants without your intervention:

- Oracle Cluster Registry Configuration Tool (`ocrconfig`)
- Cluster Configuration Tool (`clscfg`)

When the `root.sh` script has been run on all nodes, click the OK button to close the dialog box. Run the `cluvfy` utility to verify the post crs installation.

Note: Make sure you run the above mentioned scripts serially on each node in the proposed order.

End of Installation



End of Installation

When the configuration scripts have been run on both nodes, the Configuration Assistants page is displayed. The ONS Configuration Assistant and Private Interconnect Configuration Assistant are run and their progress is displayed here. The Cluster Verification Utility is then run to test the viability of the new installation. When the Next button is clicked, the End of Installation screen appears. Click **Exit** to leave OUI.

Verifying the Oracle Clusterware Installation

- Check for Oracle Clusterware processes with the `ps` command.
- Check the Oracle Clusterware startup entries in the `/etc/inittab` file.

```
# cat /etc/inittab
# Run xdm in runlevel 5
x:5:respawn:/etc/X11/prefdm -nodaemon
h1:35:respawn:/etc/init.d/init.evmd run >/dev/null
2>&1 </dev/null
h2:35:respawn:/etc/init.d/init.cssd fatal >/dev/null
2>&1 </dev/null
h3:35:respawn:/etc/init.d/init.crsd run >/dev/null
2>&1 </dev/null
```


Verifying the Oracle Clusterware Installation

Before continuing with the installation of the Oracle database software, you must verify your Oracle Clusterware installation and startup mechanism. With the introduction of Oracle RAC 11*g*, cluster management is controlled by the `evmd`, `ocssd`, and `crsd` processes. Run the `ps` command on both nodes to make sure that the processes are running.

```
$ ps -ef|grep d.bin
oracle     9332  9313  0 Oct30 ?        00:00:38 .../evmd.bin
root       9347  8498  0 Oct30 ?        00:25:20 .../crsd.bin reboot
oracle     9897  9414  0 Oct30 ?        00:09:40 .../ocssd.bin
oracle    18387     1  0 Oct30 ?        00:00:00 .../oclskd.bin

...
```

Check the startup mechanism for Oracle Clusterware. In Oracle RAC 11*g*, Oracle Clusterware processes are started by entries in the `/etc/inittab` file, which is processed whenever the run level changes (as it does during system startup and shutdown):

```
h1:35:respawn:/etc/init.d/init.evmd run >/dev/null 2>&1 </dev/null
h2:35:respawn:/etc/init.d/init.cssd fatal >/dev/null 2>&1 </dev/null
h3:35:respawn:/etc/init.d/init.crsd run >/dev/null 2>&1 </dev/null
```

Note: The processes are started at run levels 3 and 5 and are started with the *`respawn`* flag.

Verifying the Oracle Clusterware Installation (continued)

This means that if the processes abnormally terminate, they are automatically restarted. If you kill the Oracle Clusterware processes, they automatically restart or, worse, cause the node to reboot. For this reason, stopping Oracle Clusterware by killing the processes is not recommended. If you want to stop Oracle Clusterware without resorting to shutting down the node, you should run the `crsctl` command:

```
# /u01/crs11g/bin/crsctl stop
```

If you encounter difficulty with your Oracle Clusterware installation, it is recommended that you check the associated log files. To do this, check the directories under the Oracle Clusterware Home:

$ORA_CRS_HOME/log/*hostname*: This directory contains the `alert.log` file for the nodes Clusterware.

$ORA_CRS_HOME/log/*hostname*/crsd/: This directory contains the log files for the CRSD process.

$ORA_CRS_HOME/log/*hostname*/cssd/: This directory contains the log files for the CSSD process.

$ORA_CRS_HOME/log/*hostname*/evmd/: This directory contains the log files for the EVMD process.

$ORA_CRS_HOME/log/*hostname*/client/: Log files for OCR are written here.

When you have determined that your Oracle Clusterware installation is successful and fully functional, you may start the Oracle Database 11*g* software installation.

Summary

In this lesson, you should have learned how to:

- Describe the installation of Oracle RAC 11*g*
- Perform RAC preinstallation tasks
- Perform cluster setup tasks
- Install Oracle Clusterware

Practice 1: Overview

This practice covers the following topics:

- Performing initial cluster configuration
- Installing Oracle Clusterware

ORACLE

2

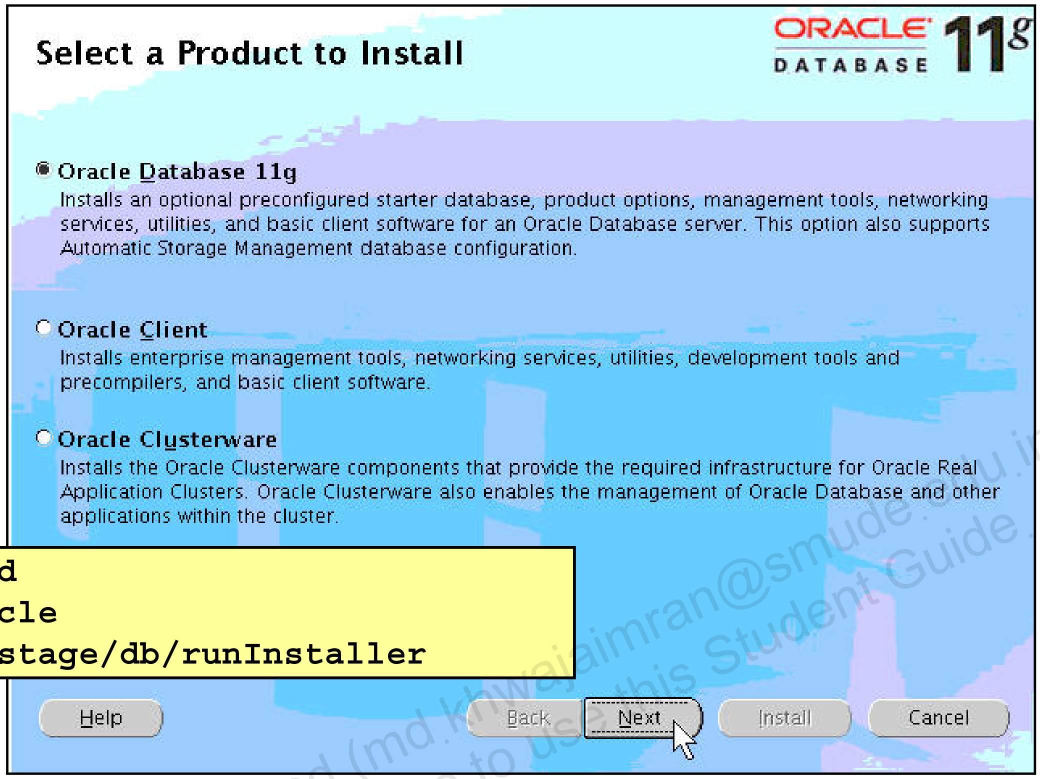
RAC Software Installation

Objectives

After completing this lesson, you should be able to:

- Install and configure Automatic Storage Management (ASM)
- Install the Oracle database software
- Perform required tasks prior to database creation

Installing Automatic Storage Management




```
$ id
oracle
$ /stage/db/runInstaller
```

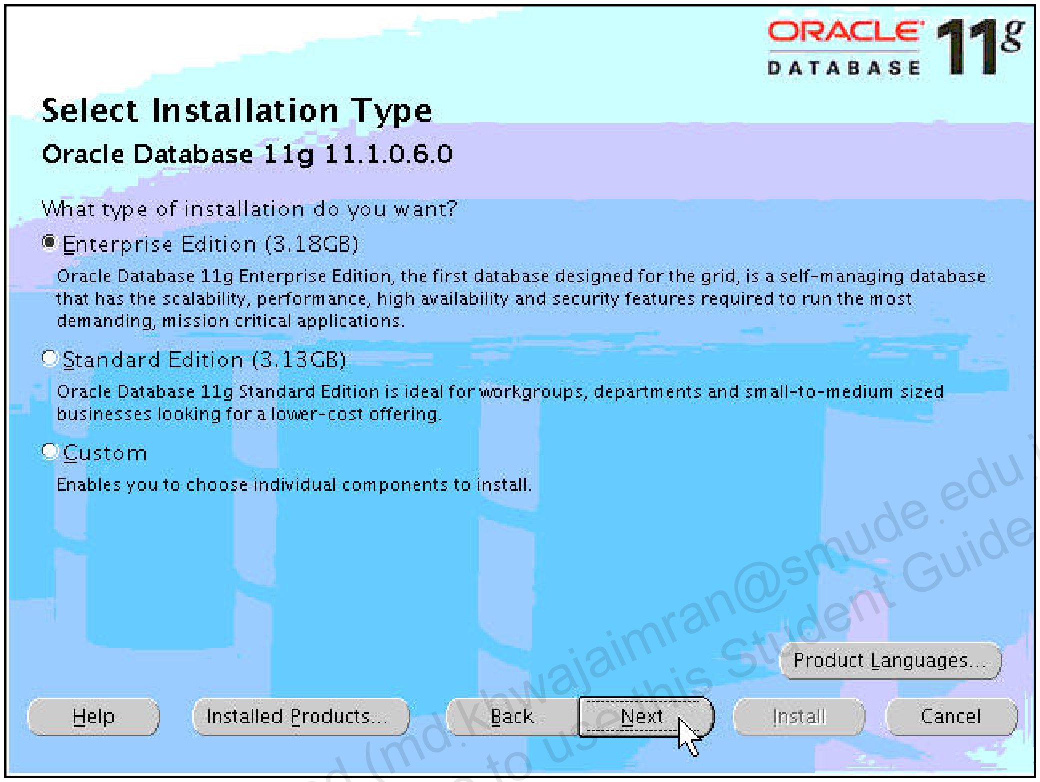

Installing Automatic Storage Management

For this installation, Automatic Storage Management (ASM) is used to manage the shared storage for the cluster database. After Oracle Clusterware is installed, run Oracle Universal Installer (OUI) from either the installation CD or a staged software location and install ASM as the `oracle` user.

```
$ id
oracle
$ cd /stage/db
$./runInstaller
```

The “Select a Product to Install” screen allows you to install and create a database, install Oracle client software, or install Oracle Clusterware. Click the **Oracle Database 11*g*** button and then click **Next**.

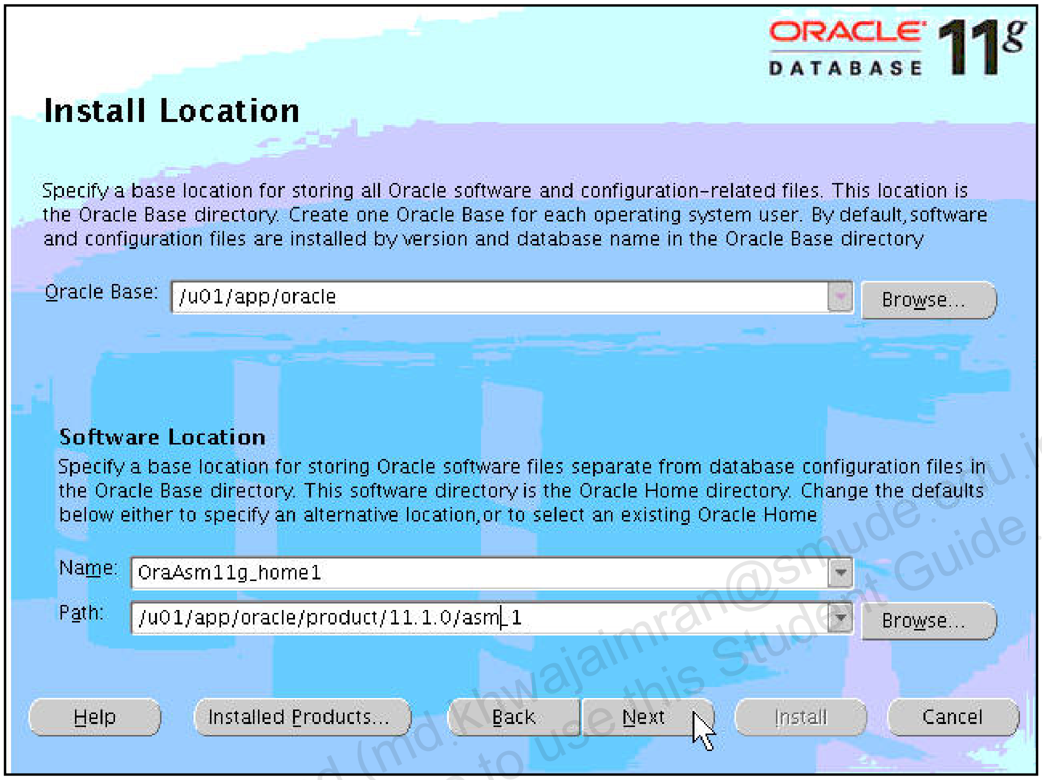
Installation Type



Installation Type

When the Installation Type screen appears, select your installation type by clicking the **Enterprise Edition** option button. Click the **Next** button to proceed.

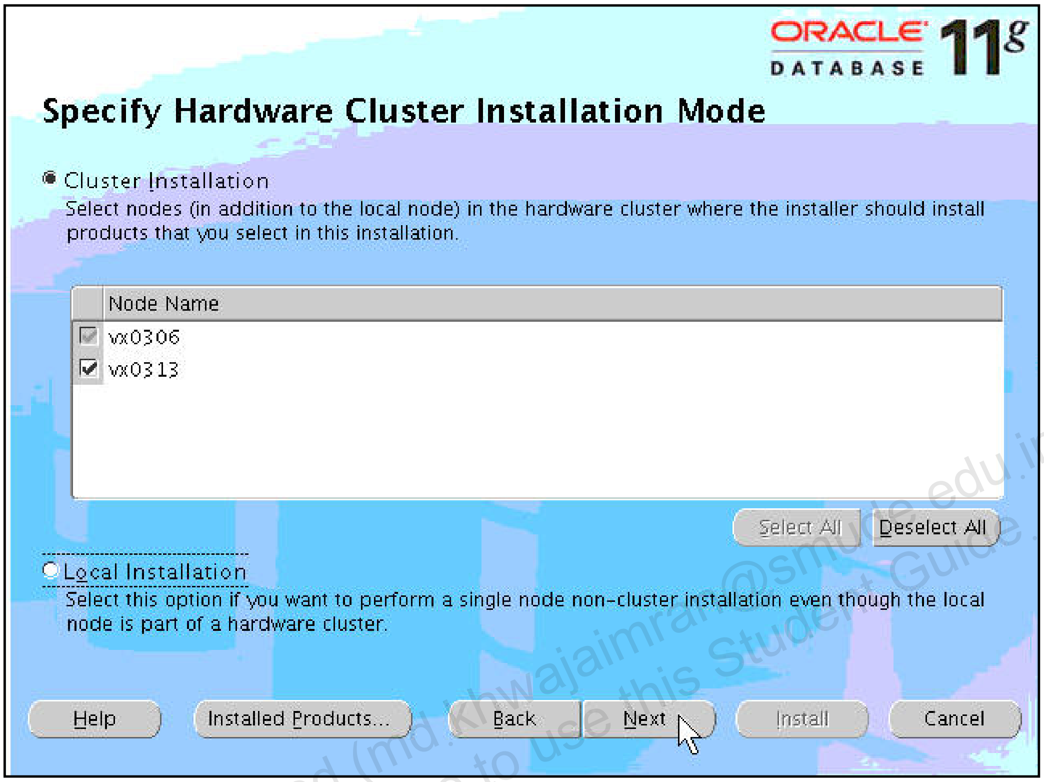
Install Location



Install Location

The next screen that appears is the "Install Location" screen. Here you specify the location of your ASM home directory and installation name. Although it is possible for ASM and the database installation to reside in the same directory and use the same files, you are installing ASM separately, into its own ORACLE_HOME to prevent the database `ORACLE_HOME` from being a point of failure for the ASM disk groups and to prevent versioning difficulties between the ASM and database file installations. Be sure to specify a name for your installation that reflects this. Then click the **Next** button to continue.

Hardware Cluster Installation Mode

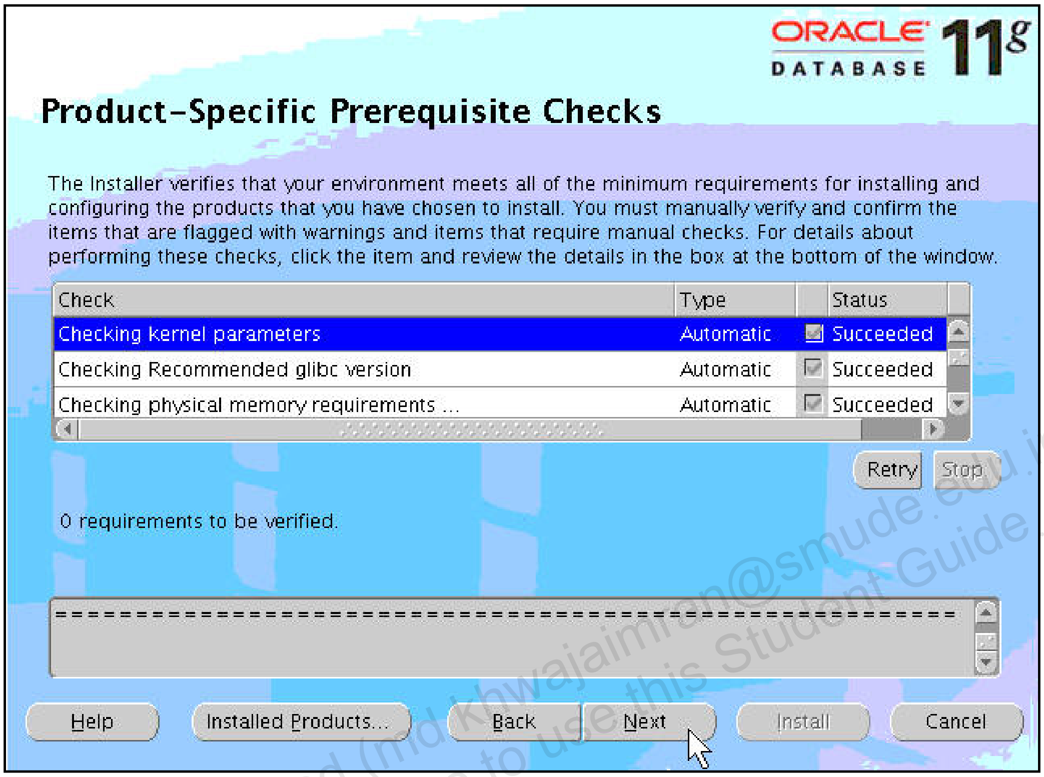


Hardware Cluster Installation Mode

When the Specify Hardware Cluster Installation Mode screen appears, click the Cluster Installation option button. Next, ensure that all nodes in your cluster are selected by clicking the Select All button. If OUI does not display the nodes properly, perform clusterware diagnostics by executing the `olsnodes -v` command from the `ORA_CRS_HOME/bin` directory, and analyze its output. Alternatively, you may use the `cluvfy` utility to troubleshoot your environment. Refer to your documentation if the detailed output indicates that your clusterware is not running properly.

When this is done, click the **Next** button to continue.

Product-Specific Prerequisite Checks

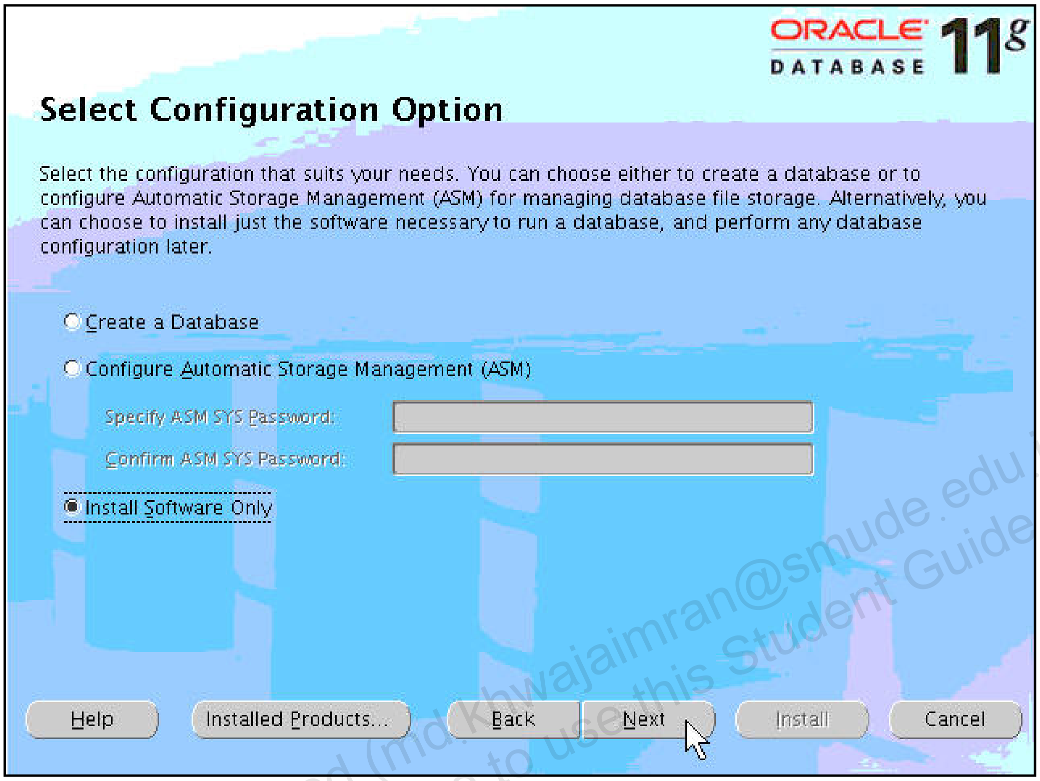


ORACLE

Product-Specific Prerequisite Checks

The Product-Specific Prerequisite Checks screen verifies the operating system requirements that must be met for the installation to be successful. After each successful check, the Succeeded check box is selected for that test. The test suite results are displayed at the bottom of the screen. Any tests that fail are also reported here. The example in the slide shows the results of a completely successful test suite. If you encounter any failures, try opening another terminal window and correct the deficiency from another terminal window. Then return to OUI, and click the Retry button to rerun the tests. It is possible to bypass the errors that are flagged by selecting the check box next to the error, but this is not recommended unless you are absolutely sure that the reported error will not affect the installation. When all tests have succeeded, click the **Next** button to continue.

Select Configuration Option



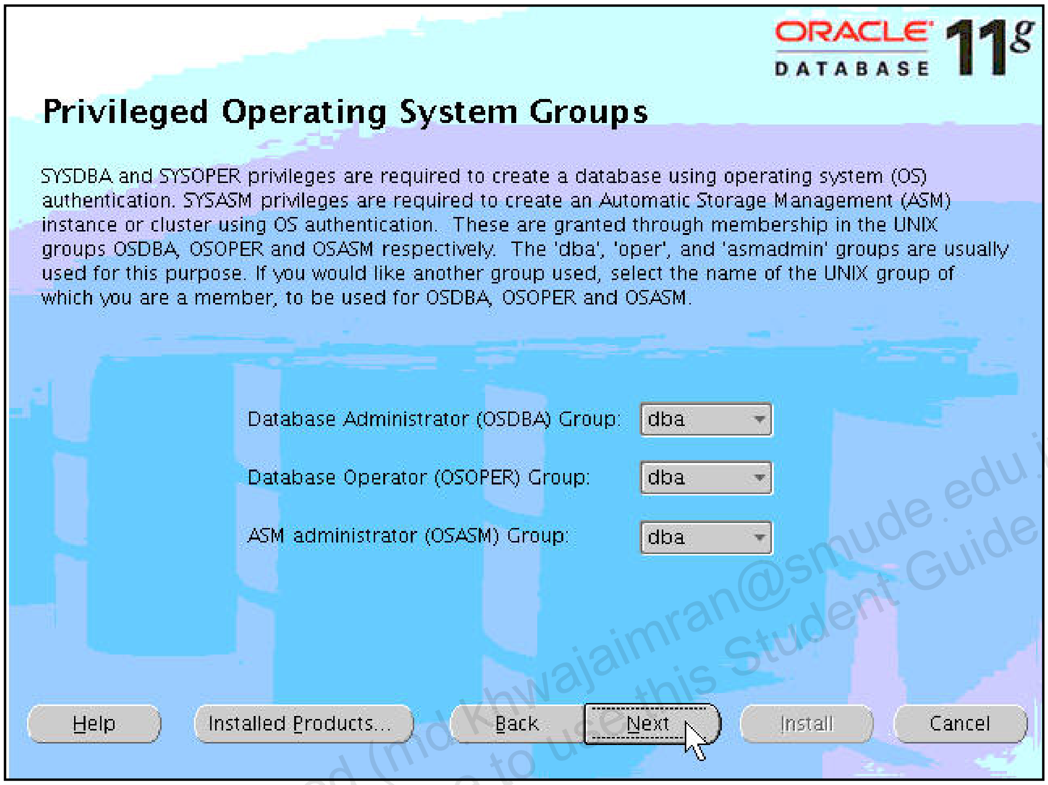
Select Configuration Option

The Select Configuration Option screen allows you to choose from the following options:

- Install database software and create a database
- Configure ASM
- Install database software only (no database creation)

This installation is concerned only with installing the ASM Home, so click the **Install Software Only button**. When you have done this, click the **Next** button to proceed.

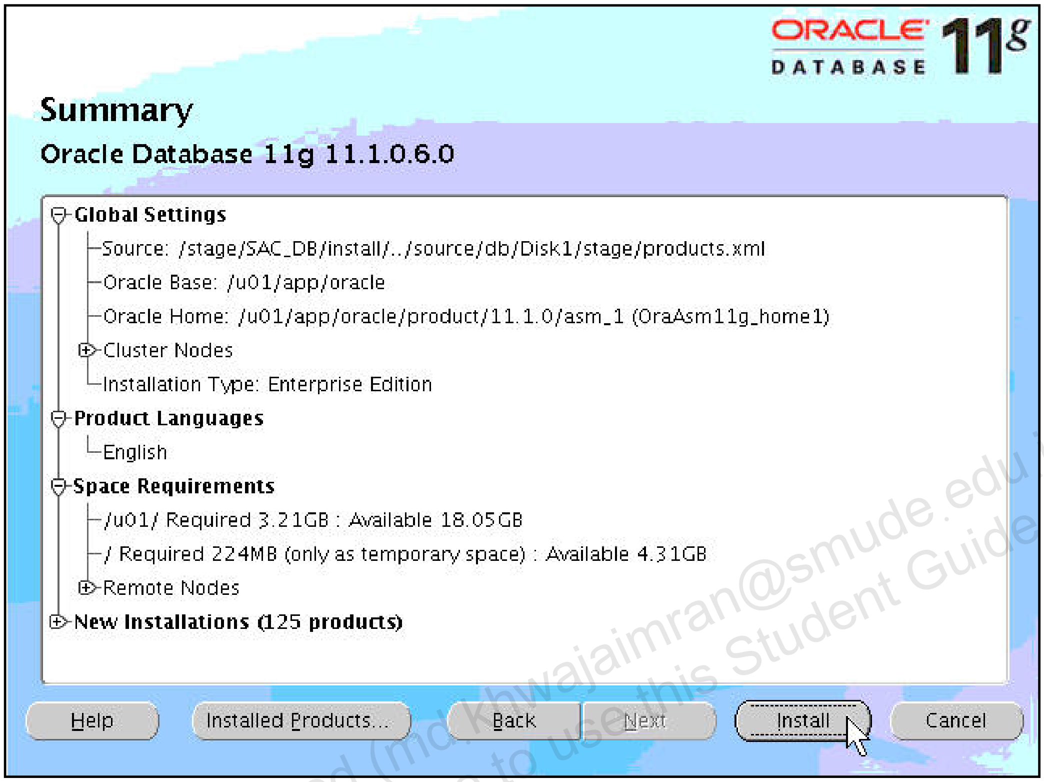
Privileged Operating System Groups



Privileged Operating System Groups

On the Privileged Operating Systems Groups page, choose the operating system groups corresponding to the Database trusted groups listed to support OS authentication. The default value is `dba` for each group: OSDBA, OSOPER, and OSASM. Click **Next** to continue.

Summary

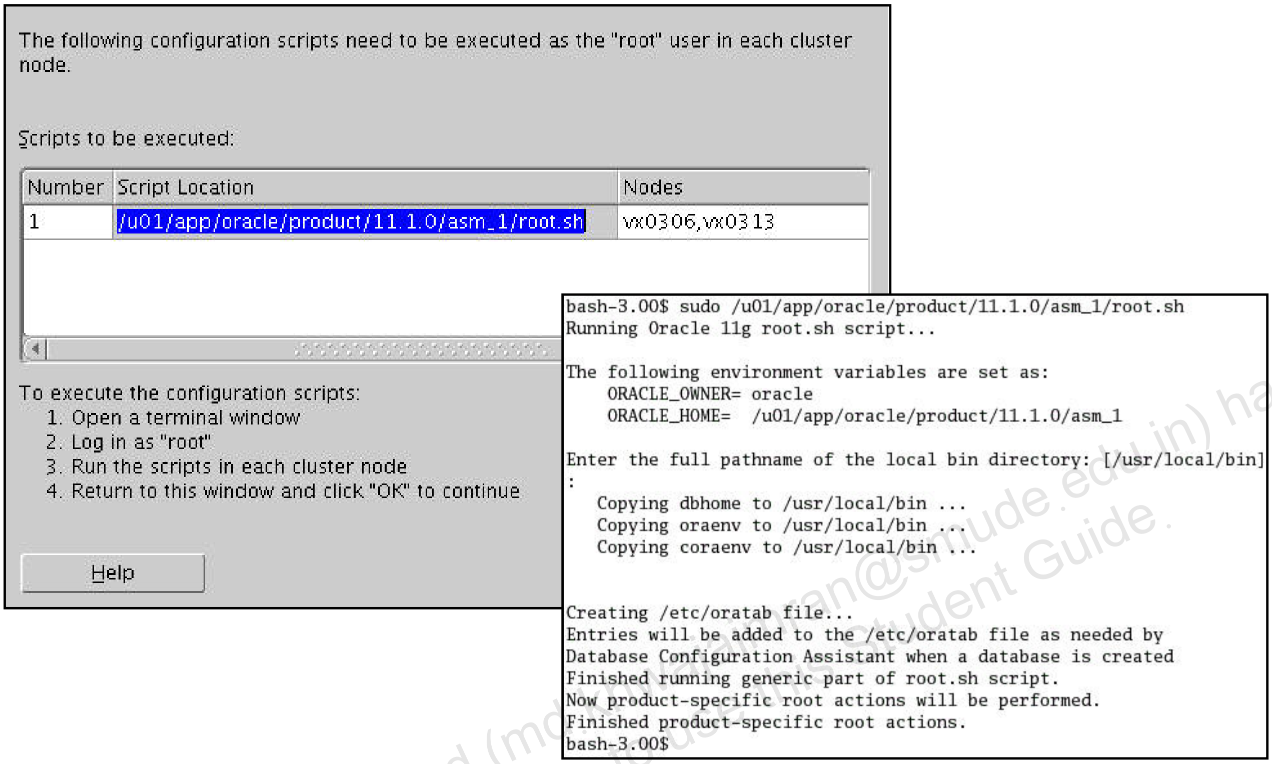


Summary

The Summary screen appears next. You may scan the installation tree to verify your choices if you like. Then click the **Install** button to proceed.

You can monitor the progress of the installation on the Install screen. After installing the files and linking the executables on the first node, the installer copies the installation to the remaining nodes. When the installation progress reaches 100%, OUI prompts you to execute configuration scripts on all nodes.

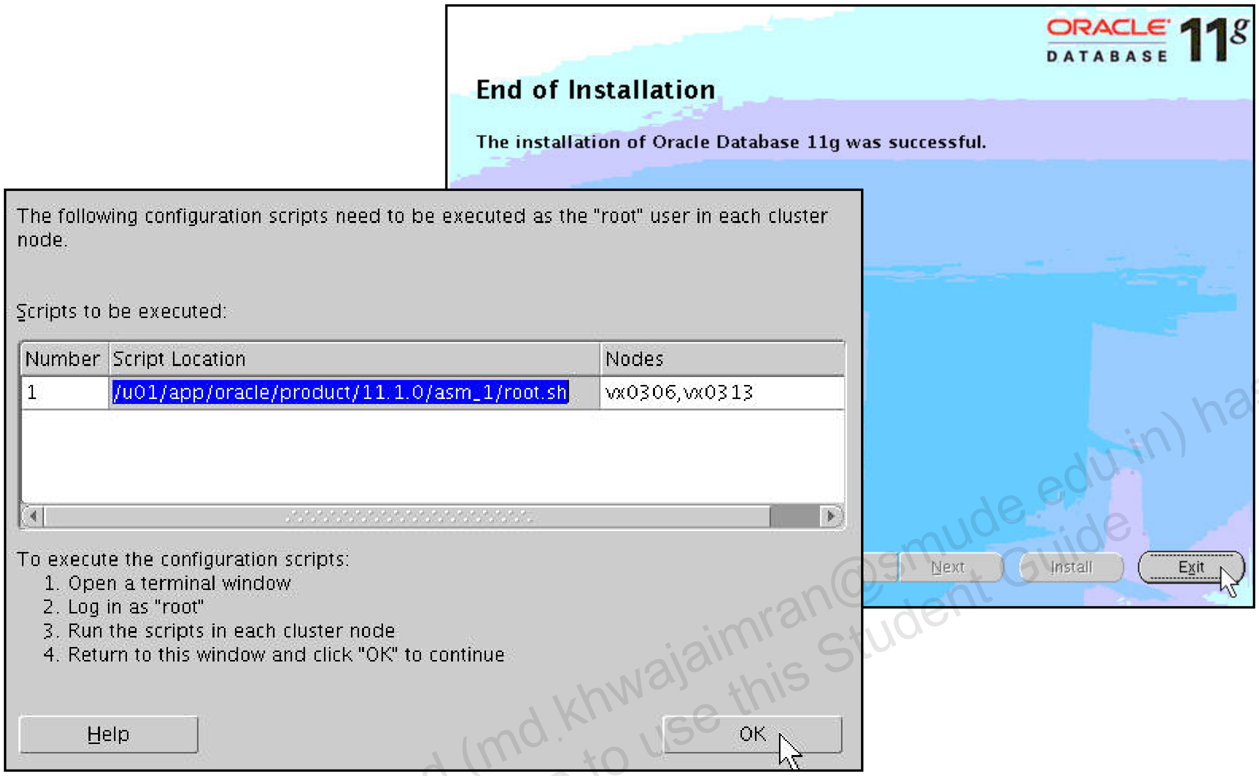
Execute Configuration Scripts



Execute Configuration Scripts

The next screen that appears prompts you to run the `root.sh` script on the specified nodes. Open a terminal window for each node listed and run the `root.sh` script as the `root` user from the specified directory.

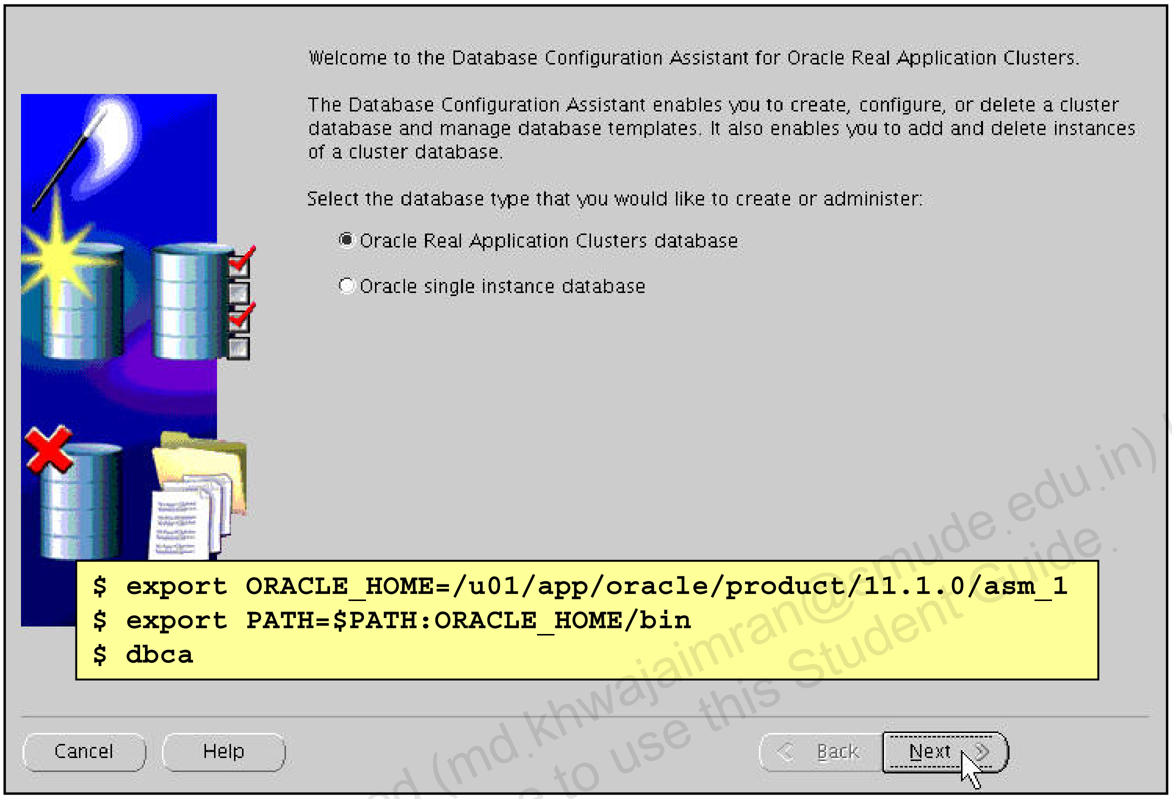
End of Installation



End of Installation

When the `root.sh` script has been executed on all nodes in the cluster, return to the Execute Configuration scripts window and click the **OK** button to continue. When the installation is finished, the End of Installation screen appears. Click the **Exit** button to quit.

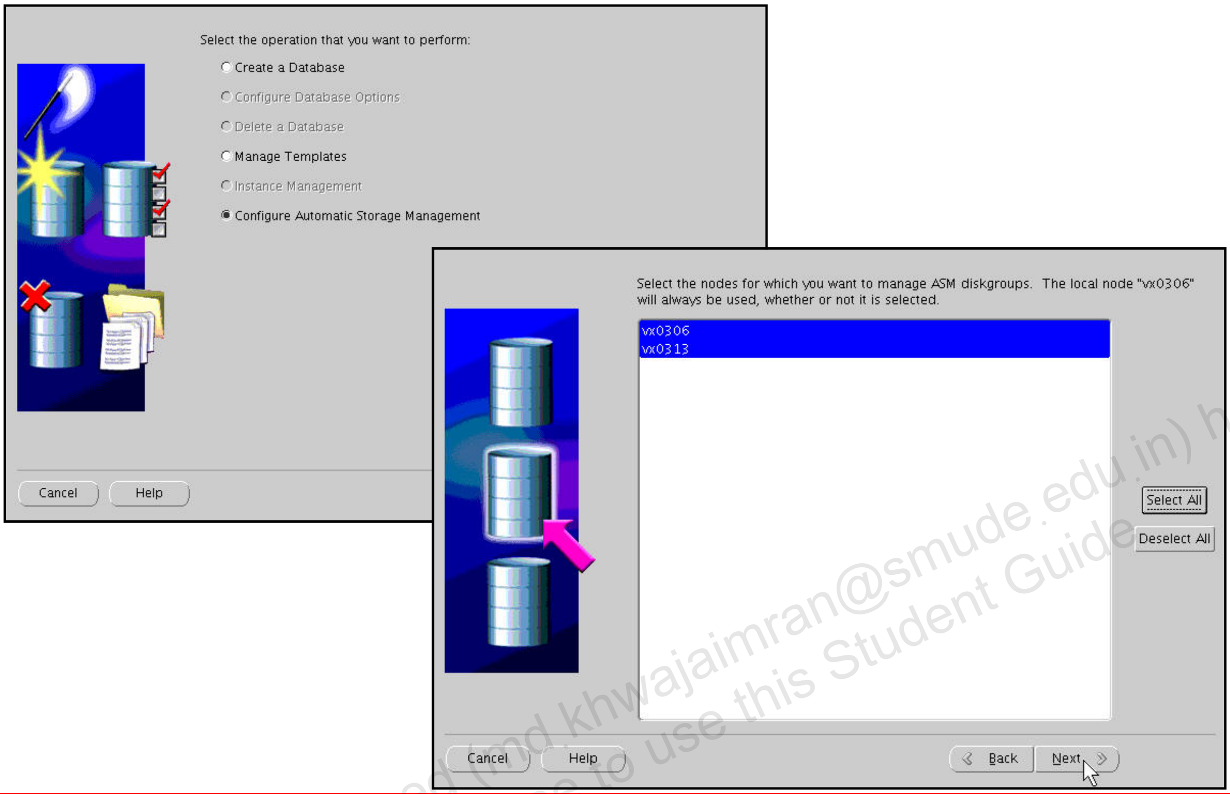
ASM Configuration



ASM Configuration

After the ASM software has been successfully installed, the ASM instances should be configured and started and disk groups should be created to support the shared storage needs of your cluster database. DBCA is used to do this quickly and accurately. Execute `dbca` from the ASM `ORACLE_HOME/bin` directory as shown in the slide. Select **Oracle Real Application Clusters database** from the Welcome screen and click **Next**.

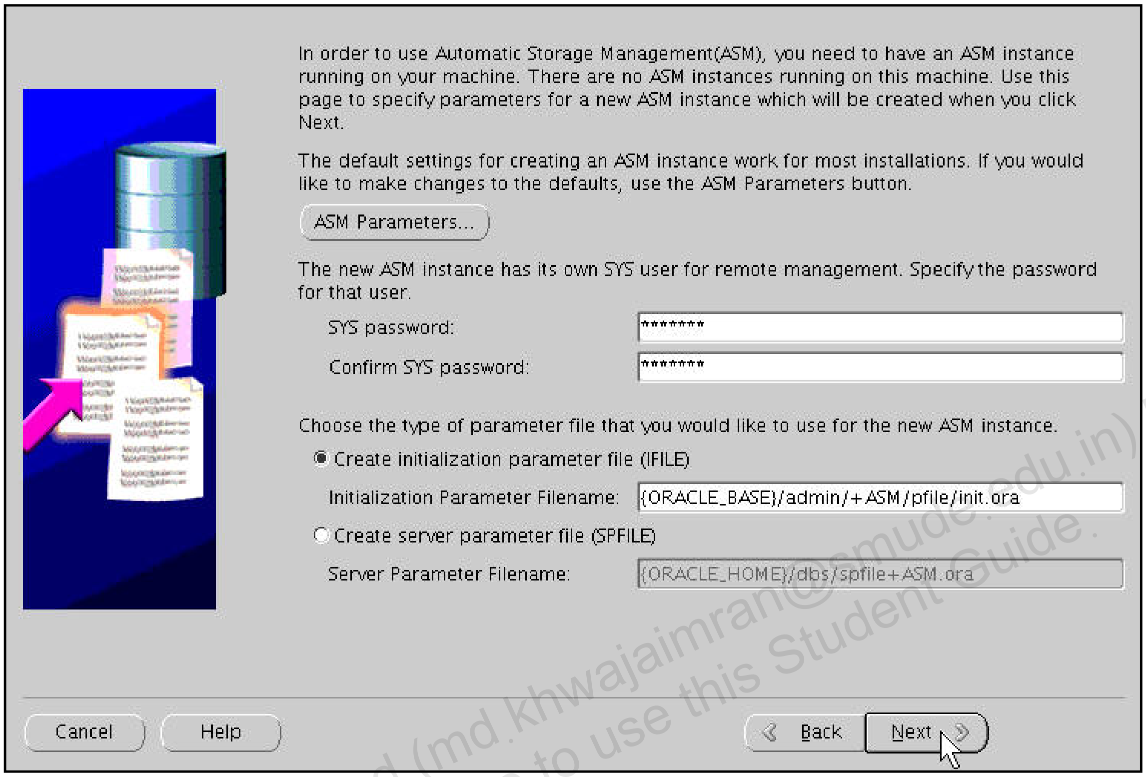
ASM Configuration



ASM Configuration (continued)

On the next screen, click **Configure Automatic Storage Management** and click the **Next** button. You now choose the nodes on which to manage ASM. Click the **Select All** button, and then click **Next** to continue.

ASM Configuration

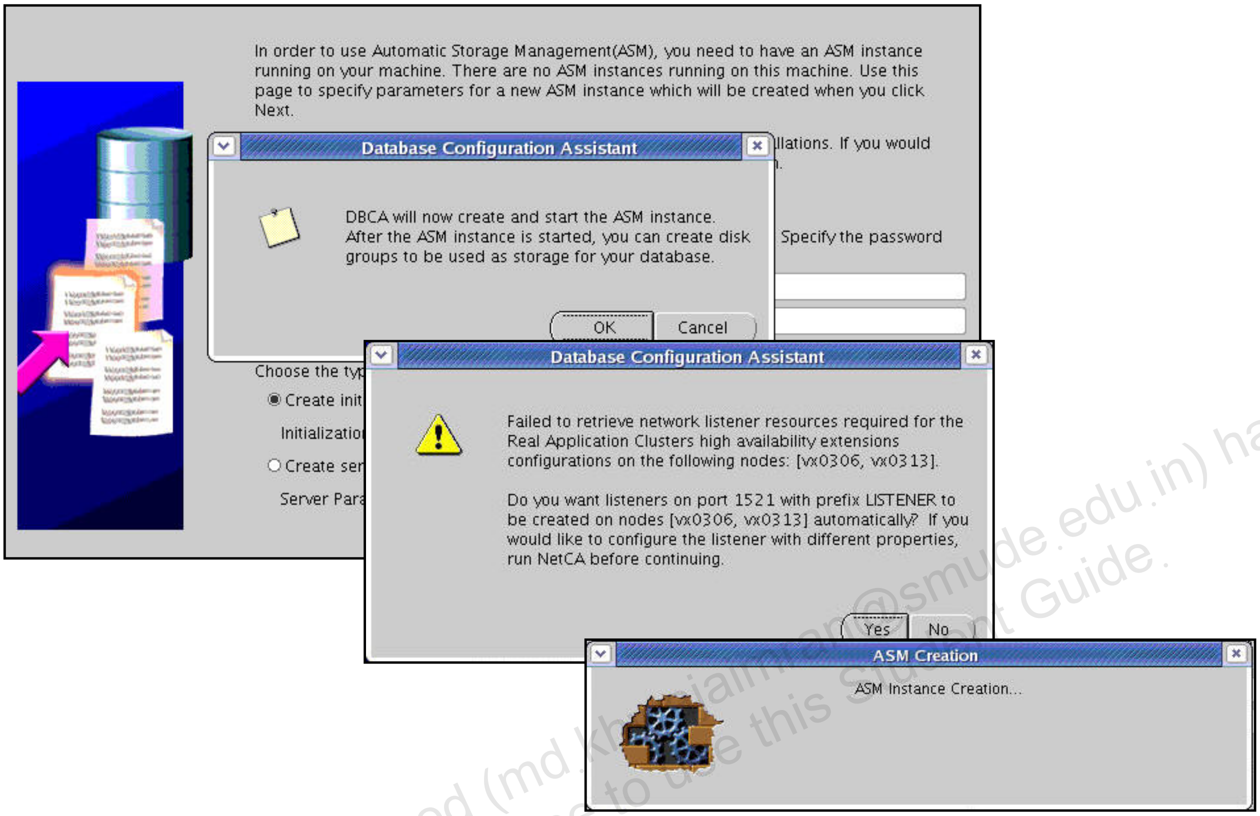


ORACLE

ASM Configuration (continued)

At this stage, no ASM instances are running on your cluster nodes. Before starting the instances, you must provide a password for the ASM `SYS` user. In addition, choose the type and location of the parameter file to be used for the ASM instances. If you require specific initialization parameter values to be set for your ASM instances, you can modify the default values by clicking the **ASM Parameters** button. Then click **Next** to continue.

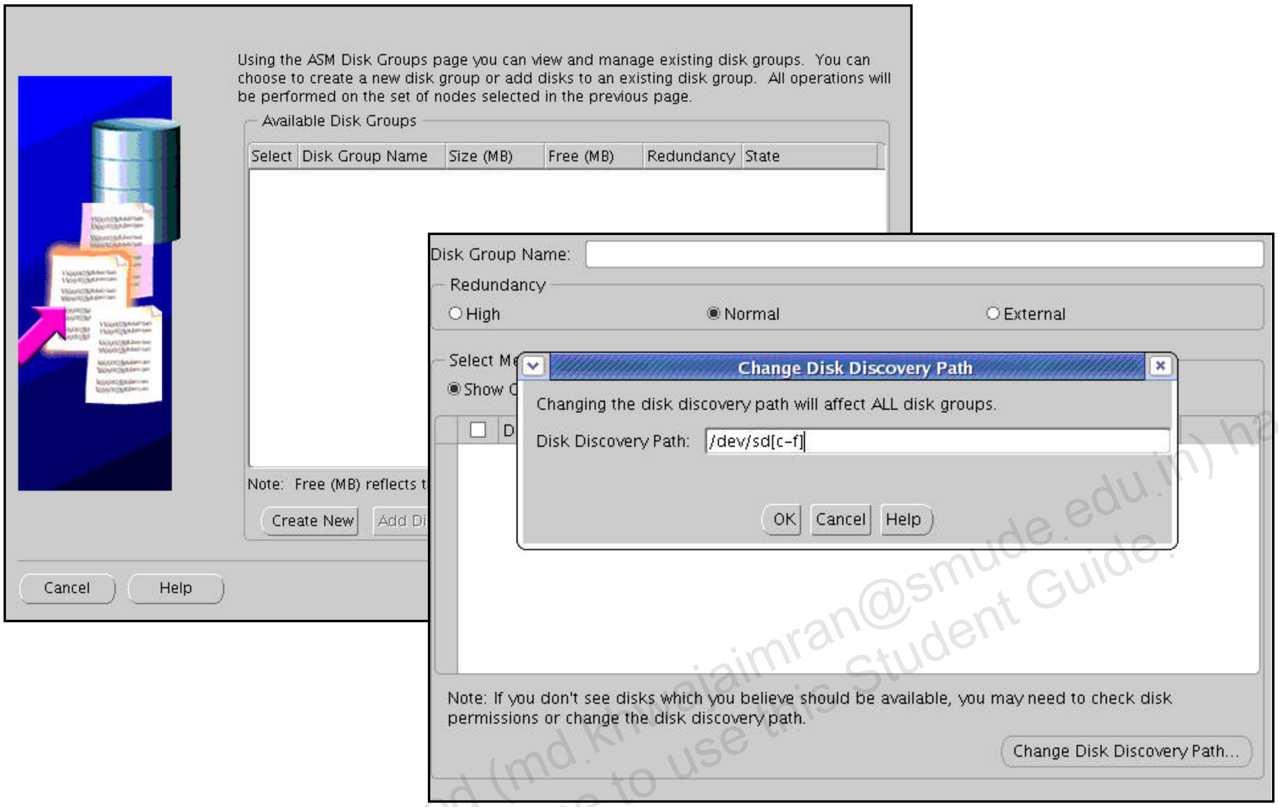
ASM Configuration



ASM Configuration (continued)

A dialog box informs you that listeners are not running on the cluster nodes and asks if you want to start them now. When you click **Yes** to start the listeners, the ASM instances are started on the cluster nodes.

Creating ASM Disk Groups

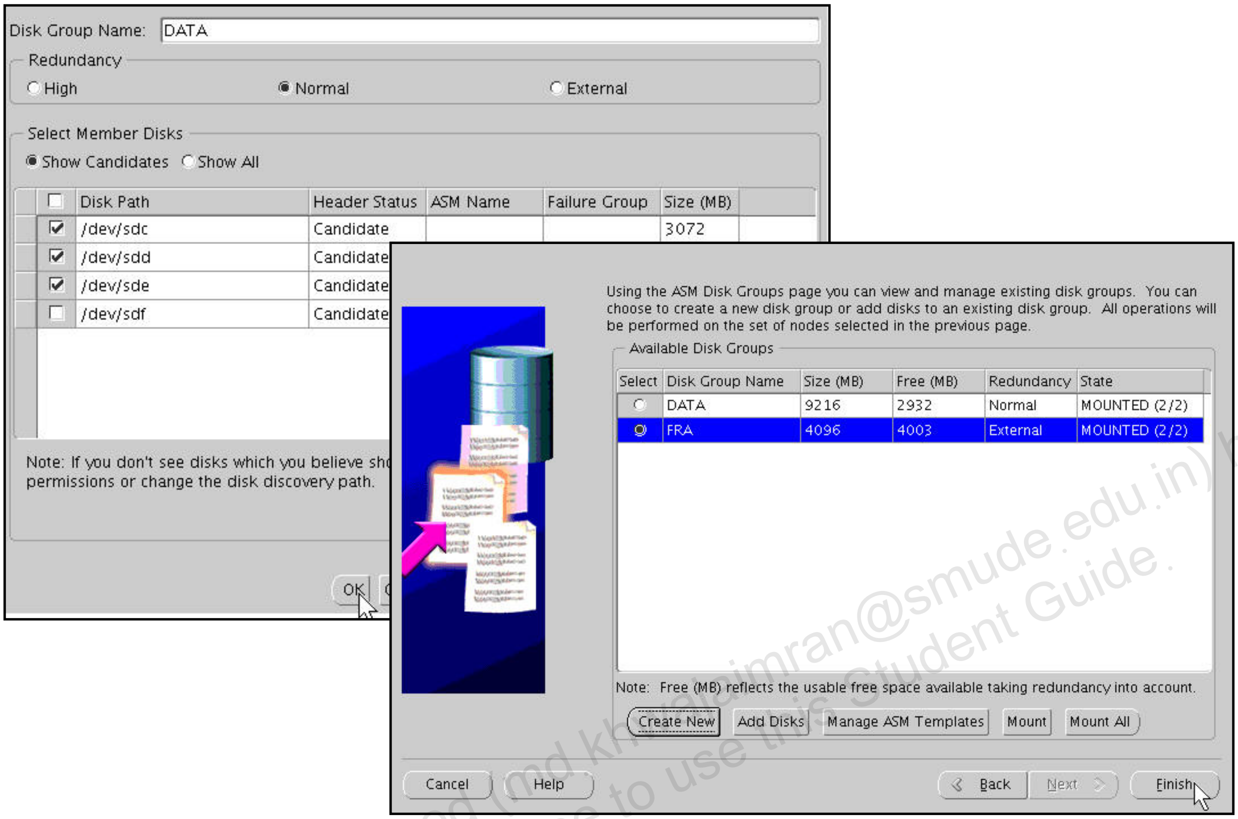


ORACLE

Creating ASM Disk Groups

When the instances are started, the ASM Disk Groups page is displayed. Click the **Create New** button to create a disk group. On the disk group creation page, click the **Change Disk Discovery Path** button to refine the search string that is used by DBCA to find candidate disk devices. Click the **OK** button after providing the discovery path to your disks.

Creating ASM Disk Groups



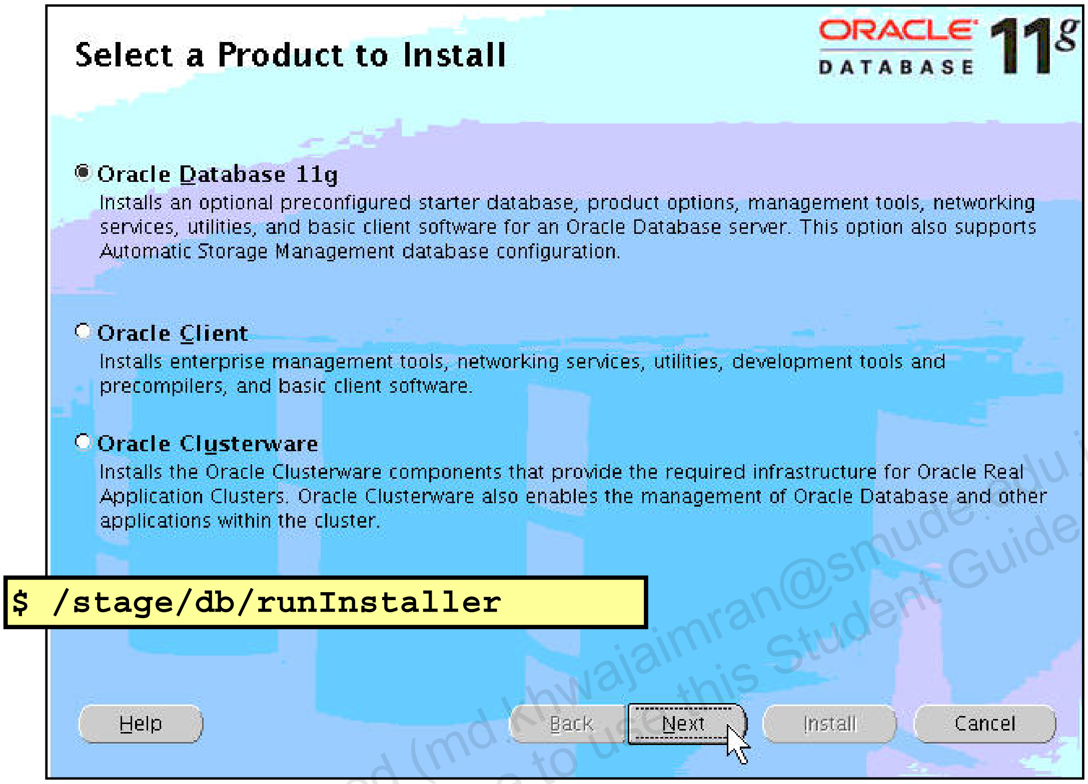
ORACLE

Creating ASM Disk Groups (continued)

A list of candidate disks and their relative size in megabytes is displayed. In the example in the slide, a DATA disk group is created using normal redundancy. When you click **OK**, the ASM disk group page is redisplayed showing the status of the newly created disk group. Other disk groups can also be created at this point.

When you have finished creating all the necessary disk groups, click **Finish**. A dialog box asks if you wish to perform other operations. Click **No** to exit DBCA.

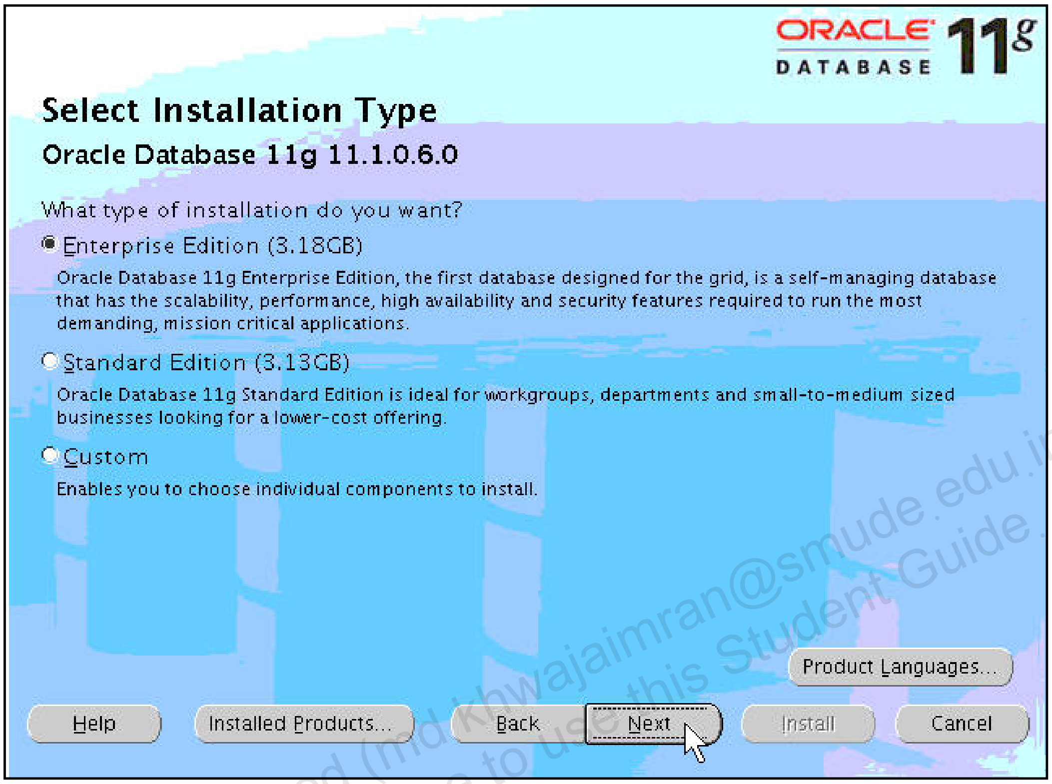
Select a Product to Install



Select a Product to Install

The OUI is used to install the Oracle Database 11*g* software. You need to run OUI as the `oracle` user. Start OUI by executing the `runInstaller` command from the `root` directory of the Oracle Database 11*g* Release 1 CD-ROM or the software staging location. The “Select a Product to Install” screen allows you to install and create a database, install Oracle client software, or install Oracle Clusterware. Click the **Oracle Database 11g** button and click **Next**.

Select Installation Type



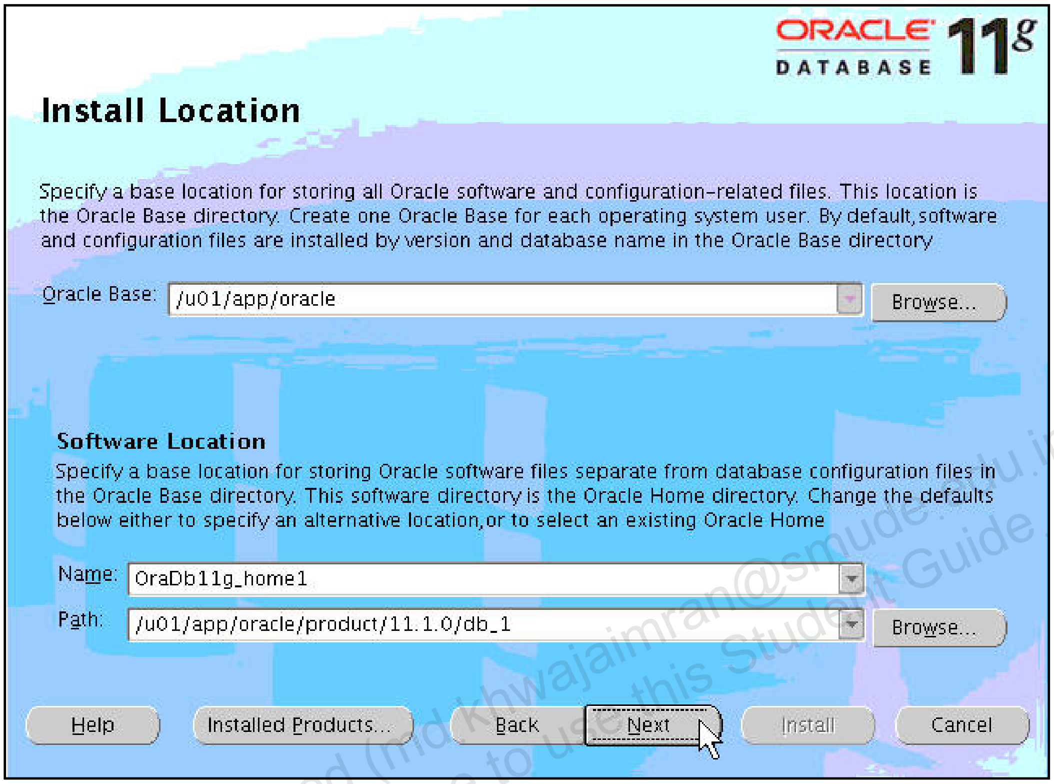
Select Installation Type

The Select Installation Type screen is displayed next. Your installation options include:

- Enterprise Edition
- Standard Edition
- Custom

For most installations, the Enterprise Edition installation is the correct choice (but Standard Edition is also supported). Selecting the Custom installation type option enables you to install only those Oracle product components that you deem necessary. For this, you must have a good knowledge of the installable Oracle components and of any dependencies or interactions that may exist between them. For this reason, it is recommended that you select the Enterprise Edition installation because it installs all components that are part of the Oracle Database 11*g* 11.1.0 distribution.

Install Location

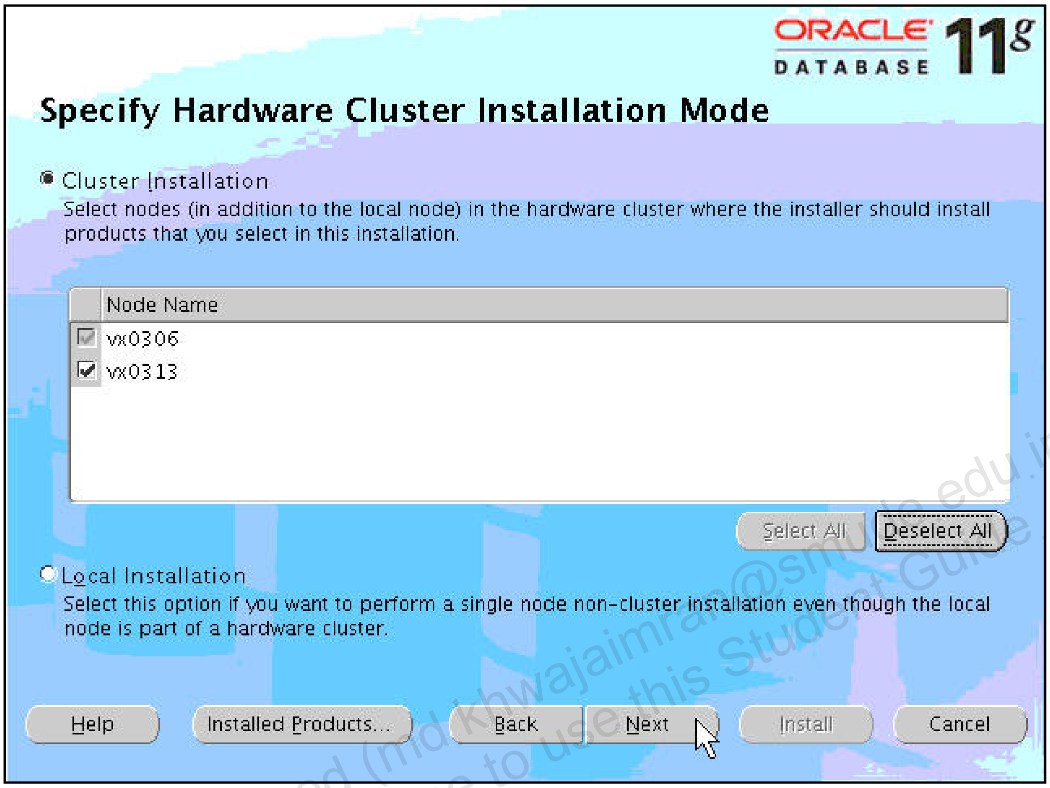


Install Location

On the Install Location screen, provide a value for ORACLE_BASE if you have not yet already done so. In the Software Location section of the page

The Name field is populated with a default or suggested installation name. Accept the suggested name or enter your own Oracle Home name. Next, in the Path field, enter the fully qualified path name for the installation, `/u01/app/oracle/product/11.2.0/db_1` in the example in the slide. After entering the information, review it for accuracy, and click the **Next** button to continue.

Specify Cluster Installation



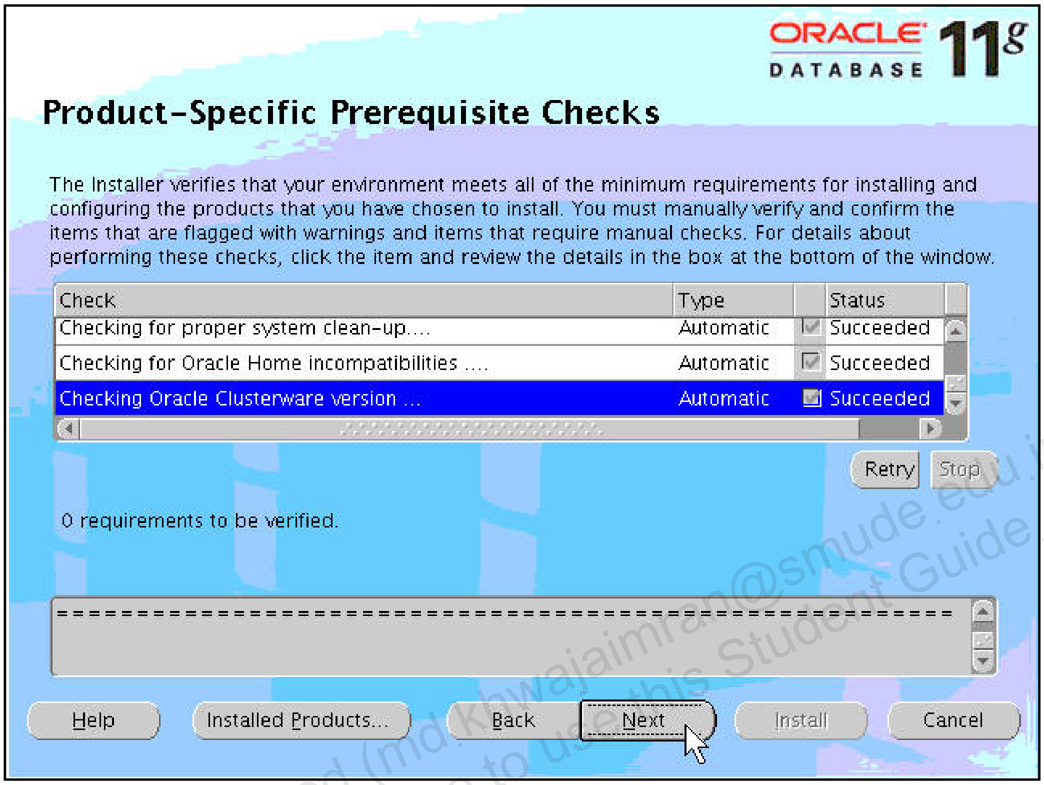
ORACLE

Specify Cluster Installation

The Specify Hardware Cluster Installation Mode screen is displayed next. Because OUI is node aware, you must indicate whether you want the installation to be copied to the recognized and selected nodes in your cluster, or whether you want a single, noncluster installation to take place. Most installation scenarios require the Cluster Installation option.

To do this, click the Cluster Installation option button and make sure that all nodes have been selected in Node Name list. Note that the local node is always selected for the installation. Additional nodes that are to be part of this installation must be selected by selecting the check boxes. If you do not see all your nodes listed here, exit OUI and make sure that Oracle Clusterware is running on all your nodes. Restart OUI. Click the **Next** button when you are ready to proceed with the installation.

Products Prerequisite Check



Products Prerequisite Check

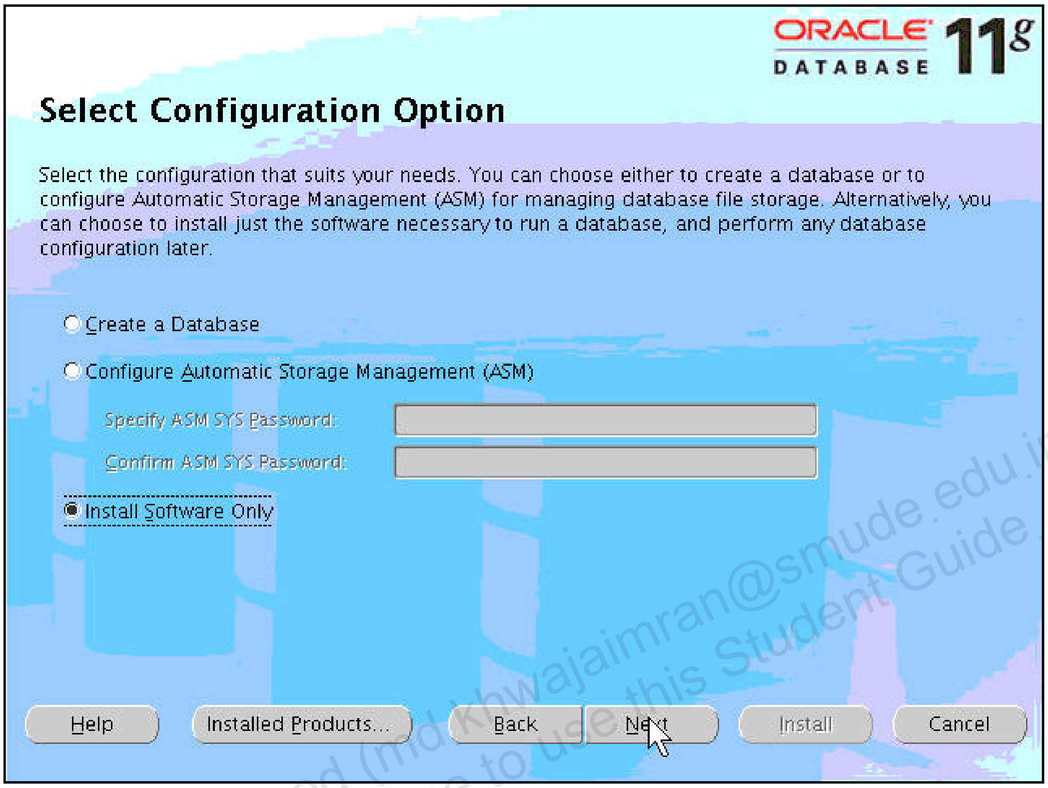
The Product-Specific Prerequisite Checks screen verifies the operating system requirements that must be met for the installation to be successful. These requirements include:

- Certified operating system check
- Kernel parameters as required by the database software
- Required operating system packages and correct revisions
- Required `glibc` and `glibc-compat` (compatibility) package versions

In addition, OUI checks whether the `ORACLE_BASE` user environment variable has been set and, if so, whether the value is acceptable.

After each successful check, the Succeeded check box is selected for that test. The test suite results are displayed at the bottom of the page. Any tests that fail are also reported here. The example in the slide shows the results of a completely successful test suite. If you encounter any failures, try opening another terminal window and correct the deficiency. For example, if your `glibc` version is too low, acquire the correct version of the `glibc` RPM, install it from another terminal window, return to OUI, and click the **Retry** button to rerun the tests. When all tests have succeeded, click the **Next** button to continue.

Select Configuration Option



Select Configuration Option

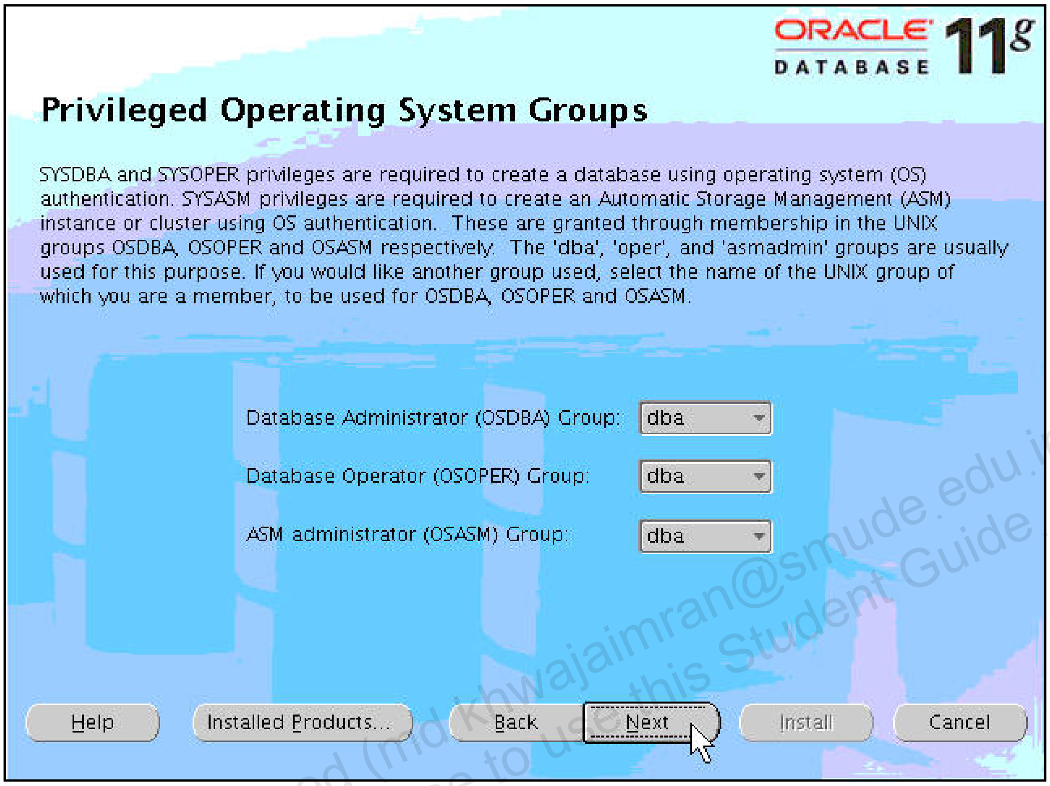
The Select Configuration Option screen appears. On this screen, you can choose to create a database as part of the database software installation or install ASM. If you choose to install a database, you must select one of the preconfigured starter database types:

- General Purpose
- Transaction Processing
- Data Warehouse
- Advanced (user customizable)

If you choose one of these options, you are queried about the specifics of your database (cluster database name, shared storage options, and so on). After OUI stops, the DBCA is launched to install your database with the information that you provided.

You may also choose to defer the database creation by clicking the **Install Software Only** option button. This option enables you to create the database by manually invoking the DBCA at some point in time after OUI finishes installing the database software. This choice provides you with more options than the standard preconfigured database models. Select the **Install Software Only** option. Click the **Next** button to continue.

Privileged Operating System Groups

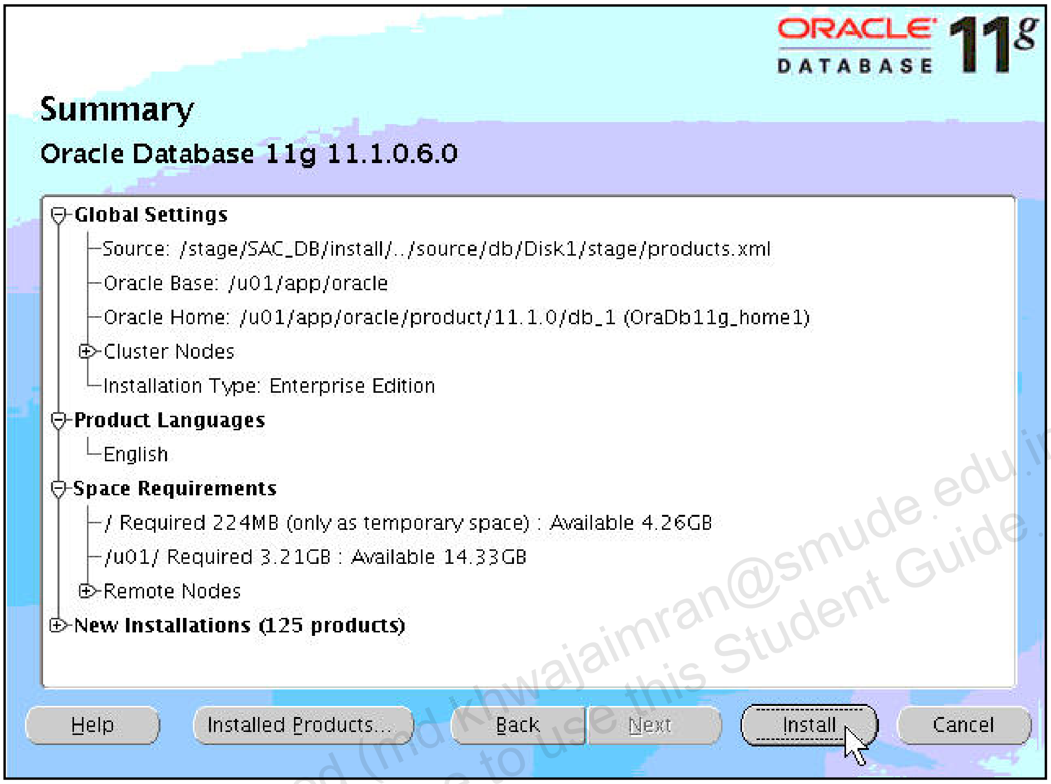


ORACLE

Privileged Operating System Groups

On the Privileged Operating Systems Groups page, choose the operating system groups corresponding to the Database trusted groups listed to support OS authentication. The default value is dba for the OSDBA, OSOPER, and OSASM groups respectively. Click **Next** to continue.

Check Summary

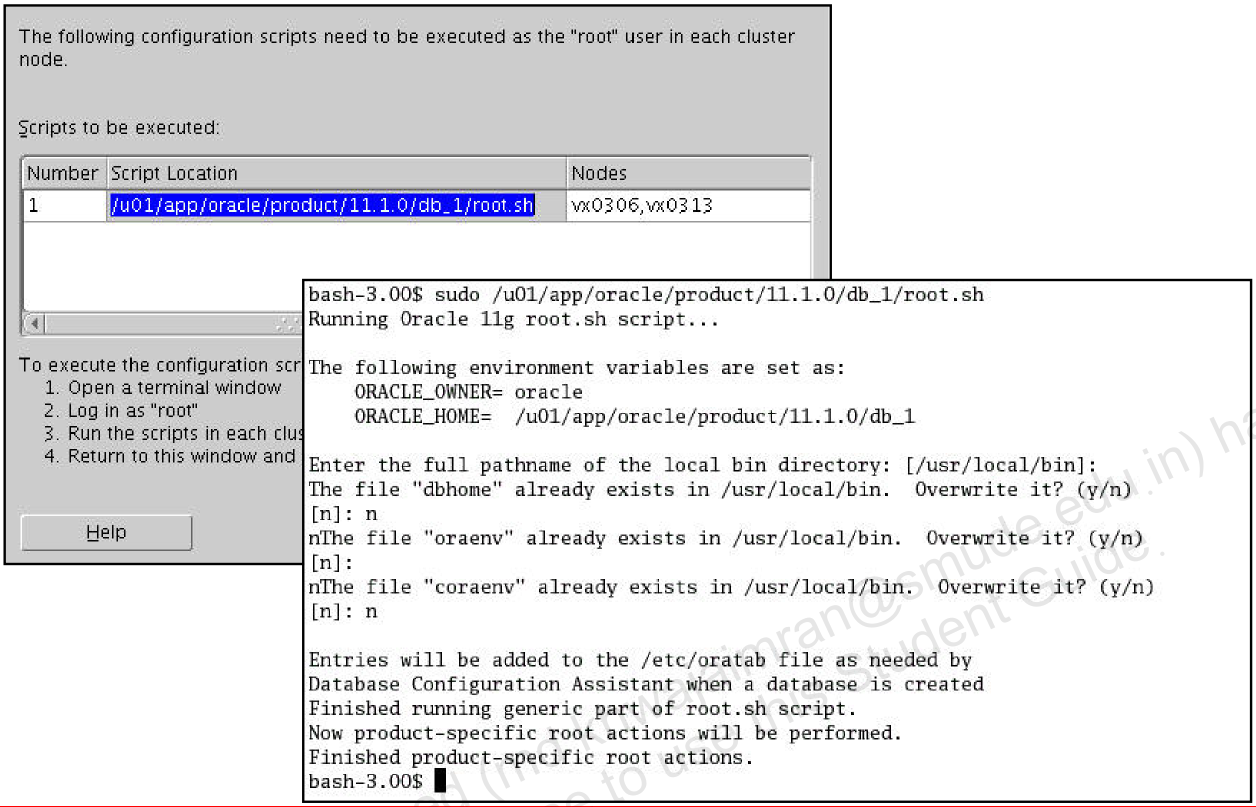


Check Summary

The Summary screen is displayed next. Review the information on this page. Node information and space requirements can be viewed here, as well as selected software components. If you are satisfied with the summary, click the Install button to proceed. If you are not, click the Back button to go back and make the appropriate changes.

On the Install screen, you can monitor the progress of the installation. During installation, OUI copies the software first to the local node and then to the remote nodes.

`root.sh` Script



`root.sh` script

At the end of the installation, OUI displays a dialog box indicating that you must run the `root.sh` script as the `root` user on all the nodes where the software is being installed. Execute the `root.sh` script on one node at a time, and then click the **OK** button to continue.

Required Tasks Prior to Database Creation

Set the environment variables for the Oracle database:

```
$ cd
$ vi .bash_profile
export ORACLE_BASE=/u01/app/oracle
export ORACLE_SID=RDB1
export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1;

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


Required Tasks Prior to Database Creation

You can now set the Oracle database–related environment variables for the `oracle` user so that they are recognized by the DBCA during database creation:

```
$ cd
$ vi .bash_profile
export ORACLE_BASE=/u01/app/oracle
export ORACLE_SID=RDB1
export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1
export PATH=$PATH:$ORACLE_HOME/bin
```

Checks Before Database Creation

- Use `cluvfy` to perform a configuration check before database creation.
- Use the `-pre` option with the `dbcfg` argument when executing `runcluvfy`.
- As `oracle`, run the command as shown below:

```
$ cd /stage/db
$ ./runcluvfy.sh stage -pre dbcfg -n vx0306,vx0313 -d
/u01/app/oracle/product/11.1.0
```

- Set the Oracle database–related environment variables for the `oracle` user.

Checks Before Database Creation

The Cluster Verify Utility can be used for the configuration check prior to database creation. Run the Cluster Verify Utility as the `oracle` user. Use the `–pre` option with the `dbcfg` argument to thoroughly analyze your cluster before creating your RAC database. Your output should be similar to the following example:

```
$ ./runcluvfy.sh stage -pre dbcfg -n vx0306,vx0313 -d
/u01/app/oracle/product/11.1.0

Performing pre-checks for database configuration

Checking node reachability...
Node reachability check passed from node "vx0306".

Checking user equivalence...
User equivalence check passed for user "oracle".

Checking administrative privileges...
User existence check passed for "oracle".
Group existence check passed for "dba".
Membership check for user "oracle" in group "dba" [as Primary] passed.
```

Checks Before Database Creation (continued)

```
Administrative privileges check passed.

Checking node connectivity...

Node connectivity check passed for subnet "138.2.204.0" with node(s)
vx0313,vx0306.
Node connectivity check passed for subnet "10.0.0.0" with node(s)
vx0313,vx0306.

Node connectivity check passed for subnet "192.168.255.0".

Suitable interfaces for VIP on subnet "138.2.204.0":
vx0313 eth0:138.2.204.33 eth0:138.2.205.171
vx0306 eth0:138.2.204.32 eth0:138.2.205.170

Suitable interfaces for the private interconnect on subnet "10.0.0.0":
vx0313 eth1:10.0.0.2
vx0306 eth1:10.0.0.1

Node connectivity check passed.

Checking CRS integrity...
Checking daemon liveness...
Liveness check passed for "CRS daemon".
Checking daemon liveness...
Liveness check passed for "CSS daemon".
Checking daemon liveness...
Liveness check passed for "EVM daemon".

Checking CRS health...
CRS health check passed.

CRS integrity check passed.

Pre-check for database configuration was successful.
```

Summary

In this lesson, you should have learned how to:

- Install and configure Automatic Storage Management (ASM)
- Install the Oracle database software
- Perform required tasks prior to database creation

Practice 2: Overview

This practice covers the following topics:

- Installing and configuring ASM
- Installing the Oracle database software

ORACLE

3

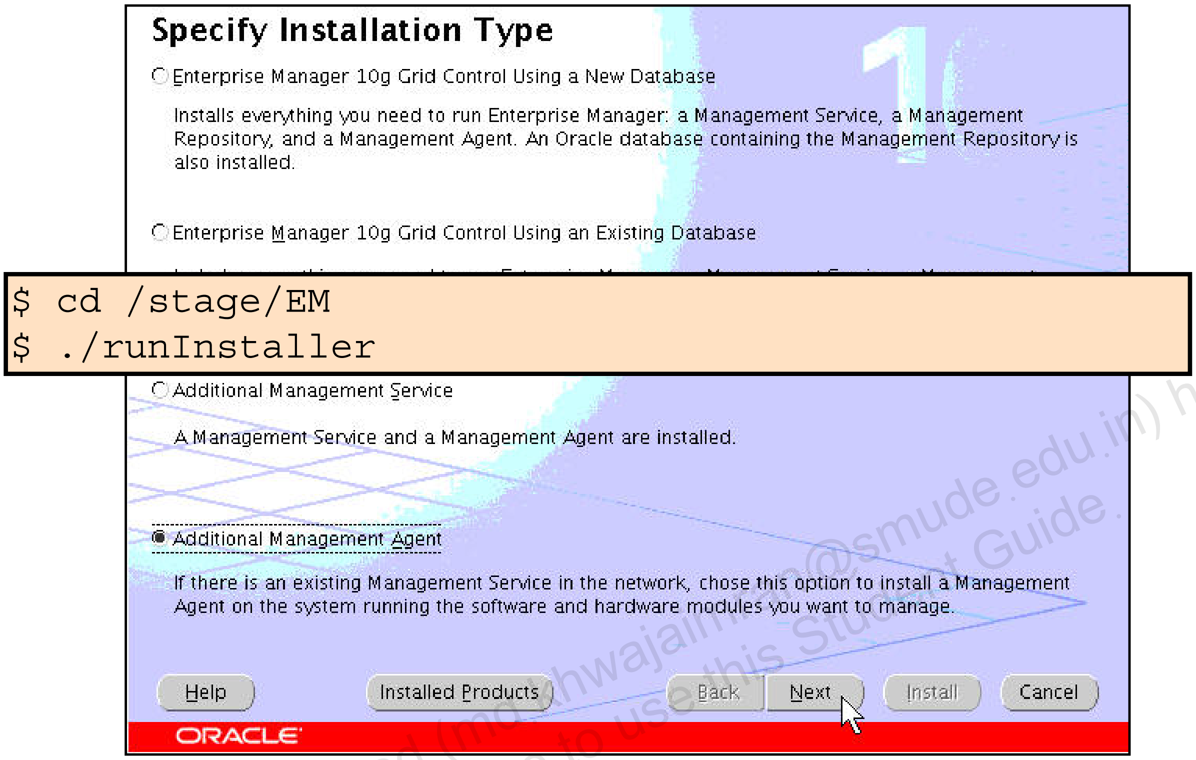
RAC Database Creation

Objectives

After completing this lesson, you should be able to:

- Install the Enterprise Manager agent on each cluster node
- Create a cluster database
- Perform required tasks after database creation

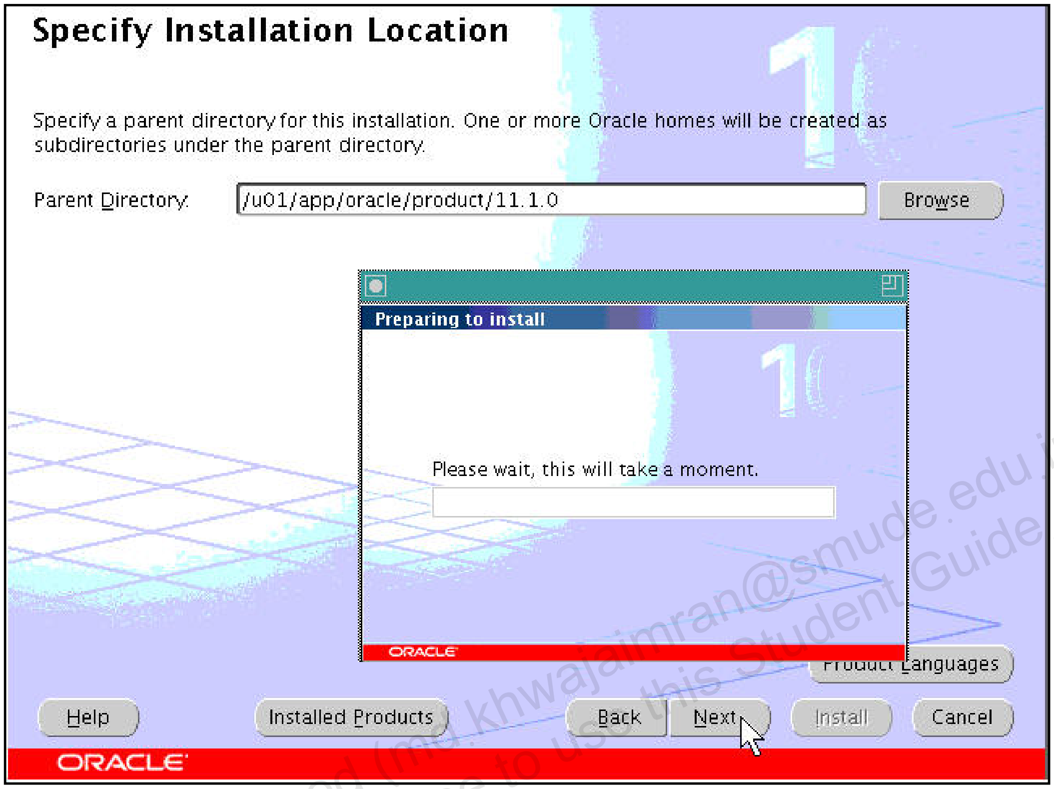
Management Agent Installation: Specify Installation Type



Management Agent Installation: Specify Installation Type

There are two management tools available for your cluster database, Database Control and Grid Control. Both tools are based on Enterprise Manager, but Grid Control is the preferred tool for deploying and managing cluster databases in an enterprise setting. To use Grid Control, the Management Agent must be installed on each managed node in your cluster. To install the Management Agent, go to the Enterprise Manager Installation CD or a software staging area and start the Oracle Universal Installer. The Installation page provides several install types from which to choose from. Click the **Additional Management Agent** option button to install an agent only.

Specify Installation Location

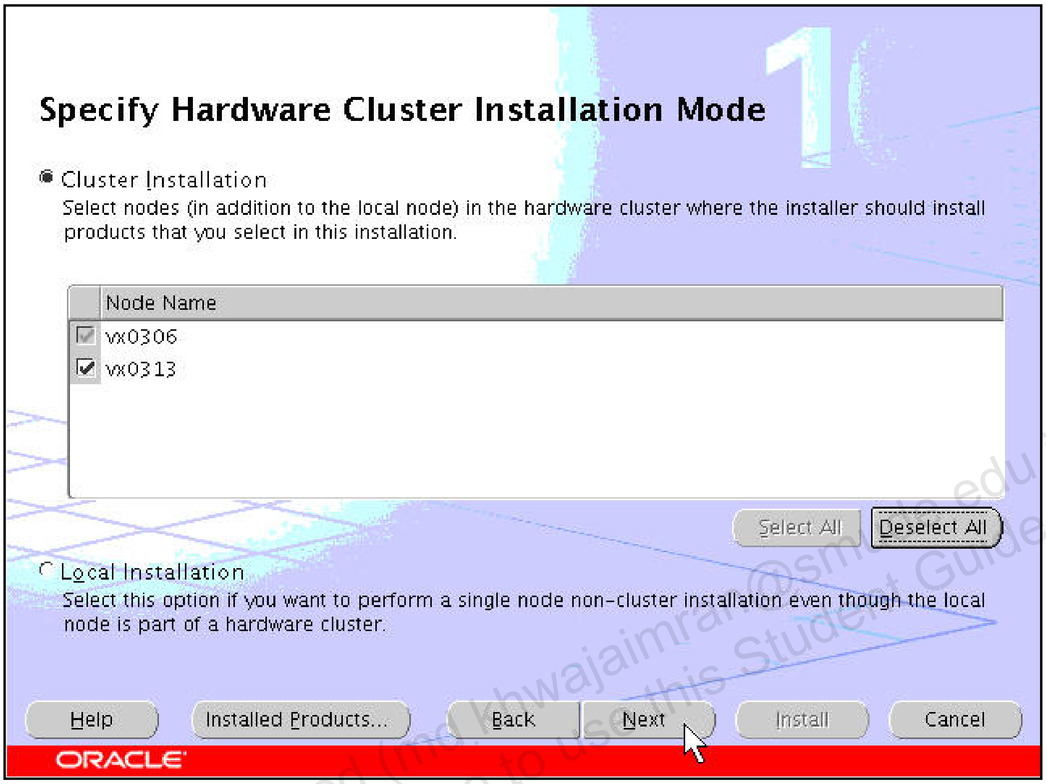


Specify Installation Location

The next page is the Specify Installation Location page. Because this cluster database installation uses an `ORACLE_BASE` set to `/u01/app/oracle`, the agent software should be installed in a subdirectory under `ORACLE_BASE`, with the other Oracle Homes. After starting the installation and specifying the installation location as shown in the example in the slide, the `11.1.0` subdirectory looks like this:

```
$ pwd
/u01/app/oracle/product/11.1.0
$ ls -l
total 20
drwxrwx---    3 oracle   oinstall          4096 Nov 12 07:59 agent10g
drwxr-x---   54 oracle   oinstall          4096 Nov 12 07:00 asm_1
drwxr-x---   54 oracle   oinstall          4096 Nov 12 07:16 db_1
```

Specify Cluster Installation Mode

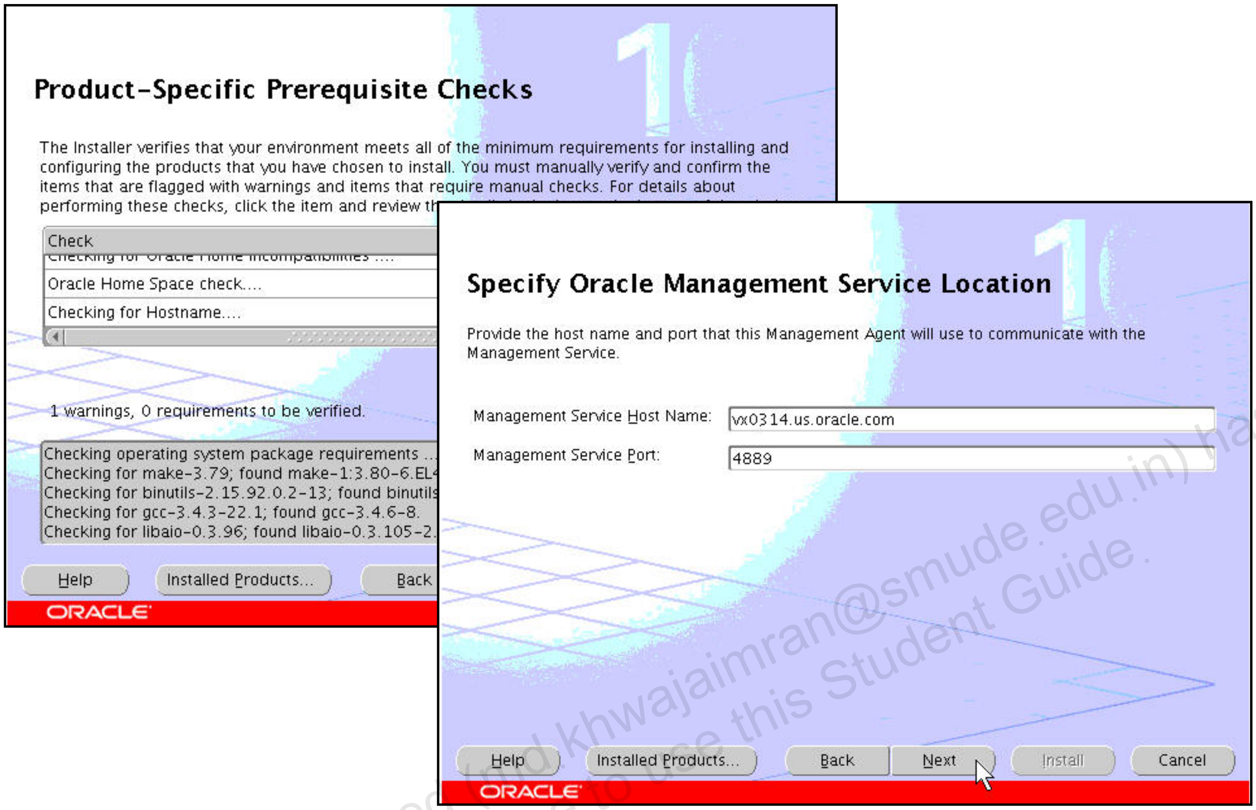


ORACLE

Specify Cluster Installation Mode

The Specify Cluster Installation Mode screen is displayed next. Because you want to install the agent on all cluster nodes, click the **Select All** button to choose all the nodes of the cluster. Each node must be check marked before continuing. If all the nodes do not appear, you must stop the installation and troubleshoot your environment. If no problems are encountered, click the **Next** button to proceed.

Prerequisite Check and OMS Location



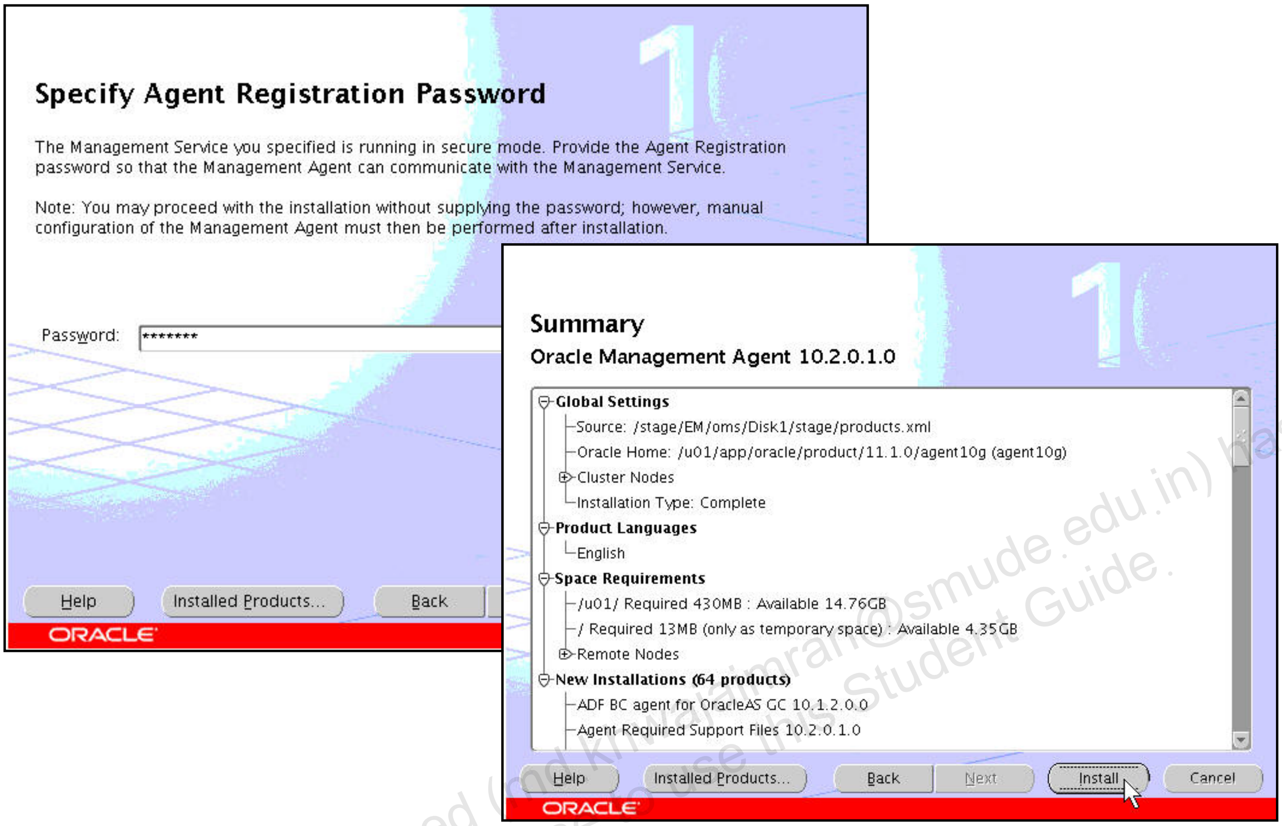
ORACLE

Prerequisite Check and OMS Location

The installer next checks the minimum requirements for the installation of the agent software. Any deficiencies are reported. You should take care of any issues identified by the installer before continuing with the installation. If the prerequisite check identifies no problems, click **Next** to continue.

The next screen requires you to identify the location of the Oracle management service. You must provide the host name and port number of an existing Grid Control console. In the example in the slide, a previously installed Grid Control console is specified on the host `ex0043.us.oracle.com`. The port number specified is the default port (4889) used by Grid Control. After you complete this step, click the **Next** button to continue.

Agent Registration Password



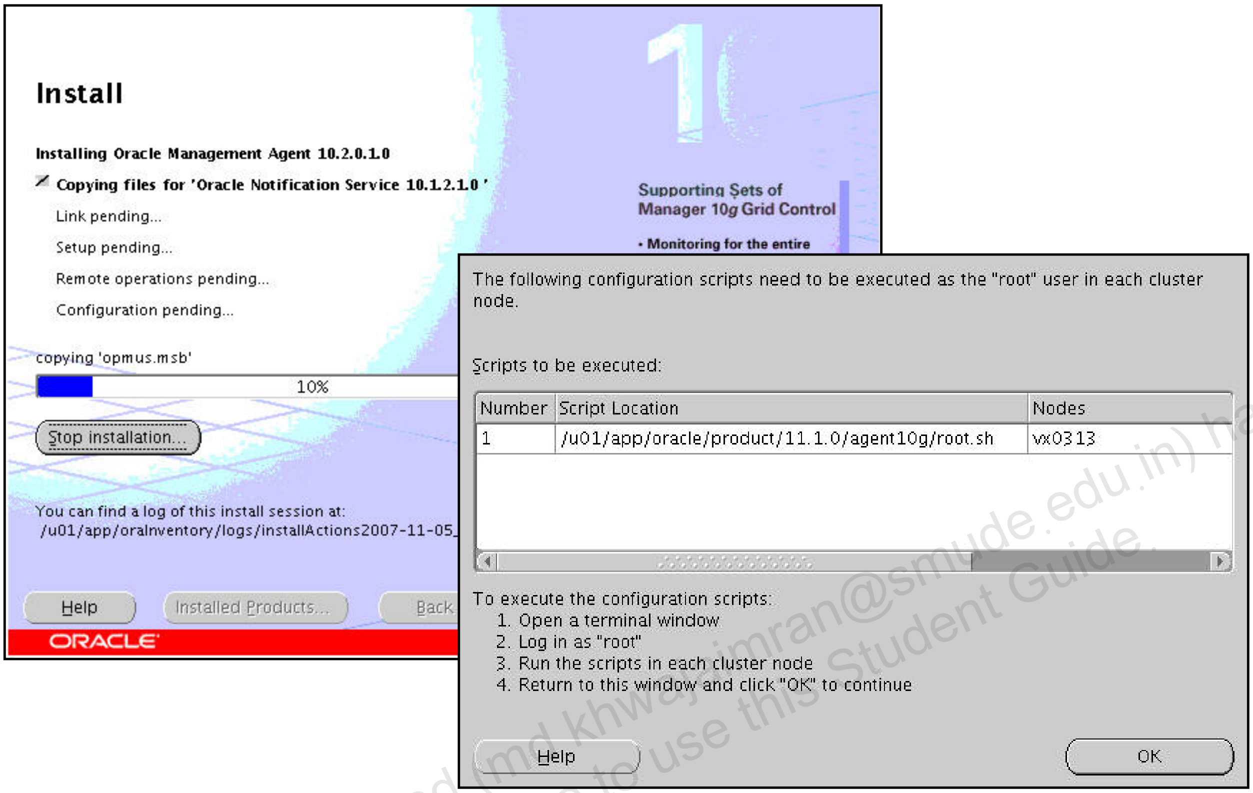
ORACLE

Agent Registration Password

On the Specify Agent Registration Password screen, you must provide the agent registration password. Do not specify an arbitrary password here, otherwise the agent registration will fail and the target nodes and their managed components will not be visible from the Grid Control server. In the example in the slide, you must provide the password for the Grid Control server located on `vx0314.us.oracle.com`. When this information is provided, click the **Next** button to continue.

The next screen is the Summary screen. Quickly review the information for accuracy and click the **Next** button to continue.

Management Agent Installation Finish

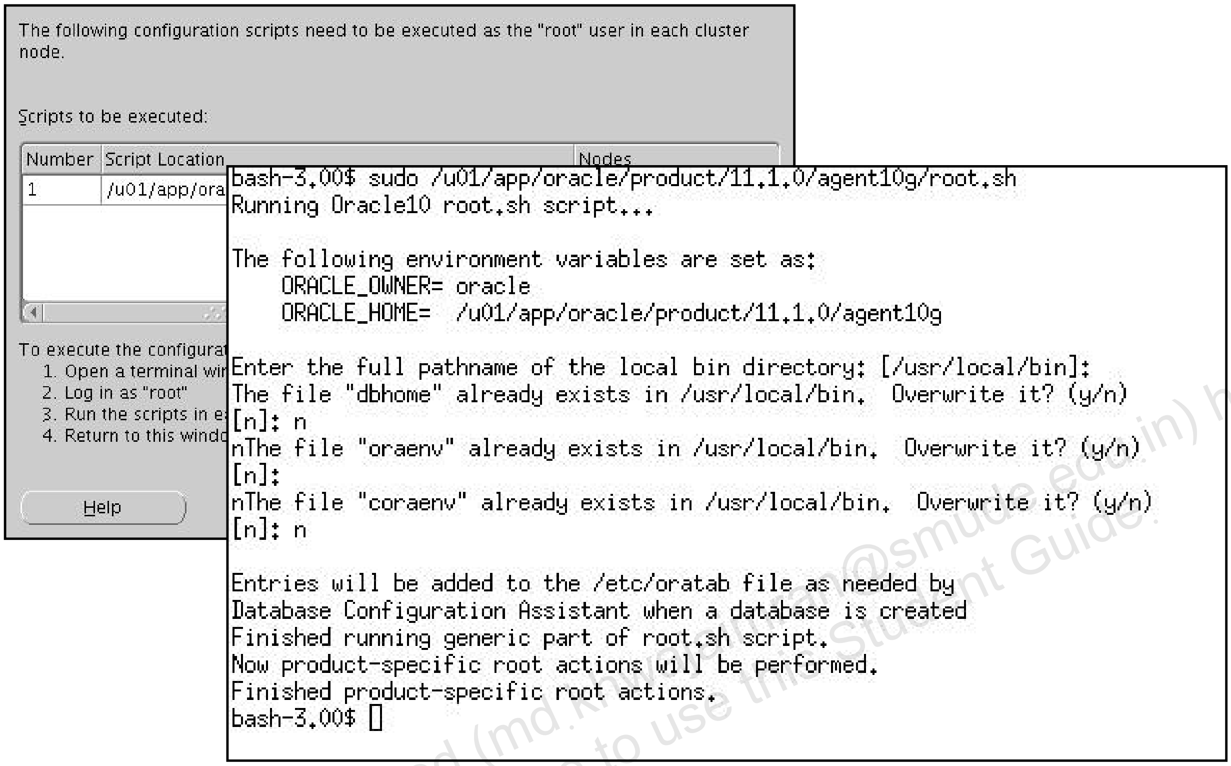


ORACLE

Management Agent Installation Finish

During the course of the agent installation, you can monitor the progress of the install from the Install screen. When the installation is complete, you will be prompted to execute the `root.sh` scripts on all nodes where the agent was installed.

Executing the `root.sh` Script



ORACLE

Executing the `root.sh` Script

When the script has been run on all nodes, close the Execute Configuration Scripts screen by clicking the Exit button and return to the installer. When the installer indicates that the agent has been installed, click the Exit button to quit the installer. You are now ready to create the cluster database.

Creating the Cluster Database


```
$ cd /u01/app/oracle/product/11.1.0/db_1/bin
$ ./dbca
```

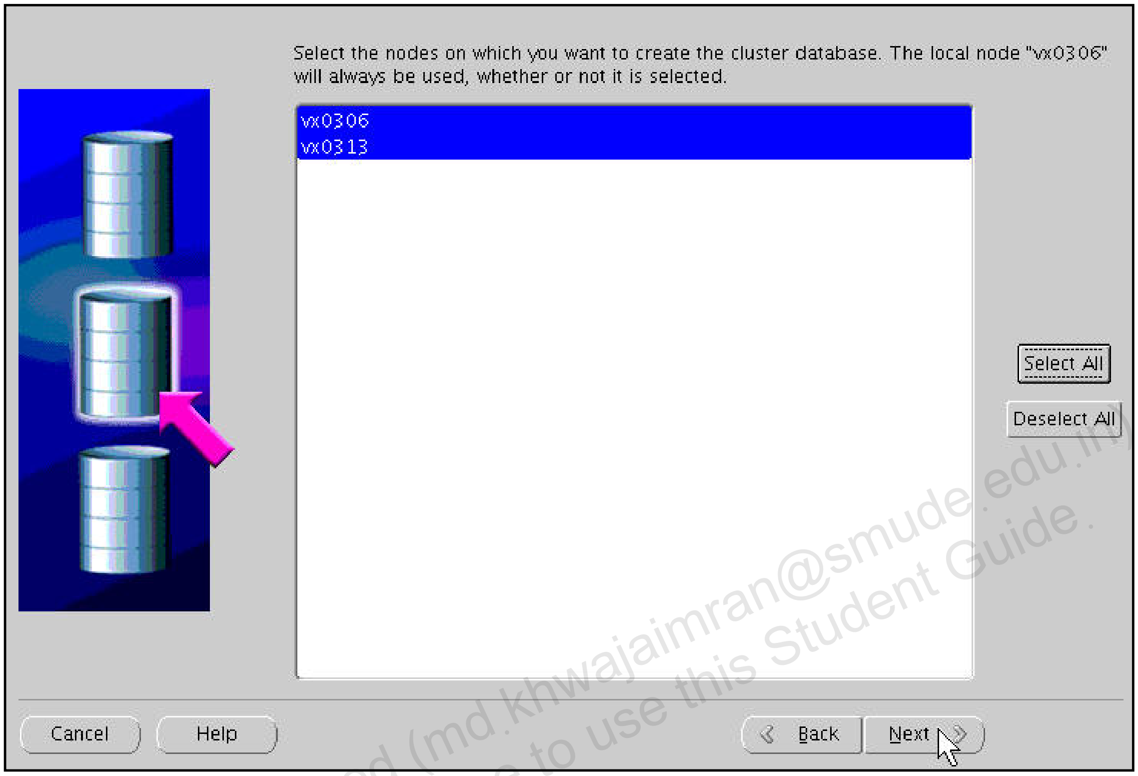

Creating the Cluster Database

This database creation assumes that the database will use ASM for its shared storage. Change directory to `$ORACLE_HOME/bin` on the installing node and execute the DBCA as shown below:

```
$ cd /u01/app/oracle/product/11.1.0/db_1/bin
$ ./dbca
```

The Welcome screen appears first. You must select the type of database that you want to install. Click the **Oracle Real Application Clusters database** option button, and then click Next. The Operations screen appears. For a first-time installation, you have two choices only. The first option enables you to create a database and the other option enables you to manage database creation templates. Click the **Create a Database** option button, and then click **Next** to continue.

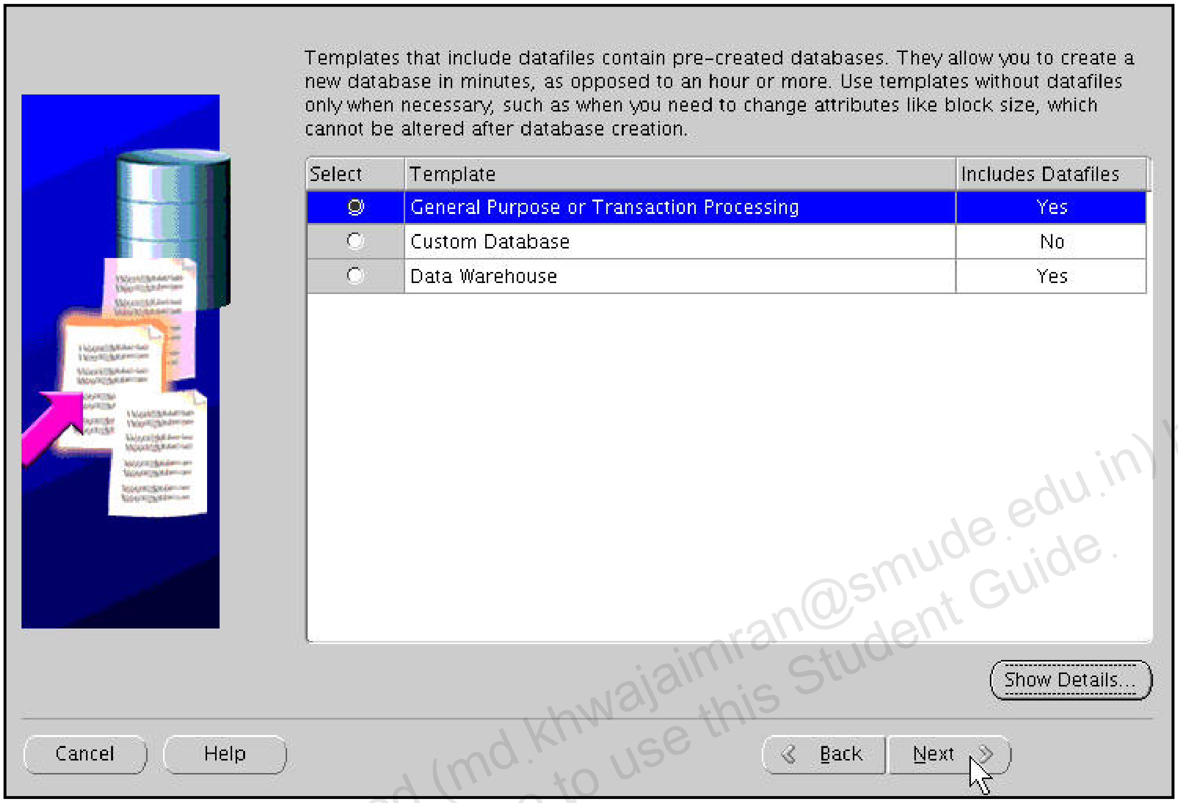
Node Selection



Node Selection

The Node Selection screen is now displayed. Because you are creating a cluster database, choose all the nodes. Click the **Select All** button to choose all the nodes of the cluster. Each node must be highlighted before continuing. If all nodes do not appear, you must stop the installation and troubleshoot your environment. If no problems are encountered, click the **Next** button to proceed.

Select Database Type



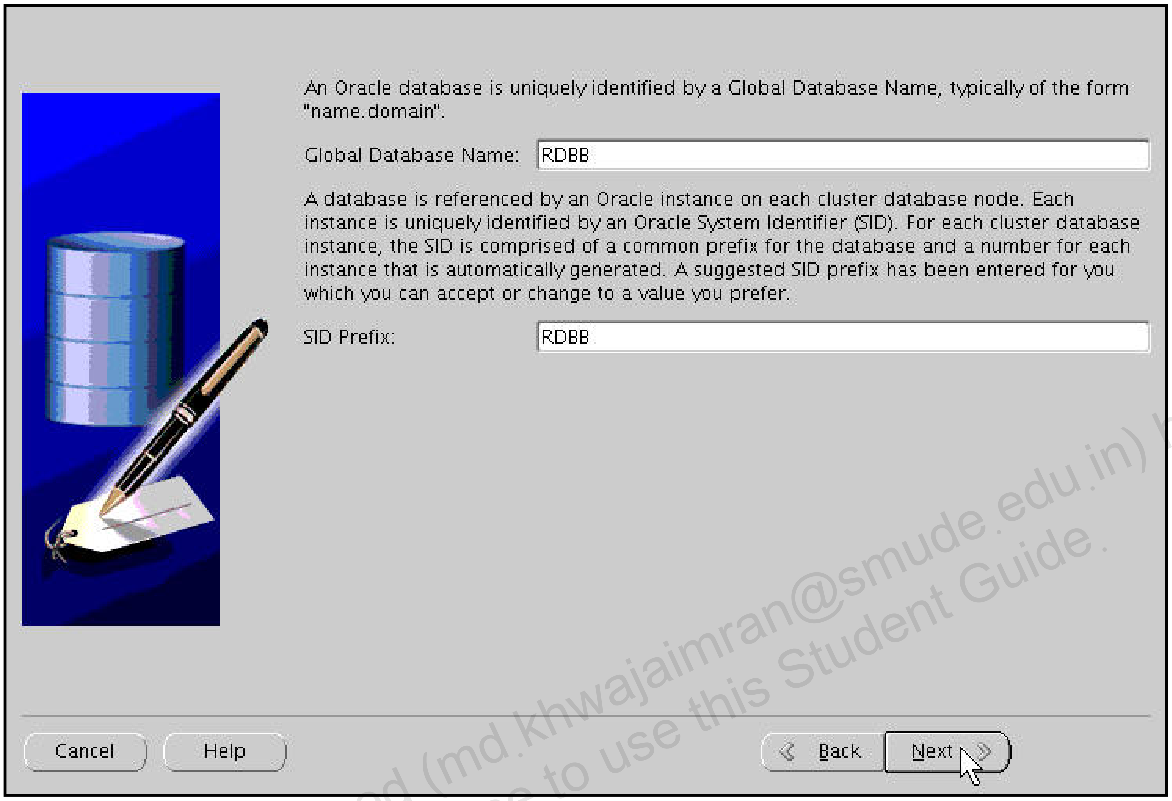
Select Database Type

The Database Templates screen appears. 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 the Next button to continue.

Database Identification



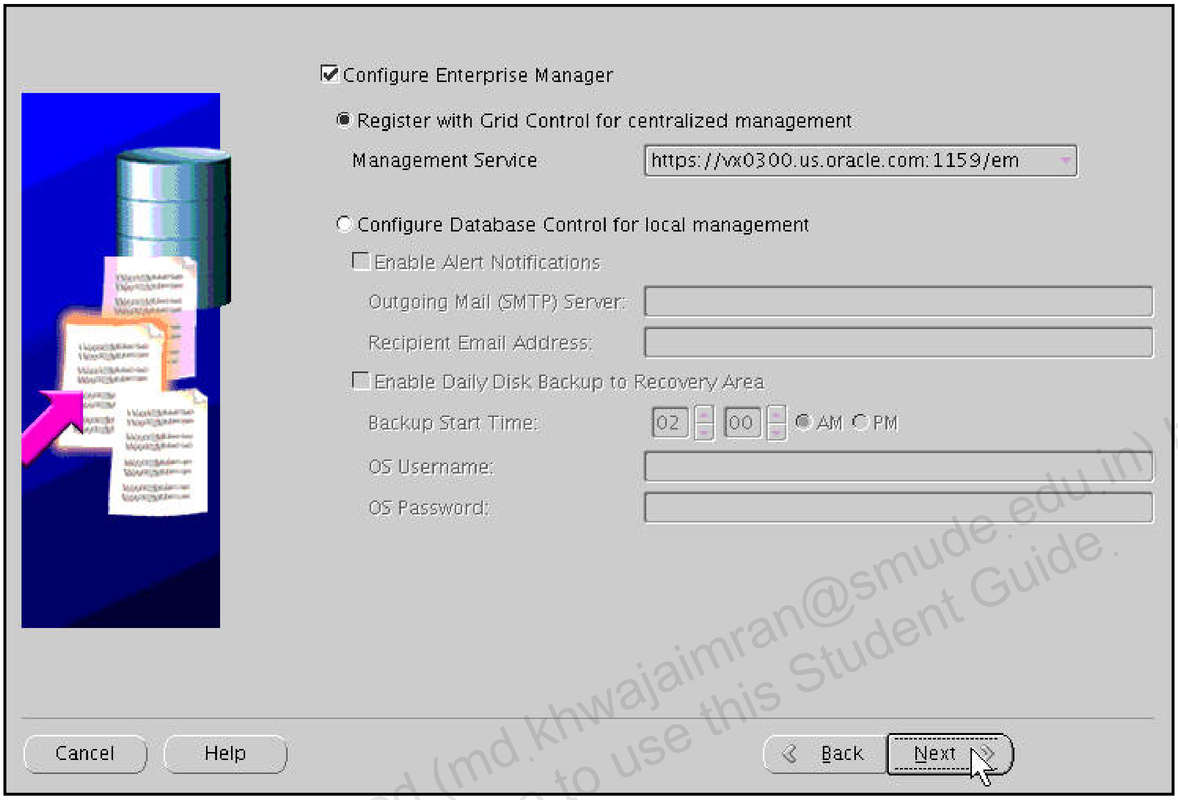
ORACLE

Database Identification

In the Database Identification screen, you must enter the database name in the Global Database Name field. The name that you enter in this screen must be unique among all the global database names used in your environment. The global database name can be up to 30 characters in length and must begin with an alphabetical character.

A system identifier (SID) prefix is required, and the DBCA suggests a name based on your global database name. This prefix is used to generate unique SID names for the two instances that make up the cluster database. For example, if your prefix is `RDBB`, the DBCA creates two instances on node 1 and node 2 named `RDBB1` and `RDBB2`, respectively. This example assumes that you have a two-node cluster. If you do not want to use the system-supplied prefix, enter a prefix of your choice. The SID prefix must begin with an alphabetical character and contain no more than 5 characters (on UNIX-based systems) or 61 characters (on Windows-based systems), or 64 characters (on Linux-based systems). Click the Next button to continue.

Cluster Database Management Method



Cluster Database Management Method

The Management Options screen is displayed. For small cluster environments, you may choose to manage your cluster with Enterprise Manager Database Control. To do this, select the "Configure the Database with Enterprise Manager" check box. If you have Grid Control installed somewhere on your network, you can select the "Use Grid Control for Database Management" option. If you select Enterprise Manager with the Grid Control option and the DBCA discovers agents running on the local node, you can select the preferred agent from a list. Grid Control can simplify database management in large, enterprise deployments.

You can also configure Database Control to send email notifications when alerts occur. If you want to configure this, you must supply a Simple Mail Transfer Protocol (SMTP) or outgoing mail server and an email address. You can also enable daily backups here. You must supply a backup start time as well as operating system user credentials for this option.

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. After making your choices, click the **Next** button to continue.

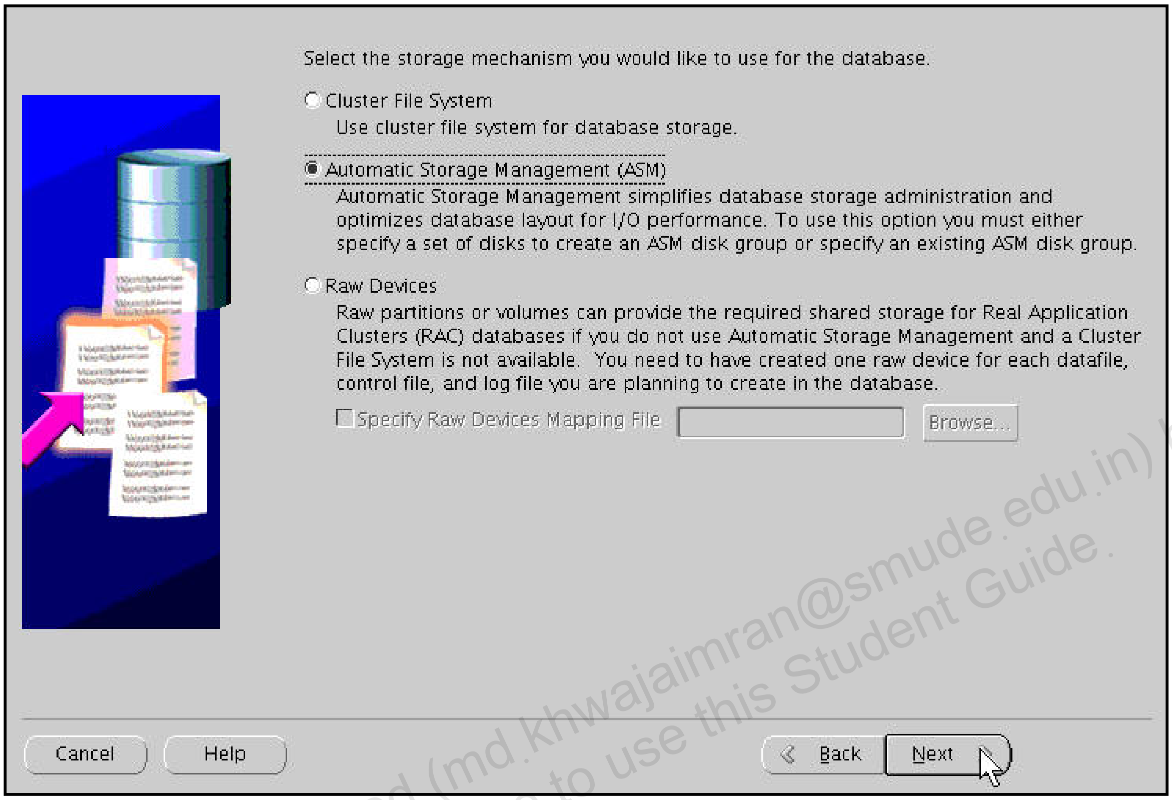
Passwords for Database Schema Owners

Passwords for Database Schema Owners

The Database Credentials screen 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 clicking the **Use the Same Password for All Accounts** option button. 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, click the Use Different Passwords option button, and then 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 the **Next** button to continue.

Storage Options for Database Files



Storage Options for Database Files

On the Storage Options screen, you must select the storage medium where your shared database files are stored. Your three choices are:

- Cluster File System
- Automatic Storage Management (ASM)
- Raw Devices

If you click the **Cluster File System** option button, you can click the **Next** button to continue.

If you click the **Automatic Storage Management (ASM)** option button, you can either use an existing ASM disk group or specify a new disk group to use. If there is no ASM instance on any of the cluster nodes, the DBCA displays the Create ASM Instance screen for you. If an ASM instance exists on the local node, the DBCA displays a dialog box prompting you to enter the password for the `SYS` user for ASM. To initiate the creation of the required ASM instance, enter the password for the `SYS` user of the ASM instance. After you enter the required information, click **Next** to create the ASM instance. After the instance is created, the DBCA proceeds to the ASM Disk Groups screen. If you have just created a new ASM instance, there is no disk group from which to select, so you must create a new one by clicking **Create New** to open the Create Disk Group screen.

Storage Options for Database Files (continued)

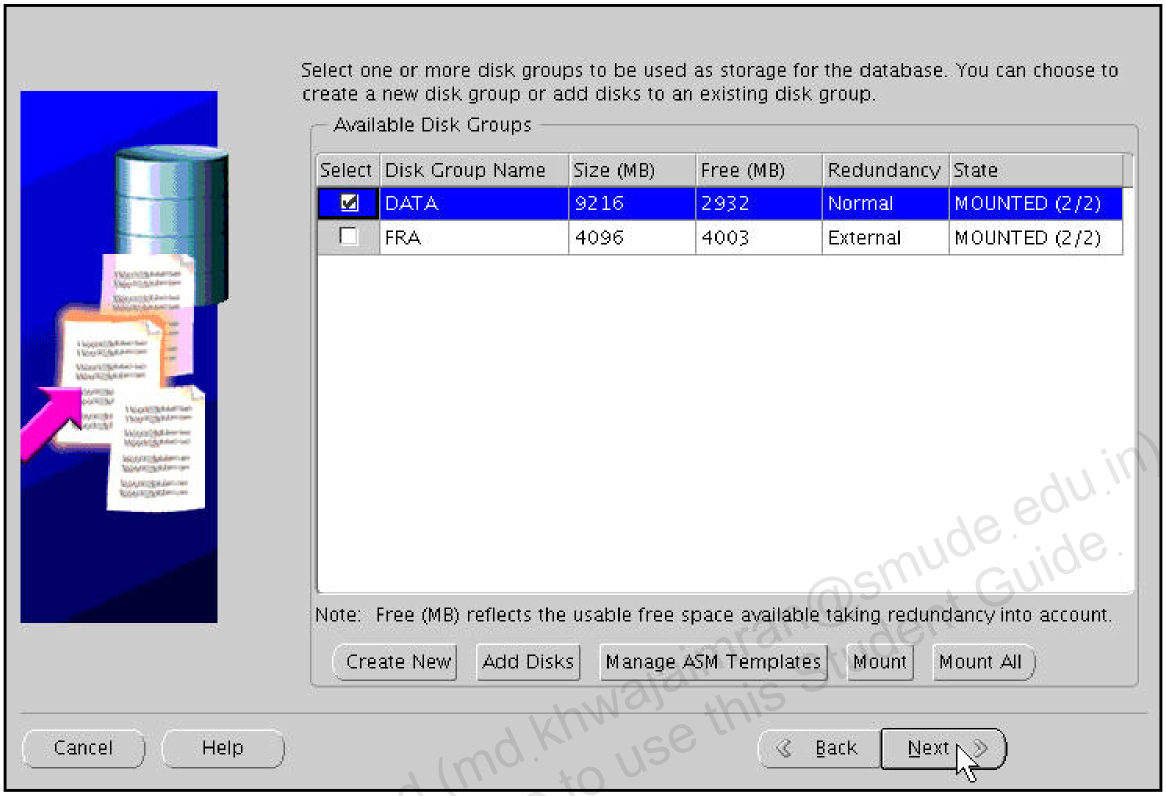
After you are satisfied with the ASM disk groups available to you, select the one that you want to use for your database files, and click **Next** to proceed to the Database File Locations screen.

If you have configured raw devices, click the corresponding button. You must provide a fully qualified mapping file name if you did not previously set the DBCA_RAW_CONFIG environment variable to point to it. You can enter your response or click the Browse button to locate it. The file should follow the format of the example below:

```
system=/dev/vg_name/rdbname_system_raw_500m
sysaux=/dev/vg_name/rdbname_sysaux_raw_800m
...
redo2_2=/dev/vg_name/rdbname_redo2_2_raw_120m
control1=/dev/vg_name/rdbname_control1_raw_110m
control2=/dev/vg_name/rdbname_control2_raw_110m
spfile=/dev/vg_name/rdbname_spfile_raw_5m
pwdfile=/dev/vg_name/rdbname_pwdfile_raw_5m
```

where VG_NAME is the volume group (if configured) and rdbname is the database name. Because this example uses a preexisting ASM disk group, click the **Automatic Storage Management** button, and then the **Next** button to continue.

ASM Disk Groups

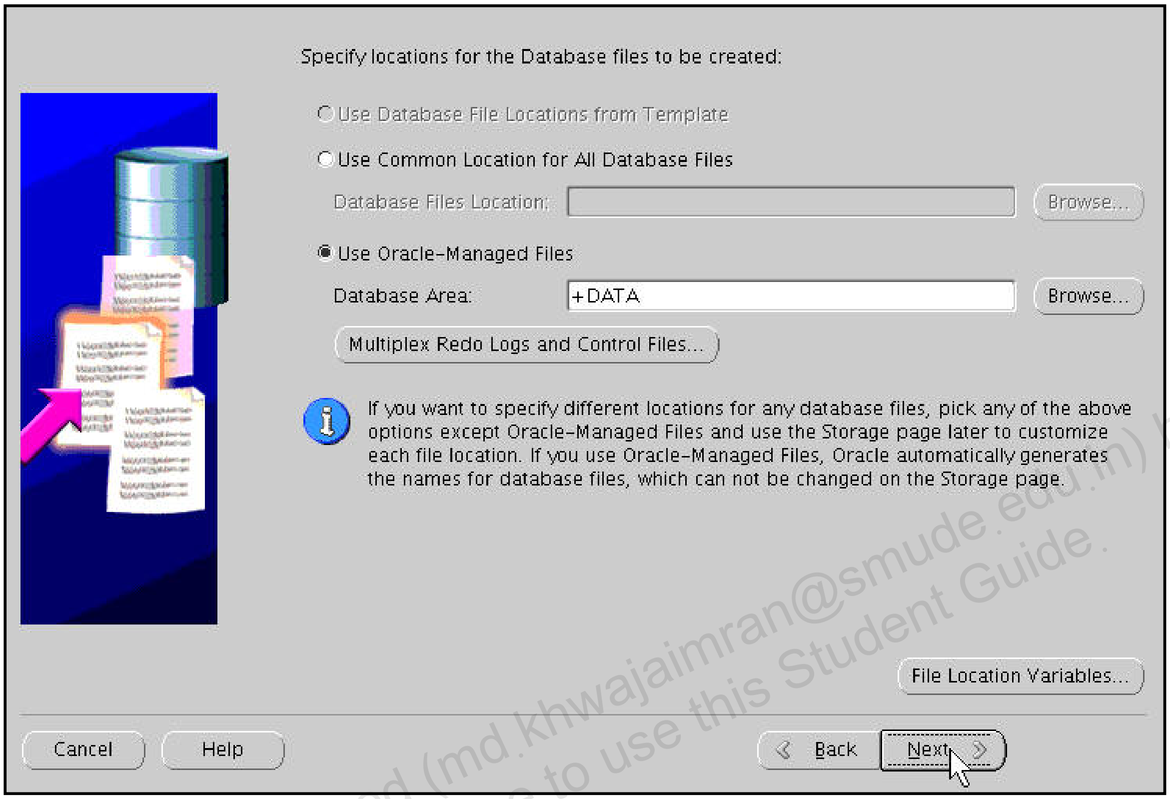


ORACLE

ASM Disk Groups

On the ASM Disk Groups screen, you can choose an existing disk group to be used for your shared storage. To select a disk group, choose the Select check box corresponding to the disk group that you want to use. Alternatively, you can also create a new disk group to be used by your cluster database. When you are finished, click the **Next** button to continue.

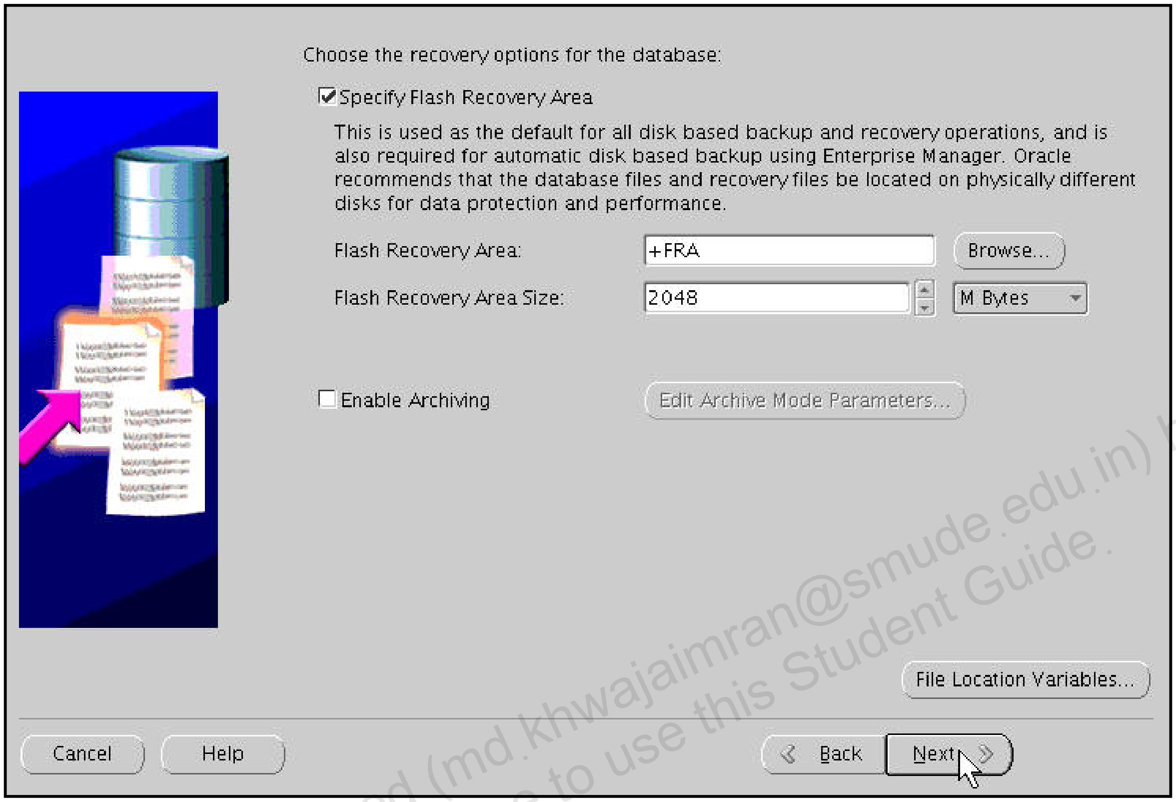
Database File Locations



Database File Locations

On the Database File Locations screen, you must indicate where the database files are created. You can choose to use a standard template for file locations, one common location, or Oracle Managed Files (OMF). This cluster database uses Oracle-managed files. Therefore, select the **Use Oracle-Managed Files** option button, and enter the disk group name preceded with "+" in the Database Area field. Alternatively, you can click the Browse button to indicate the location where the database files are to be created. When you have made your choices, click the **Next** button to continue.

Recovery Configuration

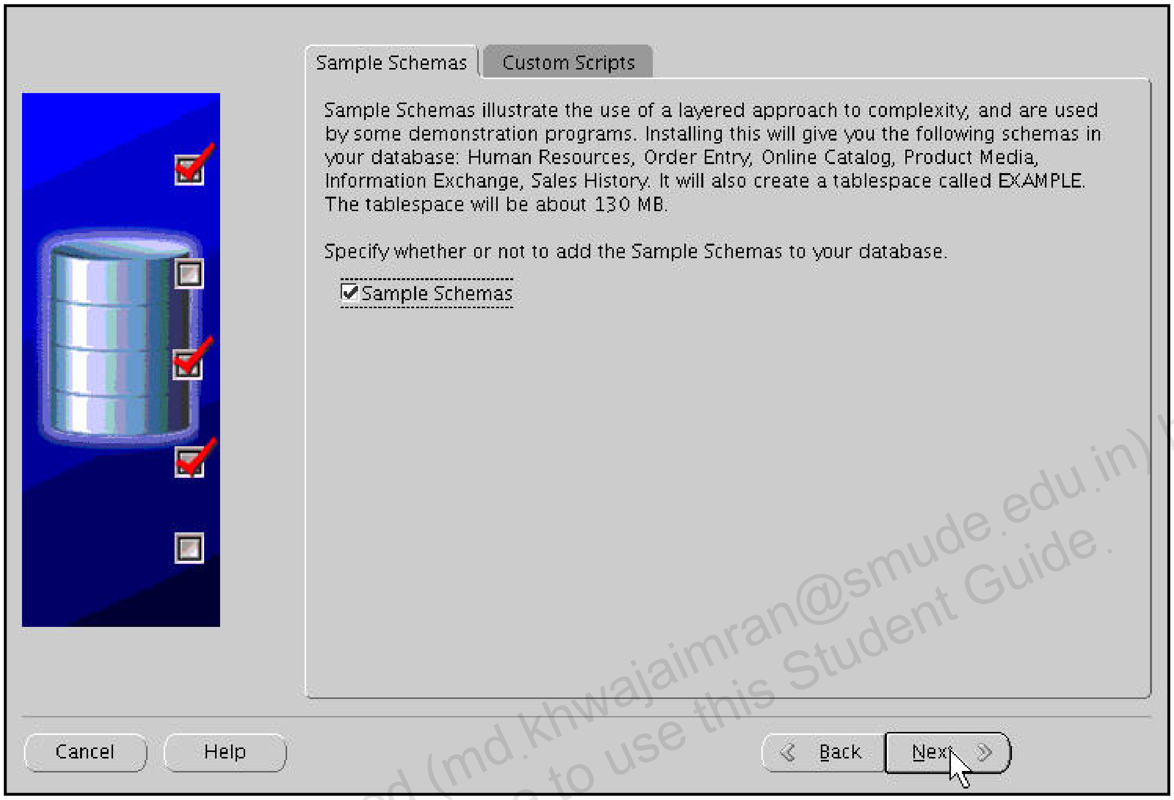


Recovery Configuration

On the Recovery Configuration screen, 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 Flash Recovery Area size on the Recovery Configuration screen. The size of the area defaults to 2048 megabytes, but you can change this figure if it is not suitable for your requirements. If you are using ASM and a single disk group, the flash recovery area defaults to the ASM Disk Group. If more than one disk group has been created, you can specify it here. If you use a cluster file system, the flash recovery area defaults to `$ORACLE_BASE/flash_recovery_area`. You may also define your own variables for the file locations if you plan to use the Database Storage screen to define individual file locations.

When you have completed your entries, click **Next,** and the Database Content screen is displayed.

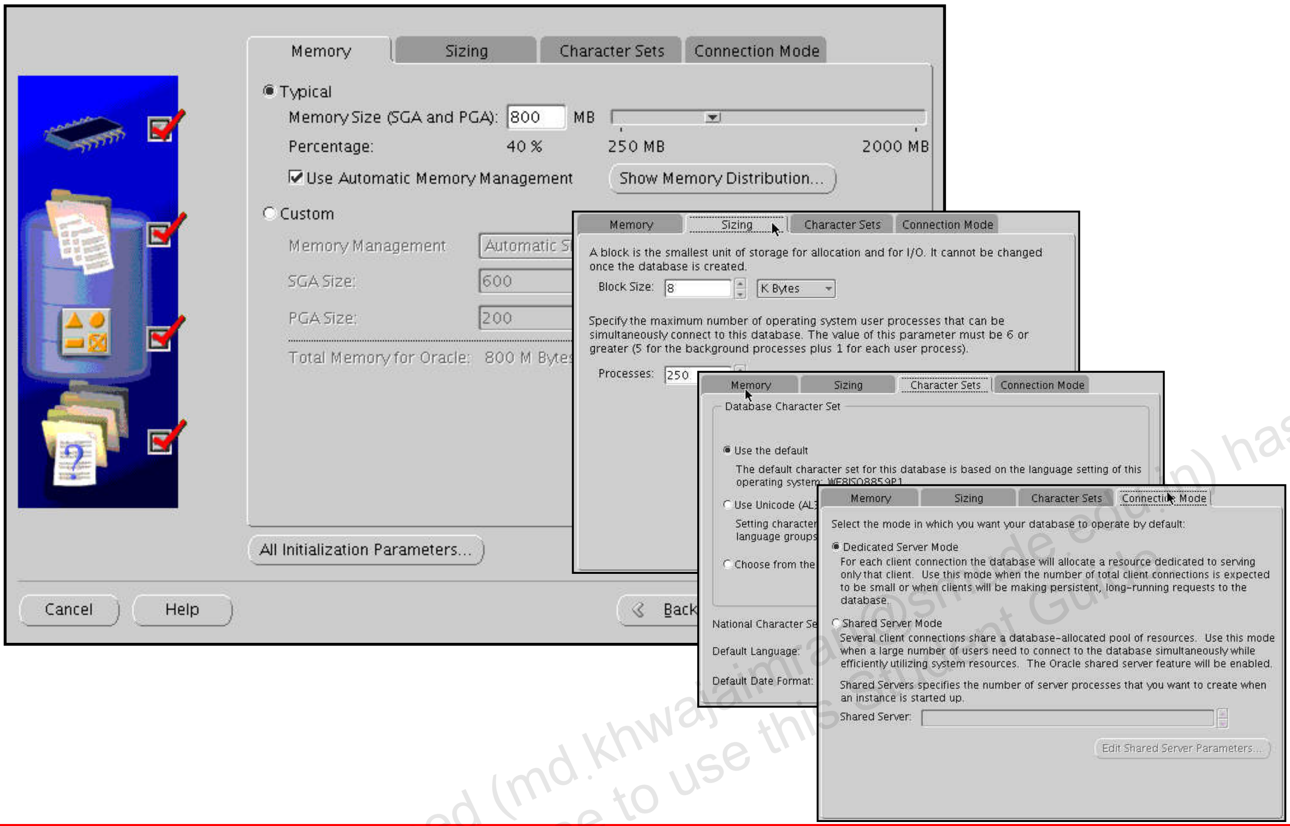
Database Content



Database Content

On the Database Content screen, you can choose to install the Sample Schemas included with the database distribution. On the Custom Scripts tabbed page, you can choose to run your own scripts as part of the database creation process. When finished, click the **Next** button to continue to the next page.

Initialization Parameters



ORACLE

Initialization Parameters

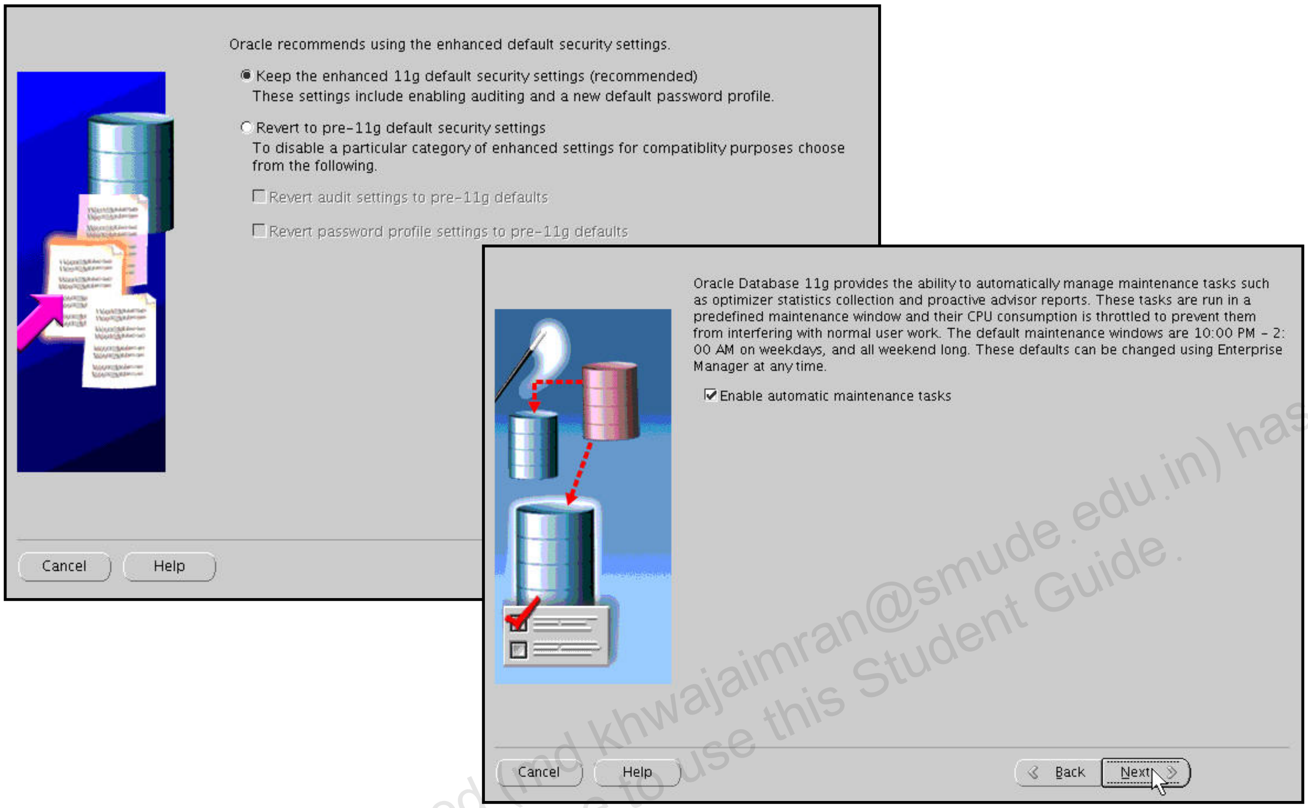
On the Initialization Parameters screen, you can set important database parameters. The parameters are grouped on four tabs:

- 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 tab, you can adjust the database block size. Note that the default is 8 KB. 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 in the four tabs, click the All Initialization Parameters button. Click the Use Automatic Memory Management button, and then click the **Next** button to continue.

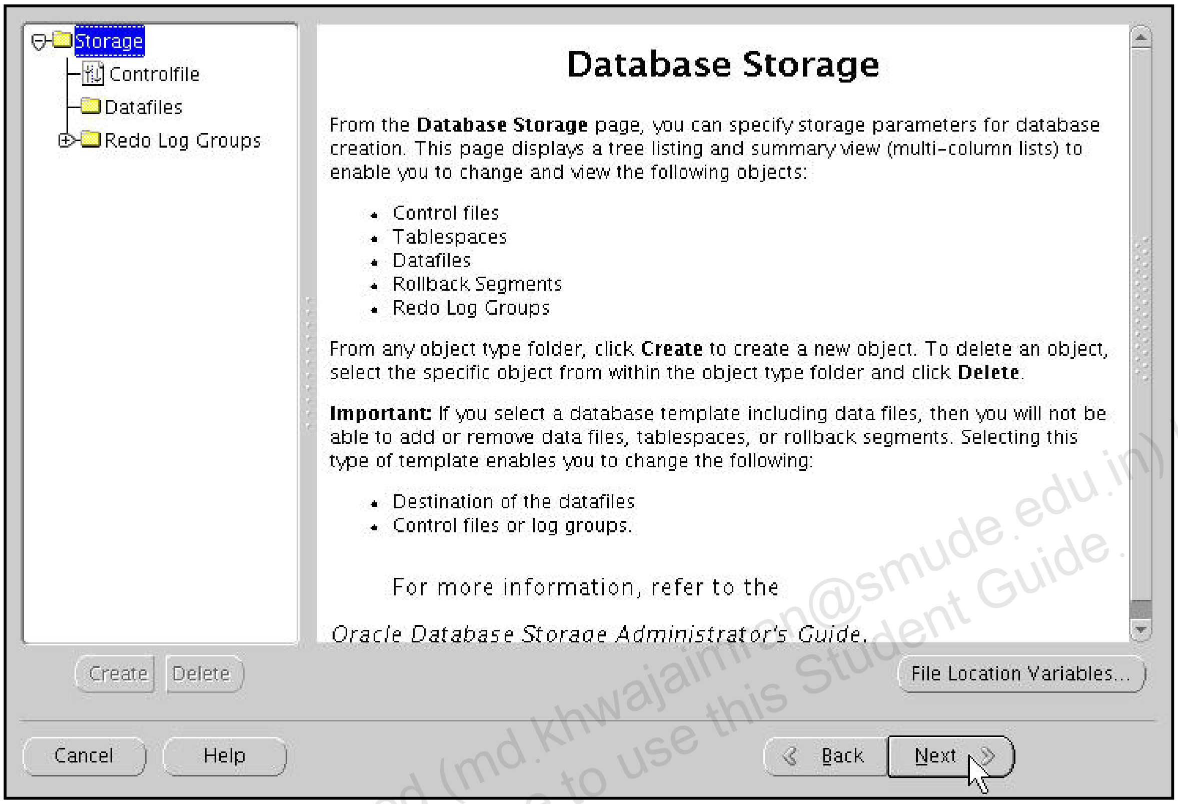
Security Settings and Maintenance Tasks



Security Settings and Maintenance Tasks

On the Security Settings page you can choose between enhanced Oracle 11*g* security and the pre-11*g* model (which requires slightly less overhead). Make your choice here and click **Next** to continue. Click **Enable automatic maintenance tasks** to schedule metrics collection and advisor runs using predefined maintenance windows and resource consumption models. Then click **Next** to continue.

Database Storage Options

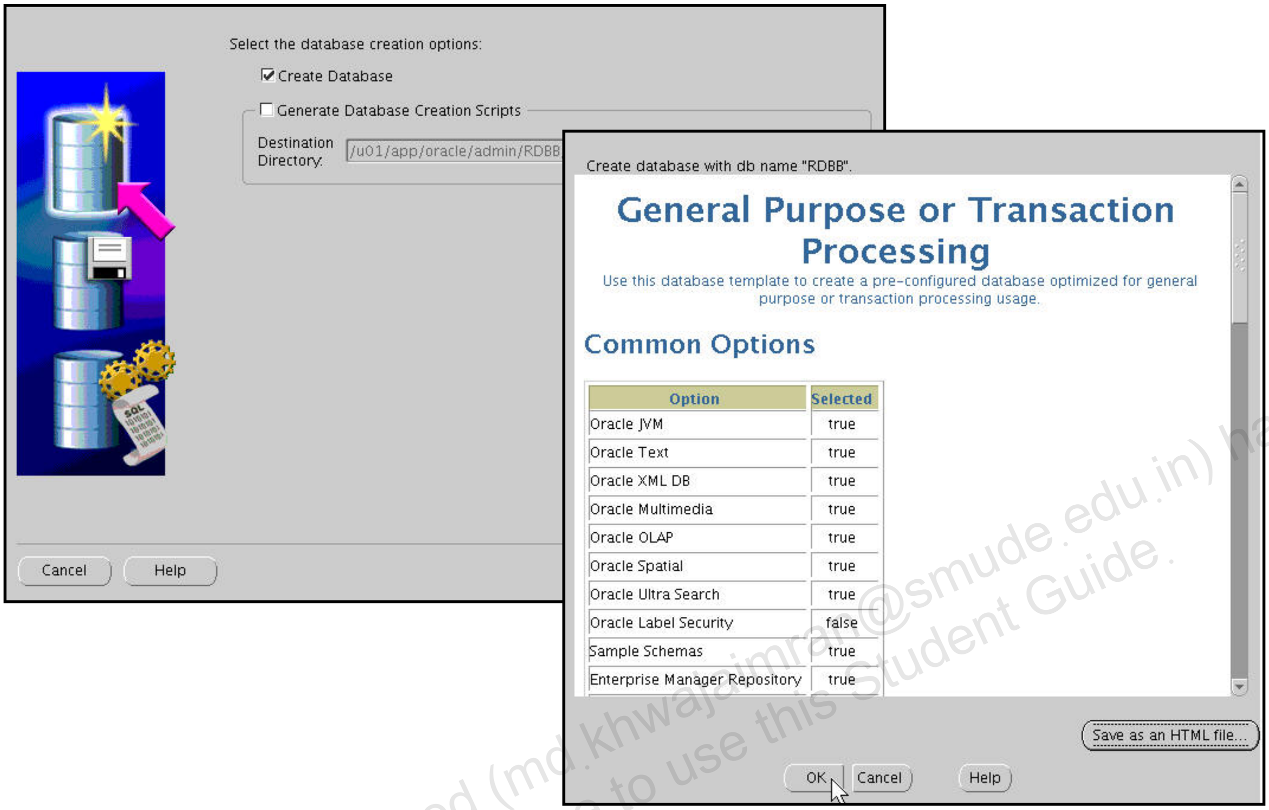


Database Storage Options

The Database Storage screen provides full control over all aspects of database storage, including table spaces, data files, and log members. Size, location, and all aspects of extent management are under your control here.

When finished, click the **Next** button to continue to the next page.

Create the Database



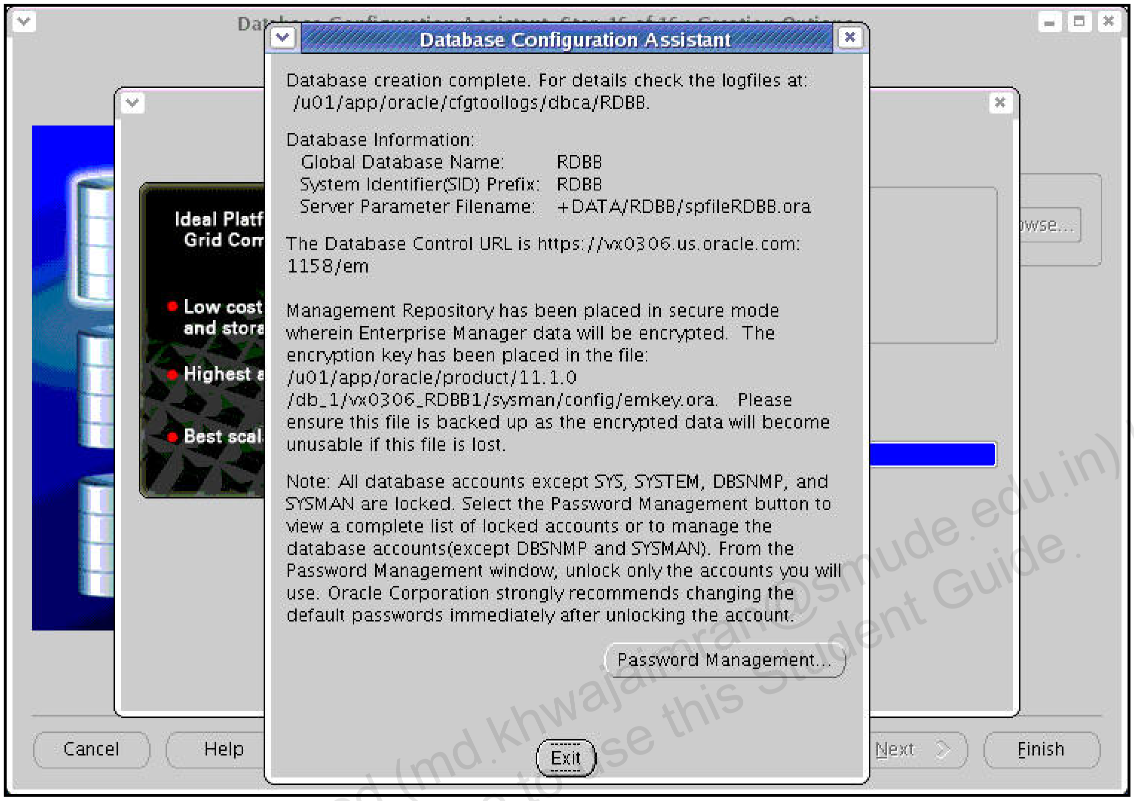
ORACLE

Create the Database

The Creation Options screen appears next. You can choose to create the database, or save your DBCA session as a database creation script by clicking the corresponding button. Select the Create Database check box, and then click the **Finish** button. The DBCA displays the Summary screen, giving you the 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 screen by clicking the **OK** button.

Monitor Progress



ORACLE

Monitor Progress

The Progress Monitor screen 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

- Verify the cluster database configuration.

```
$ srvctl config database -d racdb
vx0306 racdb1 /u01/app/.../db_1
vx0313 racdb2 /u01/app/.../db_1
```

- Back up the `root.sh` script.

```
$ cd $ORACLE_HOME
$ cp root.sh root.sh.bak
```

- Back up the voting disk.

```
$ dd if=/dev/sdb6 of=/RACdb/OCR/backup/vdisk.bak
```

- Download and install the required patch updates.

Postinstallation Tasks

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 -d db_name
```

Server Control (`SRVCTL`) displays the name of the node and the instance for the node. The following example shows two nodes, `vx0306` and `vx0313` running instances named `racdb1` and `racdb2` respectively. Execute the following command:

```
$ srvctl config database -d racdb
vx0306 racdb1 /u01/app/.../db_1
vx0313 racdb2 /u01/app/.../db_1
```

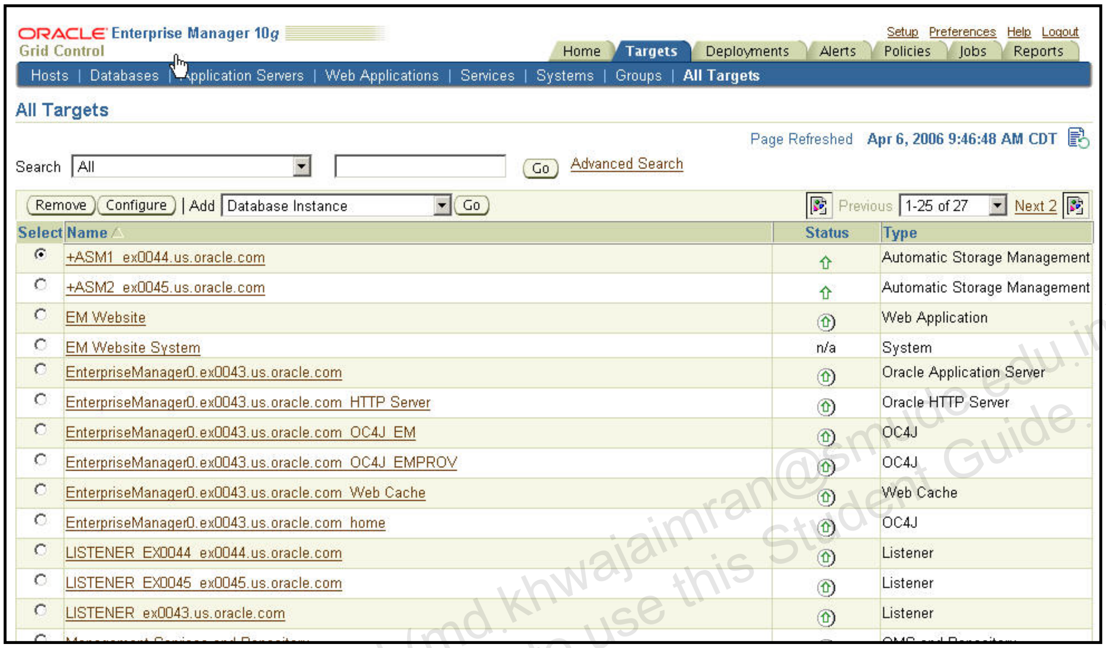
It is also recommended that you back up the `root.sh` script after you complete an installation. If you install other products in the same Oracle Home directory, the OUI updates the contents of the existing `root.sh` script during the installation. If you require information contained in the original `root.sh` script, you can recover it from the `root.sh` file copy.

After your Oracle Database 11*g* RAC installation is complete and after you are sure that your system is functioning properly, make a backup of the contents of the voting disk by using the `dd` utility.

Note: the `SRVCTL` utility is covered in more detail in the next lesson.

Check Managed Targets

http://vx0314.us.oracle.com:4889/em

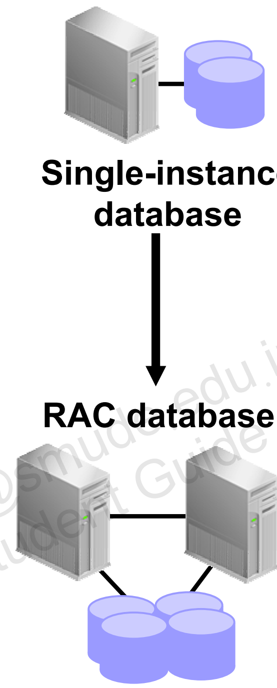


Check Managed Targets

Another post-installation task you should perform if you are using Grid Control is to check that all the managed nodes and their managed resources are properly registered and available. Open a browser and enter the address for your Grid Control console. Click the **Targets** tab to verify that all the targets appear here.

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



Single Instance to RAC Conversion

You can use the Database Configuration Assistant (DBCA) to convert from 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 following conditions:

- It is a supported hardware and operating system software configuration.
- It has shared storage: Either Oracle Cluster File System or ASM is available and accessible from all nodes.
- Your applications have no design characteristics that preclude their use with cluster database processing.

If your platform supports a cluster file system, then you can use it for RAC. You can also convert to RAC and use a nonshared file system. In either case, it is recommended that you use the Oracle Universal Installer (OUI) to perform an Oracle Database 11*g* installation that sets up the Oracle Home and inventory in an identical location on each of the selected nodes in your cluster.

Single-Instance Conversion Using the DBCA

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

1. Back up the original single-instance database.
2. Perform the preinstallation steps.
3. Set up and validate the cluster.
4. Copy the preconfigured database image.
5. Install the Oracle Database 11*g* software with Real Application Clusters.

ORACLE

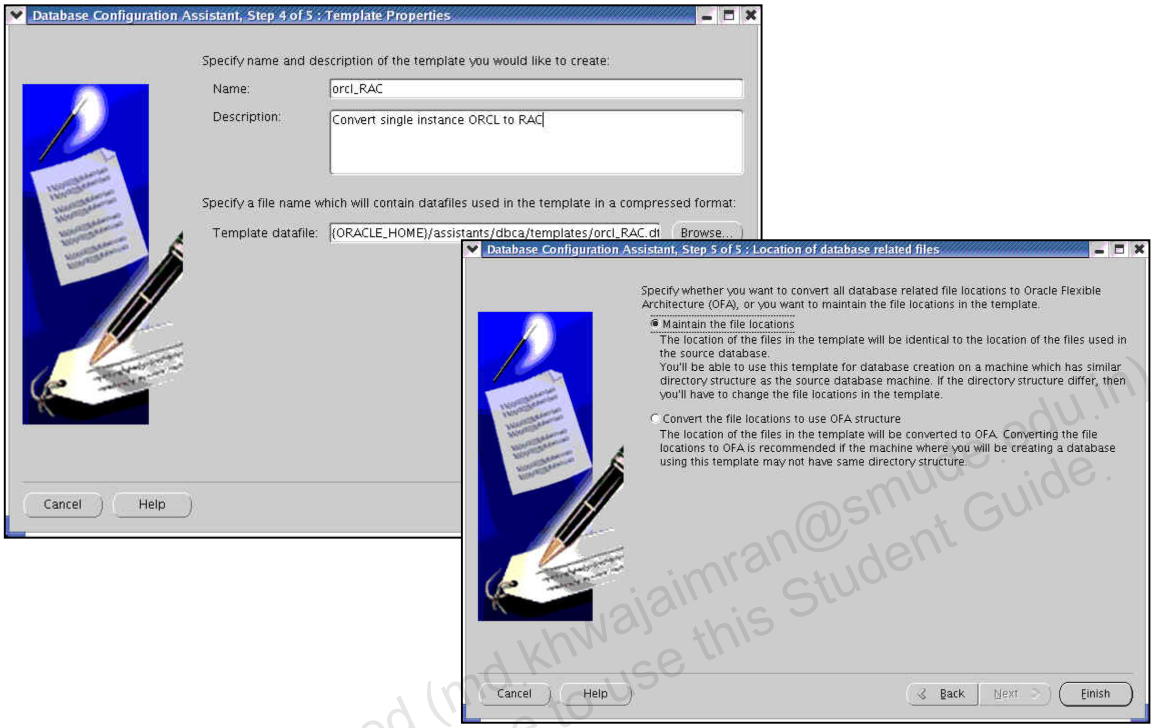
Single-Instance Conversion Using DBCA

To convert from a single-instance Oracle database that is on a noncluster computer to a RAC database, perform the steps outlined below, and in the order shown:

1. Back up the original single-instance database.
2. Perform the preinstallation steps.
3. Set up the cluster.
4. Validate the cluster.
5. Copy the preconfigured database image.
6. Install the Oracle Database 11*g* software with Real Application Clusters.

Conversion Steps

1. Back up the original single-instance database.



ORACLE

Conversion Steps

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. On the Welcome screen, click Next.
3. On the Operations screen, select Manage Templates, and click Next.
4. On the Template Management screen, select "Create a database" template and "From an existing database (structure as well as data)," and click Next. On the Source Database screen, enter the database name in the Database instance field, and click Next.
5. On the Template Properties screen, enter a template name in the Name field. By default, the template files are generated in the `ORACLE_HOME/assistants/dbca/templates` directory. 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 finished, click Next.
6. On the Location of Database Related Files screen, 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.ctl`) and a database preconfigured image file (`template_name.dfb`).

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 using `cluvfy`.
4. Copy the preconfigured database image.
   – Copy the preconfigured database image including:
     – The database structure `*.dbc` file
     – The preconfigured database image `*.dfb` file

ORACLE

Conversion Steps (continued)

2. Perform the preinstallation steps.

Several tasks must be completed before Oracle Clusterware and Oracle Database 11*g* software can be installed. Some of these tasks are common to all Oracle database installations and should be familiar to you. Others are specific to Oracle RAC 11*g*. You can review these tasks by referring to the lesson titled "Oracle Clusterware Installation and Configuration."

3. Set up and validate the cluster.

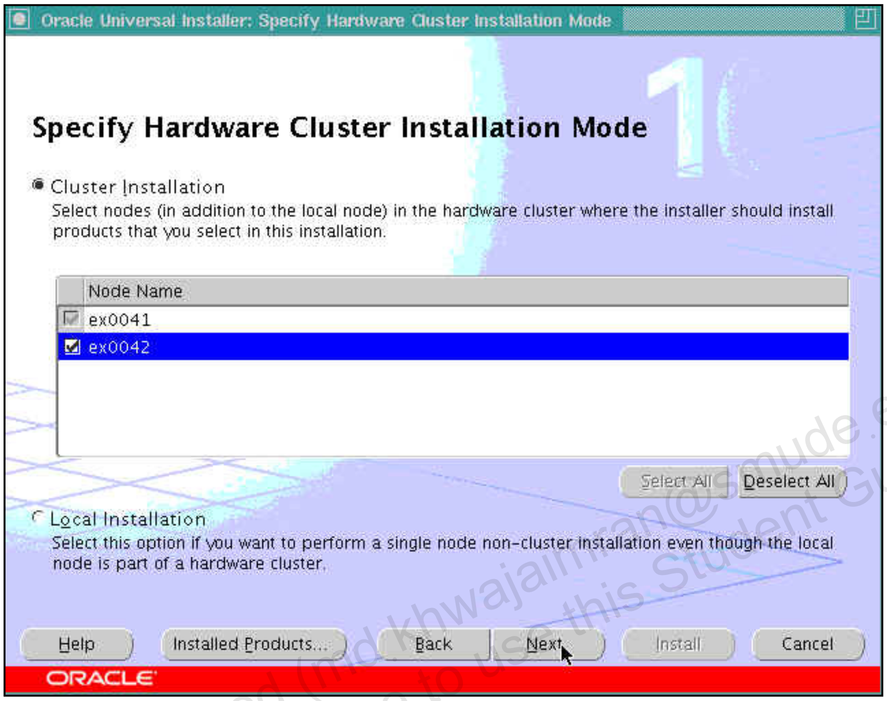
Form a cluster with the required number of nodes according to your hardware vendor's documentation. When you have configured all of the nodes in your cluster, validate cluster components by using the `cluvfy` utility, and then install Oracle Clusterware. When the clusterware is installed, validate the completed cluster installation and configuration using the Cluster Verification Utility, `cluvfy`.

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.

Conversion Steps

5. Install the Oracle Database 11*g* software with RAC.



ORACLE

Conversion Steps (continued)

5. Install the Oracle Database 11*g* software with RAC.

1. Run the OUI to perform an Oracle database installation with RAC. Select Cluster Installation Mode on the Specify Hardware Cluster Installation screen of the OUI, and select the nodes to include in your RAC database.
2. On the OUI Database Configuration Types screen, select "Advanced install." After installing the software, the OUI runs postinstallation tools such as NETCA, DBCA, and so on.
3. On the DBCA Template Selection screen, 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. If you selected raw storage on the OUI Storage Options screen, then on the DBCA File Locations tab of the Initialization Parameters screen, replace the data files, control files, and log files, and so on, with the corresponding raw device files if you did not set the `DBCA_RAW_CONFIG` environment variable. You must also replace default database files with raw devices on the Storage page.
5. After creating the RAC database, the DBCA displays the Password Management screen on which you must change the passwords for database privileged users who have `SYSDBA` and `SYSOPER` roles. When the DBCA exits, the conversion process is complete.

Single-Instance Conversion Using `rconfig`

1. Edit the `ConvertToRAC.xml` file located in the `$ORACLE_HOME/assistants/rconfig/sampleXMLs` directory.
2. Modify the parameters in the `ConvertToRAC.xml` file as required for your system.
3. Save the file under a different name.

```
$ cd $ORACLE_HOME/assistants/rconfig/sampleXMLs
$ vi ConvertToRAC.xml
$ rconfig my_rac_conversion.xml
```


Single-Instance Conversion Using `rconfig`

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

1. Go to the `$ORACLE_HOME/assistants/rconfig/sampleXMLs` directory as the `oracle` user, and open the `ConvertToRAC.xml` file using a text editor, such as `vi`.
2. Review the `ConvertToRAC.xml` file, and modify the parameters as required for your system. The XML sample file contains comment lines that provide instructions for how to configure the file.
   When you have completed making changes, save the file with the syntax `filename.xml`. Make a note of the name you select.
3. Assuming that you save your XML file as `my_rac_conversion.xml`, navigate to the `$ORACLE_HOME/bin` directory, and use the following syntax to run the `rconfig` command:
   `$ rconfig my_rac_conversion.xml`

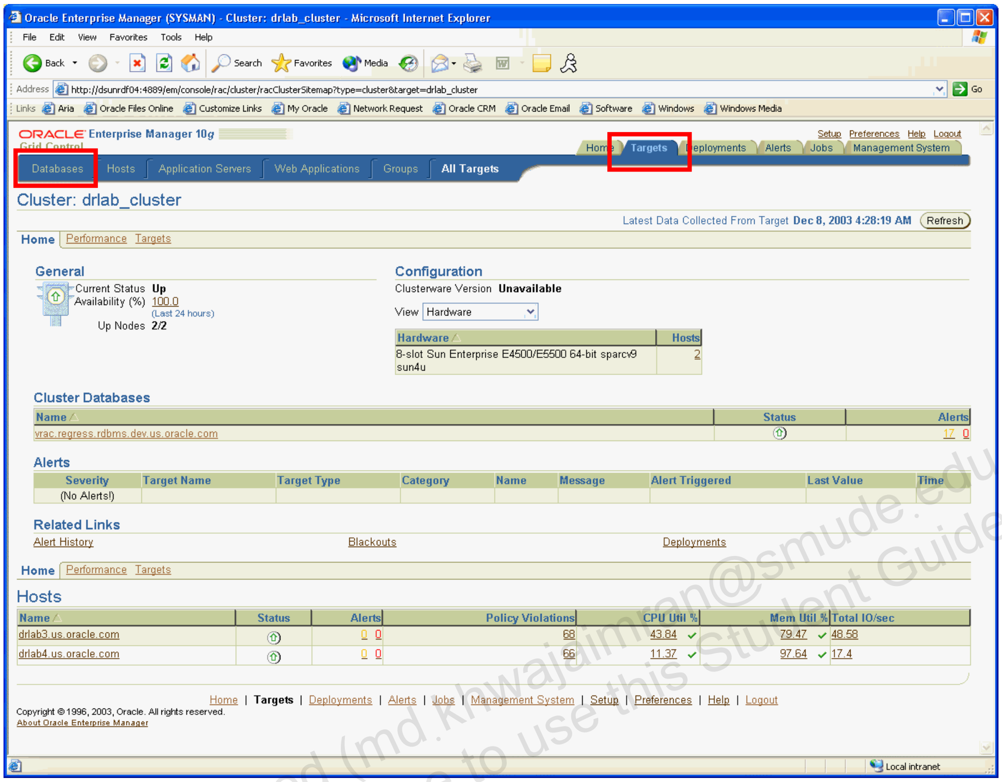
Note: The Convert verify option in the `ConvertToRAC.xml` file has three options:

- `Convert verify="YES"`: `rconfig` performs checks to ensure that the prerequisites for single-instance to RAC conversion have been met before it starts conversion.

Single-Instance Conversion Using `rconfig` (continued)

- `Convert verify="NO"`: `rconfig` does not perform prerequisite checks; it starts conversion.
- `Convert verify="ONLY"`: `rconfig` performs only prerequisite checks; it does not start conversion after completing prerequisite checks.

Single-Instance Conversion Using Grid Control



ORACLE

Single-Instance Conversion Using Grid Control

In addition to using the DBCA and `rconfig` for single-instance conversion, you can also use Enterprise Manager Grid Control to convert a single-instance database to RAC. To use this feature of Grid Control, complete the following steps:

1. Log in to Grid Control. From the Grid Control Home page, click the Targets tab.
2. On the Targets page, click the Databases secondary tab, and click the link in the Names column of the database that you want to convert to RAC.
3. On the Database Instance Home page, click the Administration secondary tab.
4. On the Administration page, in the Database Administration Change Database section, click Convert to Cluster Database.
5. Log in as the database user `SYS` with `SYSDBA` privileges to the database you want to convert, and click Next.
6. On the Convert to Cluster Database: Cluster Credentials page, provide a username and password for the `oracle` user and password of the target database that you want to convert. If the target database is using ASM, then provide the ASM SYS user and password and click Next.
7. On the Hosts page, select the host nodes in the cluster that you want to be cluster members in the RAC database installed. When you have completed your selection, click Next.

Single-Instance Conversion Using Grid Control (continued)

8. On the Convert to Database: Options page, select whether you want to use the existing listener and port number or specify a new listener and port number for the cluster. Also, provide a prefix for the cluster database instances. When you have finished entering information, click Next.
9. On the Convert to Cluster Database: Shared Storage page, either select the option to use your existing shared storage area, or select the option to have your database files copied to a new shared storage location. Also, decide whether you want to use your existing flash recovery area, or if you want to copy your flash recovery files to a new area using Oracle Managed Files. When you have finished entering information, click Next.
10. On the Convert to Cluster Database: Review page, review the options you have selected. Click Submit Job to proceed with the conversion.
11. On the Confirmation page, click View Job to check the status of the conversion.

Summary

In this lesson, you should have learned how to:
- Install the Oracle Enterprise Manager agent
- Create a cluster database
- Perform required tasks after database creation

Practice 3: Overview

This practice covers the following topics:

- Installing the Enterprise Manager agent on each cluster node
- Confirming that the services needed by the database creation process are running
- Creating a cluster database

4

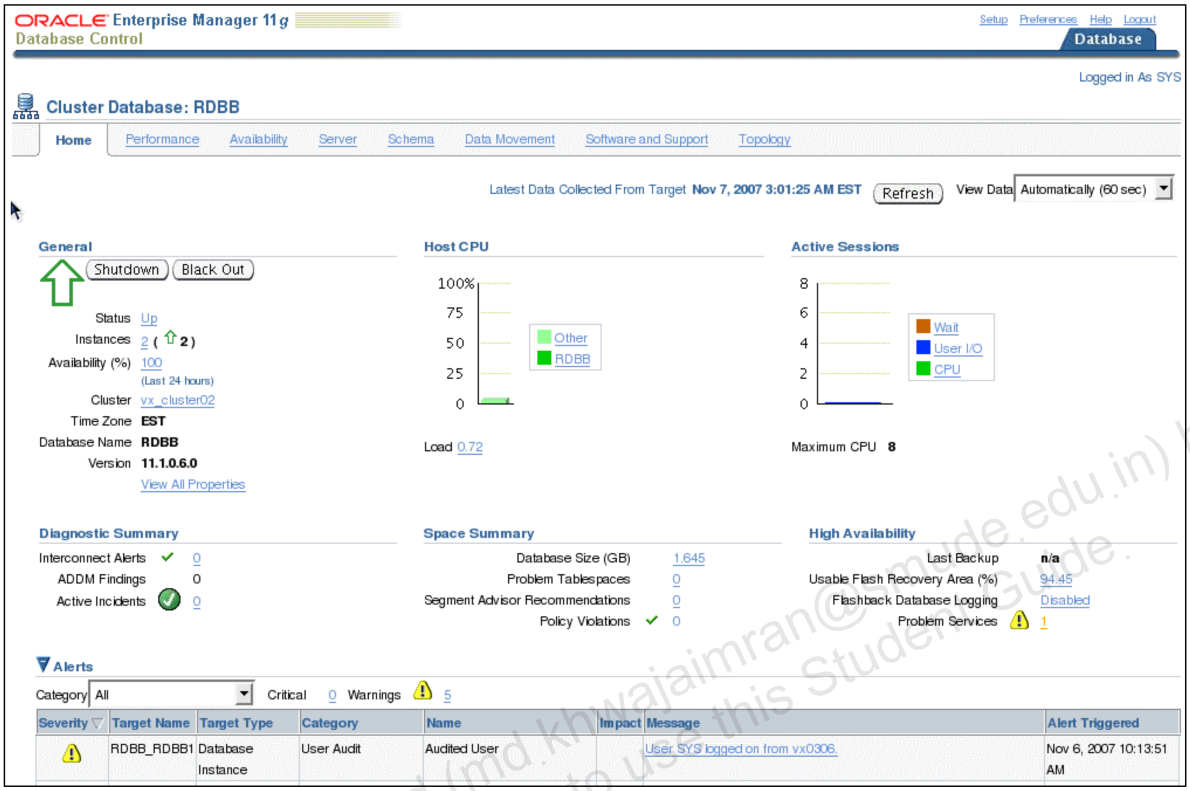
RAC Database Administration

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
- Manage ASM instances in a RAC environment

Cluster Database Home Page



Cluster Database Home Page

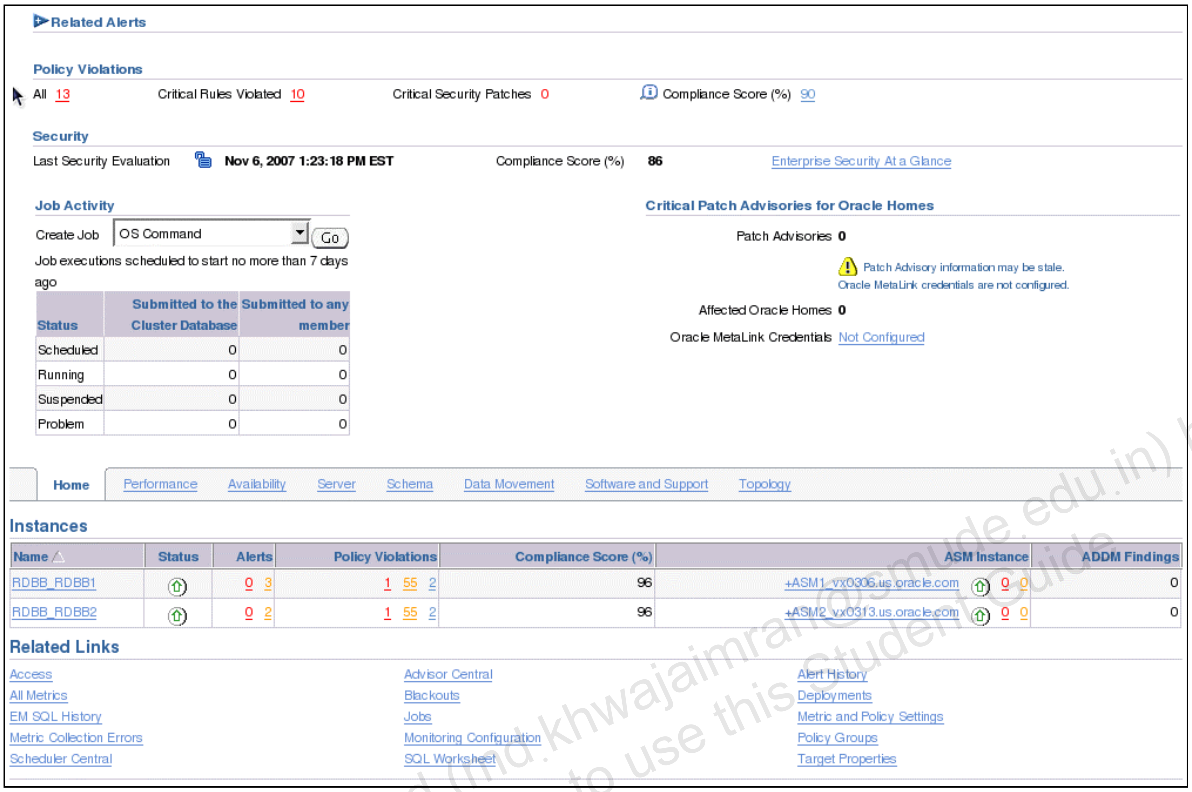
The Cluster Database home page serves as a crossroad for managing and monitoring all aspects of your RAC database. From this page, you can access the other main cluster database tabs: Performance, Availability, Server, Schema, Data Movement, Software and Support, and Topology.

On this page, you find General, High Availability, Space Summary, and Diagnostic Summary sections for information that pertains to your cluster database as a whole. The number of instances is displayed for the RAC database, in addition to the status. A RAC database is considered to be up if at least one instance has the database open. You can access the Cluster home page by clicking the Cluster link in the General section of the page.

Other items of interest include the date of the last RMAN backup, archiving information, space utilization, and an alert summary. By clicking the link next to the Flashback Database Logging label, you can go to the Recovery Settings page from where you can change various recovery parameters.

The Alerts table shows all open recent alerts. 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



ORACLE

Cluster Database Home Page (continued)

The Related Alerts table provides information about alerts for related targets, such as Listeners and Hosts, and contains details about the message, the time the alert was triggered, the value, and the time the alert was last checked.

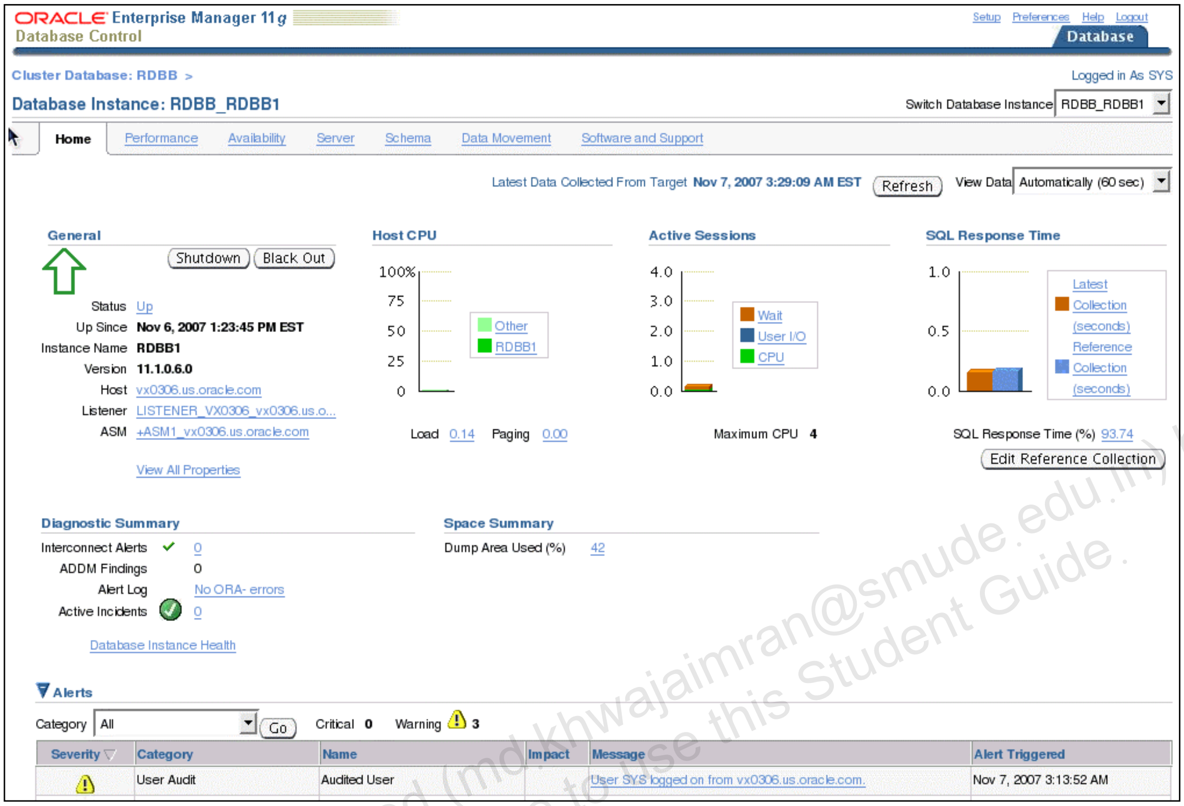
The Policy Trend Overview page (accessed by clicking the Compliance Score link) provides a comprehensive view about a group or targets containing other targets with regard to compliance over a period of time. Using the tables and graphs, you can easily watch for trends in progress and changes.

The Security At a Glance page shows an overview of the security health of the enterprise for all the targets or specific groups. This helps you to quickly focus on security issues by showing statistics about security policy violations and noting the critical security patches that have not been applied.

The Job Activity table displays a report of the job executions that shows the scheduled, running, suspended, and problem (stopped/failed) executions for all Enterprise Manager jobs on the cluster database.

The Instances table lists the instances for the cluster database, their availability, alerts, policy violations, performance findings, and related ASM Instance. Click an instance name to go to the home page for that instance. Click the links in the table to get more information about a particular alert, advice, or metric.

Cluster Database Instance Home Page



Cluster Database Instance Home Page

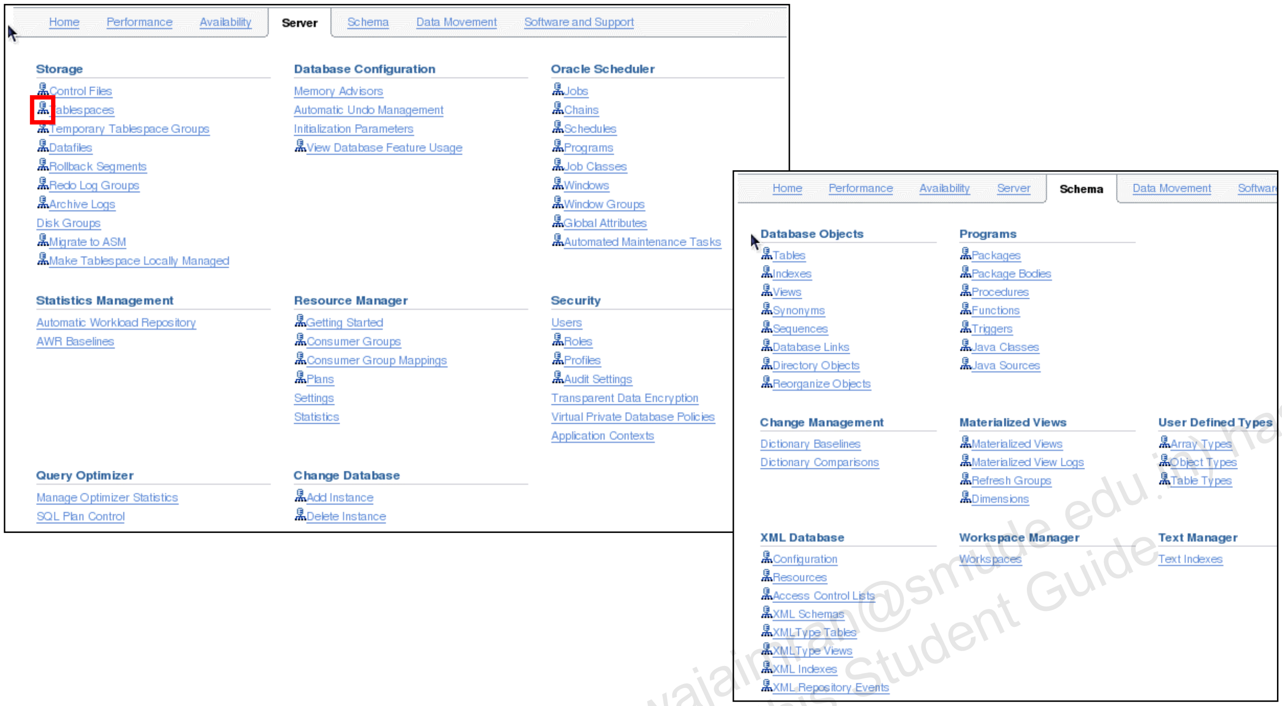
The Cluster Database Instance home page enables you to view the current state of the instance by displaying a series of metrics that portray its overall health. This page provides a launch point for the performance, administration, and maintenance of the instance environment.

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. For example, clicking the Shutdown button from this page shuts down only this one instance. However, clicking the Shutdown button from the Cluster Database home page gives you the option of shutting down all or specific instances.

By scrolling down on this page, you see the Alerts, Related Alerts, Policy Violations, Jobs Activity, and Related Links sections. These provide similar information similar to that provided in the same sections in the Cluster Database home page.

Cluster Database Instance Administration Page

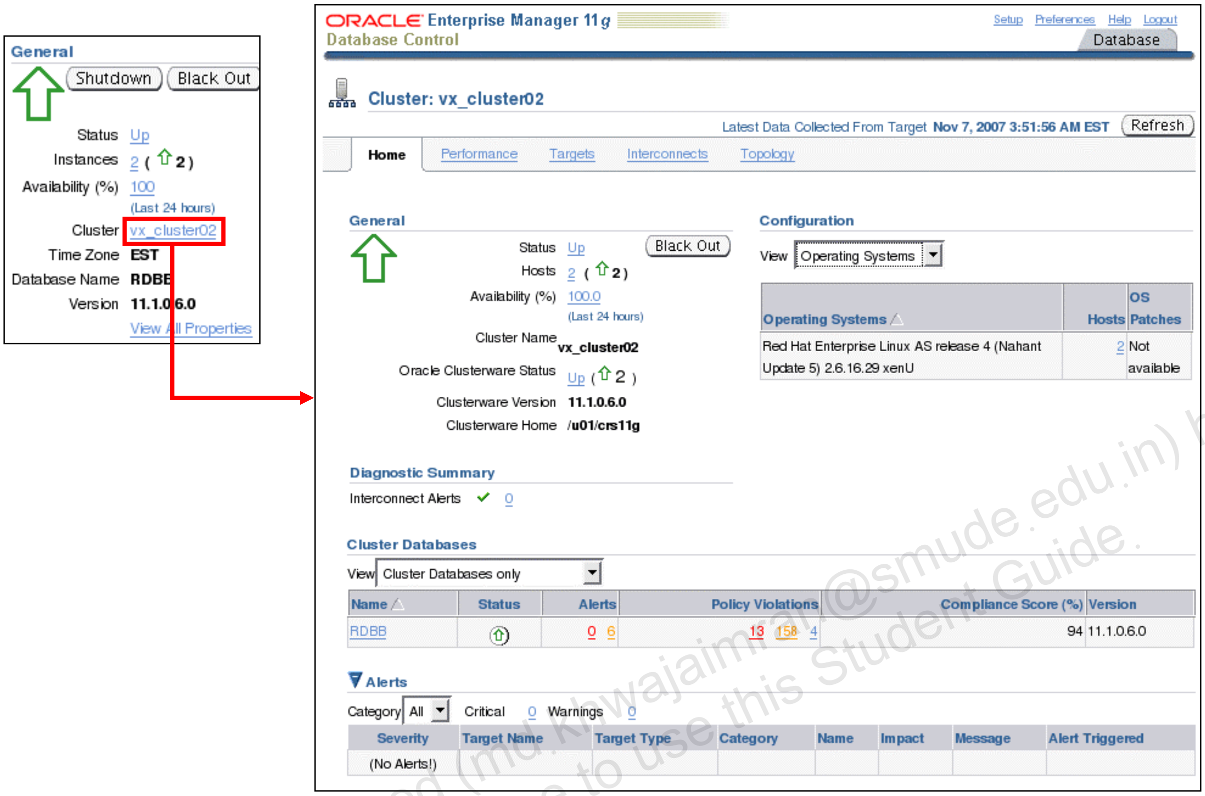


ORACLE

Cluster Database Instance Administration Page

This is the classical Administration page with an important difference with its corresponding single-instance version. As you can see it on the screenshot, each database-level related task is prefixed with a small icon representing a cluster database.

Cluster Home Page



ORACLE

Cluster Home Page

The slide shows you the Cluster home page, which can be accessed by clicking the Cluster link located in the General section of the Cluster Database home page. 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, Configuration, Cluster Databases, Alerts, and Hosts.

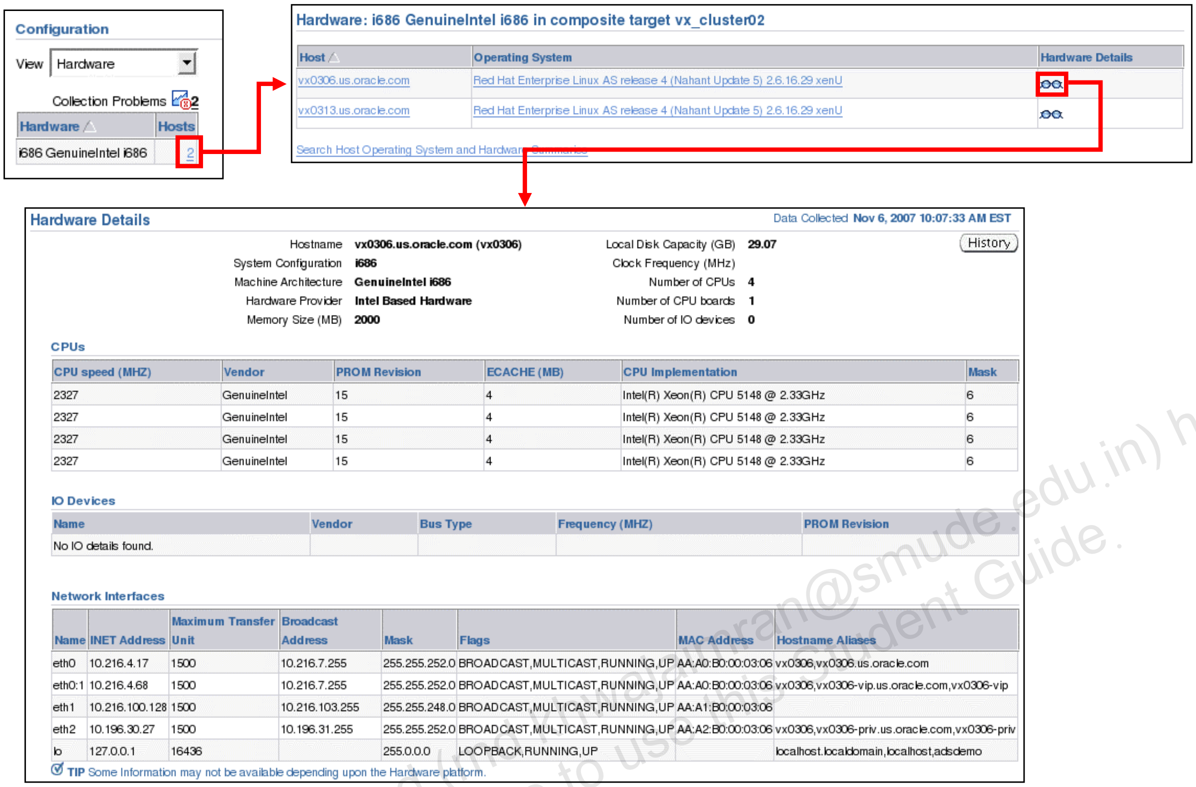
The General section provides a quick view of the status of the cluster, providing basic information such as current Status, Availability, Up nodes, and Clusterware Home and Version.

The Configuration section allows you to view the operating systems (including Hosts and OS Patches) and hardware (including Hardware configuration and Hosts) for the cluster.

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 Alerts table provides information about any alerts that have been issued along with the severity rating of each.

The Hosts table (not shown on the screenshot) displays the hosts for the cluster, their availability, corresponding alerts, CPU and memory utilization percentage, and total I/O per second.

The Configuration Section



ORACLE

The Configuration Section

The Cluster home page is invaluable for locating configuration-specific data. Locate the Configuration section on the Cluster home page. The View drop-down list allows you to inspect hardware and operating system overview information.

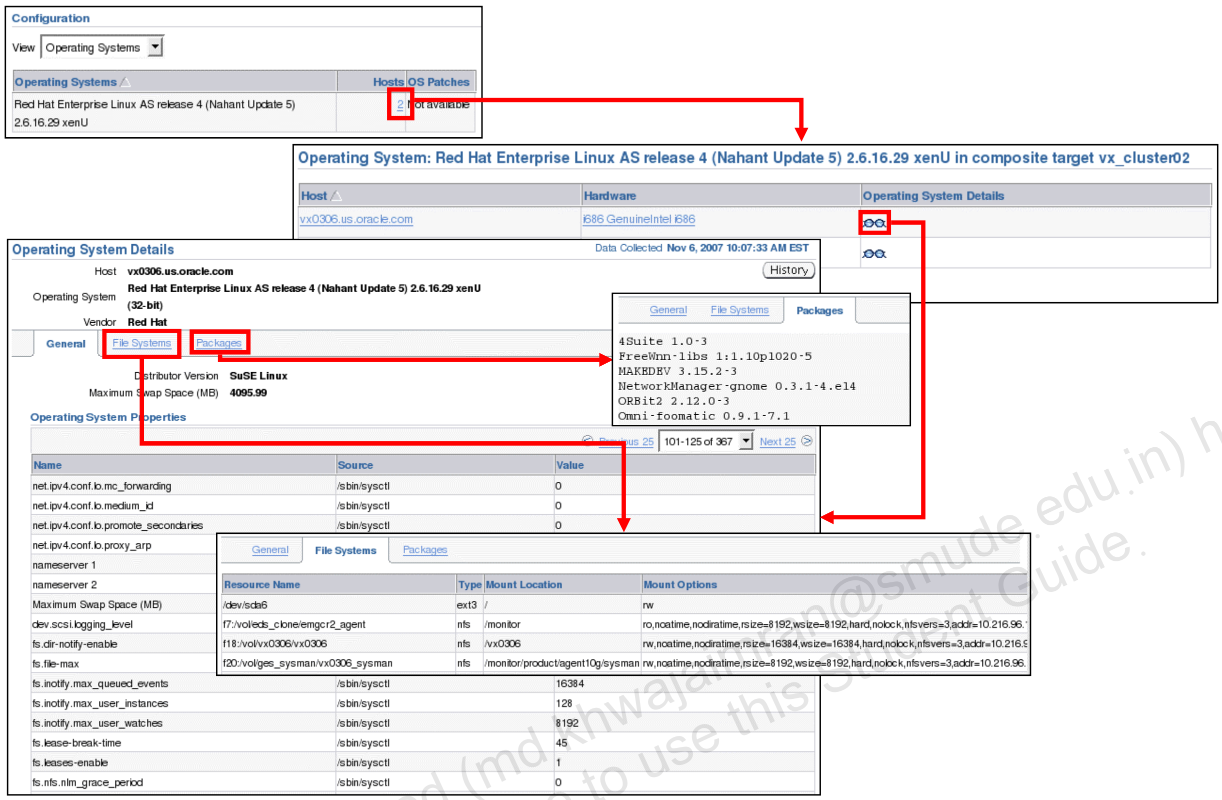
Click the Hosts link, and then click the Hardware Details link of the host that you want. On the Hardware Details page, you find detailed information regarding your CPU, disk controllers, network adapters, and so on. This information can be very useful when determining the Linux patches for your platform.

Click History to access the hardware history information for the host.

Some hardware information is not available, depending on the hardware platform.

Note: The Local Disk Capacity (GB) field shows the disk space that is physically attached (local) to the host. This value does not include disk space that may be available to the host through networked file systems.

The Configuration Section



ORACLE

The Configuration Section (continued)

The Operating System Details General page displays operating system details for a host, including:

- General information, such as the distributor version and the maximum swap space of the operating system
- Information about operating system properties

The Source column displays where Enterprise Manager obtained the value for each operating system property.

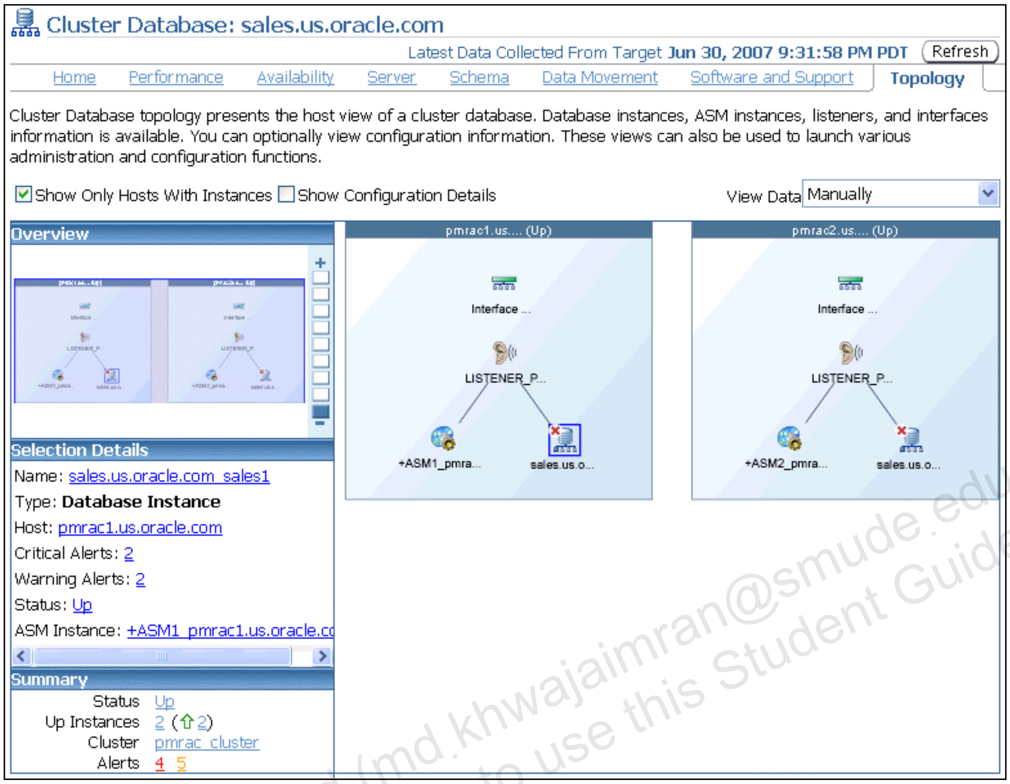
To see a list of changes to the operating system properties, click History.

The Operating System Details File Systems page displays information about one or more file systems for the selected hosts:

- Name of the file system on the host
- Type of mounted file system, for example, ufs or nfs
- Directory where the file system is mounted
- The mount options for the file system, for example ro, nosuid, or nobrowse

The Operating System Details Packages page displays information about the operating system packages that have been installed on a host.

Topology Viewer



Topology Viewer

The Oracle Enterprise Manager 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.

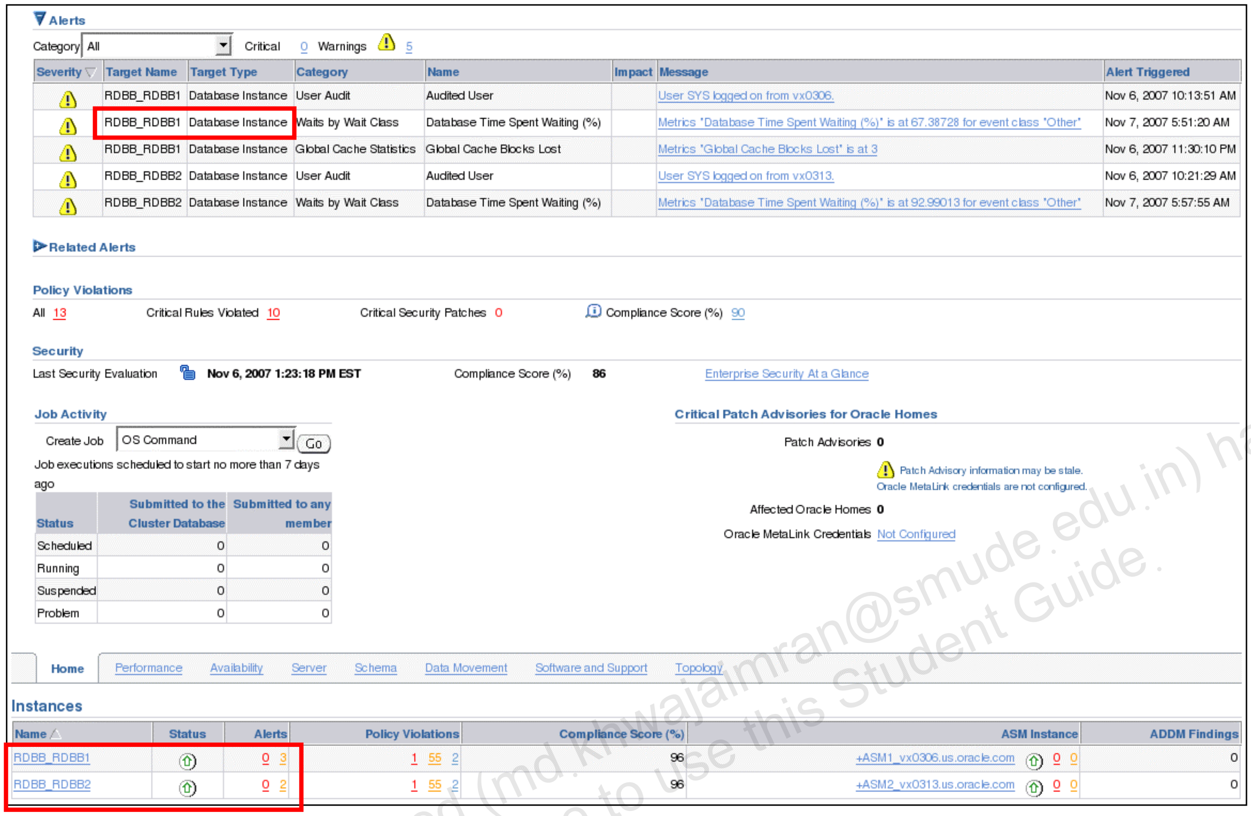
The Topology Viewer populates icons on the basis of your system configuration. If a listener is serving an instance, a line connects the listener icon and the instance icon. Possible target types are:

- Interface
- Listener
- ASM Instance
- Database Instance

If the Show Configuration Details option is not selected, the topology shows the monitoring view of the environment, which includes general information such as alerts and overall status. If you select the Show Configuration Details option, additional details are shown in the Selection Details window, which are valid for any topology view. For instance, the Listener component would also show the machine name and port number.

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

Enterprise Manager Alerts and RAC

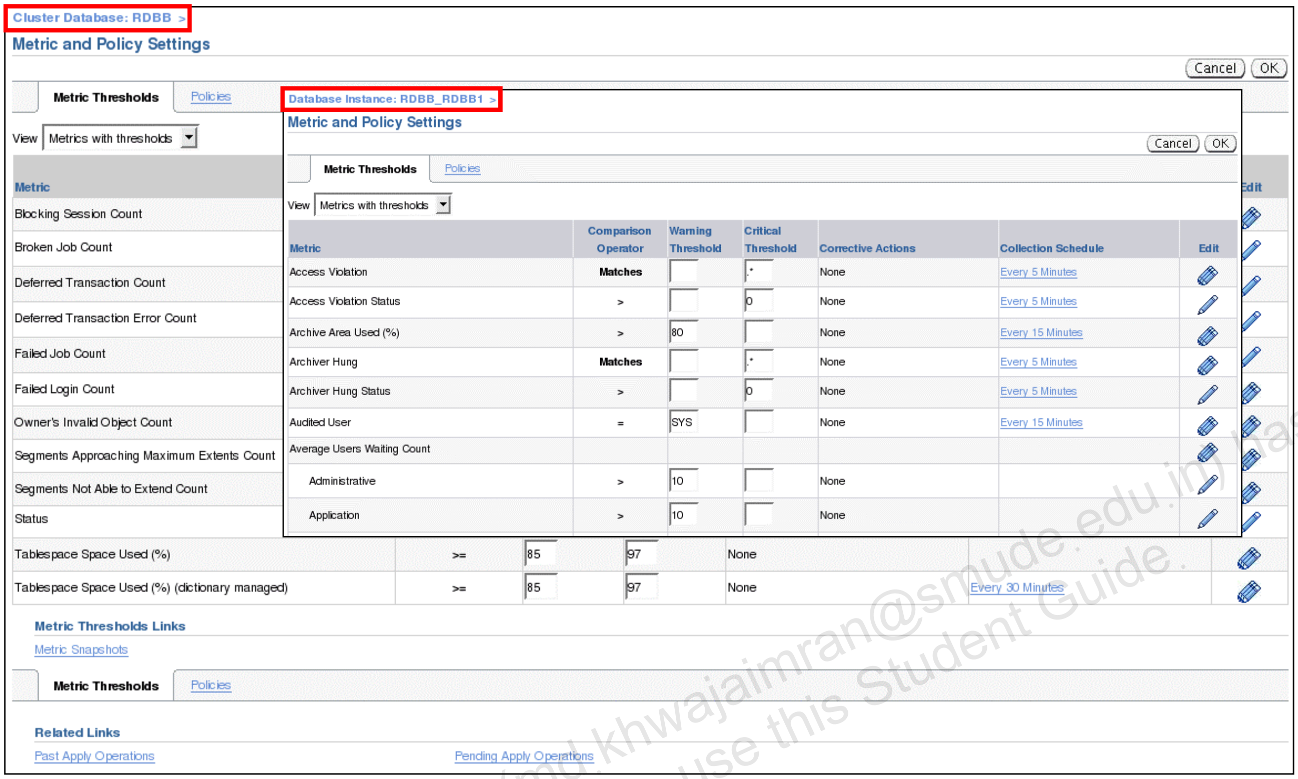


Enterprise Manager Alerts and RAC

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 Database Control home page. Notice that alerts are sorted by relative time and target name.

Enterprise Manager Metrics and RAC

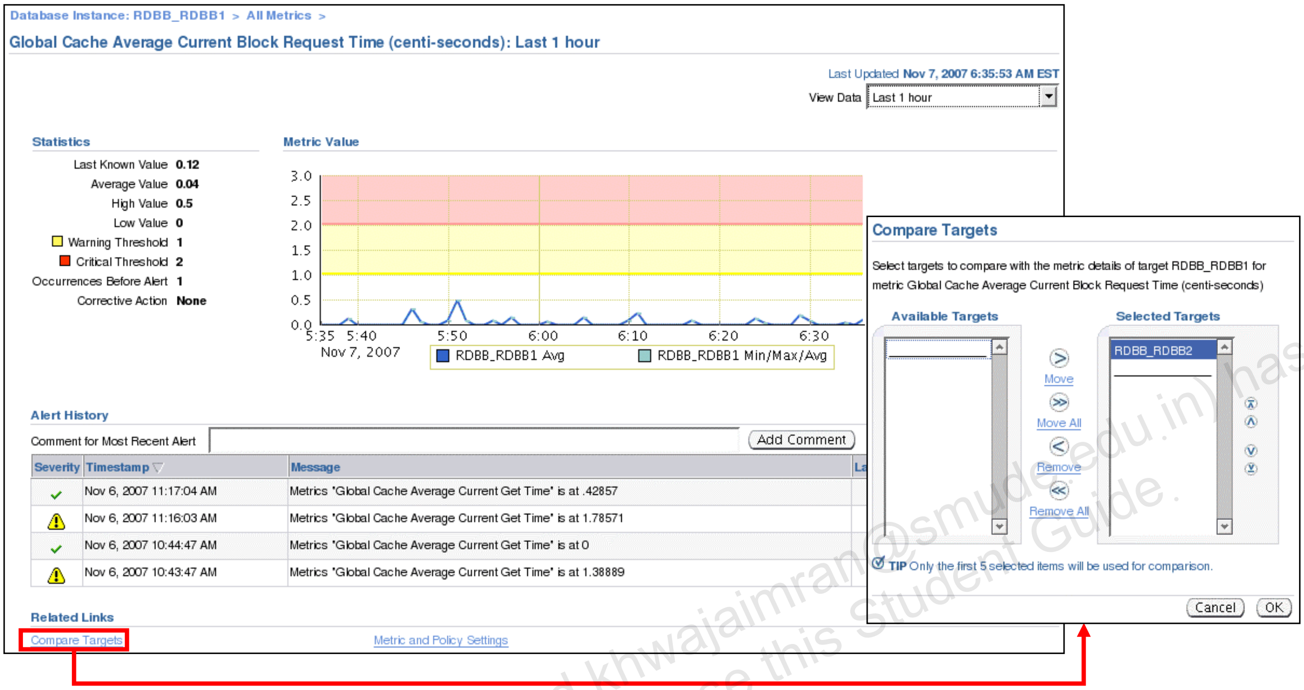


ORACLE

Enterprise Manager Metrics and RAC

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 Metrics and RAC

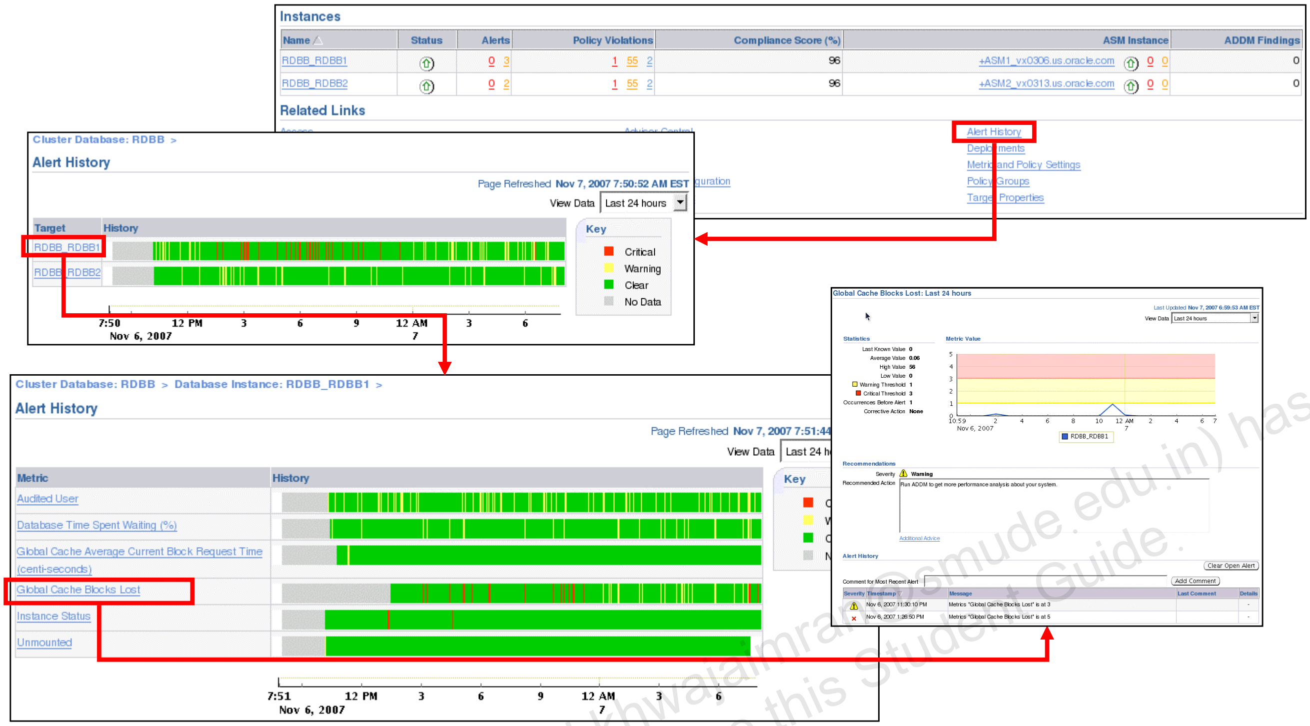


Enterprise Manager Metrics and RAC (continued)

It is also possible to view the metric across the cluster in a comparative or overlay fashion. To view this information, click the Compare Targets link at the bottom of the corresponding metric page. When the Compare Targets page appears, choose the instance targets that you want to compare by selecting them and then clicking the Move button. If you want to compare the metric data from all targets, then click the Move All button. After making your selections, click the OK button to continue.

The Metric summary page appears next. Depending on your needs, you can accept the default timeline of 24 hours or select a more suitable value from the View Data drop-down list. If you want to add a comment regarding the event for future reference, then enter a comment in the Comment for Most Recent Alert field, and then click the Add Comment button.

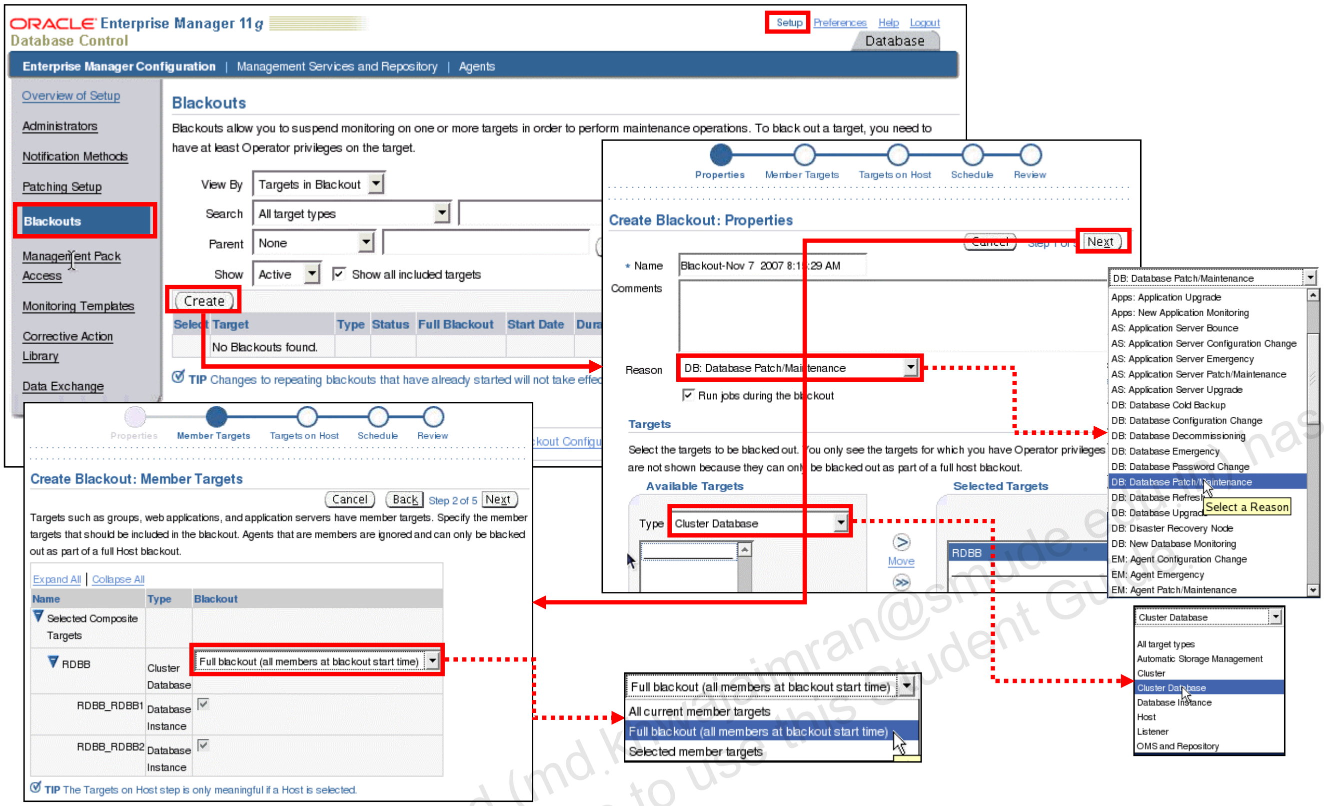
Enterprise Manager Alert History and RAC



Enterprise Manager Alert History and RAC

In a RAC environment, you can see a summary of the alert history for each participating instance directly from the Cluster Database home page. The drill-down process is shown in the slide. You click the Alert History link in the Related Links section of the Cluster Database home page. This takes you to the Alert History page on which you can see the summary for both instances in the example. You can then click one of the instance's links to go to the corresponding Alert History page for that instance. From there, you can access a corresponding alert page by choosing the alert of your choice.

Enterprise Manager Blackouts and RAC



Enterprise Manager Blackouts and RAC

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.

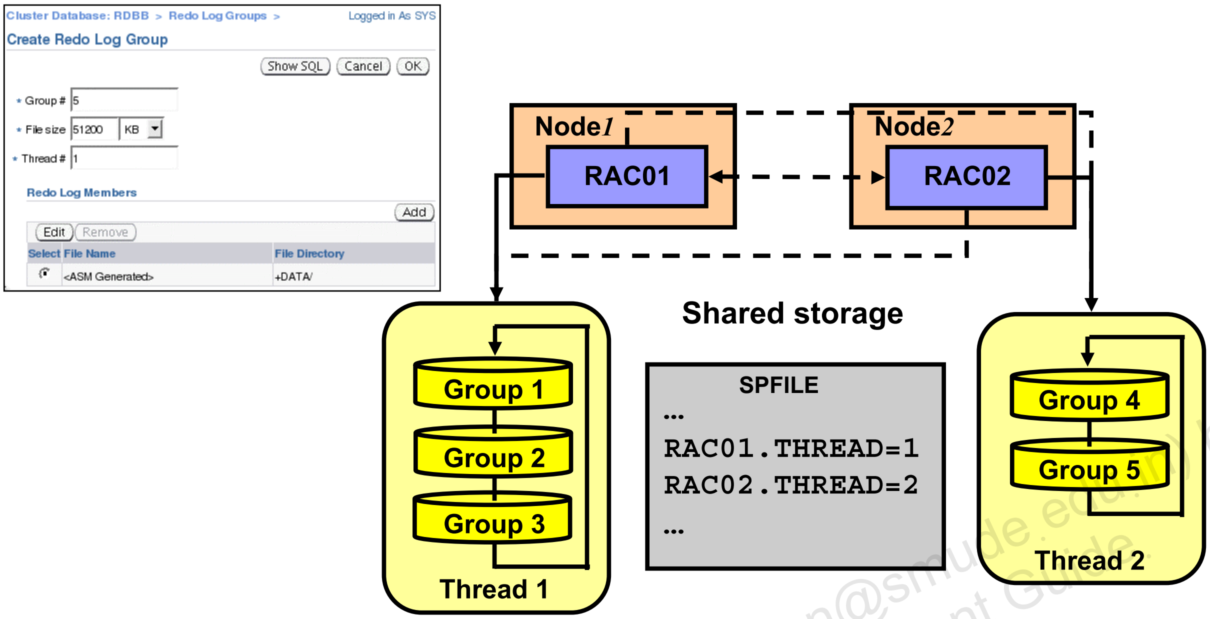
To create a blackout event, click the Setup link on top of any Enterprise Manager page. Then, click the Blackouts link on the left. The Blackouts page appears.

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. In the example in the slide, the entire cluster database RDBB is chosen. 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 as you normally would.

Redo Log Files and RAC




```
ALTER DATABASE ADD LOGFILE THREAD 2 GROUP 4;
ALTER DATABASE ADD LOGFILE THREAD 2 GROUP 5;
```

```
ALTER DATABASE ENABLE THREAD 2;
```


Redo Log Files and RAC

With Real Application Clusters (RAC), each instance writes to its own set of online redo log files, and the redo written by an instance is called a thread of redo, or thread. Thus, each redo log file group used by an instance is associated with the same thread number determined by the value of the `THREAD` initialization parameter. If you set the `THREAD` parameter to a nonzero value for a particular instance, the next time the instance is started, it will try to use that thread. Because an instance can use a thread as long as that thread is enabled, and not in use by another instance, it is recommended to set the `THREAD` parameter to a nonzero value with each instance having different values.

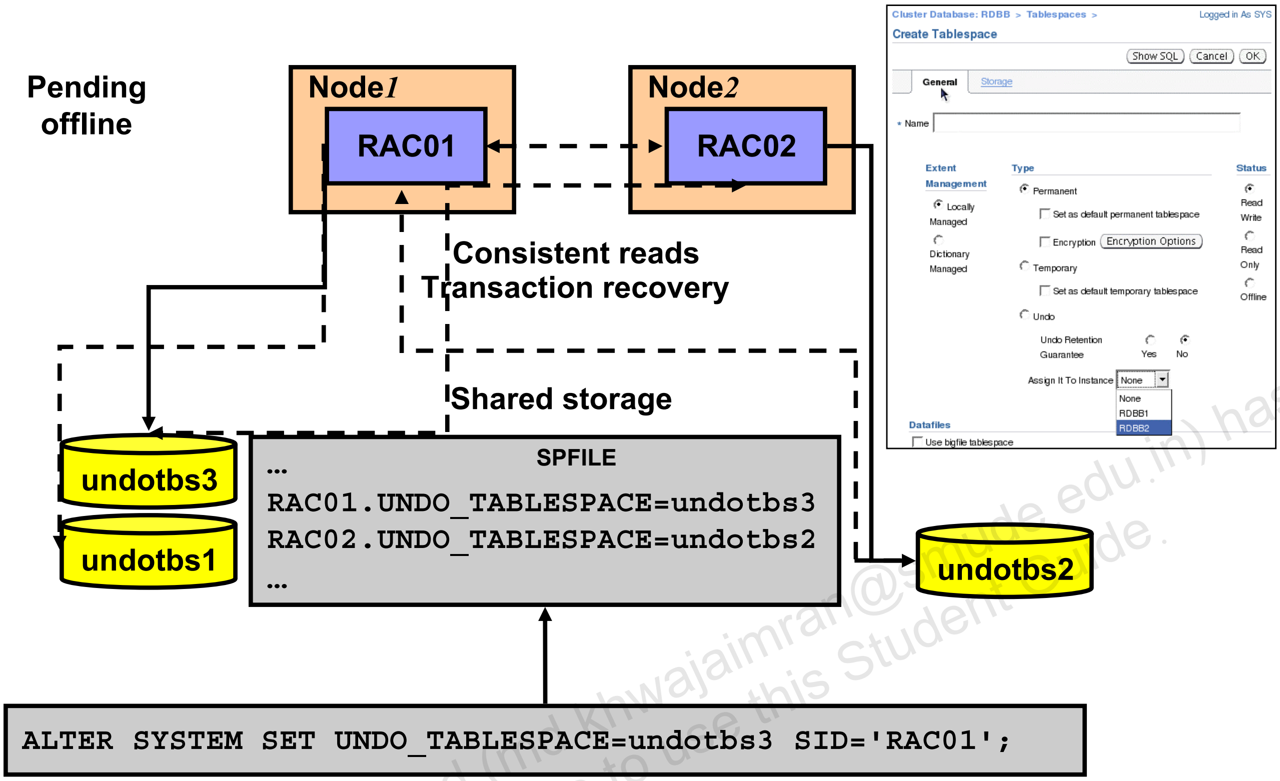
You associate a thread number with a redo log file group by using the `ALTER DATABASE ADD LOGFILE THREAD` statement. You enable a thread number by using the `ALTER DATABASE ENABLE THREAD` statement. Before you can enable a thread, it must have at least two redo log file groups.

By default, a database is created with one enabled public thread. An enabled public thread is a thread that has been enabled by using the `ALTER DATABASE ENABLE PUBLIC THREAD` statement. Such a thread can be acquired by an instance with its `THREAD` parameter set to zero. Therefore, you need to create and enable additional threads when you add instances to your database.

The maximum possible value for the `THREAD` parameter is the value assigned to the `MAXINSTANCES` parameter specified in the `CREATE DATABASE` statement.

Note: You can use Enterprise Manager to administer redo log groups in a RAC environment.

Automatic Undo Management and RAC



Automatic Undo Management in RAC

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 RAC database by specifying a different value for the `UNDO_TABLESPACE` parameter for each instance in your `SPFILE` or individual `PFILE`s. If you do not set the `UNDO_TABLESPACE` parameter, then each instance uses the first available undo tablespace. If undo tablespaces are not available, the `SYSTEM` rollback segment is used.

You can dynamically switch undo tablespace assignments by executing the `ALTER SYSTEM SET UNDO_TABLESPACE` statement with the `SID` clause. You can run this command from any instance. In the example shown in the slide, the previously used undo tablespace assigned to instance `RAC01` remains assigned to it until the `RAC01` instance's last active transaction commits. The pending offline tablespace may be unavailable for other instances until all transactions against that tablespace are committed.

Note: You cannot simultaneously use Automatic Undo Management (AUM) and manual undo management in a RAC database. It is highly recommended that you use the AUM mode.

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
  - Server Control (`SRVCTL`) utility
  - SQL*Plus
- Shutting down a RAC database means shutting down all instances accessing the database.

ORACLE

Starting and Stopping RAC Instances

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, then 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 SQL*Plus

```
[stc-raclin01] $ echo $ORACLE_SID
RACDB1
sqlplus / as sysdba
SQL> startup
SQL> shutdown

[stc-raclin02] $ echo $ORACLE_SID
RACDB2
sqlplus / as sysdba
SQL> startup
SQL> shutdown
```

OR

```
[stc-raclin01] $sqlplus / as sysdba
SQL> startup
SQL> shutdown
SQL> connect sys/oracle@RACDB2 as sysdba
SQL> startup
SQL> shutdown
```


Starting and Stopping RAC Instances with SQL*Plus

If you want to start up or shut down only one instance, and if you are connected to your local node, you must first ensure that your current environment includes the SID for the local instance.

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.

The example in the slide assumes that the alias name for the second instance is `RACDB2`. In the example, there is no need to connect to the first instance using its connect descriptor because the command is issued from the first node with the correct `ORACLE_SID`.

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 during startup and shutdown of instances by using SQL*Plus.

Starting and Stopping RAC Instances with SRVCTL

- `start/stop` syntax:

```
srvctl start|stop instance -d <db_name> -i <inst_name_list>
[-o open|mount|nomount|normal|transactional|immediate|abort>]
[-c <connect_str> | -q]
```

```
srvctl start|stop database -d <db_name>
[-o open|mount|nomount|normal|transactional|immediate|abort>]
[-c <connect_str> | -q]
```

- Examples:

```
$ srvctl start instance -d RACDB -i RACDB1,RACDB2
```

```
$ srvctl stop instance -d RACDB -i RACDB1,RACDB2
```

```
$ srvctl start database -d RACDB -o open
```


Starting and Stopping RAC Instances with SRVCTL

The `srvctl start database` command starts a cluster database, its enabled instances, and its 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 as well as 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 (OC) can restart it as a result of another planned operation. For commands that use a connect string, if you do not provide a connect string, then `SRVCTL` uses `/ as sysdba` to perform the operation. The `-q` option asks for a connect string from standard input. `SRVCTL` does not support concurrent executions of commands on the same object. Therefore, run only one `SRVCTL` command at a time for each database, service, or other object. |

To use the `START` and `STOP` options of the `SRVCTL` command, your service must be an OC-enabled, nonrunning service.

Note: For more information, see the *Oracle Clusterware Administration and Deployment Guide* and the *Oracle Real Application Clusters Administration and Deployment Guide*.

Switch Between the Automatic and Manual Policies

```
$ srvctl config database -d RACB -a
ex0044 RACB1 /u01/app/oracle/product/10.2.0/db_1
ex0045 RACB2 /u01/app/oracle/product/10.2.0/db_1
DB_NAME: RACB
ORACLE_HOME: /u01/app/oracle/product/10.2.0/db_1
SPFILE: +DGDB/RACB/spfileRACB.ora
DOMAIN: null
DB_ROLE: null
START_OPTIONS: null
POLICY:  AUTOMATIC
ENABLE FLAG: DB ENABLED
$
```

```
srvctl modify database -d RACB -y MANUAL;
```

ORACLE

Switch Between the Automatic and Manual Policies

By default, Oracle Clusterware is configured to start the VIP, listener, instance, ASM, database services, and other resources during system boot. It is possible to modify some resources to have their `AUTO_START` profile parameter set to the value `2`. This means that after node reboot, or when Oracle Clusterware is started, resources with `AUTO_START=2` need to be started manually via `srvctl`. This is designed to assist in problem troubleshooting and system maintenance. Starting with Oracle Database 10*g* Release 2, when changing resource profiles through `srvctl`, the command tool automatically modifies the profile attributes of other dependent resources given the current prebuilt dependencies. The command to accomplish this is:

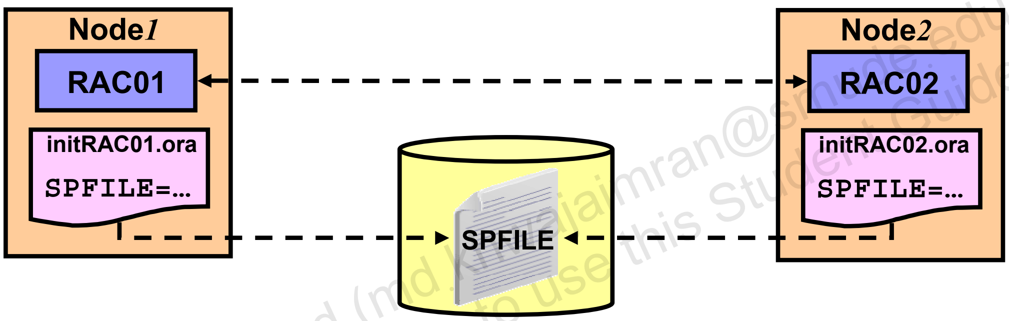
```
srvctl modify database -d <dbname> -y AUTOMATIC|MANUAL
```

To implement Oracle Clusterware and Real Application Clusters, it is best to have Oracle Clusterware start the defined Oracle resources during system boot, which is the default. The first example in the slide uses the `srvctl config database` command to display the current policy for the `RACB` database. As you can see, it is currently set to its default: `AUTOMATIC`. The second statement uses the `srvctl modify database` command to change the current policy to `MANUAL` for the `RACB` database. When you add a new database by using the `srvctl add database` command, that database is by default placed under the control of Oracle Clusterware using the `AUTOMATIC` policy. However, you can use the following statement to directly set the policy to `MANUAL`: `srvctl add database -d RACZ -y MANUAL`.

Note: You can also use this procedure to configure your system to prevent Oracle Clusterware from autorestarting failed database instances more than once.

RAC Initialization Parameter Files

- An SPFILE is created if you use the DBCA.
- The SPFILE must be created on a shared volume or shared raw device.
- All instances use the same SPFILE.
- If the database is created manually, then create an SPFILE from a PFILE.



ORACLE

Initialization Parameter Files

When you create the database, the DBCA creates an SPFILE in the file location that you specify. This location can be an Automatic Storage Management (ASM) disk group, cluster file system file, or a shared raw device. If you manually create your database, then it is recommended to create an SPFILE from a PFILE.

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

RAC uses a traditional PFILE only if an SPFILE does not exist or if you specify PFILE in your STARTUP command. Using SPFILE simplifies administration, maintaining parameter settings consistent, and guarantees parameter settings persistence across database shutdown and startup. In addition, you can configure RMAN to back up your SPFILE.

In order for each instance to use the same SPFILE at startup, each instance uses its own PFILE file that contains only one parameter called SPFILE. The SPFILE parameter points to the shared SPFILE on your shared storage. This is illustrated in the slide. By naming each PFILE using the init<SID>.ora format, and by putting them in the $ORACLE_HOME/dbs directory of each node, a STARTUP command uses the shared SPFILE.

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|*>';
```

ORACLE

SPFILE Parameter Values and RAC

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.

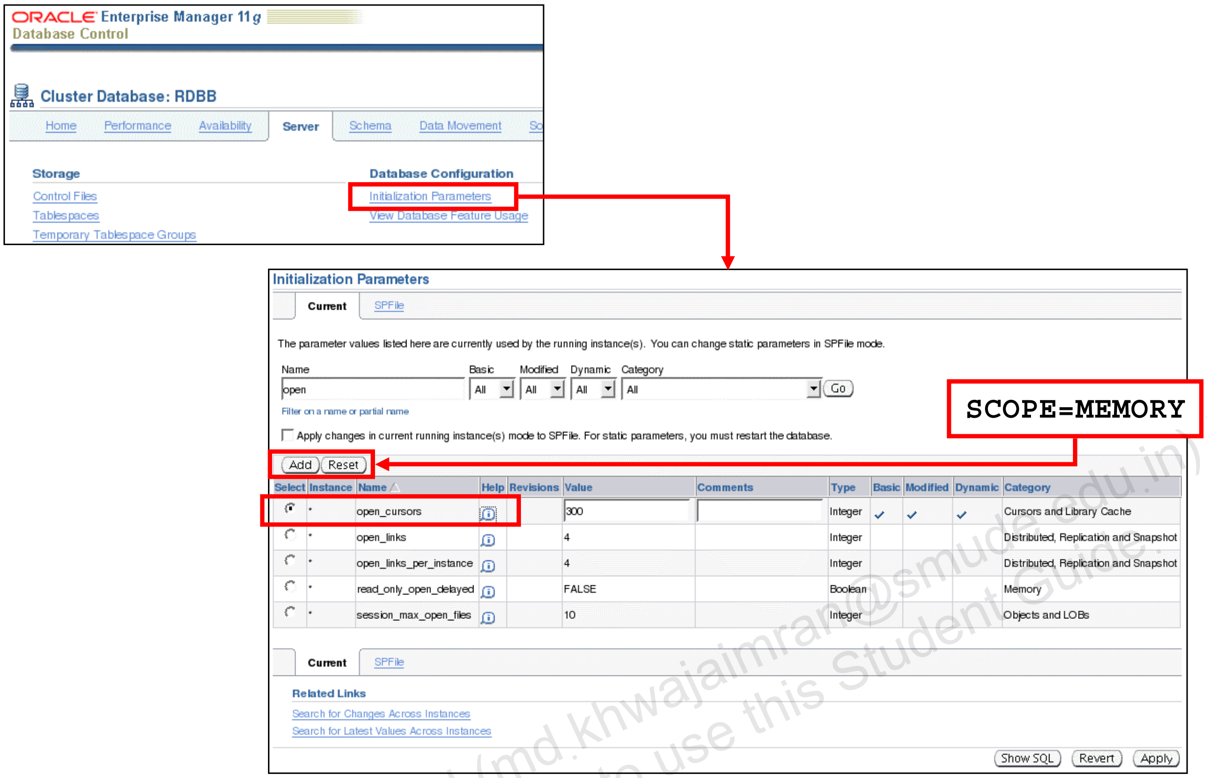
By using the SID clause, you can specify the SID of the instance where the value takes effect. Specify SID='*' if you want to change the value of the parameter for all instances. Specify SID='*sid*' if you want to change the value of the parameter only for the instance *sid*. This setting takes precedence 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.

EM and `SPFILE` Parameter Values



ORACLE

EM and `SPFILE` Parameter Values

You can access the Initialization Parameters page by clicking the Initialization Parameters link on the Cluster Database Server page.

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.

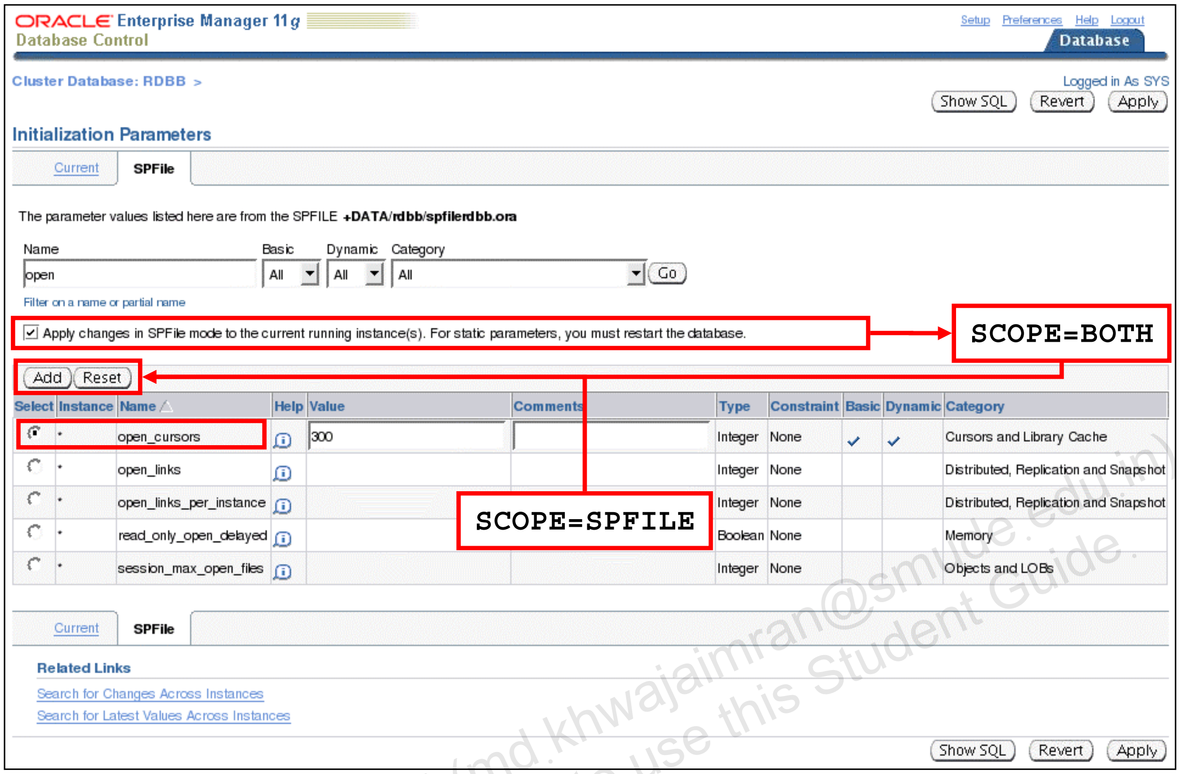
The Instance column shows the instances for which the parameter has the value listed in the table. An asterisk (`*`) indicates that the parameter has the same value for all remaining instances of the cluster database.

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 may 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



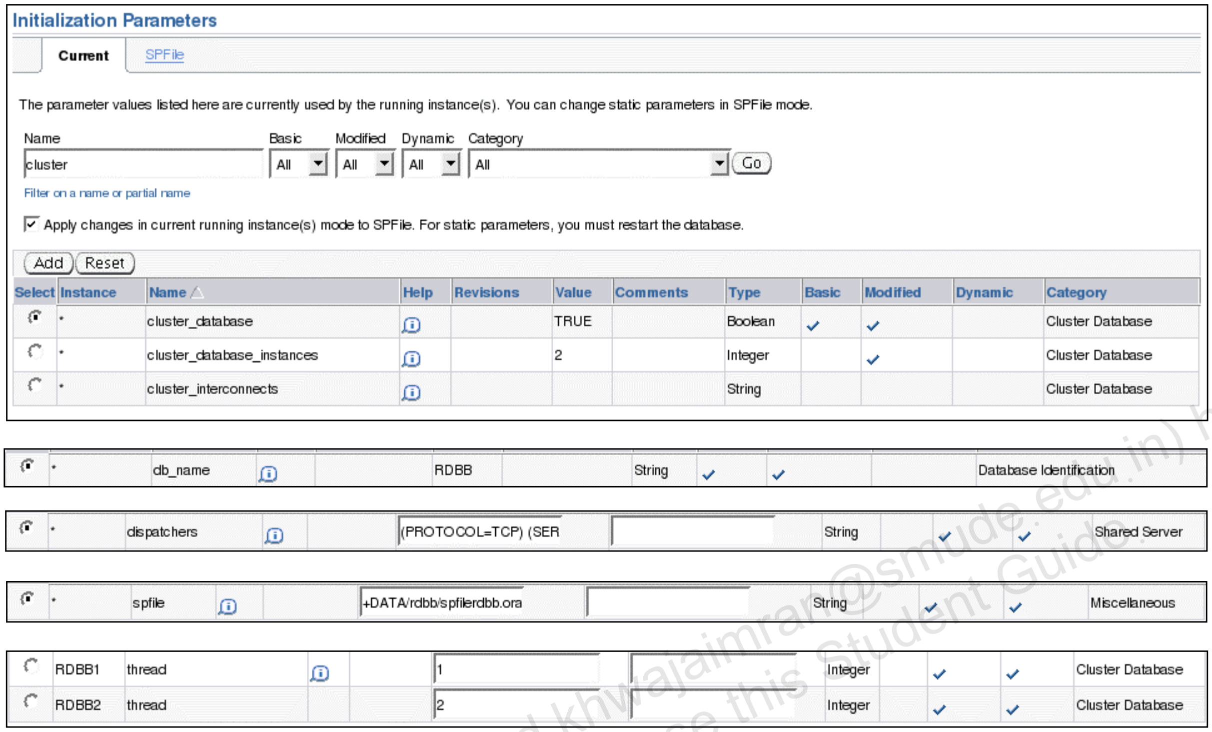
EM and SPFILE Parameter Values (continued)

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, then 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



ORACLE

RAC Initialization Parameters

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, then 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.

RAC Initialization Parameters (continued)

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.

MAX_COMMIT_PROPAGATION_DELAY: This is a RAC-specific parameter. Starting with Oracle Database 10*g* Release 2, the MAX_COMMIT_PROPAGATION_DELAY parameter is deprecated. By default, commits on one instance are immediately visible on all of the other instances (broadcast on commit propagation). This parameter is retained for backward compatibility only. This parameter specifies the maximum amount of time allowed before the system change number (SCN) held in the System Global Area (SGA) of an instance is refreshed by the log writer process (LGWR). It determines whether the local SCN should be refreshed from the SGA when getting the snapshot SCN for a query. With previous releases, you should not alter the default setting for this parameter except under a limited set of circumstances. For example, under unusual circumstances involving rapid updates and queries of the same data from different instances, the SCN might not be refreshed in a timely manner.

SPFILE: When you use an SPFILE, all RAC database instances must use the SPFILE and the file must be on shared storage.

THREAD: If specified, this parameter must have unique values on all instances. The THREAD parameter specifies the number of the redo thread to be used by an instance. You can specify any available redo thread number as long as that thread number is enabled and is not used.

Parameters That Require Identical Settings

- ACTIVE_INSTANCE_COUNT
- ARCHIVE_LAG_TARGET
- COMPATIBLE
- CLUSTER_DATABASE/CLUSTER_DATABASE_INSTANCE
- 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
- PARALLEL_MAX_SERVERS
- REMOTE_LOGIN_PASSWORD_FILE
- TRACE_ENABLED
- UNDO_MANAGEMENT

Parameters That Require Identical Settings

Certain initialization parameters that are critical at database creation or that affect certain database operations must have the same value for every instance in RAC. Specify these parameter values in the `SPFILE`, or in each `init_`*`dbname`*`.ora` file on each instance. In the list in the slide, each parameter must have the same value on all instances.

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:

- `THREAD`
- `ROLLBACK_SEGMENTS`
- `INSTANCE_NAME`
- `ASM_PREFERRED_READ_FAILURE_GROUPS`
- `INSTANCE_NUMBER`
- `UNDO_TABLESPACE`

RDBB1	instance_name	RDBB1	String			Instance Identification
RDBB2	instance_name	RDBB2	String			Instance Identification
RDBB1	instance_number	1	Integer	✓	✓	Cluster Database
RDBB2	instance_number	2	Integer	✓	✓	Cluster Database

RDBB1	thread	1		Integer	✓	✓	Cluster Database
RDBB2	thread	2		Integer	✓	✓	Cluster Database

RDBB1	undo_tablespace	UNDOTBS1		String	✓	✓	✓	Automatic Undo Management
RDBB2	undo_tablespace	UNDOTBS2		String	✓	✓	✓	Automatic Undo Management

RDBB1	asm_preferred_read_failure_groups	DATA.SITEA		String	✓	Automatic Storage Management
RDBB2	asm_preferred_read_failure_groups	DATA.SITEB		String	✓	Automatic Storage Management

ORACLE

Parameters That Require Unique Settings

If you use the `THREAD` or `ROLLBACK_SEGMENTS` parameters, it is recommended that you set unique values for them 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.

The Oracle server uses the `INSTANCE_NUMBER` parameter to distinguish among instances at startup. The Oracle server uses the `THREAD` number to assign redo log groups to specific instances. To simplify administration, use the same number for both the `THREAD` and `INSTANCE_NUMBER` parameters.

If you specify `UNDO_TABLESPACE` with Automatic Undo Management enabled, set this parameter to a unique undo tablespace name for each instance.

Using the `ASM_PREFERRED_READ_FAILURE_GROUPS` initialization parameter, you can specify a list of preferred read failure group names. The disks in those failure groups become the preferred read disks. Thus, every node can read from its local disks. The setting for this parameter is instance specific, and the values do not need to be the same on all instances.

Quiescing RAC Databases

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

```
SQL> ALTER SYSTEM QUIESCE RESTRICTED;
```

- 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.

Quiescing RAC Databases

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 non-DBA sessions become inactive, the `ALTER SYSTEM QUIESCE RESTRICTED` statement executes and the database is considered to be quiesced. In a RAC environment, this statement affects all instances.

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='JFV';

       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
```


Terminating Sessions on a Specific Instance

Starting with Oracle RAC 11gR1, you can use the `ALTER SYSTEM KILL SESSION` statement to terminate a session on a specific instance.

The slide illustrates this by terminating a session started on a different instance than the one used to terminate the problematic session.

If the session is performing some activity that must be completed (such as waiting for a reply from a remote database or rolling back a transaction), 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, then 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.

Note: You can also use the `IMMEDIATE` clause at the end of the `ALTER SYSTEM` command to immediately terminate the session without waiting for outstanding activity to complete.

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
HOST	Affects the node running the SQL*Plus session
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

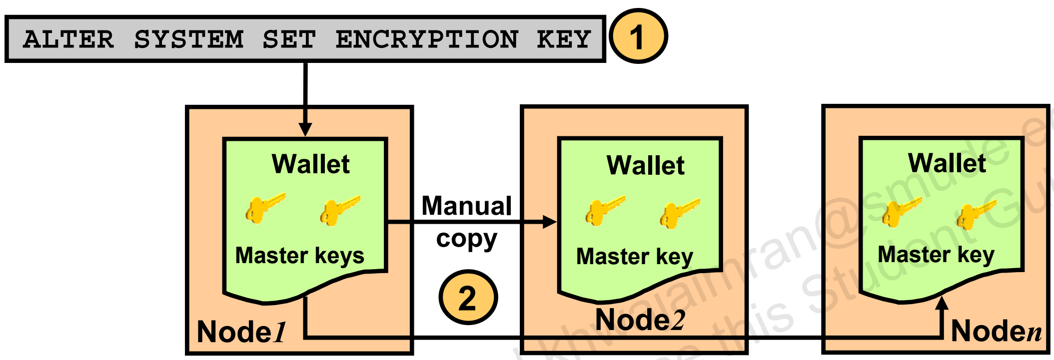
How SQL*Plus Commands Affect Instances

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

- The `ALTER SYSTEM SET CHECKPOINT LOCAL` statement affects only the instance to which you are currently connected, rather than the default instance or all instances.
- `ALTER SYSTEM CHECKPOINT LOCAL` affects the current instance.
- `ALTER SYSTEM CHECKPOINT` or `ALTER SYSTEM CHECKPOINT GLOBAL` affects 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.

Transparent Data Encryption and Wallets in RAC

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



Manual copy

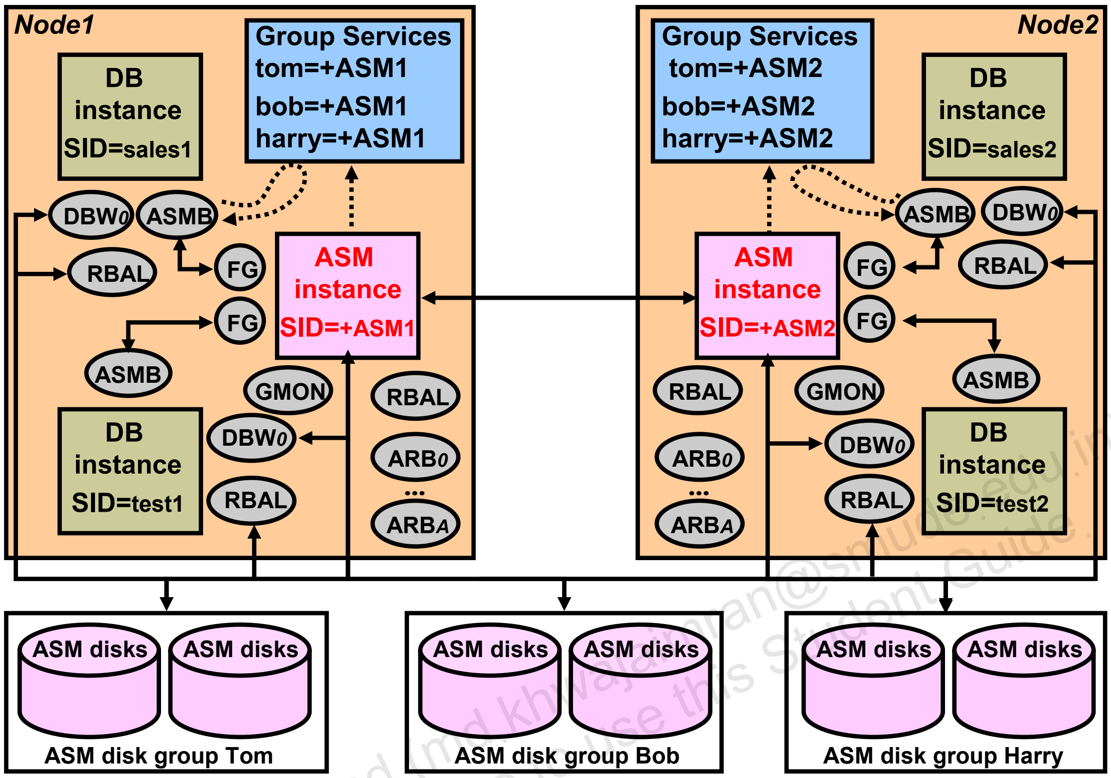
Transparent Data Encryption and Wallets in RAC

Wallets used by RAC instances for Transparent Database Encryption may be a local copy of a common wallet shared by multiple nodes, or a shared copy residing on shared storage that all of the nodes can access.

A deployment with a single wallet on a shared disk requires no additional configuration to use Transparent Data Encryption.

If you want to use local copies, you must copy the wallet and make it available to all of the other nodes after initial configuration. For systems using Transparent Data Encryption with encrypted wallets, you can use any standard file transport protocol. For systems using Transparent Data Encryption with obfuscated wallets, file transport through a secured channel is recommended. The wallet must reside in the directory specified by the setting for the `WALLET_LOCATION` or `ENCRYPTION_WALLET_LOCATION` parameter in `sqlnet.ora`. The local copies of the wallet need not be synchronized for the duration of Transparent Data Encryption usage until the server key is rekeyed through the `ALTER SYSTEM SET KEY` SQL statement. Each time you run the `ALTER SYSTEM SET KEY` statement at a database instance, you must again copy the wallet residing on that node and make it available to all of the other nodes. To avoid unnecessary administrative overhead, reserve rekeying for exceptional cases where you are certain that the server master key is compromised and that not rekeying it would cause a serious security problem. Note that RMAN does not implicitly backup wallets.

ASM: General Architecture



ASM: General Architecture

Automatic Storage Management (ASM) is part of the database kernel. One portion of the ASM code allows for the startup of a special instance called an *ASM instance*. ASM instances do not mount databases but instead manage the metadata needed to make ASM files available to ordinary database instances. Both ASM instances and database instances have access to a common set of disks called disk groups. Database instances access the contents of ASM files directly, communicating with an ASM instance only to obtain information about the layout of these files.

An ASM instance contains three new types of background processes. The first type is responsible for coordinating rebalance activity for disk groups, and is called `RBAL`. The second type actually performs the data extent movements. There can be many of these at a time, and they are called `ARB0`, `ARB1`, and so on. The third type is responsible for certain disk group–monitoring operations that maintain ASM metadata inside disk groups. The disk group monitor process is called `GMON`.

Each database instance that uses ASM has two new background processes called `ASMB` and `RBAL`. In a database instance, `RBAL` performs global opens of the disks in the disk groups. `ASMB` runs in database instances and connects to foreground processes in ASM instances. Over those connections, periodic messages are exchanged to update statistics and to verify that both instances are healthy. During operations that require ASM intervention, such as a file creation by a database foreground, the database foreground connects directly to the ASM instance to perform the operation.

ASM: General Architecture (continued)

An ASMB process is started dynamically when an ASM file is first accessed.

When started, the ASM background connects to the desired ASM instance and maintains that connection until the database instance no longer has any files open in the disk groups served by that ASM instance. Database instances are allowed to connect to only one ASM instance at a time, so they have at most one ASMB background process.

Like RAC, the ASM instances themselves may be clustered, using the existing Global Cache Services (GCS) infrastructure. There is usually one ASM instance per node on a cluster. As with existing RAC configurations, ASM requires that the operating system make the disks globally visible to all of the ASM instances, irrespective of node.

Database instances communicate only with ASM instances on the same node. If there are several database instances for different databases on the same node, they must share the same single ASM instance on that node.

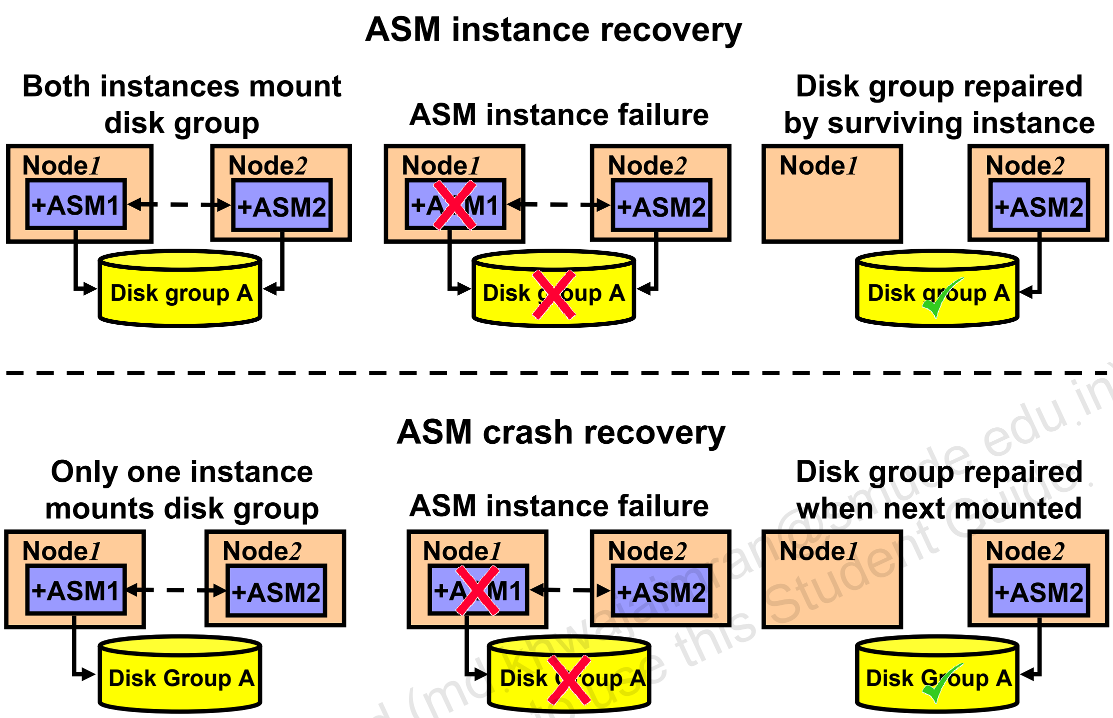
A disk group can contain files for many different Oracle databases. Thus, multiple database instances serving different databases can access the same disk group even on a single system without RAC. Alternatively, one Oracle database may also store its files in multiple disk groups managed by the same ASM instance.

Group Services is used to register the connection information needed by the database instances to find ASM instances. When an ASM instance mounts a disk group, it registers the disk group and connect string with Group Services. The database instance knows the name of the disk group, and can therefore use it to look up connection information for the correct ASM instance. Group Services is a functionality provided by Oracle Clusterware, which is automatically installed on every node that runs Oracle Database 11*g*.

Note: If an ASM instance fails, all Oracle database instances dependent on that ASM instance also fail. Note that a file system failure usually crashes a node. In a single–ASM instance configuration, if the ASM instance fails while disk groups are open for update, then after the ASM instance reinitializes, it reads the disk group's log and recovers all transient changes. With multiple ASM instances sharing disk groups, if one ASM instance fails, then another ASM instance automatically recovers transient ASM metadata changes caused by the failed instance.

The failure of a database instance does not affect ASM instances.

ASM Instance and Crash Recovery in RAC



ASM Instance and Crash Recovery in RAC

Each disk group is self-describing, containing its own file directory, disk directory, and other data such as metadata logging information. ASM automatically protects its metadata by using mirroring techniques even with external redundancy disk groups.

With multiple ASM instances mounting the same disk groups, if one ASM instance fails, another ASM instance automatically recovers transient ASM metadata changes caused by the failed instance. This situation is called ASM instance recovery, and is automatically and immediately detected by the global cache services.

With multiple ASM instances mounting different disk groups, or in the case of a single ASM instance configuration, if an ASM instance fails when ASM metadata is open for update, then the disk groups that are not currently mounted by any other ASM instance are not recovered until they are mounted again. When an ASM instance mounts a failed disk group, it reads the disk group log and recovers all transient changes. This situation is called ASM crash recovery.

Therefore, when using ASM clustered instances, it is recommended to have all ASM instances always mounting the same set of disk groups. However, it is possible to have a disk group on locally attached disks that are visible to only one node in a cluster, and have that disk group mounted on only that node where the disks are attached.

Note: The failure of an Oracle database instance is not significant here because only ASM instances update ASM metadata.

ASM Instance Initialization Parameters and RAC

- `CLUSTER_DATABASE`: This parameter must be set to `TRUE`.
- `ASM_DISKGROUP`:
  - Multiple instances can have different values.
  - Shared disk groups must be mounted by each ASM instance.
- `ASM_DISKSTRING`:
  - Multiple instances can have different values.
  - With shared disk groups, every instance should be able to see the common pool of physical disks.
- `ASM_POWER_LIMIT`: Multiple instances can have different values.

ORACLE

ASM Instance Initialization Parameters and RAC

In order to enable ASM instances to be clustered together in a RAC environment, each ASM instance initialization parameter file must set its `CLUSTER_DATABASE` parameter to `TRUE`. This enables the global cache services to be started on each ASM instance.

Although it is possible for multiple ASM instances to have different values for their `ASM_DISKGROUPS` parameter, it is recommended for each ASM instance to mount the same set of disk groups. This enables disk groups to be shared among ASM instances for recovery purposes. In addition, all disk groups used to store one RAC database must be shared by all ASM instances in the cluster.

Consequently, if you are sharing disk groups among ASM instances, their `ASM_DISKSTRING` initialization parameter must point to the same set of physical media. However, this parameter does not need to have the same setting on each node. For example, assume that the physical disks of a disk group are mapped by the OS on node A as `/dev/rdsk/c1t1d0s2`, and on node B as `/dev/rdsk/c2t1d0s2`. Although both nodes have different disk string settings, they locate the same devices via the OS mappings. This situation can occur when the hardware configurations of node A and node B are different—for example, when nodes are using different controllers as in the example above. ASM handles this situation because it inspects the contents of the disk header block to determine the disk group to which it belongs, rather than attempting to maintain a fixed list of path names.

ASM and SRVCTL with RAC

- SRVCTL enables you to manage ASM from an Oracle Clusterware (OC) perspective:
  – Add an ASM instance to OC.
  – Enable an ASM instance for OC automatic restart.
  – Start up an ASM instance.
  – Shut down an ASM instance.
  – Disable an ASM instance from OC automatic restart.
  – Remove an ASM instance configuration from the OCR.
  – Get some status information.
  – Set ASM instance dependency to database instance.
- The DBCA allows you to create ASM instances as well as helps you to add and enable them with OC.

ASM and SRVCTL with RAC

You can use SRVCTL to perform the following ASM administration tasks:

- The ADD option adds Oracle Cluster Registry (OCR) information about an ASM instance to run under Oracle Clusterware (OC). This option also enables the recourse.
- The ENABLE option enables an ASM instance to run under OC for automatic startup, or restart.
- The DISABLE option disables an ASM instance to prevent OC inappropriate automatic restarts. DISABLE also prevents any startup of that ASM instance using SRVCTL.
- The START option starts an OC-enabled ASM instance. SRVCTL uses the SYSDBA connection to perform the operation.
- The STOP option stops an ASM instance by using the shutdown normal, transactional, immediate, or abort option.
- The CONFIG option displays the configuration information stored in the OCR for a particular ASM instance.
- The STATUS option obtains the current status of an ASM instance.
- The REMOVE option removes the configuration of an ASM instance.
- The MODIFY INSTANCE command can be used to establishes a dependency between an ASM instance and a database instance.

Note: Adding and enabling an ASM instance is automatically performed by the DBCA when creating the ASM instance.

ASM and SRVCTL with RAC: Examples

- Start an ASM instance on the specified node:

```
$ srvctl start asm -n clusnode1
```

- Stop an ASM instance on the specified node:

```
$ srvctl stop asm -n clusnode1 -o immediate
```

- Add OCR data about an existing ASM instance:

```
$ srvctl add asm -n clusnode1 -i +ASM1 -o /ora/ora10
```

```
$ srvctl modify instance -d crm -i crm1 -s +asm1
```

- Disable OC management of an ASM instance:

```
$ srvctl disable asm -n clusnode1 -i +ASM1
```

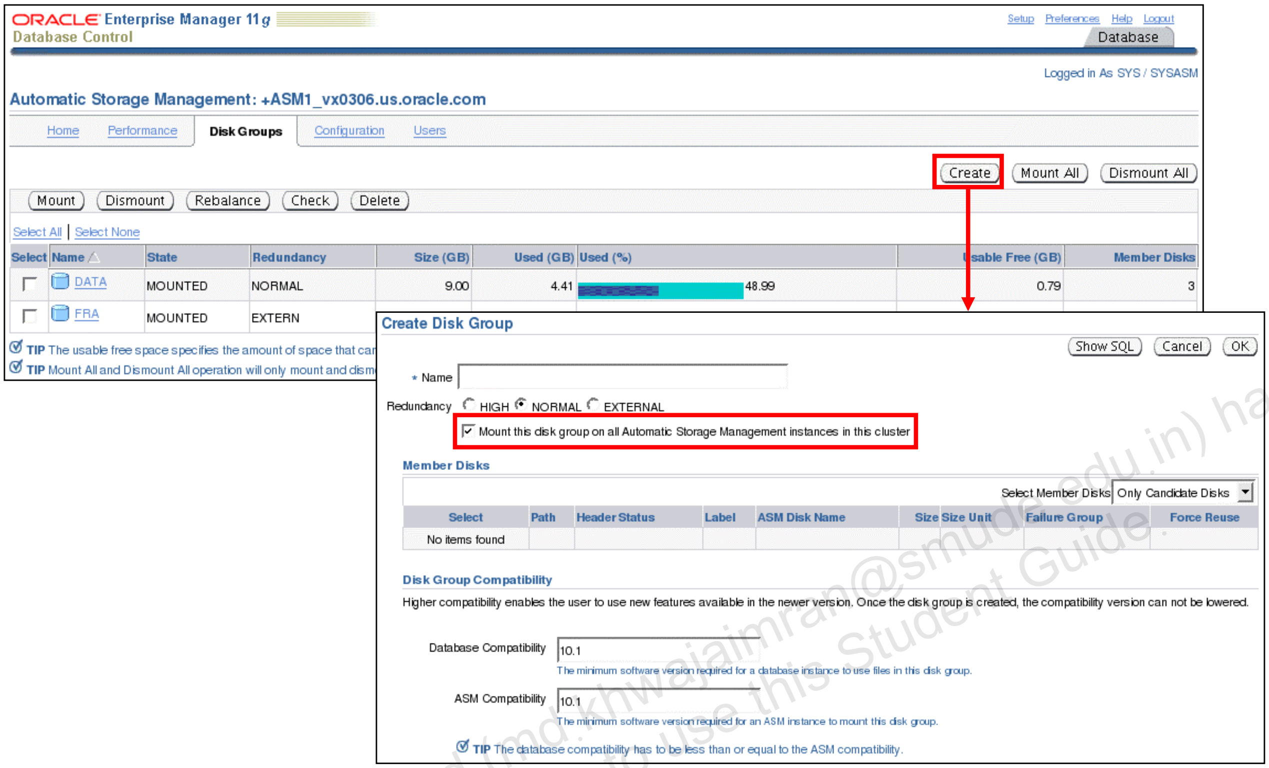

ASM and SRVCTL with RAC: Examples

Here are some examples:

- The first example starts up the only existing ASM instance on the CLUSNODE1 node. The -o option allows you to specify in which mode you want to open the instance: open is the default, but you can also specify mount, or nomount.
- The second example is of an immediate shutdown of the only existing ASM instance on CLUSNODE1.
- The third example adds to the OCR the OC information for +ASM1 on CLUSNODE1. You need to specify the ORACLE_HOME of the instance. Although the following should not be needed in case you use the DBCA, if you manually create ASM instances, you should also create OC dependency between database instances and ASM instances to ensure that the ASM instance starts up before starting database instance, and to allow database instances to be cleanly shut down before ASM instances. To establish the dependency, you have to use a command similar to the following: `srvctl modify instance -d crm -i crm1 -s +asm1` for each corresponding instance.
- The fourth example prevents OC to automatically restart +ASM1.

Note: For more information, refer to the *Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide*.

ASM Disk Groups with EM in RAC



ASM Disk Groups with EM in RAC

When you add a new disk group from an ASM instance, this disk group is not automatically mounted by other ASM instances. If you want to mount the newly added disk group on all ASM instances, for example, by using SQL*Plus, then you need to manually mount the disk group on each ASM instance.

However, if you are using Enterprise Manager (EM) to add a disk group, then the disk group definition includes a check box to indicate whether the disk group is automatically mounted to all the ASM clustered database instances. This is also true when you mount and dismount ASM disk groups by using Database Control where you can use a check box to indicate which instances mount or dismount the ASM disk group.

Disk Group Performance Page and RAC

ORACLE

Disk Group Performance Page and RAC

On the Automatic Storage Management Performance page, click the Disk Group I/O Cumulative Statistics link in the Additional Monitoring Links section. On the Disk Group I/O Cumulative Statistics page, click the corresponding disk group name.

A performance page is displayed showing clusterwide performance information for the corresponding disk group.

By clicking one of the proposed links—for example, I/O Throughput on the slide—you can see an instance-level performance details graph as shown at the bottom of the slide.

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
- Manage ASM instances in a RAC environment

Practice 4: Overview

This practice covers manipulating redo threads.

5

Managing Backup and Recovery in RAC

ORACLE

Objectives

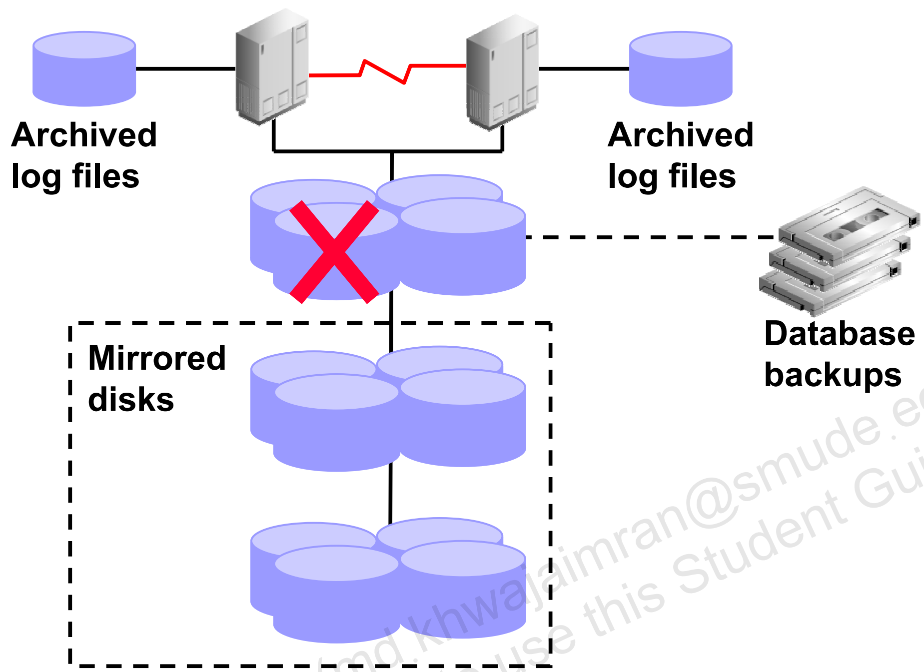
After completing this lesson, you should be able to:

- Configure the RAC database to use `ARCHIVELOG` mode and the flash recovery area
- Configure RMAN for the RAC environment

Objectives

RAC backup and recovery is almost identical to other Oracle database backup and recovery operations. This is because you are backing up and recovering a single database. The main difference is that with RAC you are dealing with multiple threads of redo log files.

Protecting Against Media Failure

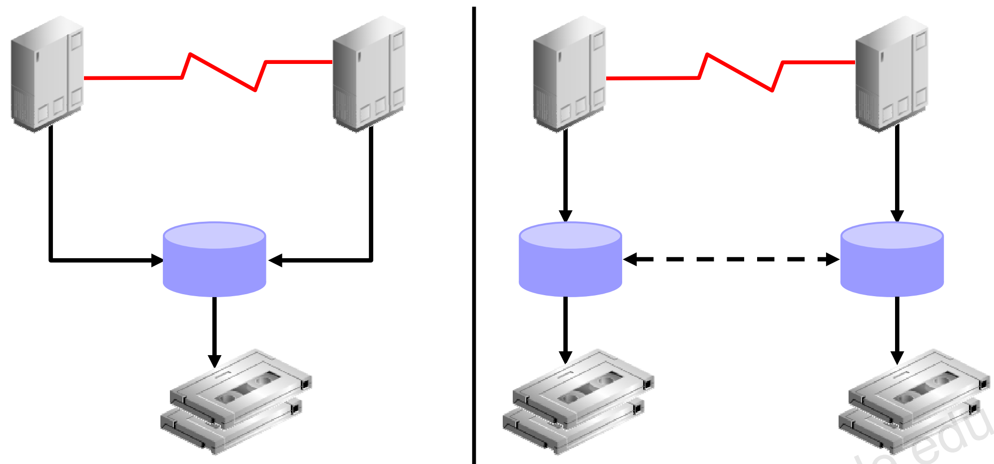


Protecting Against Media Failure

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 non-clustered 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 non-clustered 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.

Archived Log File Configurations



Cluster file system scheme: Archive logs from each instance are written to the same file location.

Local archive with NFS scheme: Each instance can read mounted archive destinations of all instances.

Archived Log File Configurations

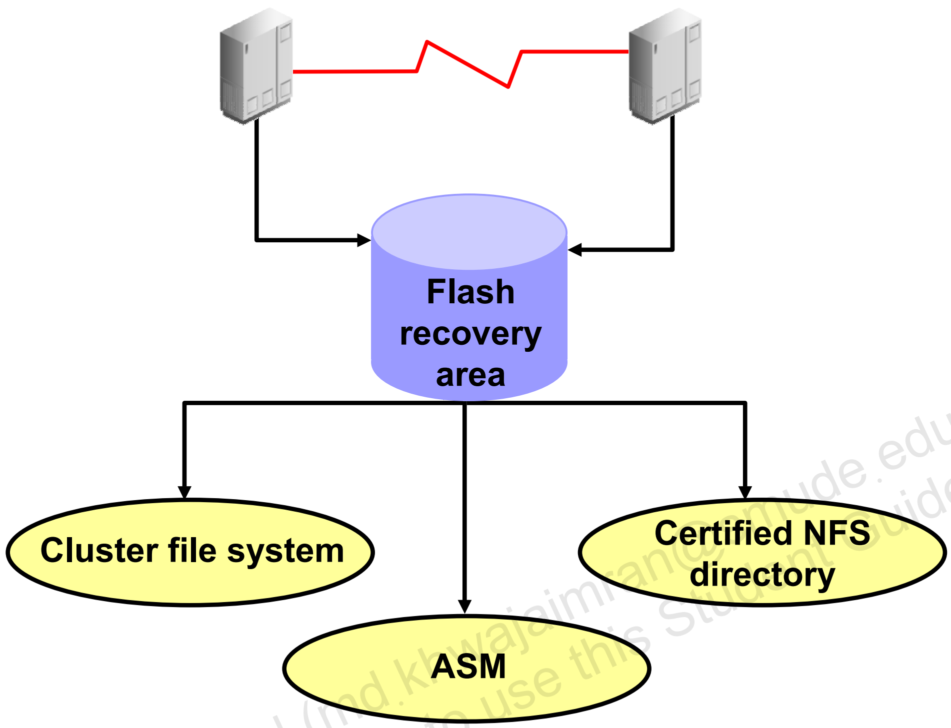
During backup and recovery operations involving archived log files, the Oracle server determines the file destinations and names from the control file. If you use RMAN, the archived log file path names can also be stored in the optional recovery catalog. However, the archived log file path names do not include the node name, so RMAN expects to find the files it needs on the nodes where the channels are allocated.

If you use a cluster file system, your instances can all write to the same archive log destination. This is known as the cluster file system scheme. Backup and recovery of the archive logs are easy because all logs are located in the same directory.

If a cluster file system is not available, then Oracle recommends that local archive log destinations be created for each instance with NFS-read mount points to all other instances. This is known as the local archive with network file system (NFS) scheme. During backup, you can either back up the archive logs from each host or select one host to perform the backup for all archive logs. During recovery, one instance may access the logs from any host without having to first copy them to the local destination.

Using either scheme, you may want to provide a second archive destination to avoid single points of failure.

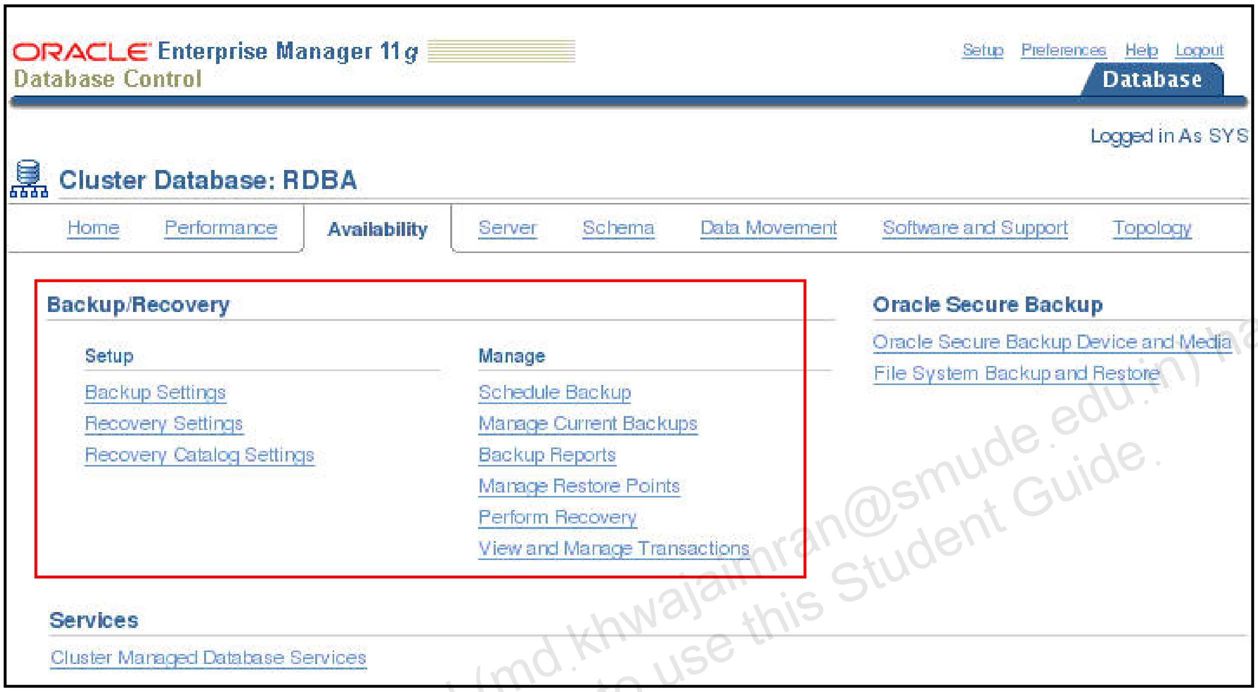
RAC and the Flash Recovery Area



RAC and the Flash Recovery Area

To use a flash 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 flash recovery area must be shared among all the instances of a RAC database.

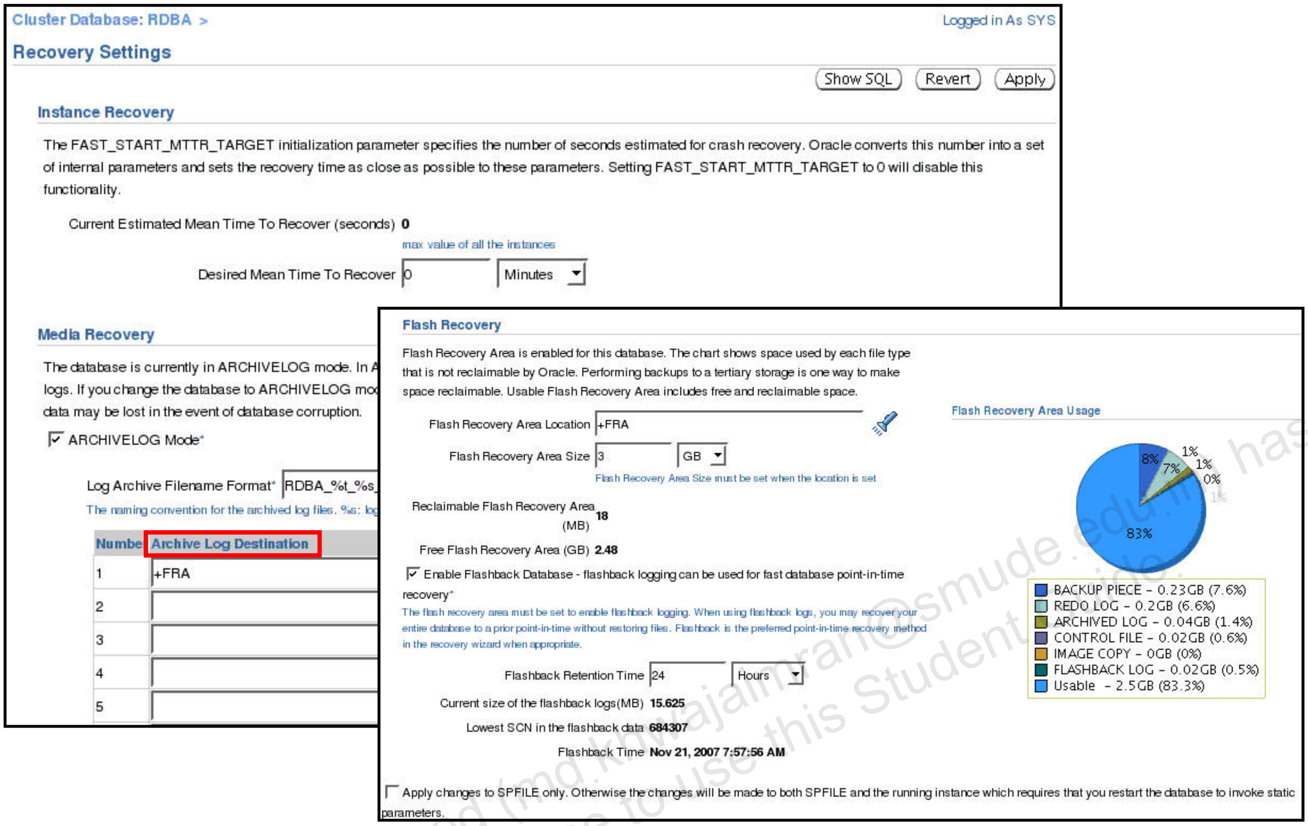
RAC Backup and Recovery Using EM



RAC Backup and Recovery Using EM

You 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 use RMAN to perform a range of backup and recovery operations, such as scheduling backups, performing recovery when necessary, and configuring backup and recovery settings. There are also links related to Oracle Secure Backup and Service management.

Configure RAC Recovery Settings with EM



Configure RAC Recovery Settings with EM

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 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 flash recovery area in a RAC environment.

Archived Redo File Conventions in RAC

Parameter	Description	Example
%r	Resetlogs identifier	log_1_62_23452345
%R	Padded resetlogs identifier	log_1_62_0023452345
%s	Log sequence number, not padded	log_251
%S	Log sequence number, left-zero-padded	log_0000000251
%t	Thread number, not padded	log_1
%T	Thread number, left-zero-padded	log_0001

Archived Redo File Conventions in RAC

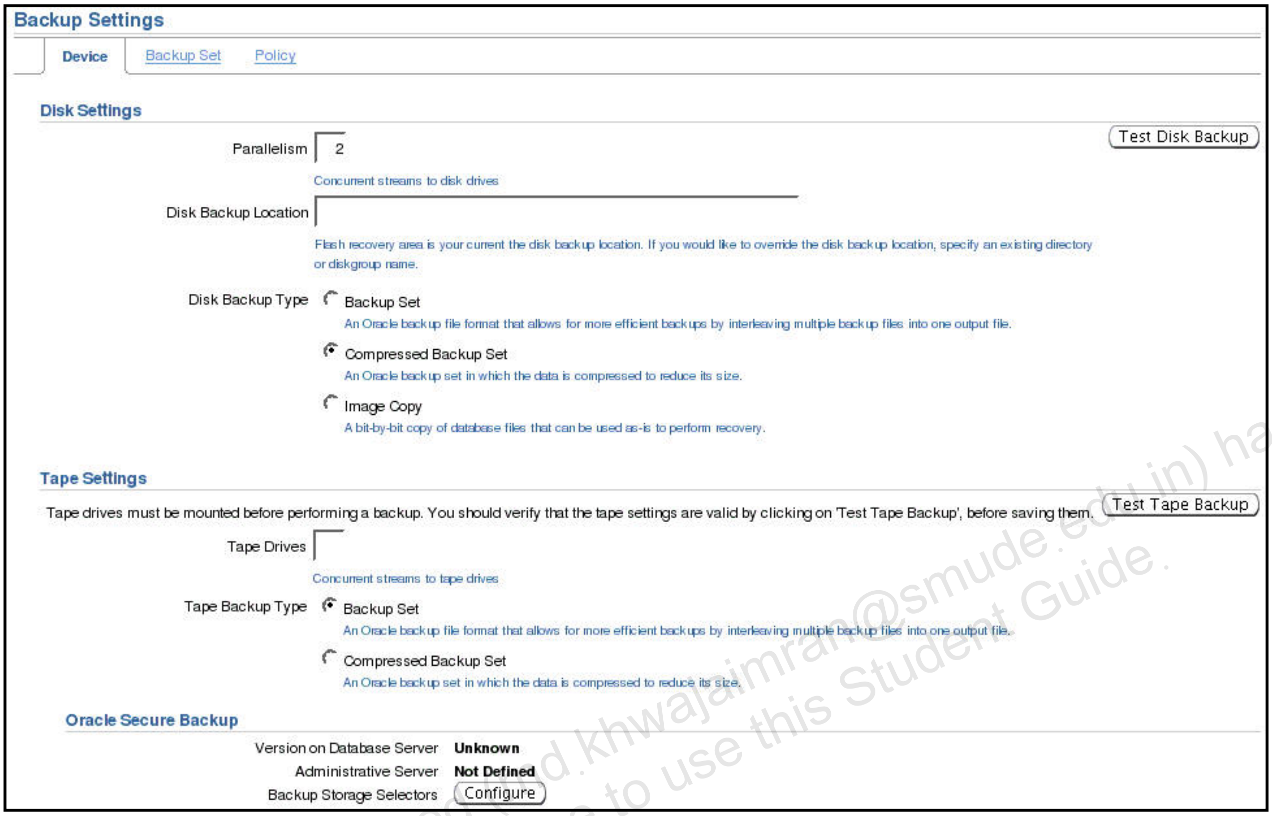
For any archived redo log configuration, uniquely identify the archived redo logs with the LOG_ARCHIVE_FORMAT parameter. The format of this parameter is operating system specific and it can include text strings, one or more variables, and a file name extension.

All of the thread parameters, in either upper or lower case, 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, then 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
...
```

Configure RAC Backup Settings with EM



ORACLE

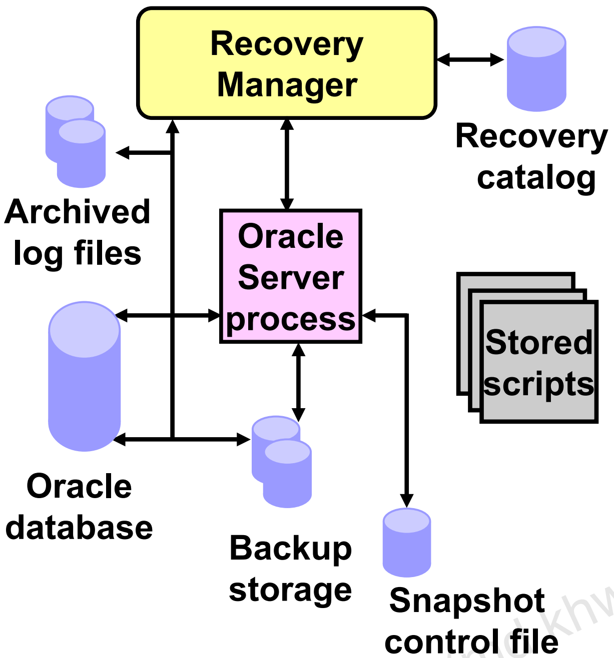
Configure RAC Backup Settings with EM

Persistent backup settings can be configured using Enterprise Manager. On the Database Control home page, click the Availability tab, 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 and vendor-specific media management parameters.

Oracle Recovery Manager



RMAN provides the following benefits for Real Application Clusters:

- Can read cluster files or raw partitions with no configuration changes
- Can access multiple archive log destinations

Oracle Recovery Manager

Oracle Recovery Manager (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 exist on all your nodes. The snapshot control file is only needed on the nodes on which RMAN performs backups. The snapshot control file does not need to be globally available to all instances in a RAC environment though.

You can use either a cluster file or a shared raw device as well as 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.

Configure 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/11.1.0/dbs/scf/snap_prod.cf
```

  - **You can use ASM, a shared file system location or a shared block device if you prefer:**

```
RMAN> CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'+FRA/SNAP/snap_prod.cf';
RMAN> CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'/ocfs/oradata/dbs/scf/snap_prod.cf';
RMAN> CONFIGURE SNAPSHOT CONTROLFILE NAME TO
'/dev/sdj2';
```


Configure RMAN Snapshot Control File Location

The snapshot control file is a temporary file that RMAN creates to resynchronize from a read-consistent version of the control file. RMAN needs a snapshot control file only when resynchronizing with the recovery catalog or when making a backup of the current control file.

In a RAC database, the snapshot control file is created on the node that is making the backup. You need to configure a default path and file name for these snapshot control files that are valid on every node from which you might initiate an RMAN backup.

Run the following RMAN command to determine the configured location of the snapshot control file:

```
SHOW SNAPSHOT CONTROLFILE NAME
```

You can change the configured location of the snapshot control file. For example, on UNIX-based systems you can specify the snapshot control file location as `snap_prod.cf` located in the ASM disk group `+FRA` by entering the following at the RMAN prompt:

```
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '+FRA/SNAP/snap_prod.cf'
```

This command globally sets the configuration for the location of the snapshot control file throughout your cluster database.

Note: The `CONFIGURE` command creates persistent settings across RMAN sessions.

Configure Control File and SPFILE Autobackup

- RMAN automatically creates a control file and SPFILE backup after `BACKUP` or `COPY`:

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;
```

- Change default location:

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR
DEVICE TYPE DISK TO '+FRA';
```

- Location must be available to all nodes in your RAC database.

Configure Control File and SPFILE Autobackup

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 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 name 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.

Note: RMAN performs the control file autobackup on the first allocated channel. When you allocate multiple channels with different parameters (especially if you allocate a channel with the `CONNECT` command), you must determine which channel will perform the automatic backup. Always allocate the channel for the connected node first.

Crosschecking on Multiple RAC Clusters Nodes

When cross-checking on multiple nodes make sure 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 cross-check 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, the backups are marked `EXPIRED`.**

Crosschecking on Multiple RAC Clusters Nodes

When cross-checking 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 cross-check operations.

If you cannot configure the cluster so that each node can access all backups, then, during restore and cross-check 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 cross-check because no channel was configured on the node that can access those backups, the backups are marked `EXPIRED` in the RMAN repository after the cross-check.

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

- During backup, 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.

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT='sys/rac@RACDB1';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT='sys/rac@RACDB2';
CONFIGURE CHANNEL 3 DEVICE TYPE sbt CONNECT='sys/rac@RACDB3';
```

or

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE CHANNEL DEVICE TYPE sbt CONNECT='sys/rac@BR';
```

ORACLE

Channel Connections to Cluster Instances

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, the channels are allocated at a node as decided by the load-balancing algorithm.

During backup, the instances to which the channels connect must be either all mounted or all open. For example, if the `RACDB1` instance has the database mounted but the `RACDB2` and `RACDB3` instances have the database open, the backup fails.

Note: In some cluster database configurations, some nodes of the cluster 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. For example, if you have a three-node cluster, and if node 1 has faster read/write access to data files 7, 8, and 9 than do the other nodes, then node 1 has greater node affinity to those files than nodes 2 and 3 and RMAN will take advantage of this automatically.

RMAN Channel Support for the Grid

- RAC allows the use of nondeterministic connect strings.
- 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;
```


RMAN Channel Support for the Grid

In Oracle Database 11*g*, 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. In Oracle Database 11*g*, 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 autolocates 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.

RMAN Default Autolocation

Recovery Manager automatically discovers which nodes of a RAC configuration can access the files that you want to back up or restore. Recovery Manager autolocates 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, then 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, then 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

Three 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.**

Distribution of Backups

When configuring the backup options for RAC, you have three possible configurations:

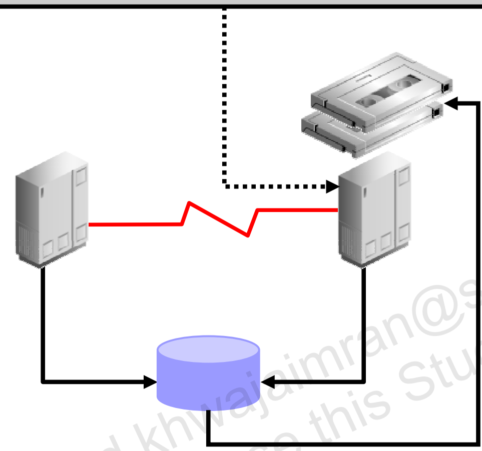
- **Network backup server:** A dedicated backup server performs and manages backups for the cluster and the cluster database. None of the nodes have local backup appliances.
- **One local drive:** One node has access to a local backup appliance and performs and manages backups for the cluster database. 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`.

One Local Drive CFS Backup Scheme

```
RMAN> CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
RMAN> CONFIGURE DEFAULT DEVICE TYPE TO sbt;
```

```
RMAN> BACKUP DATABASE PLUS ARCHIVELOG DELETE INPUT;
```



One Local Drive CFS Backup Scheme

In a cluster file system backup scheme, each node in the cluster has read access to all the data files, archived redo logs, and SPFILEs. This includes Automated Storage Management (ASM), cluster file systems, and Network Attached Storage (NAS).

When backing up to only one local drive in the cluster file system backup scheme, it is assumed that only one node in the cluster has a local backup appliance such as a tape drive. In this case, run the following one-time configuration commands:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
```

Because any node performing the backup has read/write access to the archived redo logs written by the other nodes, the backup script for any node is simple:

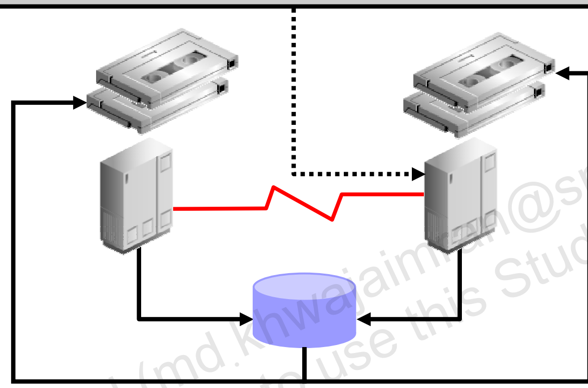
```
BACKUP DATABASE PLUS ARCHIVELOG DELETE INPUT;
```

In this case, the tape drive receives all data files, archived redo logs, and SPFILEs.

Multiple Drives CFS Backup Scheme

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 2;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT 'usr1/pwd1@n1';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT 'usr2/pwd2@n2';
```

```
BACKUP DATABASE PLUS ARCHIVELOG DELETE INPUT;
```



Multiple Drives CFS Backup Scheme

When backing up to multiple drives in the cluster file system backup scheme, it is assumed that each node in the cluster has its own local tape drive. Perform the following one-time configuration so that one channel is configured for each node in the cluster. This is a one-time configuration step. For example, enter the following at the RMAN prompt:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 2;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT 'user1/passwd1@node1';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT 'user2/passwd2@node2';
```

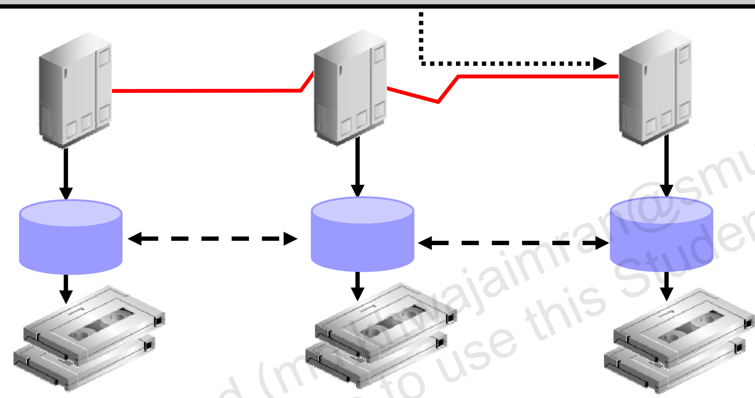
Similarly, you can perform this configuration for a device type of `DISK`. The following backup script, which you can run from any node in the cluster, distributes the data files, archived redo logs, and SPFILE backups among the backup drives:

```
BACKUP DATABASE PLUS ARCHIVELOG DELETE INPUT;
```

Non-CFS Backup Scheme

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT 'usr1/pwd1@n1';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT 'usr2/pwd2@n2';
CONFIGURE CHANNEL 3 DEVICE TYPE sbt CONNECT 'usr3/pwd3@n3';
```

```
BACKUP DATABASE PLUS ARCHIVELOG DELETE INPUT;
```



Non-CFS Backup Scheme

In a noncluster file system environment, each node can back up only its own local archived redo logs. For example, node 1 cannot access the archived redo logs on node 2 or node 3 unless you configure the network file system for remote access. To configure NFS, distribute the backup to multiple drives. However, if you configure NFS for backups, then you can back up to only one drive.

When backing up to multiple drives in a noncluster file system backup scheme, it is assumed that each node in the cluster has its own local tape drive. You can perform a similar one-time configuration as the one shown in the slide to configure one channel for each node in the cluster. Similarly, you can perform this configuration for a device type of `DISK`. Develop a production backup script for whole database backups that you can run from any node. With the `BACKUP` example, the data file backups, archived redo logs, and `SPFILE` backups are distributed among the different tape drives. However, channel 1 can read only the logs archived locally on `/arc_dest_1`. This is because the autolocation feature restricts channel 1 to back up only the archived redo logs in the `/arc_dest_1` directory. Because node 2 can read files only in the `/arc_dest_2` directory, channel 2 can back up only the archived redo logs in the `/arc_dest_2` directory, and so on. The important point is that all logs are backed up, but they are distributed among the different drives.

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.

Restoring and Recovering

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 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.

Summary

In this lesson, you should have learned how to:

- Configure RAC recovery settings with EM
- Configure RAC backup settings with EM
- Initiate archiving
- Configure RMAN
- Perform RAC backup and recovery using EM

Note

Backup and recovery procedures for OCR and voting disk are described in the *Oracle Clusterware Administration* lesson covered later in this course.

Practice 5: Overview

This practice covers the following topics:

- **Configuring the RAC database to use `ARCHIVELOG` mode and the flash recovery area**
- **Configuring RMAN for the RAC environment**

6

RAC Performance Tuning

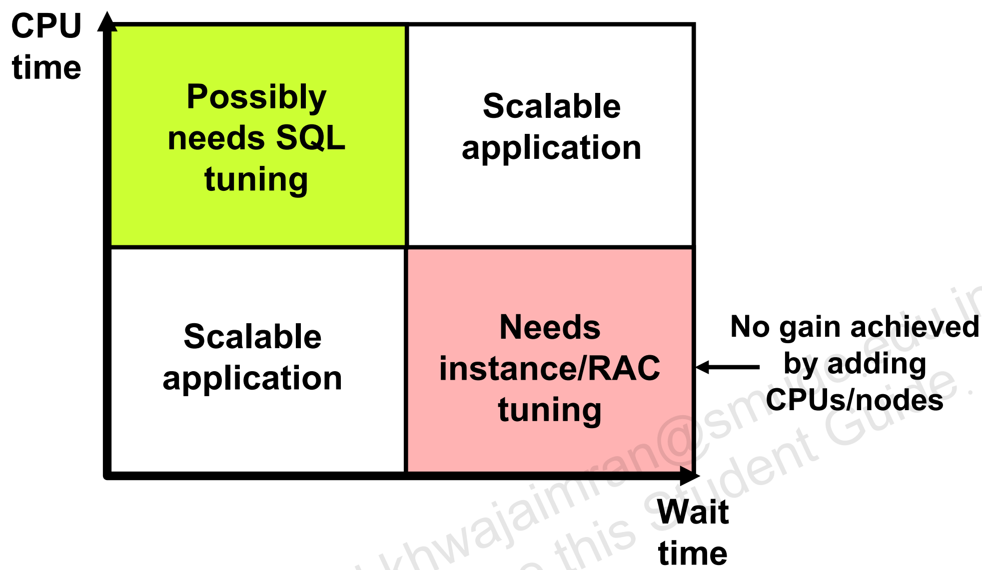
Objectives

After completing this lesson, you should be able to:

- Determine RAC-specific tuning components
- Tune instance recovery in RAC
- Determine RAC-specific wait events, global enqueues, and system statistics
- Implement the most common RAC tuning tips
- Use the Cluster Database Performance pages
- Use the Automatic Workload Repository (AWR) in RAC
- Use Automatic Database Diagnostic Monitor (ADDM) in RAC

ORACLE

CPU and Wait Time Tuning Dimensions



CPU and Wait Time Tuning Dimensions

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. On the other hand, 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 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.

Note: Automatic Workload Repository (AWR) reports display CPU time together with wait time in the **Top 5 Timed Events** section, if the CPU time portion is among the top five events.

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.

ORACLE

RAC-Specific Tuning

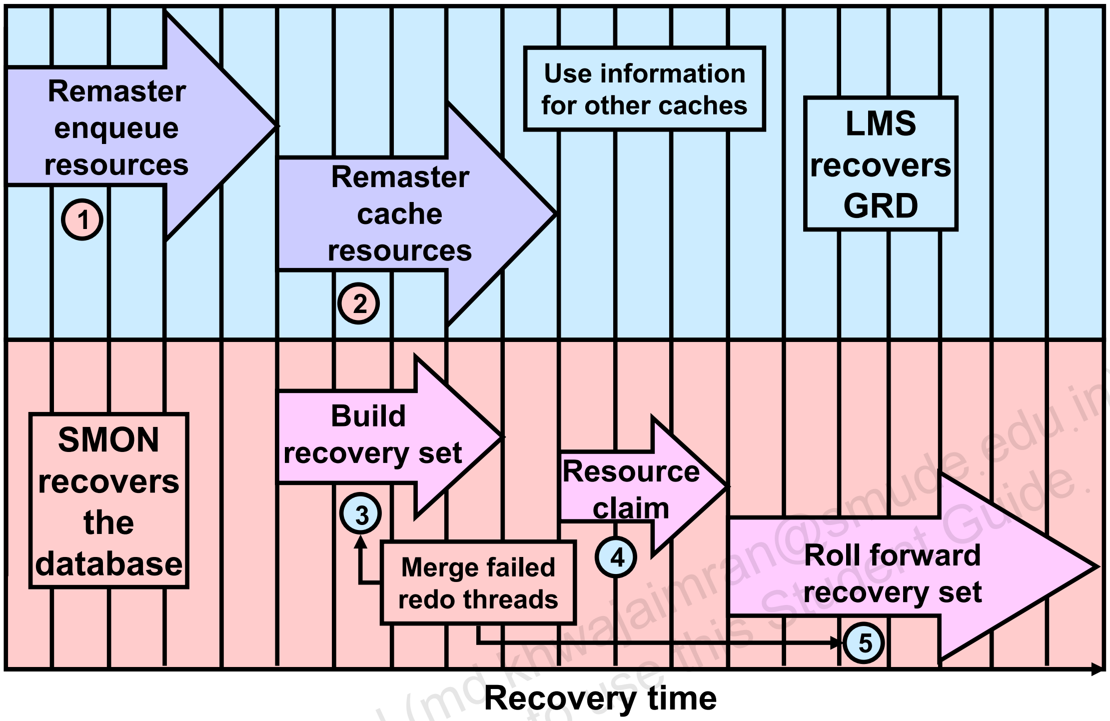
Although there are specific tuning areas for RAC, such as instance recovery and interconnect traffic, you get most benefits by tuning your system like a single-instance system. At least, this must be your starting point.

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).

RAC and Instance or Crash Recovery



RAC and Instance or Crash Recovery

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 (GES) 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.

RAC and Instance or Crash Recovery (continued)

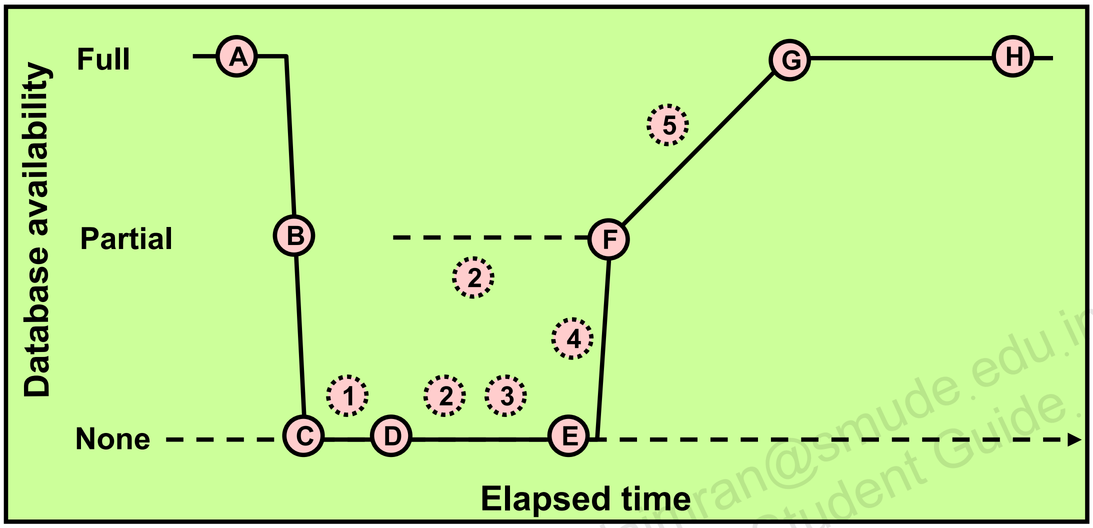
4. Buffer space for recovery is allocated and the resources that were identified in the previous reading of the redo logs are claimed as recovery resources. This is done to avoid other instances to access those resources.
5. All resources required for subsequent processing have been acquired and the 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. Refer to the section "Write to Disk Coordination: Example" in this lesson for more information about past images.

After all blocks have been recovered and the recovery resources have been released, the system is again fully available.

In summary, the recovered database or the recovered portions of the database becomes 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



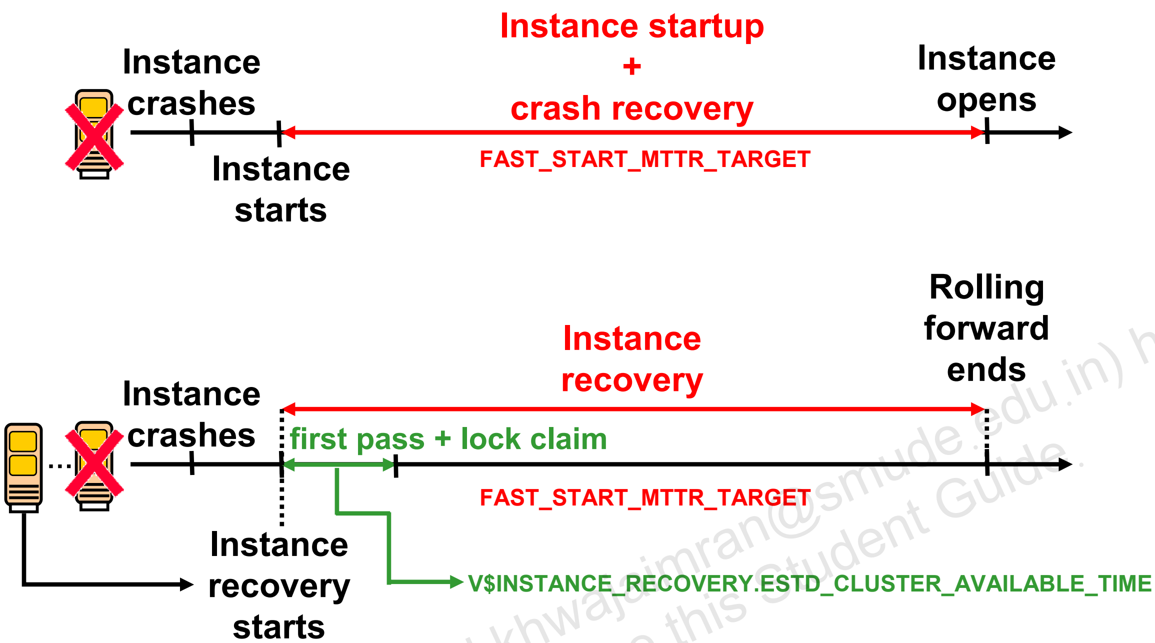
Instance Recovery and Database Availability

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

A. Real Application Clusters is running on multiple nodes.
B. Node failure is detected.
C. The enqueue part of the GRD is reconfigured; resource management is redistributed to the surviving nodes. This operation occurs relatively quickly.
D. The cache part of the GRD is reconfigured and SMON reads the redo log of the failed instance to identify the database blocks that it needs to recover.
E. SMON issues the GRD requests to obtain all the database blocks it needs for recovery. After the requests are complete, all other blocks are accessible.
F. The Oracle server performs roll forward recovery. Redo logs of the failed threads are applied to the database, and blocks are available right after their recovery is completed.
G. The Oracle server performs rollback recovery. Undo blocks are applied to the database for all uncommitted transactions.
H. Instance recovery is complete and all data is accessible.

Note: The dashed line represents the blocks identified in step 2 in the previous slide. Also, the dotted steps represent the ones identified in the previous slide.

Instance Recovery and RAC



Instance Recovery and RAC

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 autotune 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 that are 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 that it is instance recovery for single-instance death.

Note: If you really want to have small instance recovery time by setting FAST_START_MTTR_TARGET, you can safely ignore the alert log messages about raising its value.

Instance Recovery and RAC

- Use parallel instance recovery.
- Increase PARALLEL_EXECUTION_MESSAGE_SIZE.
- Set PARALLEL_MIN_SERVERS.
- Use Async I/O.
- Increase the size of the default buffer cache.

Instance Recovery and RAC (continued)

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.
- Increase PARALLEL_EXECUTION_MESSAGE_SIZE from its default of 2,148 bytes to 4 KB or 8 KB. This should provide better recovery slave performance.
- Set PARALLEL_MIN_SERVERS to CPU_COUNT-1. This will prespawn recovery slaves at startup time.
- 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.

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.

Analyzing Cache Fusion Impact in RAC

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 Services - Workload Characteristics

Avg global enqueue get time (ms):	4.5
Avg global cache cr block receive time (ms):	0.6
Avg global cache current block receive time (ms):	1.1
Avg global cache cr block build time (ms):	0.0
Avg global cache cr block send time (ms):	0.1
Global cache log flushes for cr blocks served %:	3.2
Avg global cache cr block flush time (ms):	4.0
Avg global cache current block pin time (ms):	0.4
Avg global cache current block send time (ms):	0.1
Global cache log flushes for current blocks served %:	2.9
Avg global cache current block flush time (ms):	35.5

Typical Latencies for RAC Operations

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.

Wait Events for RAC

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

ORACLE

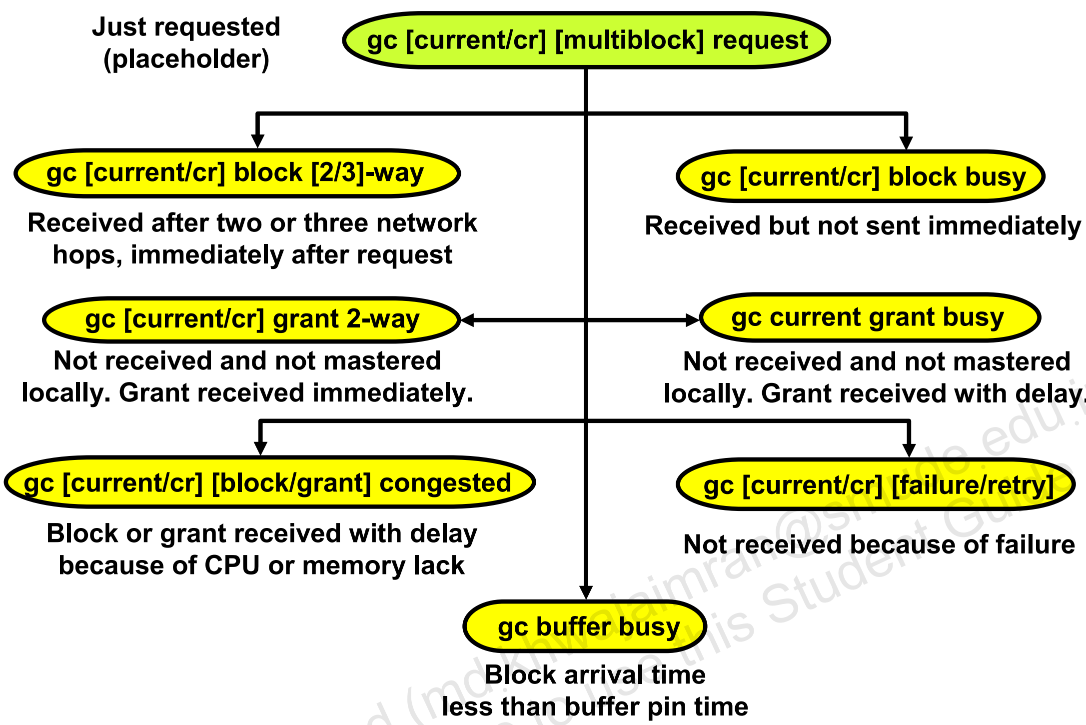
Wait Event Views

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



Global Cache Wait Events: Overview

The main global cache wait events for Oracle Database 11*g* are described briefly in the slide:

- **`gc current/cr request`:** These wait events are relevant only while a gc request for a cr or current buffer is in progress. They act as placeholders until the request completes.
- **`gc [current/cr] [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.

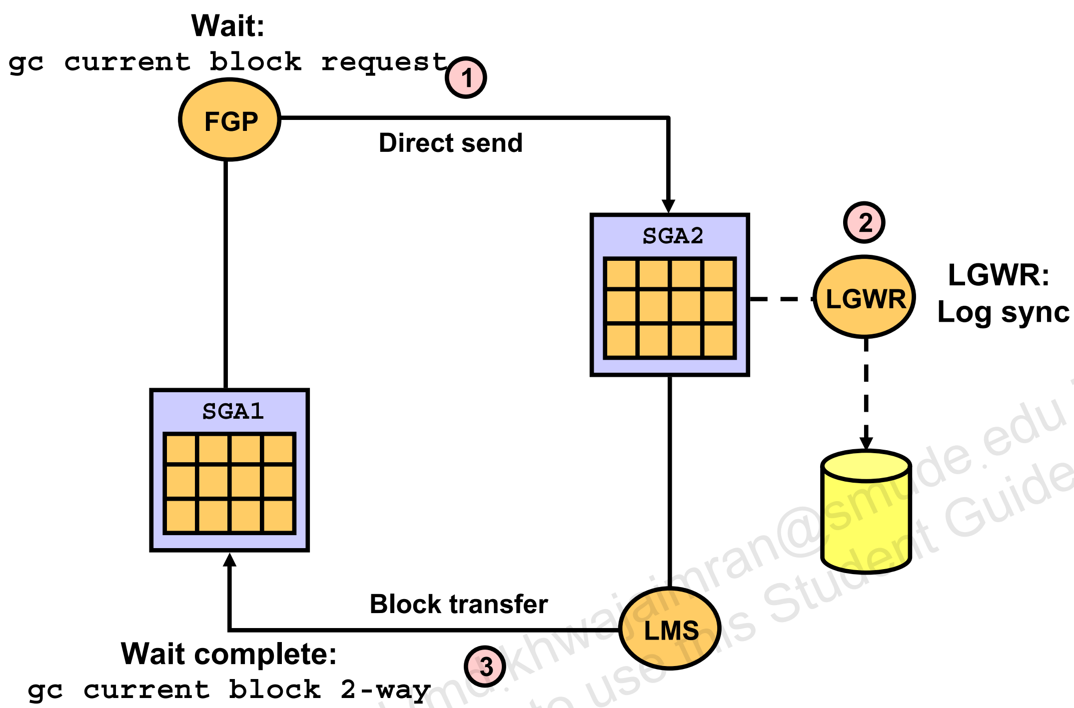
Note: For dynamic remastering, two events are of the most practical importance: `gc remaster` and `gc quiesce`. They can be symptoms of the effect of remastering on the running processes.

Global Cache Wait Events: Overview (continued)

- **`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 the *Oracle Database Reference* guide.

`2-way` Block Request: Example



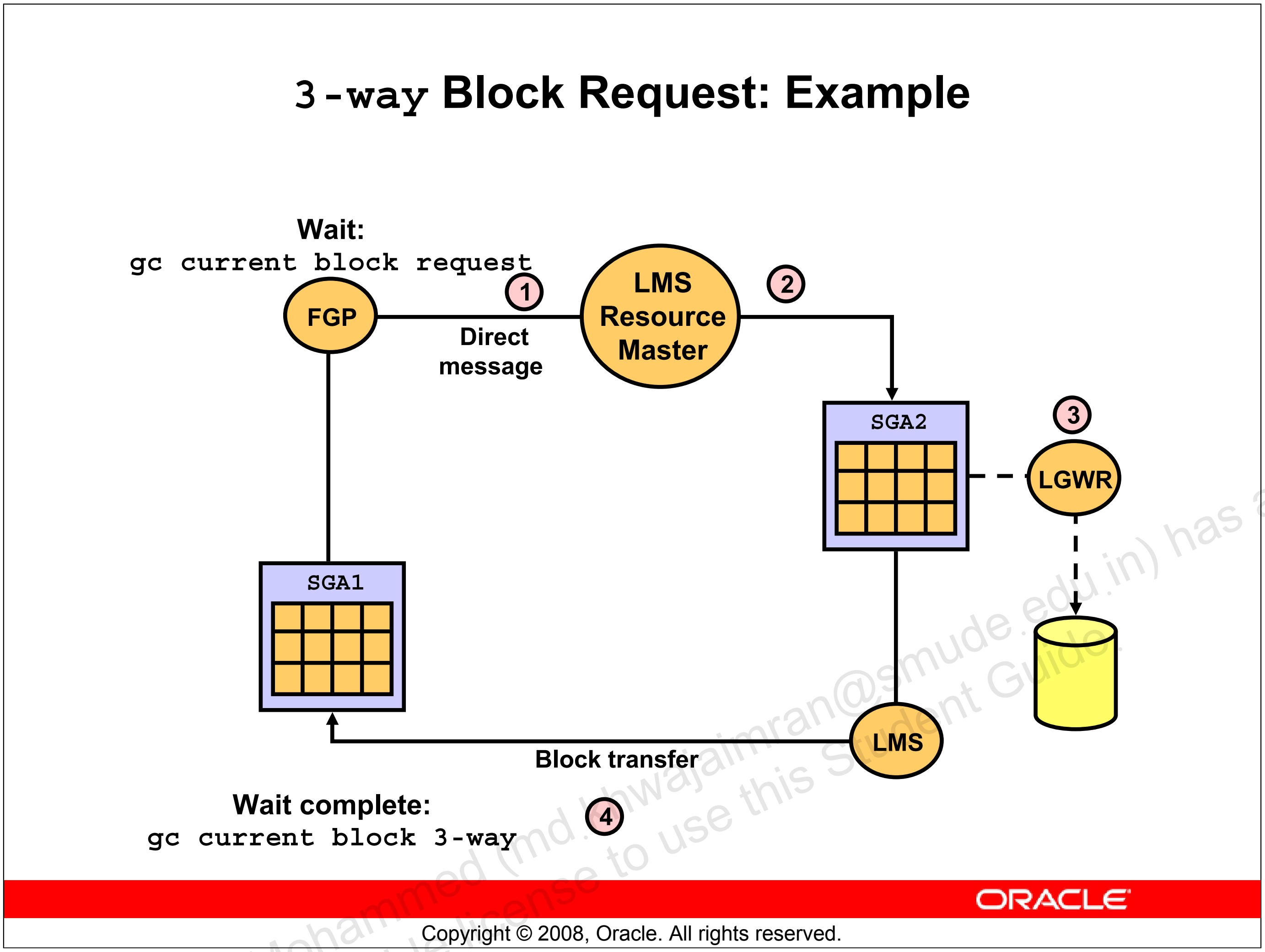
`2-way` Block Request: Example

This slide shows you what typically happens when the master instance requests a block that is not cached locally. Here it is supposed that the master instance is called `SGA1`, and `SGA2` contains the requested block. The scenario is as follows:

1. `SGA1` sends a direct request to `SGA2`. So `SGA1` waits on the `gc current block request` event.
2. When `SGA2` receives the request, its local LGWR process may need to flush some recovery information to its local redo log files. For example, if the cached block is frequently changed, and the changes have not been logged yet, LMS would have to ask LGWR to flush the log before it can ship the block. This may add a delay to the serving of the block and may show up in the requesting node as a busy wait.
3. Then, `SGA2` sends the requested block to `SGA1`. When the block arrives in `SGA1`, the wait event is complete, and is reflected as `gc current block 2-way`.

Note: Using the notation `R` = time at requestor, `W` = wire time and transfer delay, and `S` = time at server, the total time for a round-trip would be:

`R`(send) + `W`(small msg) + `S`(process msg,process block,send) + `W`(block) + `R`(receive block)



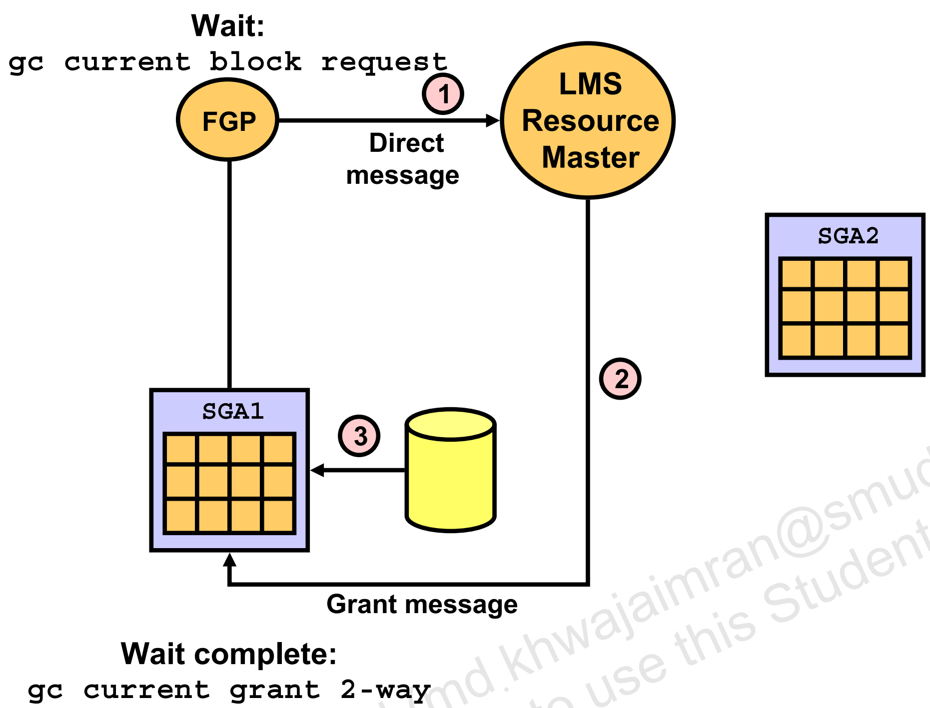
3-way Block Request: Example

This is a modified scenario for a cluster with more than two nodes. It is very similar to the previous one. However, the master for this block is on a node that is different from that of the requestor, and where the block is cached. Thus, the request must be forwarded. There is an additional delay for one message and the processing at the master node:

R(send) + W(small msg) + S(process msg,send) + W(small msg) + S(process msg,process block,send) + W(block) + R(receive block)

While a remote read is pending, any process on the requesting instance that is trying to write or read the data cached in the buffer has to wait for a `gc buffer busy`. The buffer remains globally busy until the block arrives.

`2-way` Grant: Example



`2-way` Grant: Example

In this scenario, a grant message is sent by the master because the requested block is not cached in any instance.

If the local instance is the resource master, the grant happens immediately. If not, the grant is always 2-way, regardless of the number of instances in the cluster.

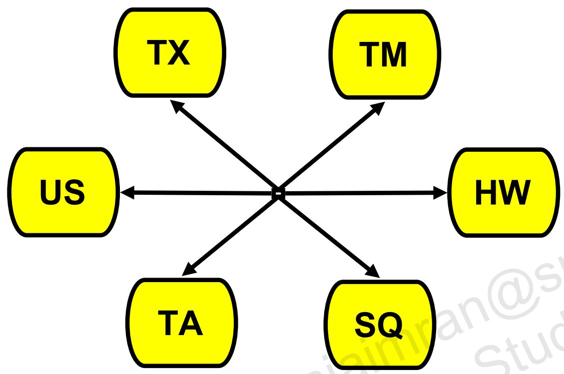
The grant messages are small. For every block read from the disk, a grant has to be received before the I/O is initiated, which adds the latency of the grant round-trip to the disk latency:

`R`(send) + `W`(small msg) + `S`(process msg,send) + `W`(small msg) + `R`(receive block)

The round-trip looks similar to a 2-way block round-trip, with the difference that the wire time is determined by a small message, and the processing does not involve the buffer cache.

Global Enqueue Waits: Overview

- Enqueues are synchronous.
- Enqueues are global resources in RAC.
- The most frequent waits are for:



- The waits may constitute serious serialization points.

Global Enqueue Waits: Overview

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 cases above, the waits are synchronous and may constitute serious serialization points that can be exacerbated in a RAC environment.

Note: In Oracle Database 11*g*, 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.

Session and System Statistics

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 the *Oracle Database Reference* guide.

Most Common RAC Tuning Tips

- Application tuning is often the most beneficial.
- Reducing long full-table scans in OLTP systems
- Using Automatic Segment Space Management
- Increasing sequence caches
- Using partitioning to reduce interinstance traffic
- Avoiding unnecessary parsing
- Minimizing locking usage
- Removing unselective indexes
- Configuring interconnect properly

Most Common RAC Tuning Tips

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 of the fact that 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 that another instance has cached the block is high.
- 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 multiinstance 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.

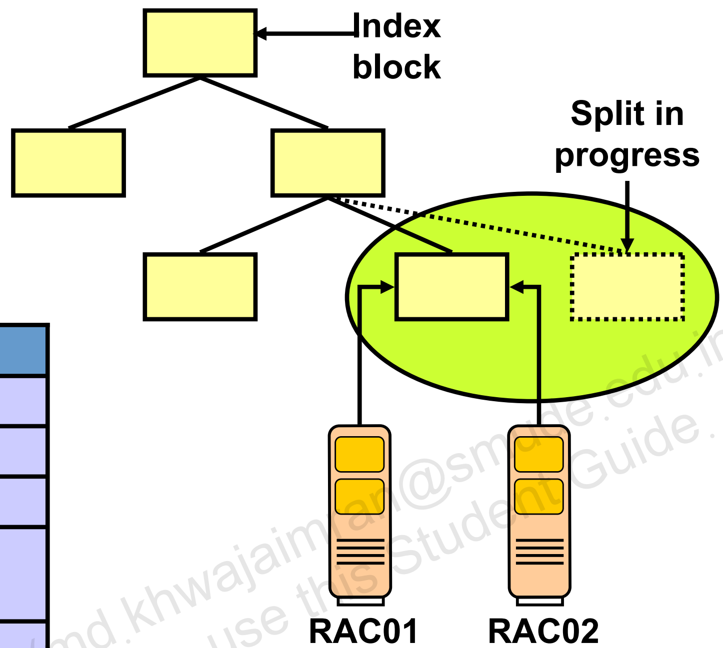
Most Common RAC Tuning Tips (continued)

- 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 using 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 interinstance 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 GBE can scale up to ten thousand 8-KB blocks per second before saturation.

Index Block Contention: Considerations

Wait events
enq: TX - index contention
gc buffer busy
gc current block busy
gc current split

System statistics
Leaf node splits
Branch node splits
Exchange deadlocks
gcs refuse xid gcs ast xid
Service ITL waits



Index Block Contention: Considerations

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:

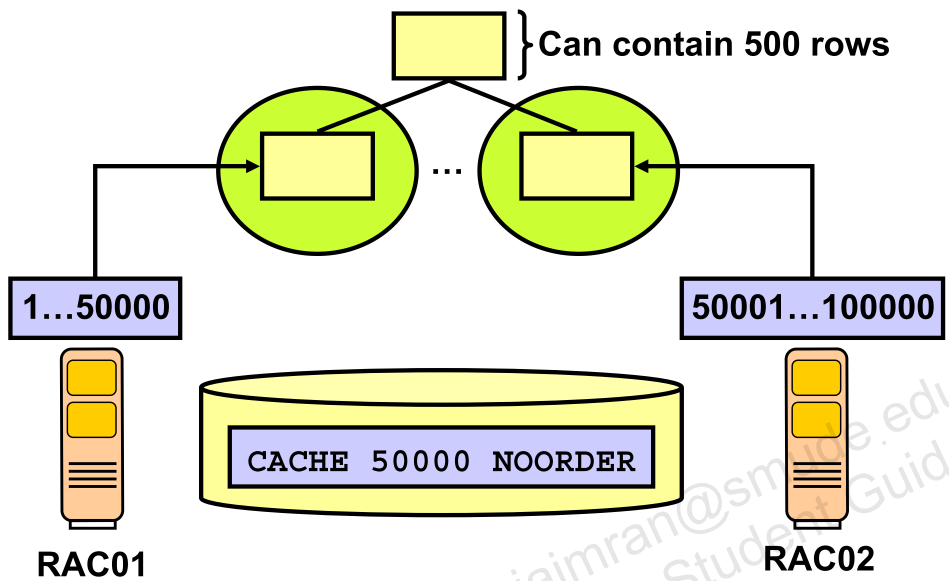
- Ordered, monotonically increasing key values in the index (right-growing trees)
- Frequent leaf block splits
- Low tree depth: All leaf block access go through the root block.

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
- Use natural keys as opposed to surrogate keys
- Use reverse key indexes

Oracle Sequences and Index Contention



Oracle Sequences and Index Contention

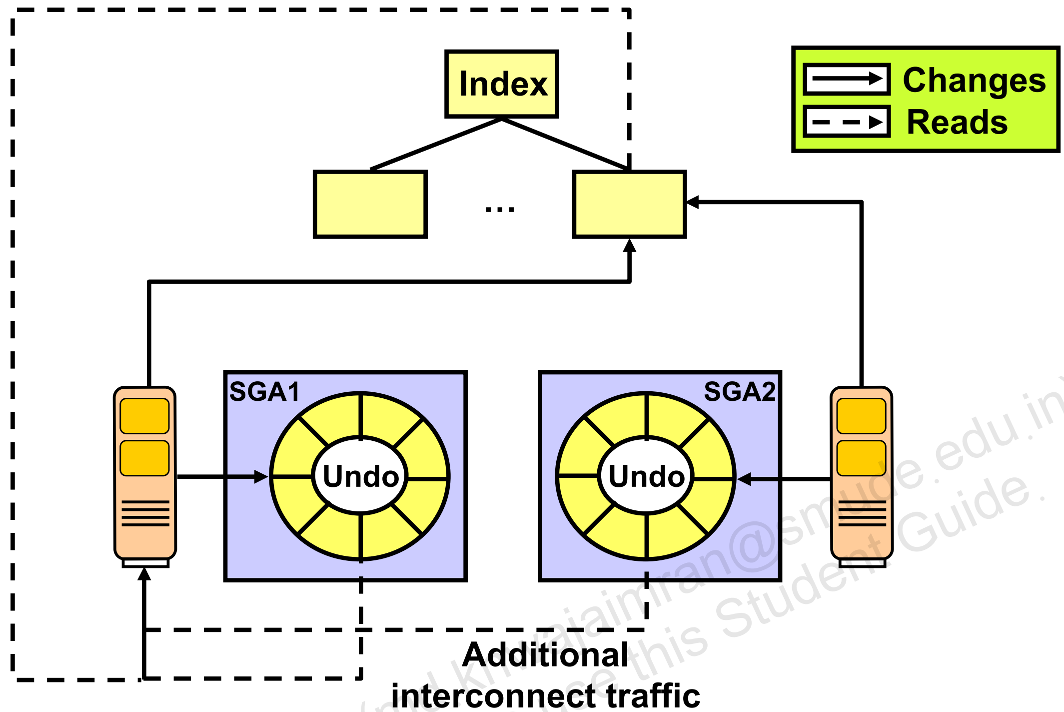
Indexes with key values generated by sequences tend to be subject to leaf block contention when the insert rate is high. That is because the index leaf block holding the highest key value is changed for every row inserted, as the values are monotonically ascending. In RAC, this may lead to a high rate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

Undo Block Considerations



Undo Block Considerations

Excessive undo block shipment and contention for undo buffers usually happens when index blocks containing active transactions from multiple instances are read frequently.

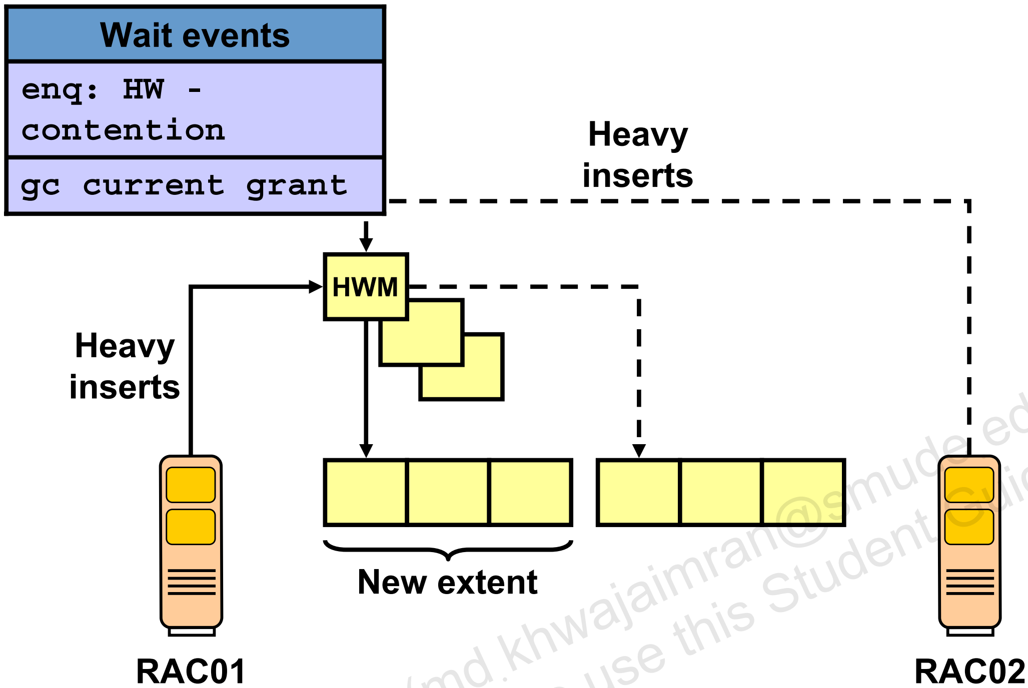
When a `SELECT` statement needs to read a block with active transactions, it has to undo the changes to create a CR version. If the active transactions in the block belong to more than one instance, there is a need to combine local and remote undo information for the consistent read. Depending on the 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 interinstance 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



High-Water Mark Considerations

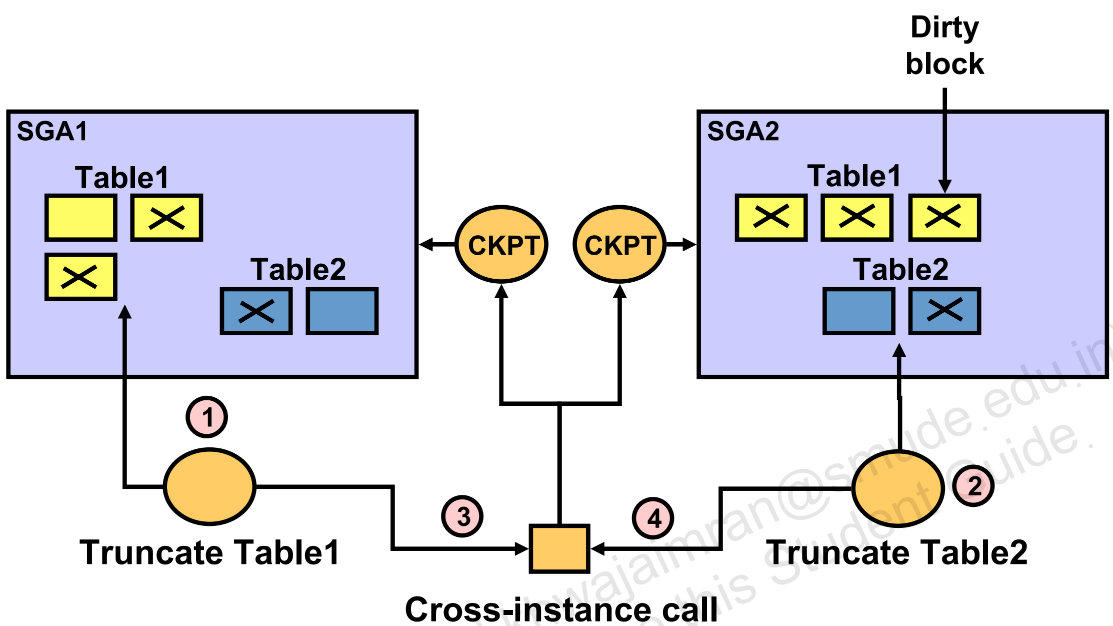
A certain combination of wait events and statistics presents itself in applications where the insertion of data is a dominant business function and new blocks have to be allocated frequently to a segment. If data is inserted at a high rate, new blocks may have to be made available after unfruitful searches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



Concurrent Cross-Instance Calls: Considerations

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.

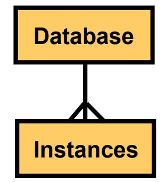
In a RAC environment, truncating tables concurrently from different instances does not scale well, especially if, in conjunction, you are also using direct read operations such as parallel queries.

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



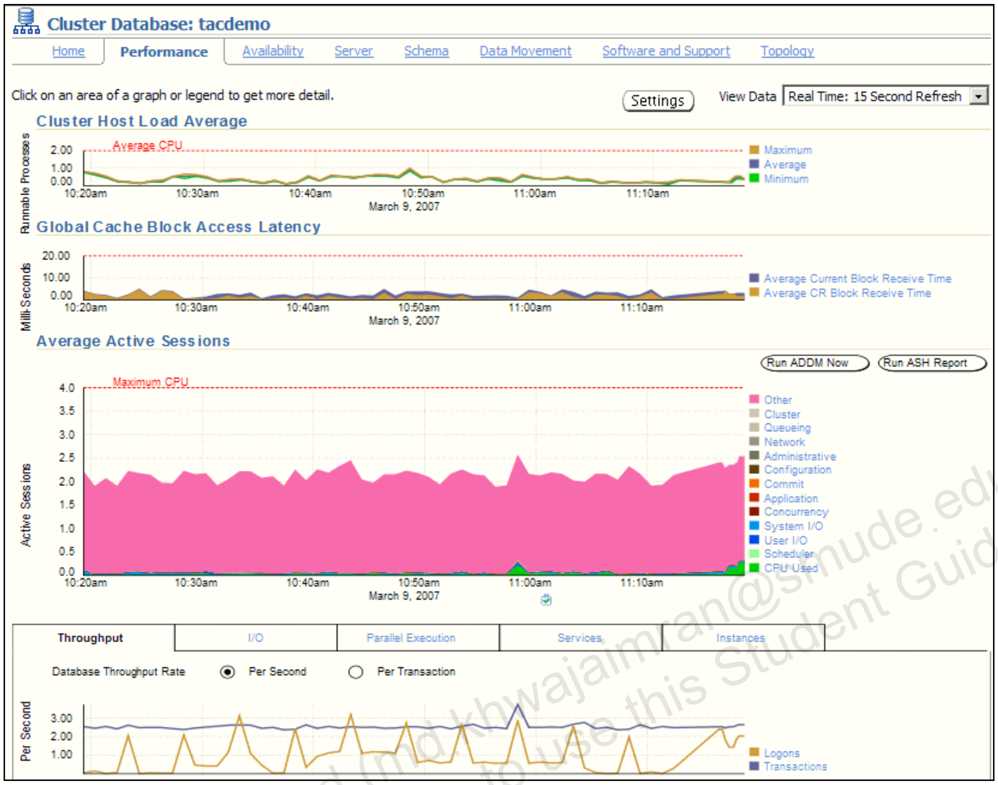
Directly from Database Control and Grid Control:
- **Status of each node in the cluster**
- **Aggregated alert messages across all the instances**
- **Review issues that are affecting the entire cluster or each instance**
- **Monitor cluster cache coherency statistics**
- **Determine if any of the services for the cluster database are having availability problems**
- **Review any outstanding Clusterware interconnect alerts**

Monitoring RAC Database and Cluster Performance

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 the following:

- View the overall system status, such as the number of nodes in the cluster and their current status so that you do not have to access each individual database instance for details
- View alert messages aggregated across all the instances with lists for the source of each alert message
- Review 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 if any of the services for the cluster database are having availability problems. A service is determined 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

Cluster Database Performance Page

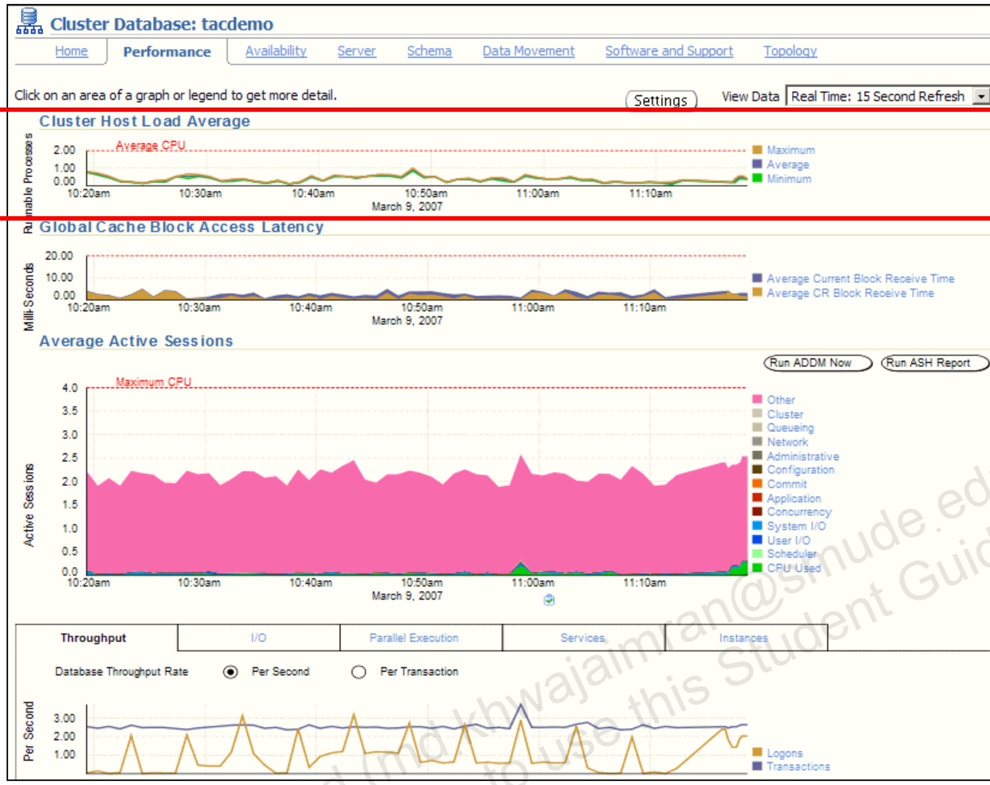
The Cluster Database Performance page provides a quick glimpse of the performance statistics for a database. Enterprise Manager accumulates data from each instance over specified periods of time (known as *collection-based data*). Enterprise Manager also provides current data from each instance (known 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 shows a partial view of the Cluster Database Performance page. You access this page by clicking the Performance tab on the Cluster Database Home page.

Determining Cluster Host Load Average



ORACLE

Determining Cluster Host Load Average

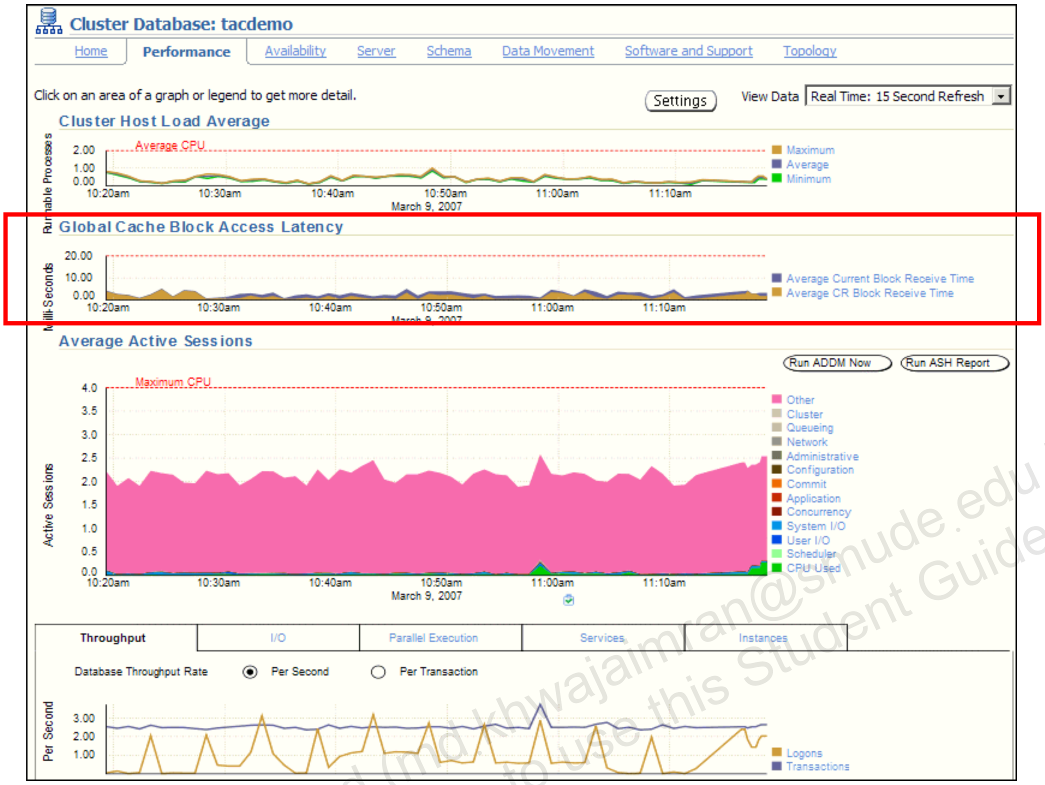
The Cluster Host Load Average chart on the Cluster Database Performance page shows potential problems that are outside the database. The chart shows maximum, average, and minimum load values for available nodes in the cluster for the previous hour.

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, 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 label Average, 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 in 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



Determining Global Cache Block Access Latency

The Global Cache Block Access Latency chart shows the latency for each different type of data block requests: 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.

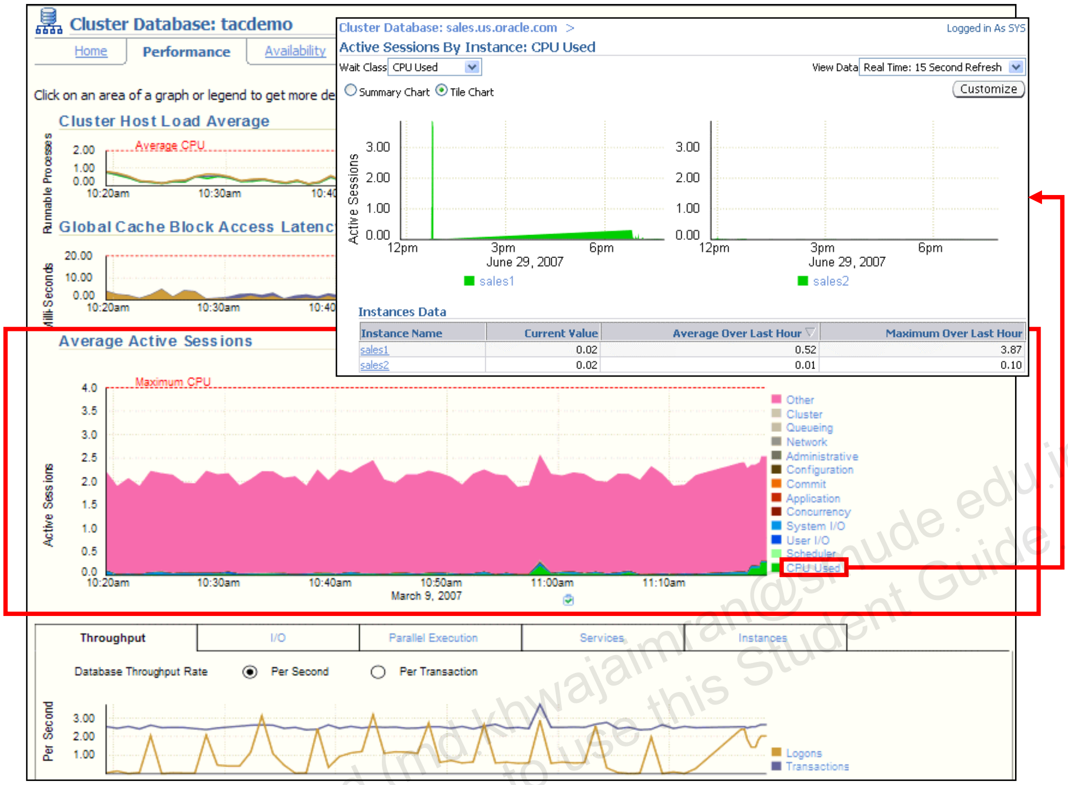
You can click either metric for the Global Cache Block Access Latency chart to view more detailed information about that type of cached block.

If the Global Cache Block Access Latency chart shows high latencies (high elapsed times), the cause can be 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, the first step in resolving the problem should be to go to the Cluster Cache Coherency page for more detailed information.

Determining Average Active Sessions



Instance Name	Current Value	Average Over Last Hour	Maximum Over Last Hour
sales1	0.02	0.52	3.87
sales2	0.02	0.01	0.10

ORACLE

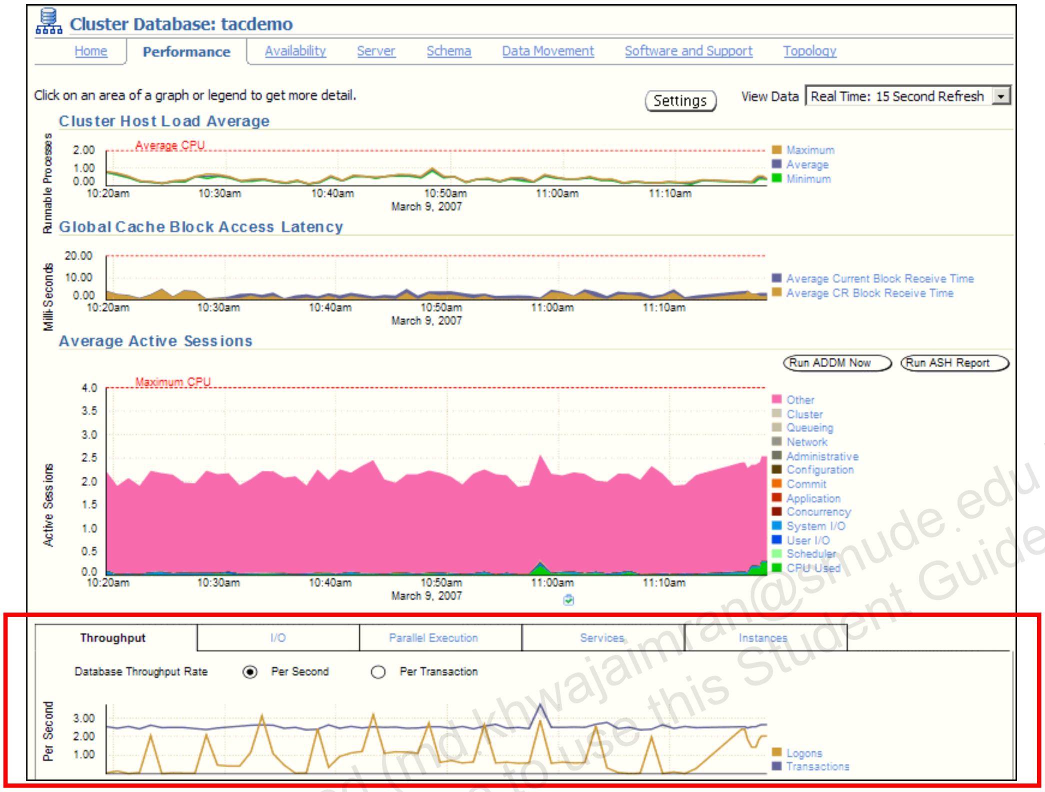
Determining Average Active Sessions

The Average Active Sessions chart in 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 to wait time helps 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, 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. 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 using tile charts, or you can combine the data into a single summary chart.

Determining Database Throughput



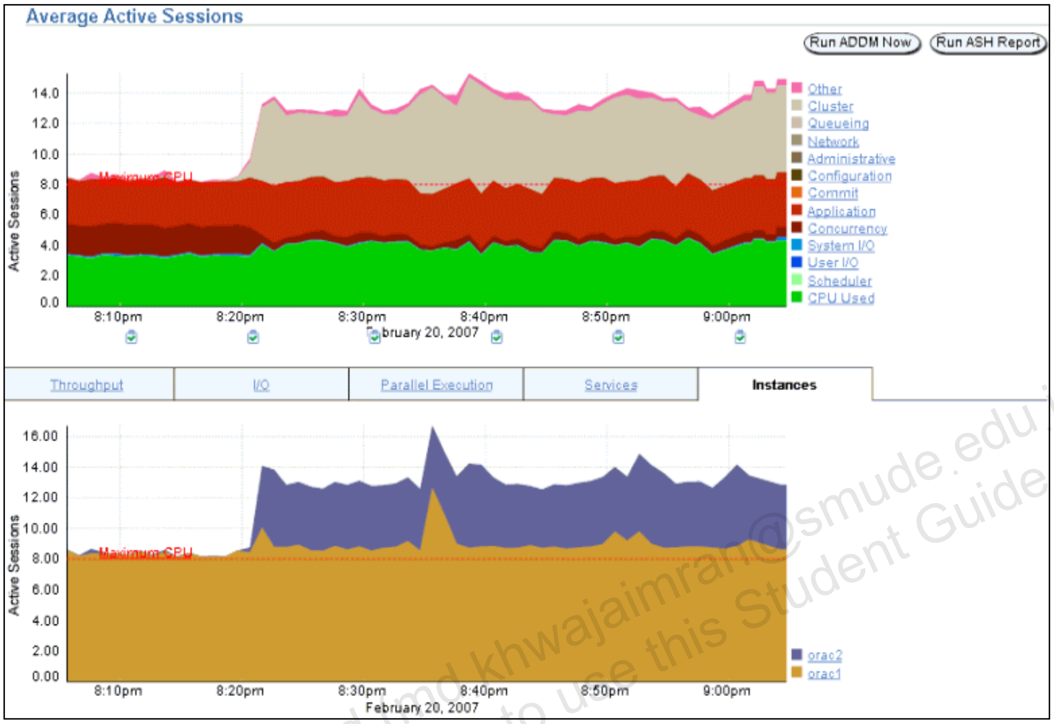
Determining Database Throughput

The last chart on the Performance page monitors the usage of various database resources. By clicking the Throughput tab at the top of this chart, you can 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 amount of physical reads compared to the redo size per second. The Per Transaction view shows the amount of physical reads compared to the redo size per transaction. Logons is the number of users that are logged on to the database.

You can also obtain information at the instance level by clicking one of the legends to the right of the charts to access the “Database Throughput by Instance” page. 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 that is consuming the greatest resources. Click an instance name legend just under the chart to go to the Top Sessions subpage of the Top Consumers page for that instance.

Determining Database Throughput



Determining Database Throughput (continued)

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, which summarizes any resource contention that appears in the Average Active Sessions chart. You can thus quickly determine how much of the database work is being performed on each instance.

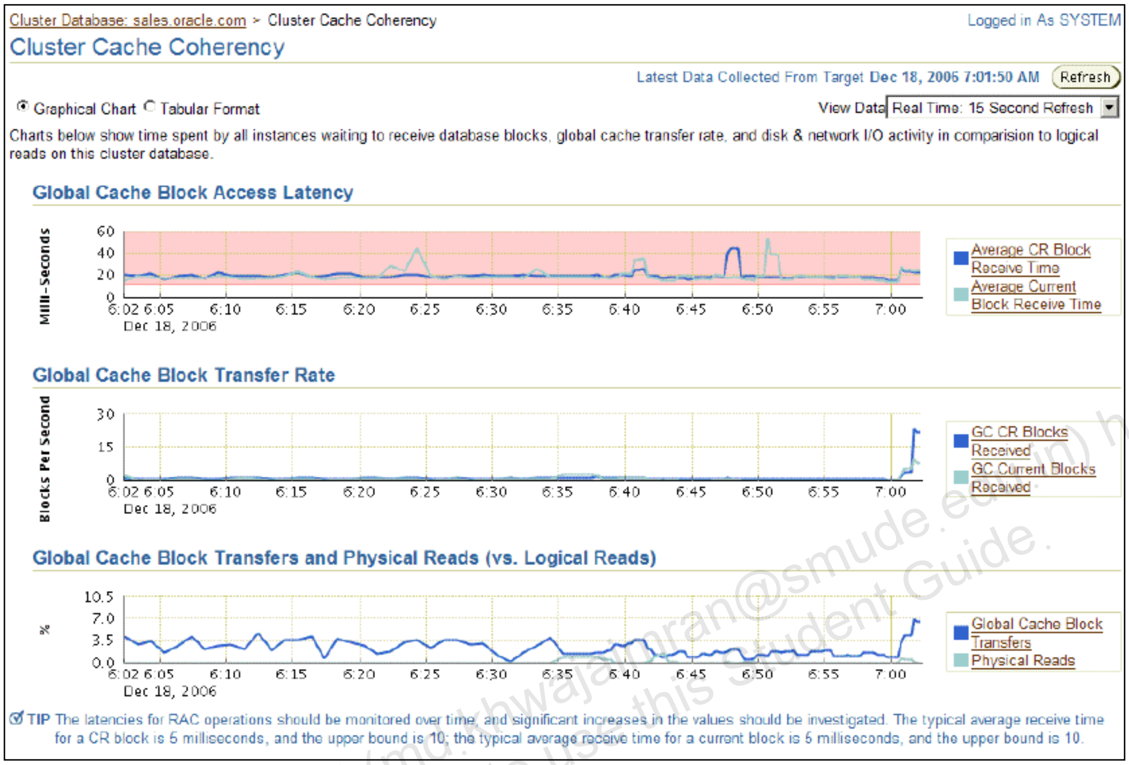
You can also obtain information at the instance level by clicking one of the legends to the right of the chart to access the Top Sessions page, where you can view real-time data showing the sessions that consume the greatest system resources. In the graph in the slide, the `orac2` instance after 8:20 PM consistently shows more active sessions than the `orac1` instance.

Accessing the Cluster Cache Coherency Page

Block Class

Segment name

Segment name



ORACLE

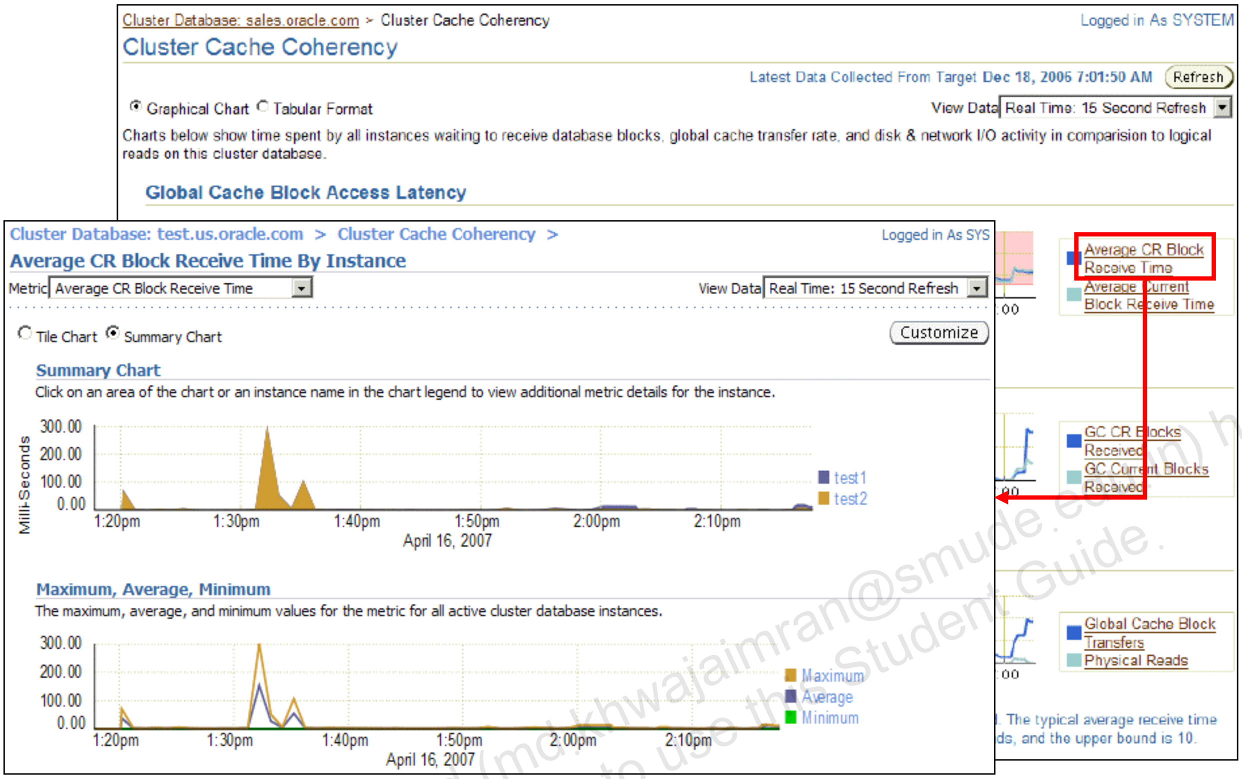
Accessing the Cluster Cache Coherency Page

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 the instances that 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

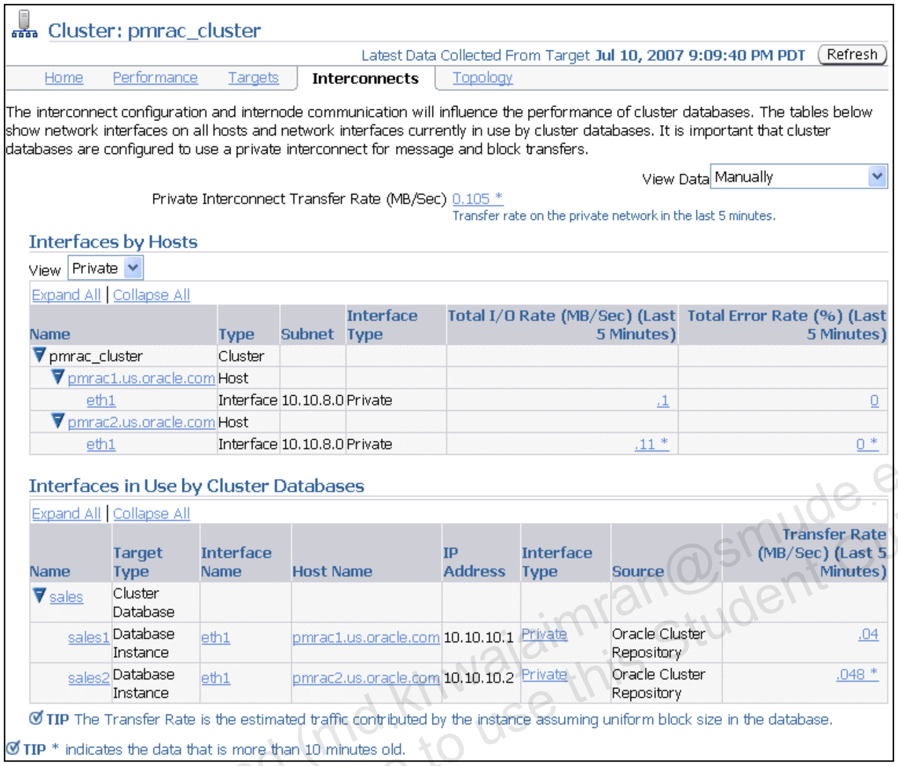


ORACLE

Accessing the Cluster Cache Coherency Page (continued)

- **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 Cluster Interconnects Page



ORACLE

Viewing Cluster Interconnects Page

The Cluster Interconnects page is useful for monitoring the interconnect interfaces, determining configuration issues, and identifying transfer rate-related issues, including excess traffic. This page helps determine the load added by individual instances and databases on the interconnect. Sometimes you can immediately identify interconnect delays that are due to applications outside Oracle.

You can use this page to perform the following tasks:

- Viewing all interfaces that are configured across the cluster
- Viewing statistics for the interfaces, such as absolute transfer rates and errors
- Determining the type of interfaces, such as private or public
- Determining whether the instance is using a public or private network
- Determining which database instance is currently using which interface
- Determining how much the instance is contributing to the transfer rate on the interface

The Private Interconnect Transfer Rate value shows a global view of the private interconnect traffic, which is the estimated traffic on all the private networks in the cluster. The traffic is calculated as the summary of the input rate of all private interfaces that are known to the cluster.

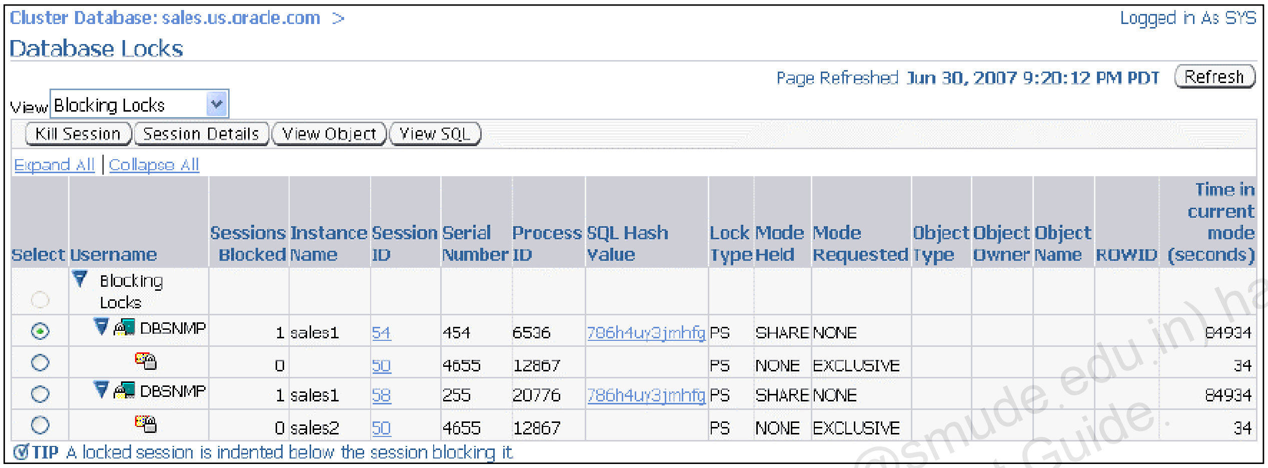
Viewing Cluster Interconnects Page (continued)

On the Cluster Interconnects page, you can access the Hardware Details page, where you obtain more information about all the network interfaces defined on each node of your cluster.

Similarly, you can access the Transfer Rate metric page, which collects the internode communication traffic of a cluster database instance. The critical and warning thresholds of this metric are not set by default. You can set them according to the speed of your cluster interconnects.

Note: You can query the `V$CLUSTER_INTERCONNECTS` view to see information about the private interconnect.

Viewing the Database Locks Page



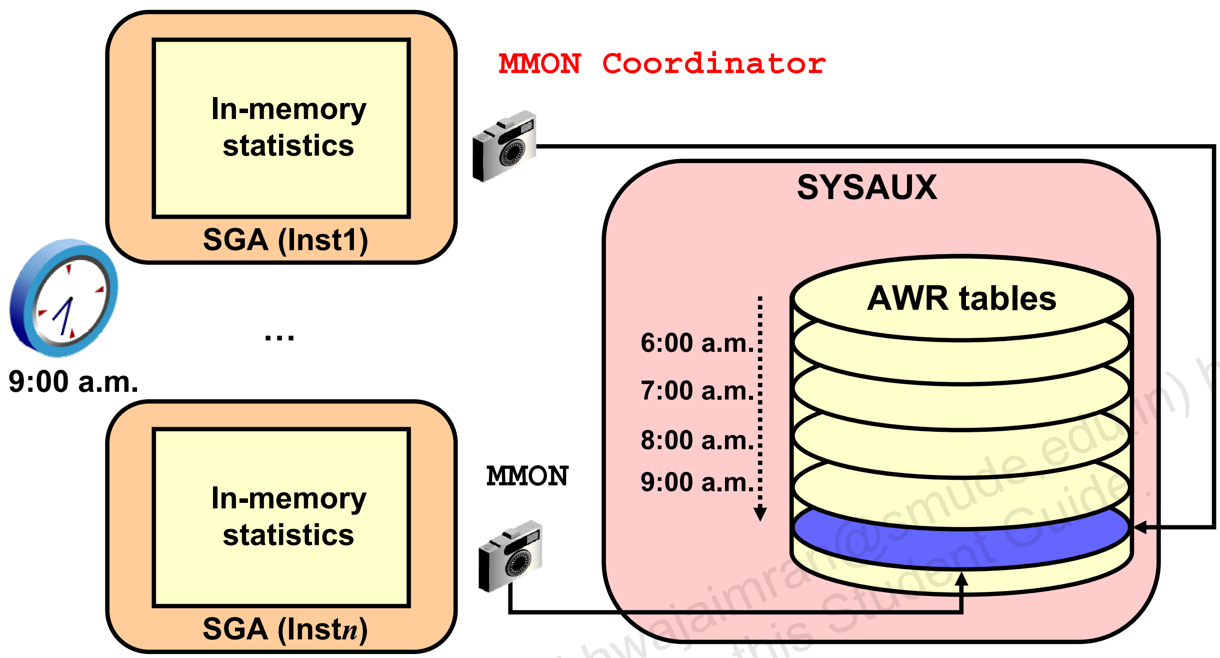
Select	Username	Sessions Blocked	Instance Name	Session ID	Serial Number	Process ID	SQL Hash Value	Lock Type	Mode Held	Mode Requested	Object Type	Object Owner	Object Name	ROWID	Time in current mode (seconds)
	Blocking Locks														
◉	DBSNMP	1	sales1	54	454	6536	786h4uy3jmhfg	PS	SHARE	NONE					84934
○		0		50	4655	12867		PS	NONE	EXCLUSIVE					34
○	DBSNMP	1	sales1	58	255	20776	786h4uy3jmhfg	PS	SHARE	NONE					84934
○		0	sales2	50	4655	12867		PS	NONE	EXCLUSIVE					34

Viewing the Database Locks Page

Use the Database Locks page to determine if 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



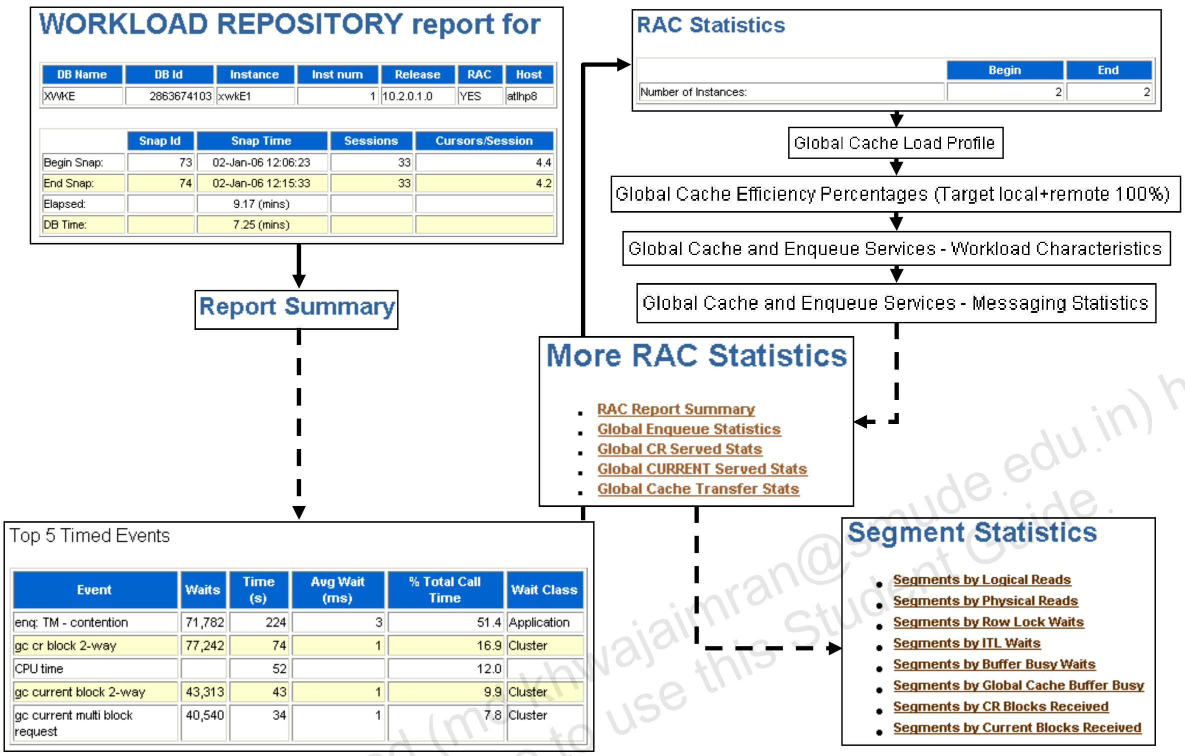
AWR Snapshots in RAC

In Oracle Database 11**g**, AWR has been enhanced to provide more comprehensive support of RAC. 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

AWR Reports and RAC: Overview

The RAC-related statistics in an AWR report are organized in different sections. A RAC statistics section appears after the Top 5 Timed Events. This section contains:

- 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
- **Global Cache Load Profile:** Essentially lists the number of blocks and messages that are sent and received, as well as the number of fusion writes
- **Global Cache Efficiency Percentages:** Indicate the percentage of buffer is divided into buffers received from the disk, local cache, and remote caches. Ideally, the percentage of disk buffer access should be close to zero.
- **GCS and GES Workload Characteristics:** 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.

AWR Reports and RAC: Overview (continued)

- **GCS and GES Messaging Statistics:** The most important statistic here is the *average message sent queue time on ksxp,* which gives a good indicator of how well the IPC works. Average numbers should be less than 1 ms.

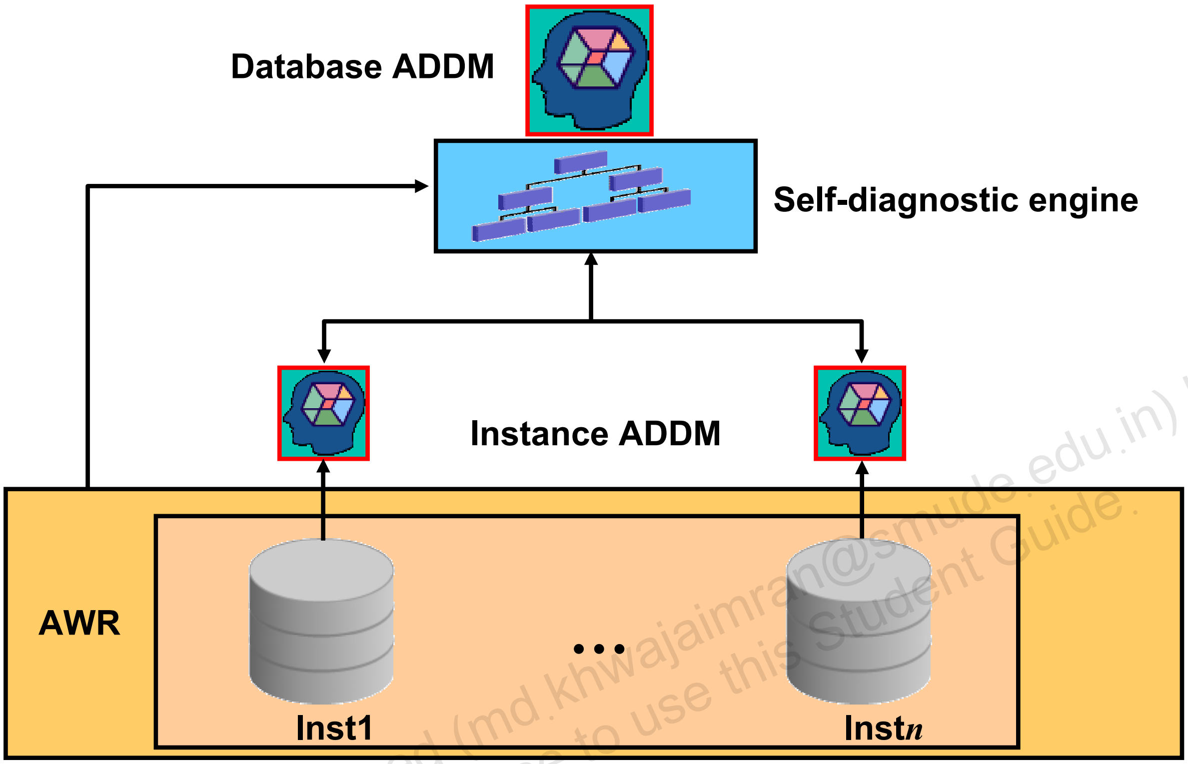
Additional RAC statistics are then organized in the following sections:

- The Global Enqueue Statistics section contains data extracted from `V$GES_STATISTICS`.
- The Global CR Served Stats section contains data from `V$CR_BLOCK_SERVER`.
- The Global CURRENT Served Stats section contains data from `V$CURRENT_BLOCK_SERVER`.
- The Global Cache Transfer Stats section contains data from `V$INSTANCE_CACHE_TRANSFER`.

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, see the *Oracle Database Reference*.

Automatic Database Diagnostic Monitor for RAC



Automatic Database Diagnostic Monitor for RAC

Oracle Database 11*g* offers an extension to the set of functionality that increases the database's manageability by offering clusterwide analysis of performance. A special mode of Automatic Database Diagnostic Monitor (ADDM) analyzes a RAC database cluster and reports on issues that affect the entire cluster as well as on those that affect individual instances. This mode is called *database ADDM* (as opposed to instance ADDM, which already existed with Oracle Database 10*g*).

Database ADDM for RAC is not simply a report of reports. Rather, it has independent analysis that is appropriate for RAC.

Note: The Database ADDM report is generated on the AWR snapshot coordinator.

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
- Eliminates need to investigate *n* reports to spot common problems

ORACLE

Automatic Database Diagnostic Monitor for RAC (continued)

In Oracle Database 11*g*, 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 equivalent to Oracle Database 10*g* ADDM and is now 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. You must, however, maintain the same database version throughout the entire time period.

Database ADDM runs automatically after each snapshot is taken. The automatic instance ADDM runs are the same as in Oracle Database 10*g*. 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 etc.)
- Top consumers of multiblock requests
- Lost blocks
- Information about interconnect devices (warns about using PUBLIC interfaces)
- Throughput of devices: How much of it is used by Oracle and for what purpose (GC, locks, PQ)

ORACLE

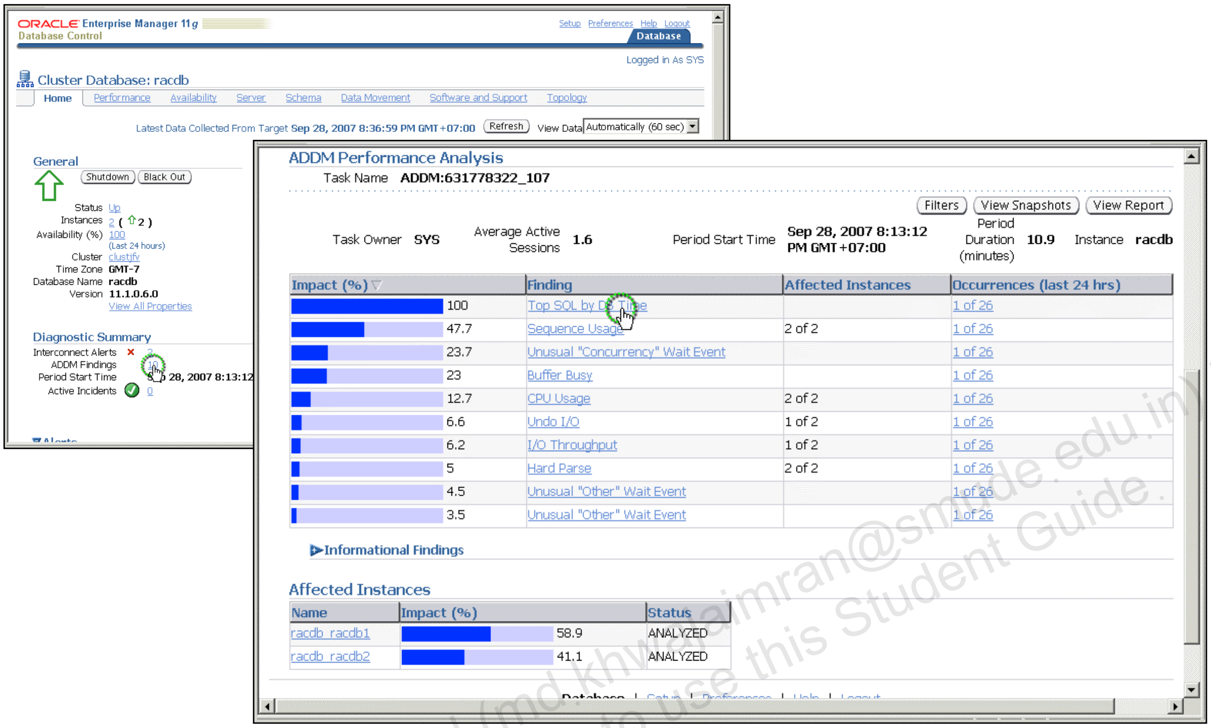
What Does ADDM Diagnose for RAC?

Data sources are:

- Wait events (especially Cluster class and buffer busy)
- ASH
- 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



EM Support for ADDM for RAC

Oracle Database 11*g* 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 Affected Instances chart shows how much each instance is impacted by these findings. The display indicates the percentage impact for each instance.

Summary

In this lesson, you should have learned how to:

- Determine RAC-specific tuning components
- Tune instance recovery in RAC
- 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 in RAC
- Use Automatic Database Diagnostic Monitor in RAC

ORACLE

Practice 6: Overview

This practice covers studying a scalability case by using the ADDM.

7

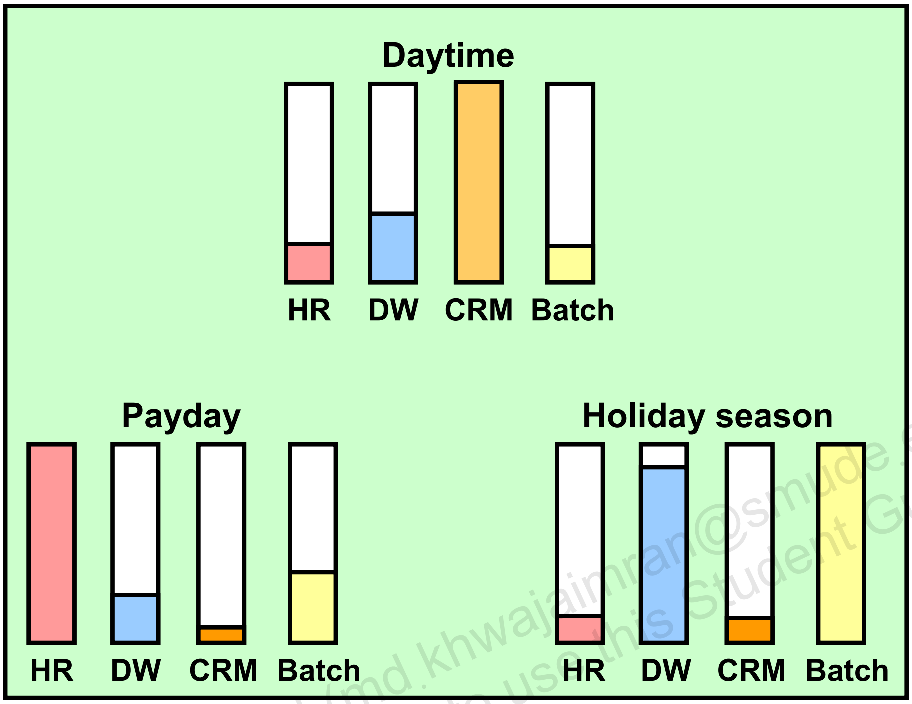
Services

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
- Set performance-metric thresholds on services
- Configure services aggregation and tracing

Traditional Workload Dispatching

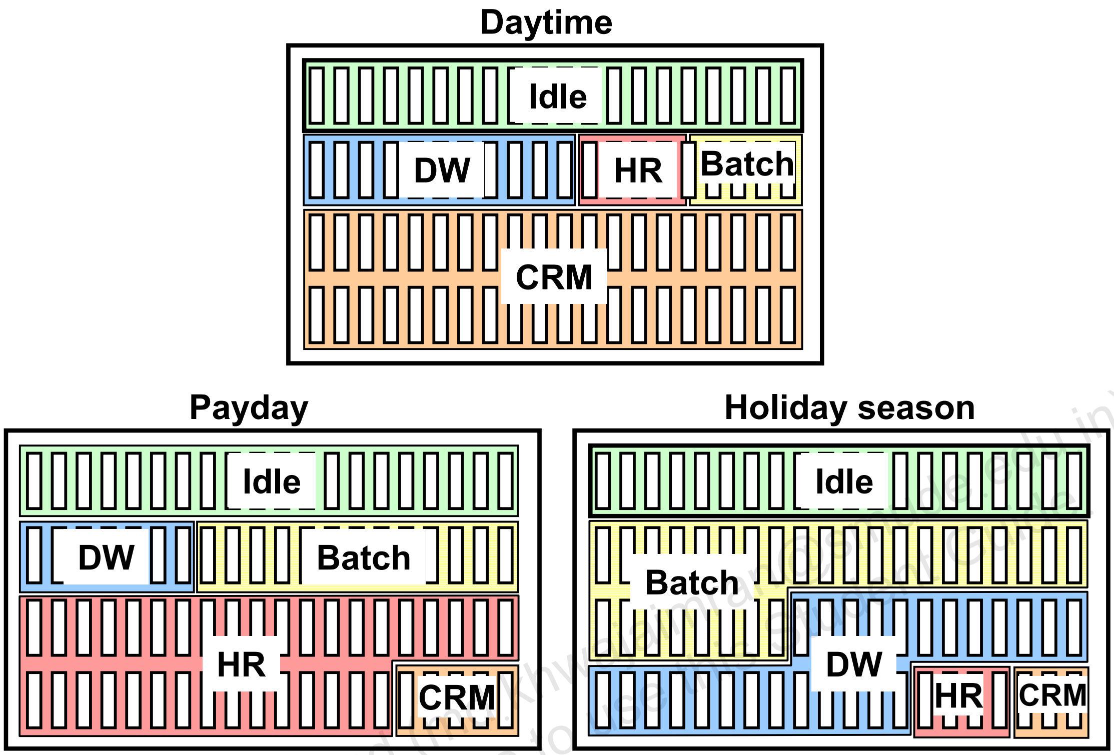


Traditional Workload Dispatching

In a standard environment, isolated computing units of different sizes are permanently dedicated to specific applications such as Human Resources, Data Warehouses, Customer Relationship Management, and Retail Batches.

These computing units need to be sized for their peak workload. As the peak workload occurs for some hours only, a considerable amount of resources is idle for a long time.

Grid Workload Dispatching



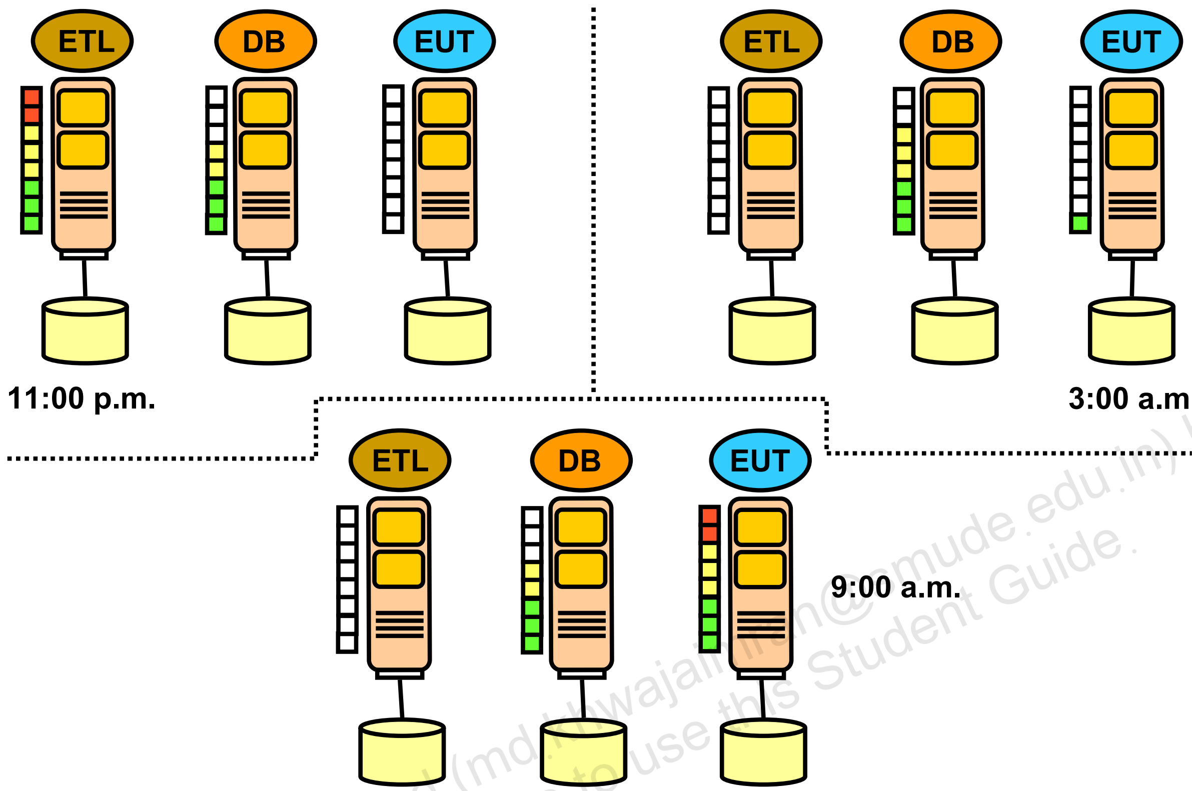
Grid Workload Dispatching

With grid computing, a global pool of computing units can be provided, and the computing units can be temporarily assigned to specific applications. Computing units can then be dynamically exchanged between applications. During business hours, more units can be used for CRM applications, and after business hours, some of them can be transferred to Retail Batches.

Grid computing minimizes unused resources. This means that overall a grid-enabled environment needs less computing power than an environment that is not grid enabled.

In the example, 25 percent of the computing resource units are idle. This unused extra capacity is there so that service levels can still be met in case of failure of components, such as nodes or instances, and also to deal with unexpected workloads. This is much better than the industry average of 70 to 90 percent idle rates when each machine is sized for its individual maximum.

Data Warehouse: Example



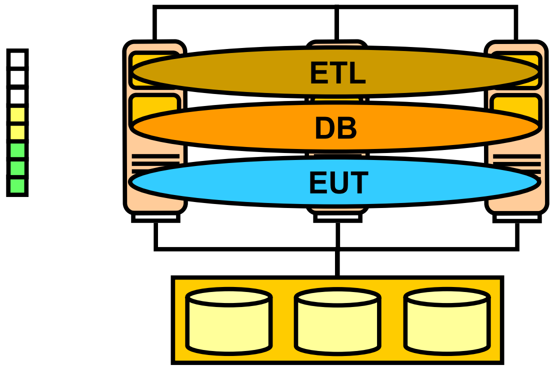
Data Warehouse: Example

Previously, building a business intelligence system required the integration of multiple server products. The result was that such systems were unnecessarily complex. The integration of multiple servers was costly. After the system was implemented, there was an ongoing administration cost in maintaining different servers and keeping the data synchronized across all servers.

- **At 11:00 p.m.:** ETL server is busy using ETL outside the database; moderate load on database; no load on end user server.
- **At 3:00 a.m.:** No load on ETL server; moderate load on database (canned reporting, aggregation, and potential data mart maintenance); no load on end user server.
- **At 9:00 a.m.:** No load on ETL server; moderate load on database; end user server is busy using analysis outside the database.

In addition, each system has to be sized according to the expected workload peaks.

RAC and Data Warehouse: An Optimal Solution

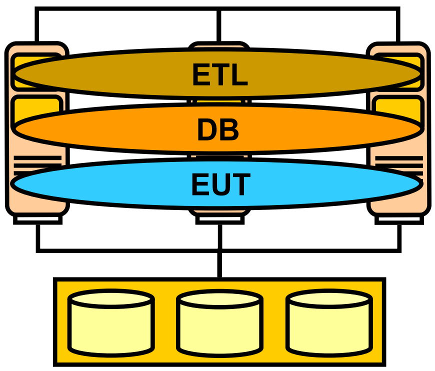


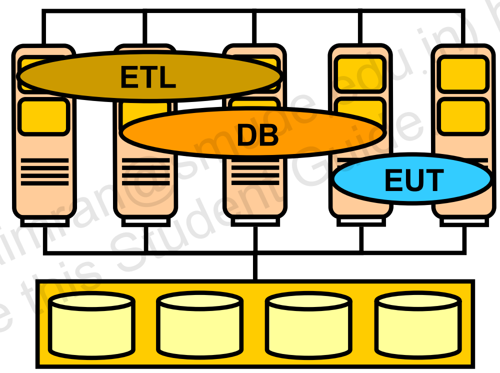
- **Maximum total workload used for system sizing:**

  Size(Workload max total) < Σ Size(workload max components)
- **The entire workload is evenly spread across all nodes at any point in time.**

Next Step

What works for a single data warehouse …





… works in a larger environment as well.

What Is a Service?

- Is a means of grouping sessions that are doing the same kind of work
- Provides a single-system image instead of a multiple-instances image
- Is a part of the regular administration tasks that provide dynamic service-to-instance allocation
- Is the base for High Availability of connections
- Provides a new performance-tuning dimension

What Is a Service?

The concept of a service was first introduced in Oracle8i as a means for the listener to perform connection load balancing between nodes and instances of a cluster. However, the concept, definition, and implementation of services have been dramatically expanded. Services are a feature for workload management that organizes the universe of work execution within the database to make that work more manageable, measurable, tunable, and recoverable. A service is a grouping of related tasks within the database with common functionality, quality expectations, and priority relative to other services. A service provides a single-system image for managing competing applications running within a single instance and across multiple instances and databases.

Using standard interfaces, such as the DBCA, Enterprise Manager, and `SRVCTL`, services can be configured, administered, enabled, disabled, and measured as a single entity.

Services provide availability. Following outages, a service is recovered quickly and automatically at surviving instances.

Services provide a new dimension to performance tuning. With services, workloads are visible and measurable. Tuning by “service and SQL” replaces tuning by “session and SQL” in the majority of systems where sessions are anonymous and shared.

Services are dynamic in that the number of instances a service runs on can be augmented when load increases, and reduced when load declines. This dynamic resource allocation enables a cost-effective solution for meeting demands as they occur.

High Availability of Services in RAC

- Services are available continuously with load shared across one or more instances.
- Additional instances are made available in response to failures.
- Preferred instances:
  - Set the initial cardinality for the service
  - Are the first to start the service
- Available instances are used in response to preferred-instance failures.

High Availability of Services in RAC

With RAC, the focus of High Availability (HA) is on protecting the logically defined application services. This focus is more flexible than focusing on high availability of instances.

Services must be location independent and the RAC HA framework is used to implement this. Services are made available continuously with load shared across one or more instances in the cluster. Any instance can offer services in response to run-time demands, failures, and planned maintenance.

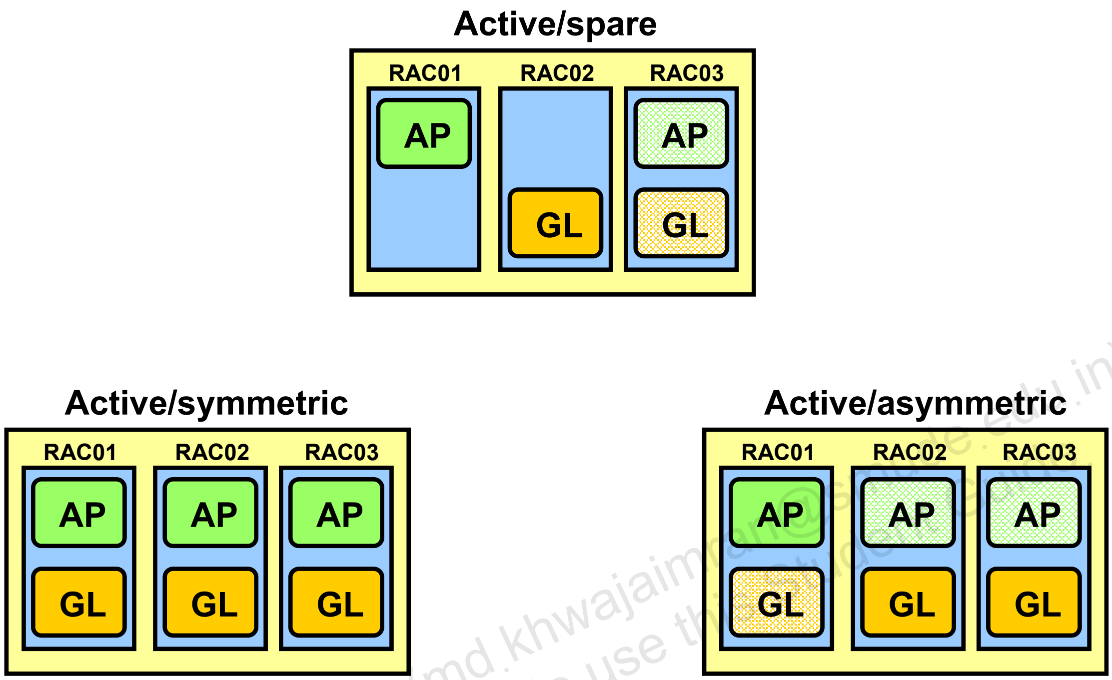
To implement the workload balancing and continuous availability features of services, Oracle Clusterware stores the HA configuration for each service in the Oracle Cluster Registry (OCR). The HA configuration defines a set of preferred and available instances that support the service.

A preferred instance set defines the number of instances (cardinality) that support the corresponding service. It also identifies every instance in the cluster that the service will run on when the system first starts up.

An available instance does not initially support a service. However, it begins accepting connections for the service when a preferred instance cannot support the service. If a preferred instance fails, then the service is transparently restored to an available instance defined for the service.

Note: An available instance can become a preferred instance and vice versa.

Possible Service Configuration with RAC



ORACLE

Possible Service Configuration with RAC

- **Active/spare:** With this service configuration, the simplest redundancy known as primary/secondary, or 1+1 redundancy is extended to the general case of N+M redundancy, where N is the number of primary RAC instances providing service, and M is the number of spare RAC instances available to provide the service. An example of this solution is a three-node configuration in which one instance provides the AP service, the second instance provides the GL service, and the third instance provides service failover capability for both services. The spare node can still be available for other applications during normal operation.
- **Active/symmetric:** With this service configuration, the same set of services is active on every instance. An example of this is illustrated in the slide, with both AP and GL services being offered on all three instances. Each instance provides service load-sharing and service failover capabilities for the other.
- **Active/asymmetric:** With this service configuration, services with lower capacity needs can be defined with single cardinality and configured as having all other instances capable of providing the service in the event of failure. The slide shows the AP service running on only one instance, and the GL service running on two instances. The first instance supports the AP services and offers failover for the GL service. Likewise, the second and third instances support the GL service and offer failover for AP. If either the first or third instance dies, then GL and AP are still offered through the second instance.

Service Attributes

- Global unique name
- Network name
- Load Balancing Advisory goal*
- Distributed transactions flag*
- Advance queuing notification characteristics for OCI and ODP.NET clients*
- Failover characteristics*
- Connection load-balancing algorithm*
- Threshold
- Priority
- High-availability configuration*

Service Attributes

When you create new services for your database, you should define each service's workload management characteristics. The characteristics of a service include:

- A unique global name to identify the service
- A Net Service name that a client uses to connect to the service
- A service goal that determines whether work requests are made to the service based on best service quality (service response time), or best throughput (how much work is completed in a unit of time), as determined by the Load Balancing Advisory
- An indicator that determines whether the service will be used for distributed transactions
- An indicator that determines whether RAC HA events are sent to OCI and ODP.NET clients that have registered to receive them through Advanced Queuing
- The characteristics of session failovers when using transparent application failover
- The method for load balancing (which you can define) of connections for each service:
  - **`SHORT`:** Use Load Balancing Advisory.
  - **`LONG`:** Use session count by service.
  - **`NONE`:** Run queue.
- Services metric thresholds (which you can define) for response time and CPU consumption
- Services to consumer groups (which you can map) instead of usernames
- How the service is distributed across instances when the system first starts

Note: Attributes highlighted with an `*` in the slide cannot be defined for single-instance environments.

Service Types

- **Application services:**
  - **Limit of 100 services per database**
- **Internal services:**
  - `SYS$BACKGROUND`
  - `SYS$USERS`
  - **Cannot be deleted or changed**

Service Types

Oracle Database 11*g* supports two broad types of services: application services and internal services. Application services are mainly functional maps to workloads. Sessions doing work for a common business function are grouped together. For Oracle E-Business suite, AP, AR, GL, MFG, WIP, BOM, and so on create a functional division of work within the database and can thus be categorized as services.

In addition to application services, the RDBMS also 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.

There is a limitation of 100 application services per database that you can create. Also, a service name is restricted to 64 characters.

Note: Shadow services are also included in the application service category. For more information about shadow services, see the lesson titled "High Availability of Connections." In addition, a service is also created for each Advanced Queue created. However, these types of services are not managed by Oracle Clusterware. Using service names to access a queue provides location transparency for the queue within a RAC database.

Service Goodness

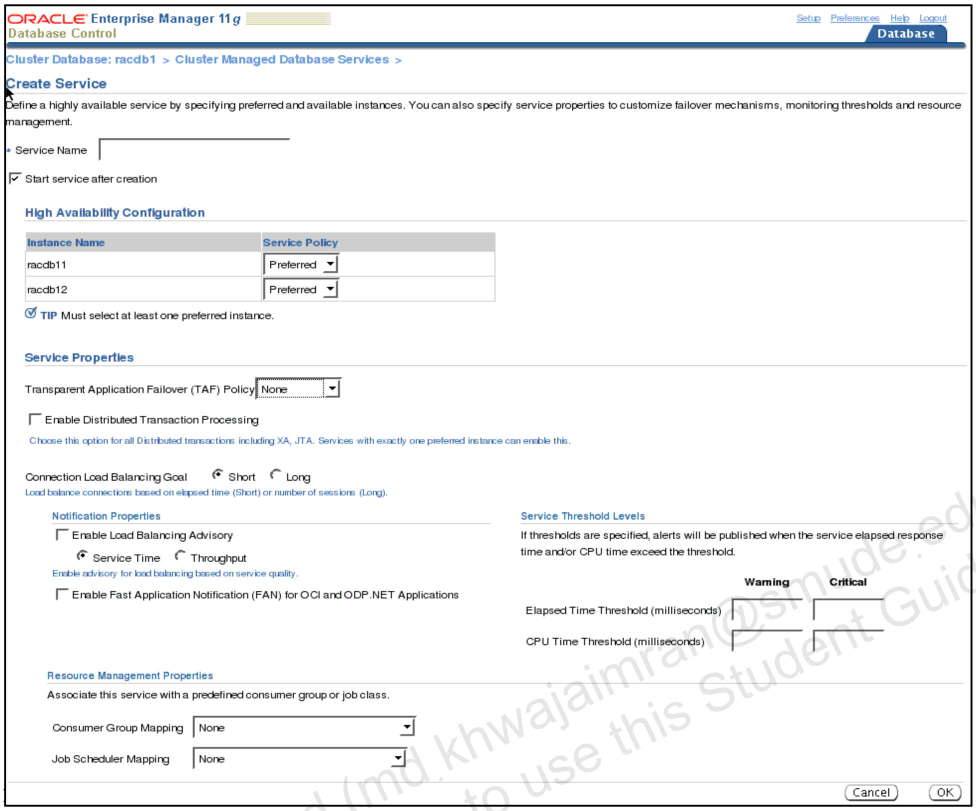
- **Value that reflects the ability of a node and instance to deliver work for a service**
- **Appropriate metrics used to compute goodness depending on the service goal:**
  - **Service time**
  - **Service throughput**
- **Automatically computed at each instance by MMNL**

Service Goodness

Service goodness is a measure of the attractiveness of an instance to provide resources for a service. MMNL calculates moving average for service time and service throughput. These values are exposed in GV$SERVICEMETRIC and GV$SERVICEMETRIC_HISTORY.

Note: MMNL is the Manageability MoNitor Light process.

Create Services with Enterprise Manager



ORACLE

Create Services with Enterprise Manager

On your Cluster Database home page, click the Availability tab. On the Availability tabbed page, click Cluster Managed Database Services. On the Cluster Managed Database Services page, click Create Service.

Use the Create Service page to configure a new service for which you do the following:

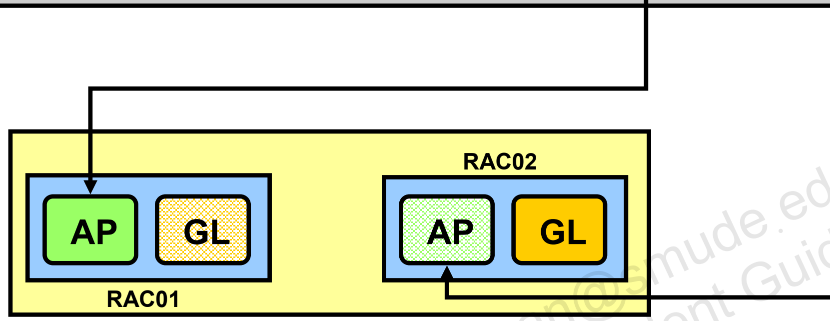
- Select the desired service policy for each instance configured for the cluster database.
- Select the desired service properties. Refer to the “Service Attributes” topic in this lesson for more information about the properties you can specify on this page. The Transparent Application Failover (TAF) policy attribute on this page does not configure server-side TAF.

Note: Although Enterprise Manager configures Oracle Clusterware resources for your newly created services, it does not generate the corresponding entries in your `tnsnames.ora` files. You have to manually edit them. For that, you can use the `srvctl config database` command with the `–t` option, which displays the TNS entries that you should use for the services created with `srvctl`. Here is an example:

```
$ srvctl config database -d xwkE -t
Example client-side TNS entry for service Standard:
Standard = (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=db_vip)
(PORT=dedicated_port))(CONNECT_DATA=(SERVICE_NAME=Standard))) …
```

Create Services with SRVCTL

```
$ srvctl add service -d PROD -s GL -r RAC02 -a RAC01
$ srvctl add service -d PROD -s AP -r RAC01 -a RAC02
```



Create Services with SRVCTL

The example in the slide shows a two-node cluster with an instance named RAC01 on one node and an instance called RAC02 on the other. The cluster database name is PROD.

Two services—AP and GL—are created and stored in the cluster repository 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.

Note that it is possible to assign more than one instance with both the `-r` and `-a` options. However, `-r` is mandatory whereas `-a` is optional.

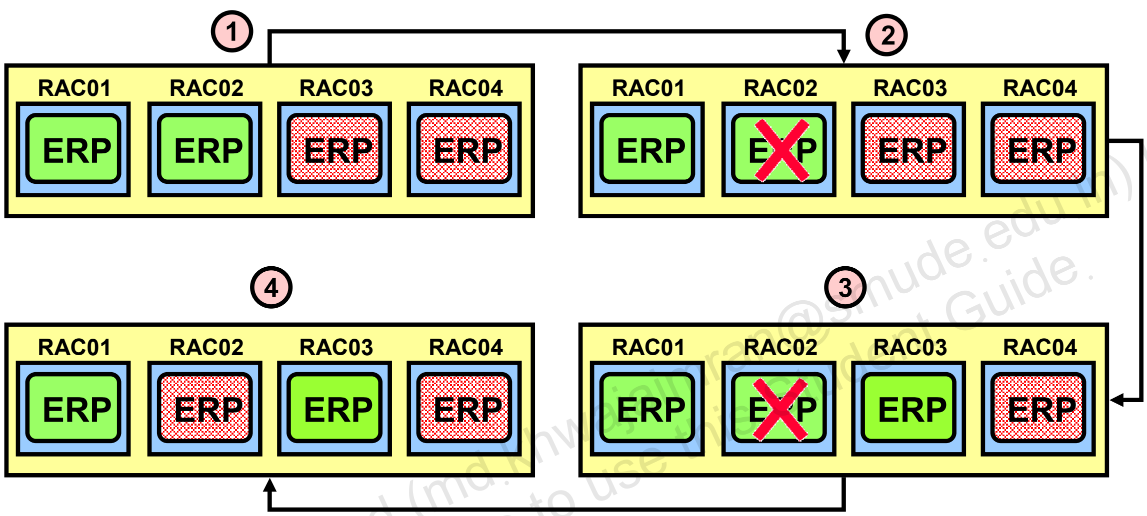
Services enable you to move beyond the simple two-node primary/secondary configuration of RAC Guard in Oracle9*i*.

With Oracle Database 11*g*, multiple primary nodes can support a service with RAC. Possible configurations for service placement are active/spare, active/symmetric, and active/asymmetric. Once a service is created, you can use the `srvctl add service` command with the `–u –r` or `–u –a` option to add a preferred or available instance to a service.

Note: You can also set up a service for Transparent Application Failover by using the `–P` option of `SRVCTL`. Possible values are `NONE`, `BASIC`, and `PRECONNECT`.

Preferred and Available Instances

```
$ srvctl add service -d PROD -s ERP \
   -r RAC01,RAC02 -a RAC03,RAC04
```



Preferred and Available Instances

In this example, it is assumed that you have a four-node cluster.

You define a service called ERP. The preferred instances for ERP are RAC01 and RAC02. The available instances for ERP are RAC03 and RAC04.

1. Initially, ERP connections are directed only to RAC01 and RAC02.
2. RAC02 is failing and goes down.
3. Oracle Clusterware detects the failure of RAC02, and because the cardinality of ERP is 2, Oracle Clusterware restores the service on one of the available instances, in this case RAC03.
4. ERP connection requests are now directed to RAC01 and RAC03, which are the instances that currently offer the service. Although Oracle Clusterware is able to restart RAC02, the ERP service does not fall back to RAC02. RAC02 and RAC04 are now the instances that are accessible if subsequent failures occur.

Note: If you want to fall back to RAC02, you can use `SRVCTL` to relocate the service. This operation can be done manually by the DBA, or by coding the `SRVCTL` relocation command using a callback mechanism to automate the fallback. However, relocating a service is a disruptive operation.

Modify Services with the DBMS_SERVICE Package

Modify a service in RAC with the following:

- SRVCTL
- Enterprise Manager
- DBMS_SERVICE.MODIFY_SERVICE

```
exec DBMS_SERVICE.MODIFY_SERVICE (
     'SELF-SERVICE', 'SELF-SERVICE.us.oracle.com',
      goal      => DBMS_SERVICE.GOAL_SERVICE_TIME,
      clb_goal  => DBMS_SERVICE.CLB_GOAL_SHORT);
```


Modify Services with the DBMS_SERVICE Package

The DBMS_SERVICE package supports the management of services in the database for the purposes of workload measurement, management, prioritization, and distributed transaction management. This package allows the creation, deletion, starting, and stopping of services in both RAC and a single instance. Additionally, it provides the ability to disconnect all sessions that connect to the instance with a service name when RAC removes that service name from the instance. Although the preferred method to create a service in a RAC environment is to use SRVCTL or Enterprise Manager, you can use the DBMS_SERVICE.CREATE_SERVICE procedure to create a service in a single-instance environment. This is because the DBMS_SERVICE package is not integrated with Oracle Clusterware to define preferred and available instances for the service.

However, you can use the DBMS_SERVICE.MODIFY_SERVICE procedure to modify some of the service's attributes in a RAC environment that cannot be modified using Enterprise Manager (for example, the FAILOVER_RETRIES parameter).

The example in the slide shows you how to use DBMS_SERVICE.MODIFY_SERVICE to set the Load Balancing Advisory goal for SELF-SERVICE. Refer to the "Service Attributes" topic in this lesson for more information about these attributes.

Note: For more information about the DBMS_SERVICE package, see the *PL/SQL Packages and Types Reference*.

Everything Switches to Services

- Data dictionary maintains services.
- The AWR measures the performance of services.
- The Database Resource Manager uses services in place of users for priorities.
- Job scheduler, Parallel Execution, and Streams queues run under services.
- RAC keeps services available within a site.
- Data Guard Broker with RAC keeps primary services available across sites.

ORACLE

Everything Switches to Services

Several database features support services. A session is tracked by the service with which it connects. In addition, performance-related statistics and wait events are also tracked by services.

The Automatic Workload Repository (AWR) manages the performance of services. It records the service performance, including SQL execution times, wait classes, and resources consumed by service. The AWR alerts the DBA when service response time thresholds are exceeded. Specific dynamic performance views report current service status with one hour of history.

In Oracle Database 11*g*, the Database Resource Manager is capable of managing services for prioritizing application workloads within an instance. In addition, jobs can now run under a service, as opposed to a specific instance. Parallel slave processes inherit the service of their coordinator.

The RAC HA framework keeps services available within a site. Data Guard Broker, in conjunction with RAC, migrates the primary service across Data Guard sites for disaster tolerance.

Use Services with Client Applications

```
ERP=(DESCRIPTION=
     (LOAD_BALANCE=on)
       (ADDRESS=(PROTOCOL=TCP)(HOST=node-1vip)(PORT=1521))
       (ADDRESS=(PROTOCOL=TCP)(HOST=node-2vip)(PORT=1521))
       (ADDRESS=(PROTOCOL=TCP)(HOST=node-3vip)(PORT=1521))
       (ADDRESS=(PROTOCOL=TCP)(HOST=node-4vip)(PORT=1521))
     (CONNECT_DATA=(SERVICE_NAME=ERP)))
```

```
url="jdbc:oracle:oci:@ERP"
```

```
url="jdbc:oracle:thin:@(DESCRIPTION=
     (LOAD_BALANCE=on)
       (ADDRESS=(PROTOCOL=TCP)(HOST=node-1vip)(PORT=1521))
       (ADDRESS=(PROTOCOL=TCP)(HOST=node-2vip)(PORT=1521))
       (ADDRESS=(PROTOCOL=TCP)(HOST=node-3vip)(PORT=1521))
       (ADDRESS=(PROTOCOL=TCP)(HOST=node-4vip)(PORT=1521))
     (CONNECT_DATA=(SERVICE_NAME=ERP)))"
```

ORACLE

Use Services with Client Applications

Applications and mid-tier connection pools select a service by using the TNS connection descriptor. The selected service must match the service that has been created using `SRVCTL` or the DBCA.

The address lists in each example in the slide use virtual IP addresses. Using the virtual IP addresses for client communication ensures that connections and SQL statements issued against a node that is down do not result in a TCP/IP timeout.

The first example in the slide shows the TNS connect descriptor that can be used to access the ERP service.

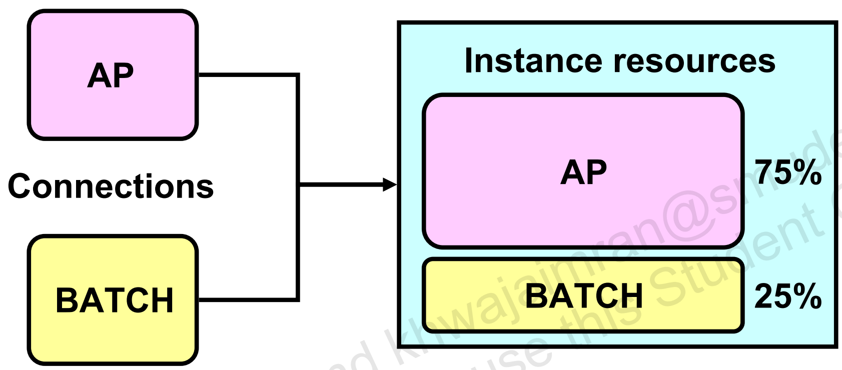
The second example shows the thick JDBC connection description using the previously defined TNS connect descriptor.

The third example shows the thin JDBC connection description using the same TNS connect descriptor.

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.

Use Services with the Resource Manager

- Consumer groups are automatically assigned to sessions based on session services.
- Work is prioritized by service inside one instance.



Use Services with the Resource Manager

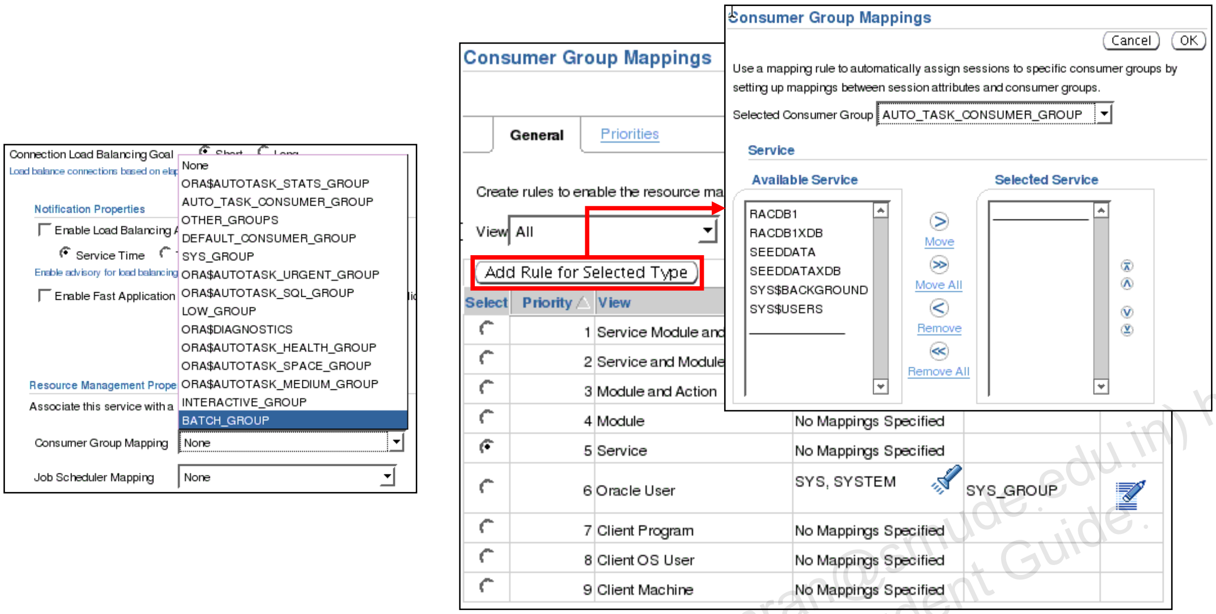
The Database Resource Manager (also called Resource Manager) enables you to identify work by using services. It manages the relative priority of services within an instance by binding services directly to consumer groups. When a client connects by using a service, the consumer group is assigned transparently 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.

Note: The Database Resource Manager applies only when the system resources are under heavy utilization. If there are free CPU cycles, BATCH (in the slide example) could get more than 25 percent.

Services and Resource Manager with EM



Services and Resource Manager with EM

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 clicking the Consumer Group Mappings link on the Server page.

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 shown in the right part of the slide).

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 on the Create Service page (as shown in the left part of the slide).

Services and the Resource Manager: Example

```
exec DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA;
exec DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(
      CONSUMER_GROUP => 'HIGH_PRIORITY',
      COMMENT => 'High priority consumer group');
exec DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING(
      ATTRIBUTE => DBMS_RESOURCE_MANAGER.SERVICE_NAME,
      VALUE => 'AP',
      CONSUMER_GROUP => 'HIGH_PRIORITY');
exec DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA;
```

```
exec -
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER_GROUP(-
 GRANTEE_NAME => 'PUBLIC',
 CONSUMER_GROUP => 'HIGH_PRIORITY',
 GRANT_OPTION => FALSE);
```


Services and the Resource Manager: Example

Assume that your site has two consumer groups called HIGH_PRIORITY and LOW_PRIORITY. These consumer groups map to a resource plan for the database that reflects either the intended ratios or the intended resource consumption.

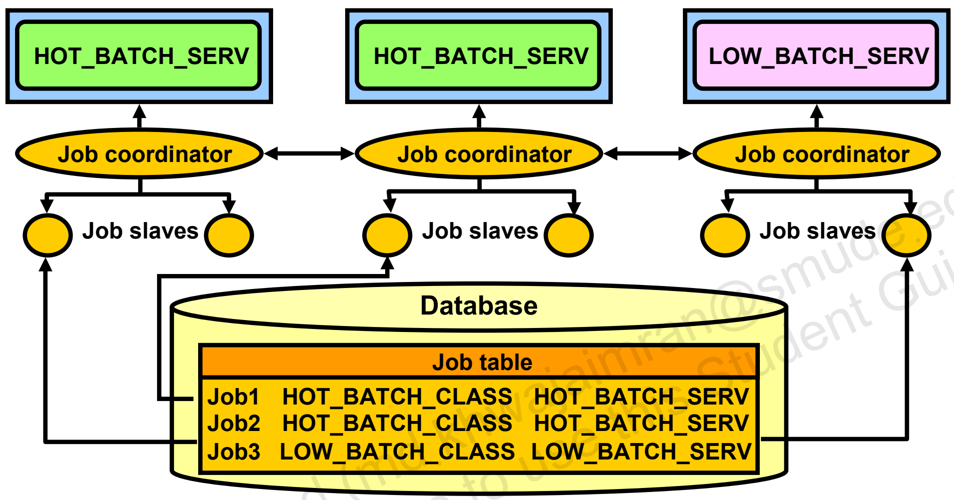
Before mapping services to consumer groups, you must first create the consumer groups and the resource plan for these consumer groups. The resource plan can be priority based or ratio based. The PL/SQL calls shown in the slide are used to create the HIGH_PRIORITY consumer group, and map the AP service to the HIGH_PRIORITY consumer group. You can use similar calls to create the LOW_PRIORITY consumer groups and map the BATCH service to the LOW_PRIORITY consumer group.

The last PL/SQL call in the example in the slide is executed because sessions are automatically assigned only to consumer groups for which they have been granted switch privileges. A similar call should be executed for the LOW_PRIORITY consumer group.

Note: For more information about the Database Resource Manager, refer to the *Oracle Database Administrator's Guide* and *PL/SQL Packages and Types Reference*.

Use Services with the Scheduler

- Services are associated with Scheduler classes.
- Scheduler jobs have service affinity:
  - High Availability
  - Load balancing



ORACLE

Use Services with the Scheduler

Just as in other environments, the Scheduler in a RAC environment uses one job table for each database and one job coordinator for each instance. The job coordinators communicate with each other to keep information current.

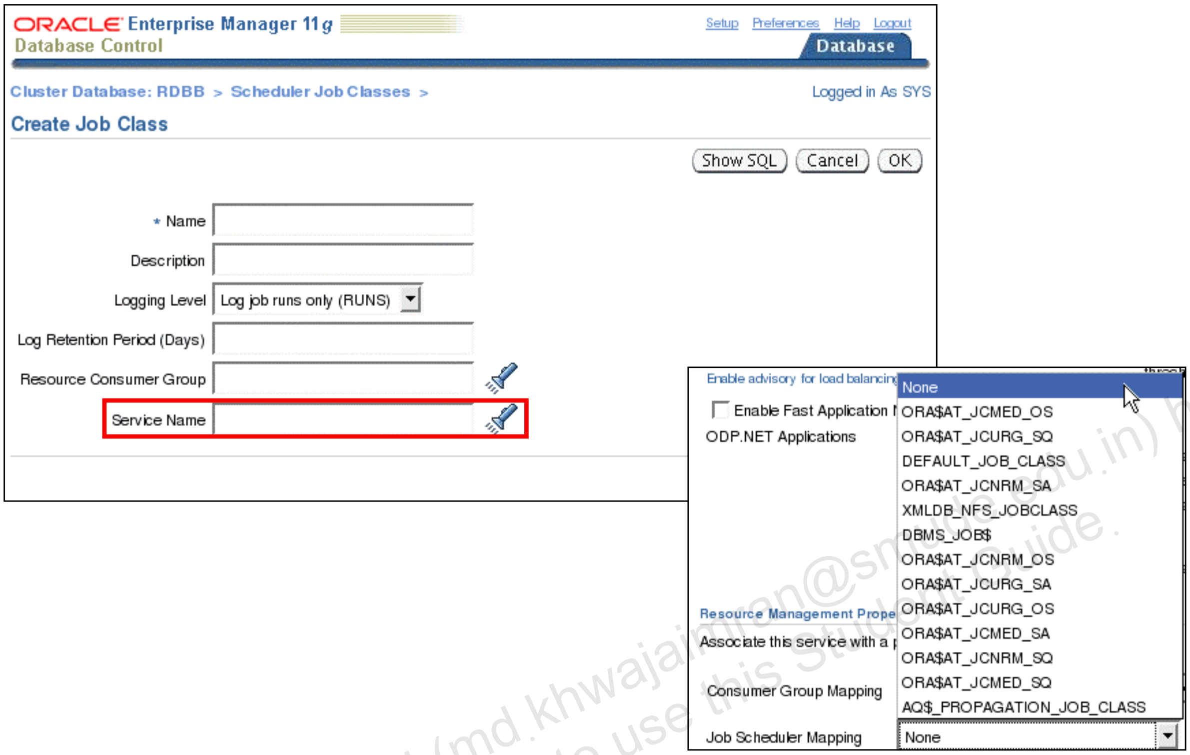
The Scheduler can use the services and the benefits they offer in a RAC environment. The service that a specific job class uses is defined when the job class is created. During execution, jobs are assigned to job classes and job classes run within services. Using services with job classes ensures that 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



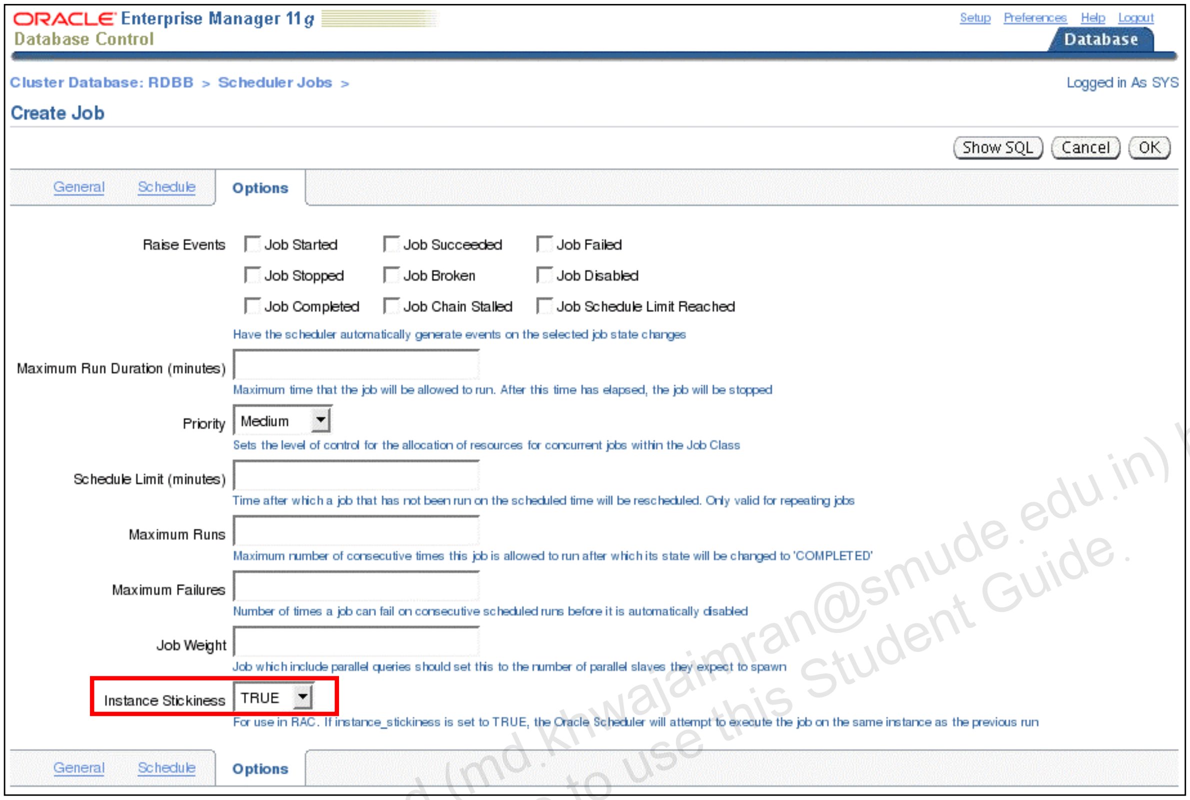
Services and the Scheduler with EM

To configure a job to run under a specific service, click the Job Classes link in the Database Scheduler section of the Server page. 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

Services and the Scheduler with EM (continued)

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, click 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.

Services and the Scheduler: Example

```
DBMS_SCHEDULER.CREATE_JOB_CLASS(
 JOB_CLASS_NAME          => 'HOT_BATCH_CLASS',
 RESOURCE_CONSUMER_GROUP => NULL              ,
 SERVICE                 => 'HOT_BATCH_SERV'       ,
 LOGGING_LEVEL => DBMS_SCHEDULER.LOGGING_RUNS,
 LOG_HISTORY   => 30, COMMENTS => 'P1 batch');
```

```
DBMS_SCHEDULER.CREATE_JOB(
 JOB_NAME => 'my_report_job',
 JOB_TYPE => 'stored_procedure',
 JOB_ACTION => 'my_name.my_proc();',
 NUMBER_OF_ARGUMENTS => 4, START_DATE => SYSDATE+1,
 REPEAT_INTERVAL => 5, END_DATE => SYSDATE+30,
 JOB_CLASS => 'HOT_BATCH_CLASS', ENABLED => TRUE,
 AUTO_DROP => false, COMMENTS => 'daily status');
```

ORACLE

Services and the Scheduler: Example

In this PL/SQL example, you define a batch queue, HOT_BATCH_CLASS, managed by the Scheduler. You associate the HOT_BATCH_SERV service to the HOT_BATCH_CLASS queue. It is assumed that you had already defined the HOT_BATCH_SERV service.

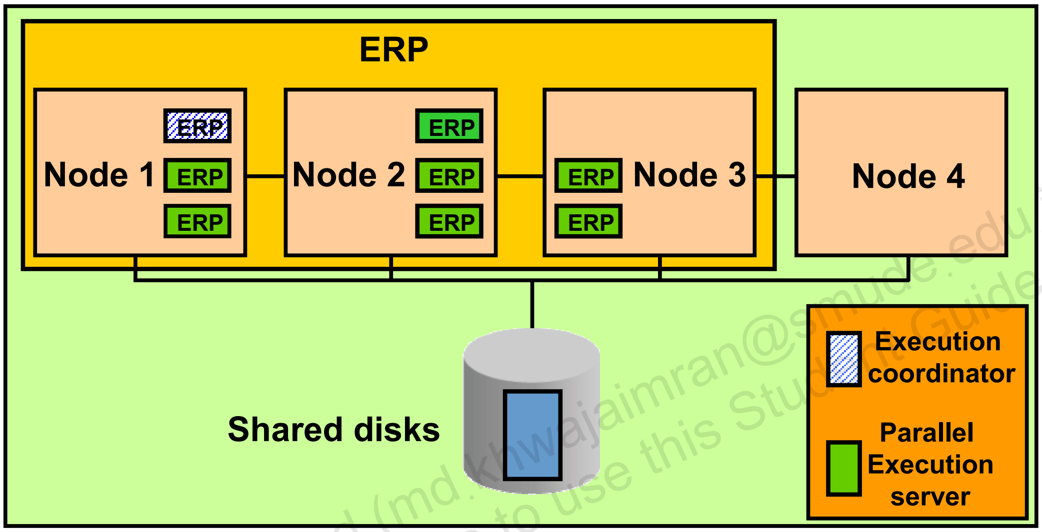
After the class is defined, you can define your job. In this example, the MY_REPORT_JOB job executes in the HOT_BATCH_CLASS job class at instances offering the HOT_BATCH_SERV service.

In this example, you do not assign a resource consumer group to the HOT_BATCH_CLASS job class. However, it is possible to assign a consumer group to a class. Regarding services, this allows you to combine Scheduler jobs and service prioritization by using the Database Resource Manager.

Note: For more information about the Scheduler, refer to the *Oracle Database Administrator's Guide* and *PL/SQL Packages and Types Reference*.

Use Services with Parallel Operations

- Slaves inherit the service from the coordinator.
- Slaves only execute on instances running the service.



Use Services with Parallel Operations

For parallel query and parallel DML operations, the parallel query slaves inherit the service from the query coordinator for the duration of the operation. ERP is the name of the service used by the example shown in the slide.

By default—and starting with Oracle RAC 11gR1—services restrict the set of instances that are used by a parallel query. Connecting via a service and then issuing a parallel query can use only those instances that are part of the service that was specified during the connection. This is implemented by automatically modifying the `INSTANCE_GROUPS` parameter to reflect `SERVICE_NAMES` and by using the service name used to connect to select the instance group to use for parallel operations (unless you specified `PARALLEL_INSTANCE_GROUP` to a different value).

To override this behavior, set a value for the `INSTANCE_GROUPS` and `PARALLEL_INSTANCE_GROUP` initialization parameters. In that case, a slave appears to belong under the service even on an instance that does not support the service, if that slave is being used by a query coordinator that was started on an instance that does support that service. At the end of the execution, the slaves revert to the default database service. Note, however, that `INSTANCE_GROUPS` is a deprecated initialization parameter and is retained only for backward compatibility purposes.

Use Services with Metric Thresholds

- You can define service-level thresholds:
  - ELAPSED_TIME_PER_CALL
  - CPU_TIME_PER_CALL
- Server-generated alerts are triggered on threshold violations.
- You can react on generated alerts:
  - Change priority.
  - Relocate services.
  - Add instances for services.

```
SELECT service_name, elapsedpercall, cpupercall
FROM   V$SERVICEMETRIC;
```

ORACLE

Use Services with Metric Thresholds

Service-level thresholds permit the comparison of actual service levels against the accepted minimum required level. This provides accountability with respect to delivery or failure to deliver an agreed service level.

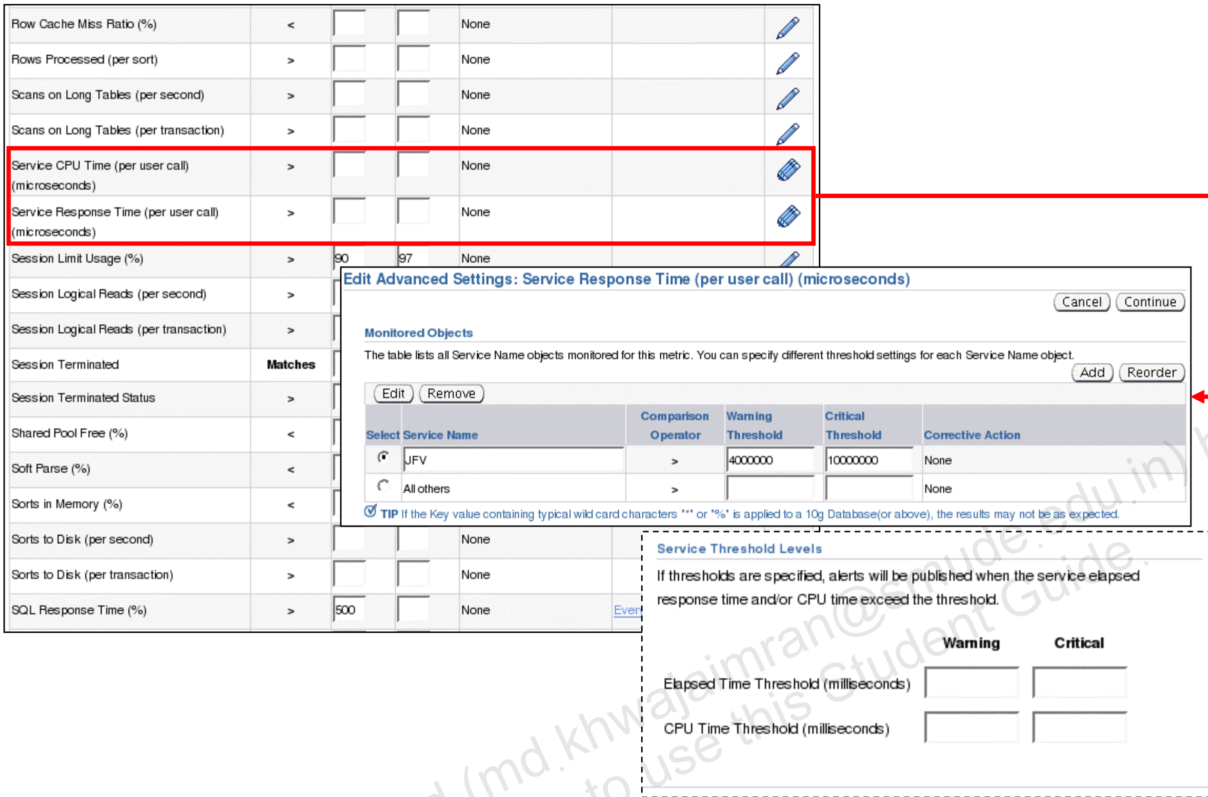
You can explicitly specify two metric thresholds for each service on a particular instance:

- **The response time for calls, `ELAPSED_TIME_PER_CALL`:** The response time goal indicates a desire for the elapsed time to be, at most, a certain value. The response time represents the wall clock time. It is a fundamental measure that reflects all delays and faults that block the call from running on behalf of the user.
- **CPU time for calls:** `CPU_TIME_PER_CALL`

The AWR monitors the service 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 relocating, expanding, shrinking, starting, or stopping a service. Using automated tasks, you can automate the reaction. This enables you to maintain service quality despite changes in demand.

Note: The `SELECT` statement shown in the slide gives you the accumulated instance statistics for elapsed time and for CPU-used metrics for each service for the most recent 60-second interval. For the last-hour history, look at `V$SERVICEMETRIC_HISTORY`.

Change Service Thresholds by Using EM



ORACLE

Change Service Thresholds by Using EM

You can set threshold values for your services from the Database Instance Metric and Policy Settings page. You can access this page from the Database Instance home page by clicking the Metric and Policy Settings link in the Related Links section.

Using the Metric and Policy Settings page, you can set the Service CPU Time (per user call) and Service Response Time (per user call) metrics for your services. If you modify the critical and warning values on this page, the thresholds apply to all services of the instance. If you want different thresholds for different services, click the corresponding icon on the last column of the table as shown in the slide. This takes you to the corresponding Edit Advanced Settings page. There, you can click Add to add rows to the Monitored Objects table. Each row represents a particular service in this case.

Note: You can directly set service thresholds from the Create Service page as shown at the bottom of the slide.

Services and Metric Thresholds: Example

```
exec 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 => 15,
  CONSECUTIVE_OCCURRENCES => 3,
  INSTANCE_NAME => 'I0n',
  OBJECT_TYPE => dbms_server_alert.object_type_service,
  OBJECT_NAME => 'ERP');
```

Thresholds must be set on each instance supporting the service.

Services and Metric Thresholds: Example

In this example, thresholds are added for the ERP service for the `ELAPSED_TIME_PER_CALL` metric. This metric measures the elapsed time for each user call for the corresponding service. The time must be expressed in microseconds.

A warning alert is raised by the server whenever the average elapsed time per call for the ERP service over a 15-minute period exceeds 0.5 seconds three consecutive times.

A critical alert is raised by the server whenever the average elapsed time per call for the ERP service over a 15-minute period exceeds 0.75 seconds three consecutive times.

Note: The thresholds must be created for each RAC instance that potentially supports the service.

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_NAMES`
  - `MODULE`
  - `ACTION`
  - Combination of `SERVICE_NAME, MODULE, ACTION`
- **This is useful for tuning systems that use shared sessions.**

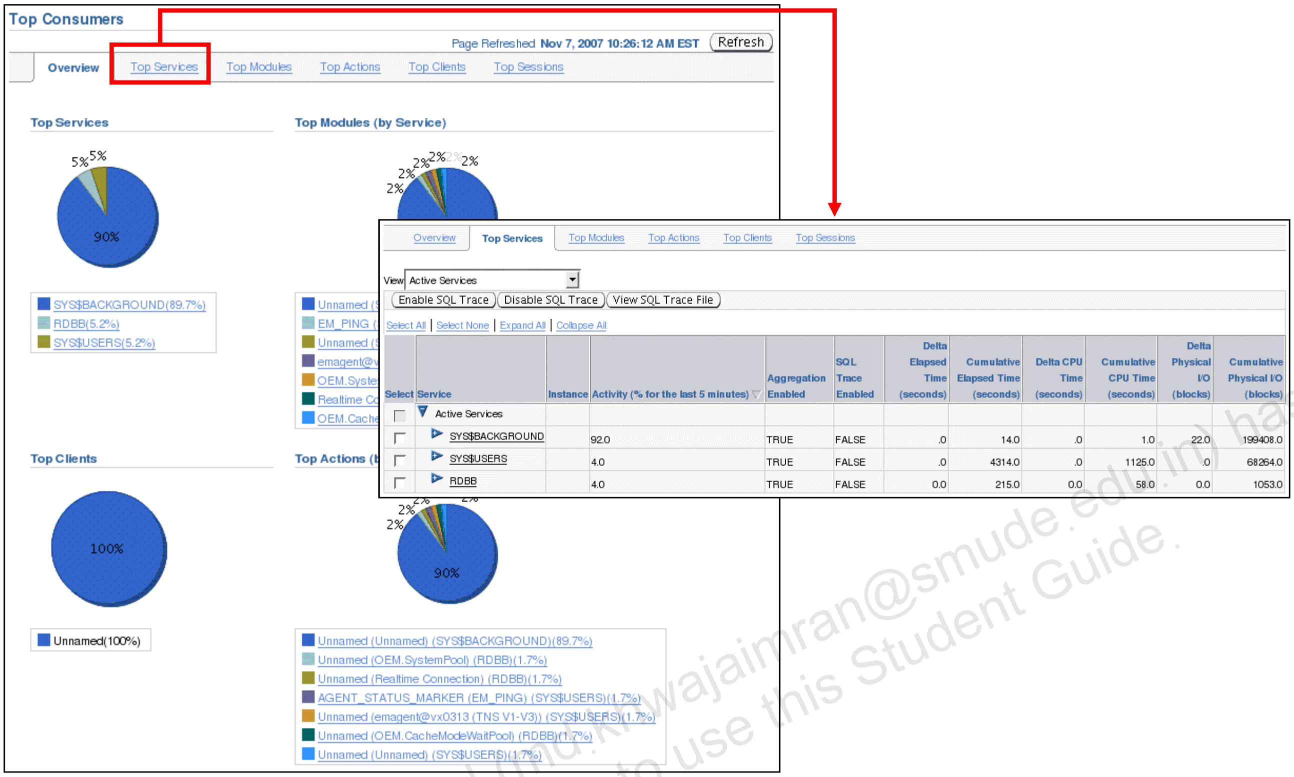
Service Aggregation and Tracing

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



Top Services Performance Page

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.

Service Aggregation Configuration

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 using the services 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. Furthermore, 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 Aggregation: Example

- Collect statistics on service and module:

```
exec DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(-
     'AP', 'PAYMENTS');
```

- Collect statistics on service, module, and action:

```
exec DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(-
     'AP', 'PAYMENTS', 'QUERY_DELINQUENT');
```

- Trace all sessions of an entire service:

```
exec DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE('AP');
```

- Trace on service, module, and action:

```
exec DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE(-
     'AP', 'PAYMENTS', 'QUERY_DELINQUENT');
```

ORACLE

Service Aggregation: Example

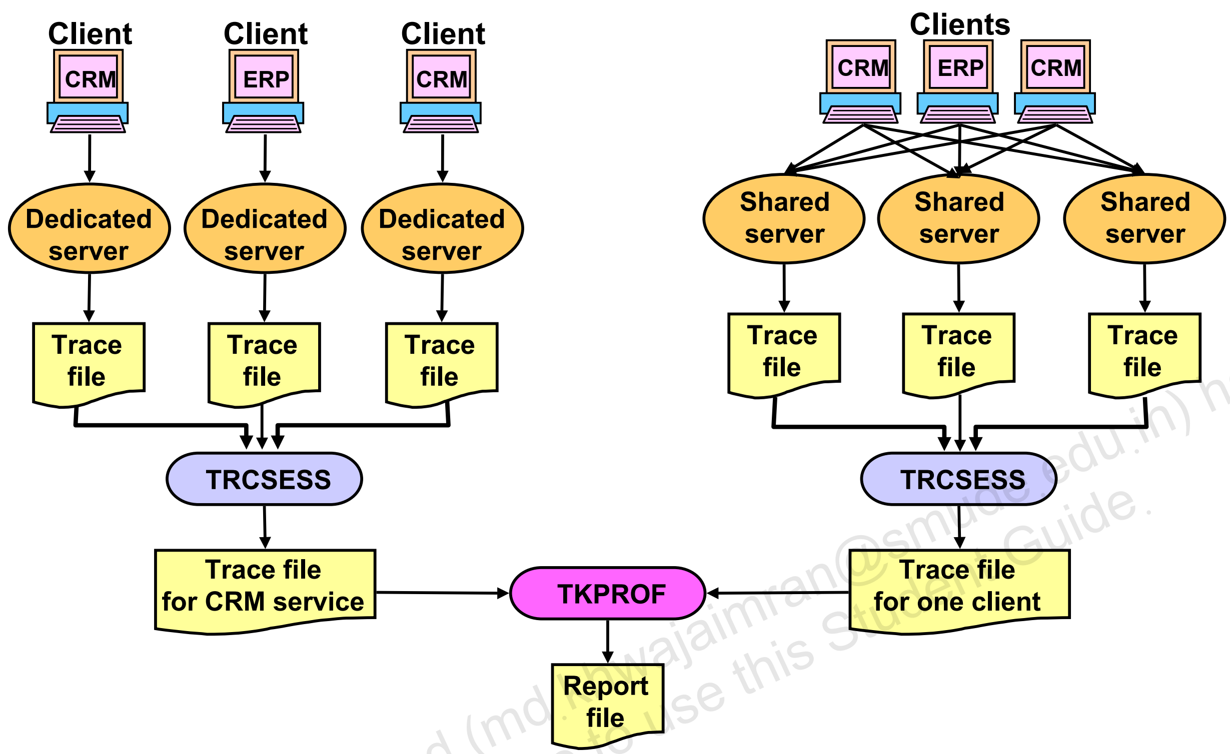
The first piece of sample code begins collecting statistics for the `PAYMENTS` module within the `AP` service. The second example collects statistics only for the `QUERY_DELINQUENT` program that runs in the `PAYMENTS` module under the `AP` service. This enables statistics collection on specific tasks that run in the database.

In the third code box, all sessions that log in under the `AP` service are traced. A trace file is created for each session that uses the service, regardless of the module and action. To be precise, you can trace only specific tasks within a service. This is illustrated in the last example, where all sessions of the `AP` service that execute the `QUERY_DELINQUENT` action within the `PAYMENTS` module are traced.

Tracing by service, module, and action enables you to focus your tuning efforts on specific SQL, rather than sifting through trace files with SQL from different programs. Only the SQL statements that define this task are recorded in the trace file. This complements collecting statistics by service, module, and action because relevant wait events for an action can be identified.

Note: For more information about the `DBMS_MONITOR` package, refer to the *PL/SQL Packages and Types Reference*.

`trcsess` Utility



`trcsess` Utility

The `trcsess` utility consolidates trace output from selected trace files on the basis of several criteria: session ID, client ID, service name, action name, and module name. After `trcsess` merges the trace information into a single output file, the output file can be processed by `tkprof`.

When using the `DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE` procedure, tracing information is present in multiple trace files and you must use the `trcsess` tool to collect it into a single file.

The `trcsess` utility is useful for consolidating the tracing of a particular session or service for performance or debugging purposes.

Tracing a specific session is usually not a problem in the dedicated server model because a single dedicated process serves a session during its lifetime. All the trace information for the session can be seen from the trace file belonging to the dedicated server serving it. However, tracing a service might become a complex task even in the dedicated server model.

Moreover, in a shared-server configuration, a user session is serviced by different processes from time to time. The trace pertaining to the user session is scattered across different trace files belonging to different processes. This makes it difficult to get a complete picture of the life cycle of a session.

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`
- **Twenty-eight statistics for services**

Service Performance Views

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.

There are more than 300 performance-related statistics that 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.

Generalized Trace Enabling

- For all sessions in the database:

```
EXEC dbms_monitor.DATABASE_TRACE_ENABLE(TRUE,TRUE);
```

```
EXEC dbms_monitor.DATABASE_TRACE_DISABLE();
```

- For a particular session:

```
EXEC dbms_monitor.SESSION_TRACE_ENABLE(session_id =>
27, serial_num => 60, waits => TRUE, binds =>
FALSE);
```

```
EXEC dbms_monitor.SESSION_TRACE_DISABLE(session_id
=> 27, serial_num => 60);
```

ORACLE

Generalized Trace Enabling

You can use tracing to debug performance problems. Trace-enabling procedures have been implemented as part of the DBMS_MONITOR package. These procedures enable tracing globally for a database.

You can use the DATABASE_TRACE_ENABLE procedure to enable instancewide session-level SQL tracing. The procedure has the following parameters:

- **WAITS:** Specifies whether wait information is to be traced
- **BINDS:** Specifies whether bind information is to be traced
- **INSTANCE_NAME:** Specifies the instance for which tracing is to be enabled. Omitting INSTANCE_NAME means that the session-level tracing is enabled for the whole database.

Use the DATABASE_TRACE_DISABLE procedure to disable SQL tracing for the whole database or a specific instance.

Similarly, you can use the SESSION_TRACE_ENABLE procedure to enable tracing for a given database session identifier (SID), on the local instance. The SID and SERIAL# information can be found from V$SESSION.

Use the SESSION_TRACE_DISABLE procedure to disable the trace for a given database session identifier (SID) and serial number.

Manage Services

- Use EM or `SRVCTL` to manage services:
  - Start: Allow connections
  - Stop: Prevent connections
  - Enable: Allow automatic restart and redistribution
  - Disable: Prevent starting and automatic restart
  - Relocate: Temporarily change instances on which services run
  - Modify: Modify preferred and available instances
  - Get status information
  - Add or remove
- Use the DBCA:
  - Add or remove
  - Modify services

Manage Services

Depending on the type of management tasks that you want to perform, you can use Enterprise Manager, the DBCA, or `SRVCTL`.

The following is a description of the management tasks related to services in a RAC environment:

- Disabling a service is used to disable a specified service on all or specified instances. The disable state is used when a service is down for maintenance to prevent inappropriate automatic Oracle Clusterware restarts. Disabling an entire service affects all the instances by disabling the service at each instance.
- Enabling a service is used to enable a service to run under Oracle Clusterware for automatic restart and redistribution. You can enable a service even if that service is stopped. `Enable` is the default value when a service is created. If the service is already enabled, then the command is ignored. Enabled services can be started, and disabled services cannot be started. Enabling an entire service affects the enabling of the service over all the instances by enabling the service at each instance.
- Starting a service is used to start a service or multiple services on the specified instance. Only enabled services can be started. The command fails if you attempt to start a service on an instance and if the number of instances that are currently running the service already reaches its cardinality.

Manage Services (continued)

- Stopping is used to stop one or more services globally across the cluster database, or on the specified instance. Only Oracle Clusterware services that are starting or have started are stopped. You should disable a service that you intend to keep stopped after you stop that service because if the service is stopped and is not disabled, then it can be restarted automatically as a result of another planned operation. This operation can force sessions to be disconnected transactionally.
- Removing a service is used to remove its configuration from the cluster database on all or specified instances. You must first stop the corresponding service before you can remove it. You can remove a service from specific instances only.
- Relocating a service is used to relocate a service from a source instance to a target instance. The target instance must be on the preferred or available list for the service. This operation can force sessions to be disconnected transactionally. The relocated service is temporary until you permanently modify the configuration.
- Modifying a service configuration is used to permanently modify a service configuration. The change takes effect when the service is restarted later. This allows you to move a service from one instance to another. Additionally, this command changes the instances that are to be the preferred and available instances for a service.
- Displaying the current state of a named service

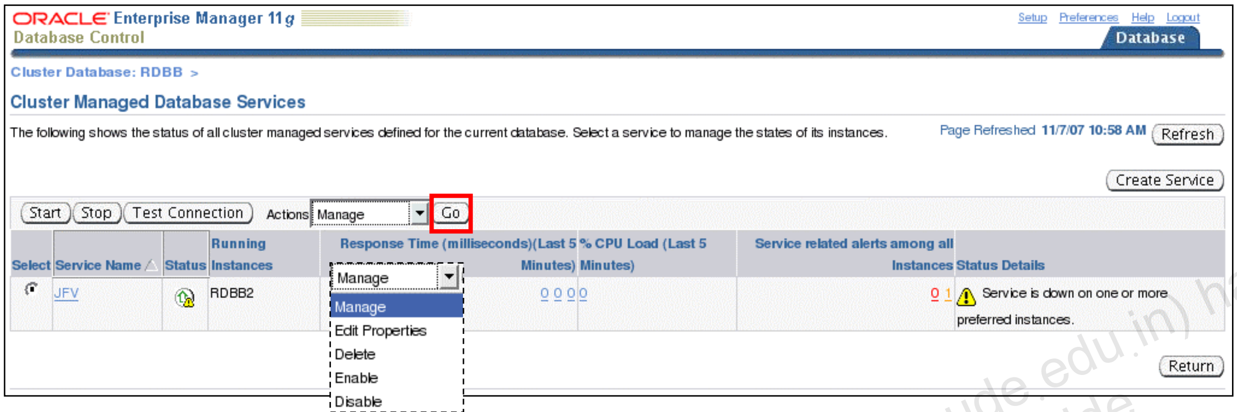
When you use the DBCA to add services, the DBCA also configures the net service entries for these services and starts them. When you use the DBCA to remove services, it stops the service, removes the Oracle Clusterware resource for the service, and removes the net service entries.

When you create a service with `SRVCTL`, you must start it with a separate `SRVCTL` command.

`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.

Note: The `srvctl stop database` command implicitly does a `srvctl stop services` (because services are dependent on database). However, a subsequent `srvctl start database` requires an explicit `srvctl start service`.

Manage Services with Enterprise Manager



ORACLE

Manage Services with Enterprise Manager

You can use Enterprise Manager to manage services within a GUI framework. The screenshot shown in the slide is the main page for administering services within RAC. It shows you some basic status information about a defined service.

To access this page, click the Cluster Managed Database Services link on the Cluster Database Availability page.

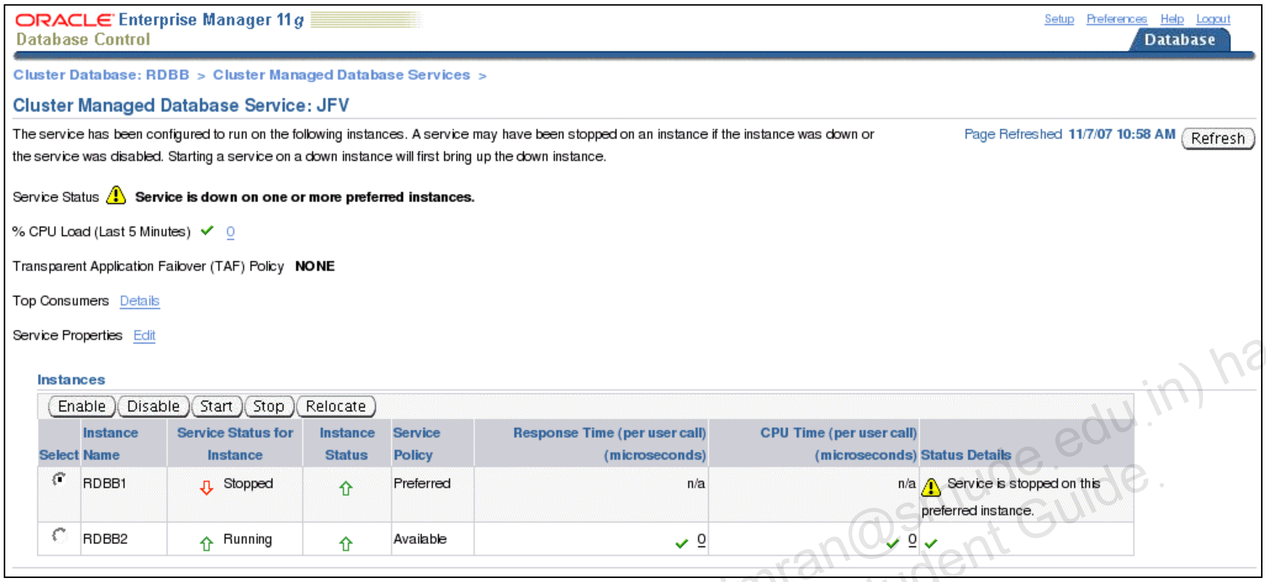
You can perform simple service management such as enabling, disabling, starting, stopping, and relocating services. All possible operations are shown in the slide.

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, choose 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.

Manage Services with Enterprise Manager



Manage Services with Enterprise Manager (continued)

To access the Cluster Managed Database Service page for an individual service, you must choose 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.

Note: This page also shows you the TAF policy set for this particular service. You can directly edit the service's properties, or link to the Top Consumers page.

Manage Services: Example

- Start a named service on all preferred instances:

```
$ srvctl start service -d PROD -s AP
```

- Stop a service on selected instances:

```
$ srvctl stop service -d PROD -s AP -i RAC03,RAC04
```

- Disable a service at a named instance:

```
$ srvctl disable service -d PROD -s AP -i RAC04
```

- Set an available instance as a preferred instance:

```
$ srvctl modify service -d PROD -s AP -i RAC05 -r
```

ORACLE

Manage Services: Example

The slide demonstrates some management tasks with services by using SRVCTL.

Assume that an AP service has been created with four preferred instances: RAC01, RAC02, RAC03, and RAC04. An available instance, RAC05, has also been defined for AP.

In the first example, the AP service is started on all preferred instances. If any of the preferred or available instances that support AP are not running but are enabled, then they are started.

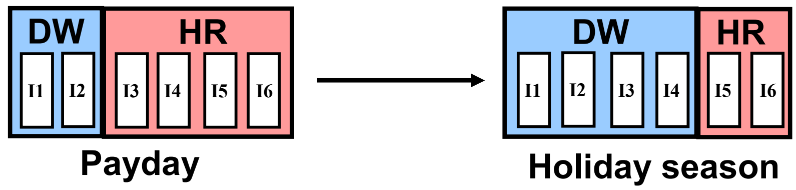
The stop command stops the AP service on instances RAC03 and RAC04. The instances themselves are not shut down, but remain running possibly supporting other services. The AP service continues to run on RAC01 and RAC02. The intention might have been to perform maintenance on RAC04, and so the AP service 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 RAC04. Thus, Oracle Clusterware will not run AP on RAC04 until the service is enabled later.

The last command in the slide changes RAC05 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 RAC04 was previously disabled.

Do not perform other service operations while the online service modification is in progress.

Note: For more information, refer to the *Oracle Real Application Clusters Administrator's Guide*.

Manage Services: Scenario




```
srvctl modify service -d PROD -s DW -n -i I1,I2,I3,I4 -a I5,I6
```

```
srvctl modify service -d PROD -s HR -n -i I5,I6 -a I1,I2,I3,I4
```

```
srvctl stop service -d PROD -s DW,HR -f
```

```
srvctl start service -d PROD -s DW,HR
```


Manage Services: Scenario

The slide presents a possible change in a service configuration with a minimum down time for your workload.

It is assumed that you have a six-node cluster where you run two services: `DW` and `HR`. Originally, `DW` is configured to have `I1` and `I2` instances as its preferred instances, and `I3`, `I4`, `I5`, and `I6` as its available instances. Similarly, `HR` is originally configured to have `I3`, `I4`, `I5`, and `I6` as its preferred instances, and `I1` and `I2` as its available instances. This initial configuration corresponds to the Payday period (as shown in the left part of the graphic).

During the Holiday season, you need to change your services configuration so that `DW` is now run on the first four instances, and `HR` on the remaining two.

From the top going down, the slide shows you the commands you need to execute to switch your services configuration.

Note that the `–n` option of the `srvctl modify service` commands is used to remove the initial configuration of your services. The changes take effect when the services are next restarted. You can also use the `–f` option for these commands so that the next `stop` command disconnects corresponding sessions. Here, you prefer using the `–f` option with the `srvctl stop service` commands, which stop the services globally on your cluster.

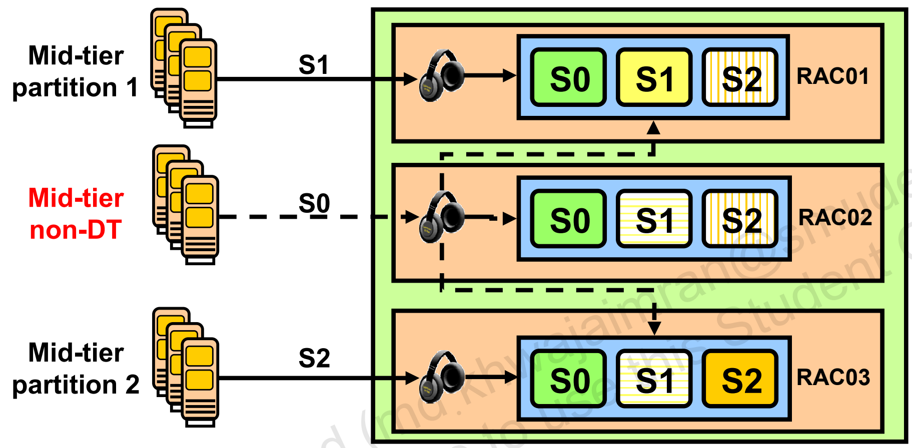
You then use the `srvctl start service` commands to use the newly created service configuration.

Using Distributed Transactions with RAC

- **Scope of application: XA or MS DTC**
- **Fully supported by default on RAC (GTX*n* background)**
- **Or all transaction branches occur on same instance:**

```
dbms_service.modify_service(service_name=>'S1', DTP=>TRUE)
```

```
dbms_service.modify_service(service_name=>'S2', DTP=>TRUE)
```



Using Distributed Transactions with RAC

When you use RAC with distributed transactions (compliant with XA standard or coordinated by Microsoft Distributed Transaction Coordinator), it is possible for two application components in the same transaction to connect to different nodes of a RAC cluster. This situation can occur on systems with automatic load balancing where the application cannot control which database nodes a distributed transaction branch gets processed.

Starting with Oracle RAC 11*g*R1, these types of distributed transactions are automatically controlled by the system through the use of new background processes called `GTX0…GTXj`. `GLOBAL_TXN_PROCESSES` specifies the initial number of `GTX`*n* background processes on an instance, and its default value is `1`. Letting the database handle distributed transactions automatically is useful for systems that process global transactions heavily.

To provide improved application performance with distributed transaction processing in Oracle RAC, you may want to take advantage of the specialized service referred to as a *DTP service*. Using DTP services, you can direct all branches of a distributed transaction to a single instance in the cluster. For load balancing 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. The graphic in the slide presents a possible solution. Assume that you have three RAC nodes—RAC01, RAC02, and RAC03—with each node capable of servicing any nondistributed transaction coming from a middle tier.

Using Distributed Transactions with RAC (continued)

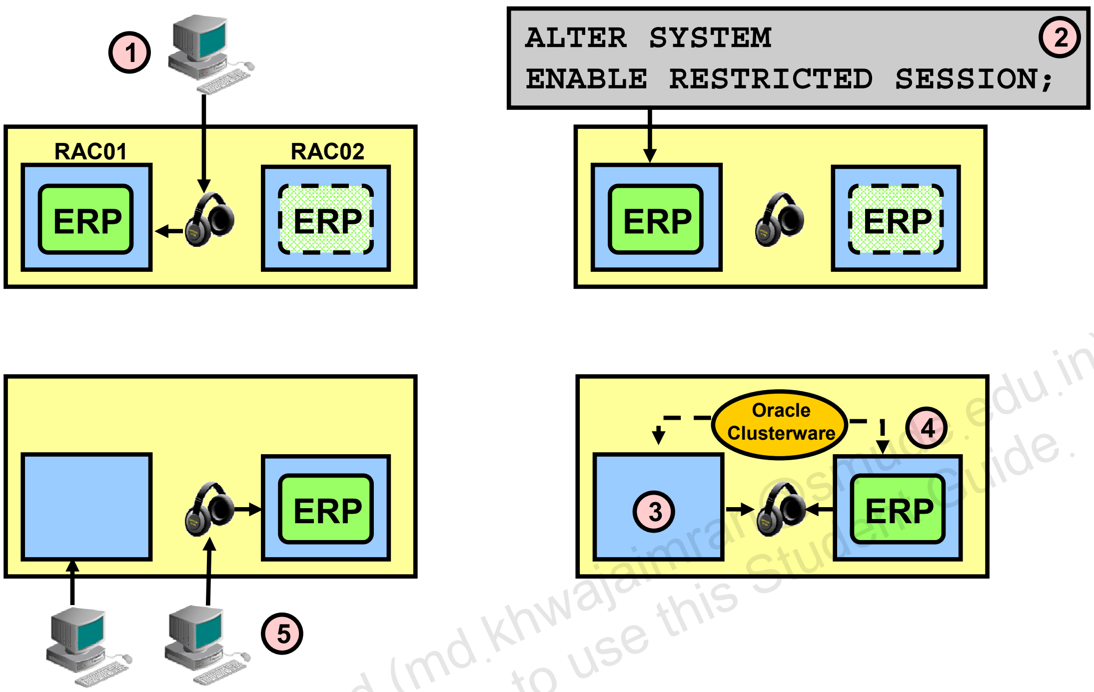
For distributed transactions from other middle tiers, they are partitioned statically via Oracle Net Services to one of these three nodes. Thus, each node publishes itself as an S0 service for nondistributed transactions. In addition, RAC01 and RAC03 publish themselves as a DTP service: S1 and S2, respectively.

As shown in the slide, a DTP service is one that has its `DTP` flag set to `TRUE`. In addition, you should always define a DTP service as a singleton service (that is, with only one preferred instance).

Each mid-tier client has Oracle Net Service Names configuration and accesses the Oracle database through Oracle Net. Each Oracle Net Service that is being used for distributed transactions is configured with one DTP service.

The Oracle server ensures the cardinality of a DTP service to be `1` across the RAC cluster, prohibiting more than one instance of the same DTP service from running in a RAC database. Each distributed transaction is processed by one of the DTP services via Oracle Net, so that all tightly coupled branches of the distributed transaction are routed to the same node of the RAC database. Different distributed transactions can be load balanced to different RAC nodes via different DTP services. If one of these database nodes fails, Oracle Clusterware and RAC automatically detect the failure and do the transaction recovery before starting the corresponding DTP service on one of the available RAC nodes. When the node comes back, the same DTP service may be relocated back automatically depending on the workload.

Restricted Session and Services



Restricted Session and Services

Whenever you put one instance of the cluster in restricted mode, Oracle Clusterware stops the services running on the restricted instance, and it starts them on available instances if they exist. That way, the listeners are dynamically informed of the changes, and they no longer attempt to route requests to the restricted instance, regardless of its current load. In effect, the listeners exempt the restricted instance from their connection load-balancing algorithm.

This feature comes with two important considerations:

- First, even users with `RESTRICTED SESSION` privilege are not able to connect remotely through the listeners to an instance that is in the restricted mode. They need to connect locally to the node supporting the instance and use the bequeath protocol.
- Second, this new feature works only when the restricted instance dynamically registers with the listeners. That is, if you configure the `listener.ora` file with `SID_LIST` entries, and you do not use dynamic registration, the listener cannot block connection attempts to a restricted instance. In this case, and because the unrestricted instances of the cluster are still accessible, the restricted instance will eventually become least loaded, and the listener will start routing connection requests to that instance. Unable to accept the connection request because of its restricted status, the instance will deny the connection and return an error. This situation has the potential for blocking access to an entire service.

Note: The listener uses dynamic service registration information before static configurations.

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

Practice 7: Overview

This practice covers the following topics:

- **Defining services by using DBCA**
- **Managing services by using Enterprise Manager**
- **Using server-generated alerts in combination with services**

8

High Availability of Connections

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)

For more information, see the following page:
http://www.oracle.com/technology/products/database/clustering/pdf/awmrac11g.pdf

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
- Run-time connection load balancing is rendered possible by using connection pools:
  - Work requests automatically balanced across the pool of connections
  - Native feature of the JDBC implicit connection cache and ODP.NET connection pool

ORACLE

Types of Workload Distribution

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:

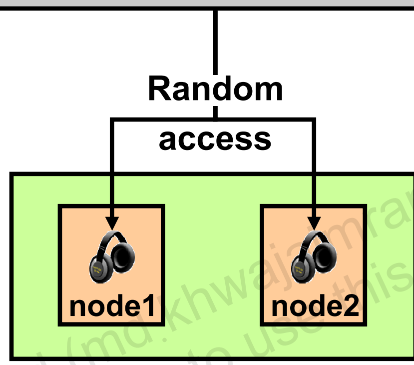
- Client-side connect-time load balancing
- Client-side connect-time failover
- Server-side connect-time load balancing

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 run-time 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 JDBC implicit connection cache feature as well as Oracle Data Provider for .NET (ODP.NET) connection pool.

Client-Side Connect-Time Load Balancing

```
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)))
```



Client-Side Connect-Time Load Balancing

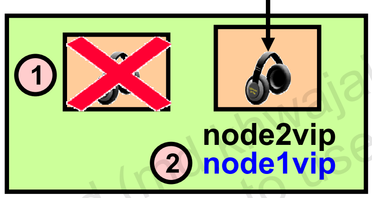
The client-side connect-time load balancing feature enables clients to randomize connection requests among a list of available listeners. Oracle Net progresses through the list of protocol addresses in a random sequence, balancing the load on the various listeners. Without this feature, Oracle Net always takes the first protocol address to attempt a connection.

You enable this feature by setting the `LOAD_BALANCE=ON` clause in the corresponding client-side TNS entry.

Note: For a small number of connections, the random sequence is not always even.

Client-Side Connect-Time Failover

```
ERP =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (LOAD_BALANCE=ON)
      (FAILOVER=ON)                             3
      (ADDRESS=(PROTOCOL=TCP)(HOST=node1vip)(PORT=1521))
      (ADDRESS=(PROTOCOL=TCP)(HOST=node2vip)(PORT=1521))
    )                                           4
    (CONNECT_DATA=(SERVICE_NAME=ERP)))
```



Client-Side Connect-Time Failover

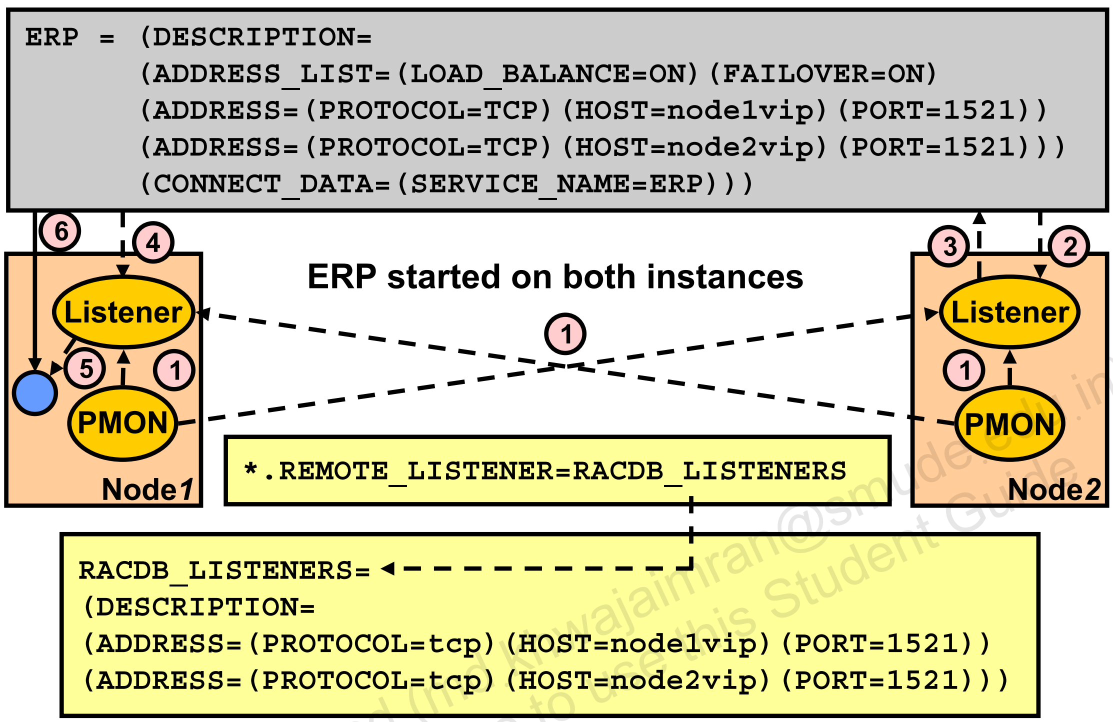
This feature enables clients to connect to another listener if the initial connection to the first listener fails. The number of listener protocol addresses in the connect descriptor determines how many listeners are tried. Without client-side connect-time failover, Oracle Net attempts a connection with only one listener. As shown by the example in the slide, client-side connect-time failover is enabled by setting the `FAILOVER=ON` clause in the corresponding client-side TNS entry.

In the example, you expect the client to randomly attempt connections to either `NODE1VIP` or `NODE2VIP`, because `LOAD_BALANCE` is set to `ON`. In the case where one of the nodes is down, the client cannot know this. If a connection attempt is made to a down node, the client needs to wait until it receives the notification that the node is not accessible, before an alternate address in the `ADDRESS_LIST` is tried.

Therefore, it is highly recommended to use virtual host names in the `ADDRESS_LIST` of your connect descriptors. If a failure of a node occurs (1), the virtual IP address assigned to that node is failed over and brought online on another node in the cluster (2). Thus, all client connection attempts are still able to get a response from the IP address, without the need to wait for the operating system TCP/IP timeout (3). Therefore, clients get an immediate acknowledgement from the IP address, and are notified that the service on that node is not available. The next address in the `ADDRESS_LIST` can then be tried immediately with no delay (4).

Note: If you use connect-time failover, do not set `GLOBAL_DBNAME` in your `listener.ora` file.

Server-Side Connect-Time Load Balancing



Server-Side Connect-Time Load Balancing

The slide shows you how listeners distribute service connection requests across a RAC cluster. Here, the client application connects to the `ERP` service. On the server side, the database is using the dynamic service registration feature. This allows the PMON process of each instance in the cluster to register service performance information with each listener in the cluster (1). Each listener is then aware of which instance has a particular service started, as well as how that service is performing on each instance.

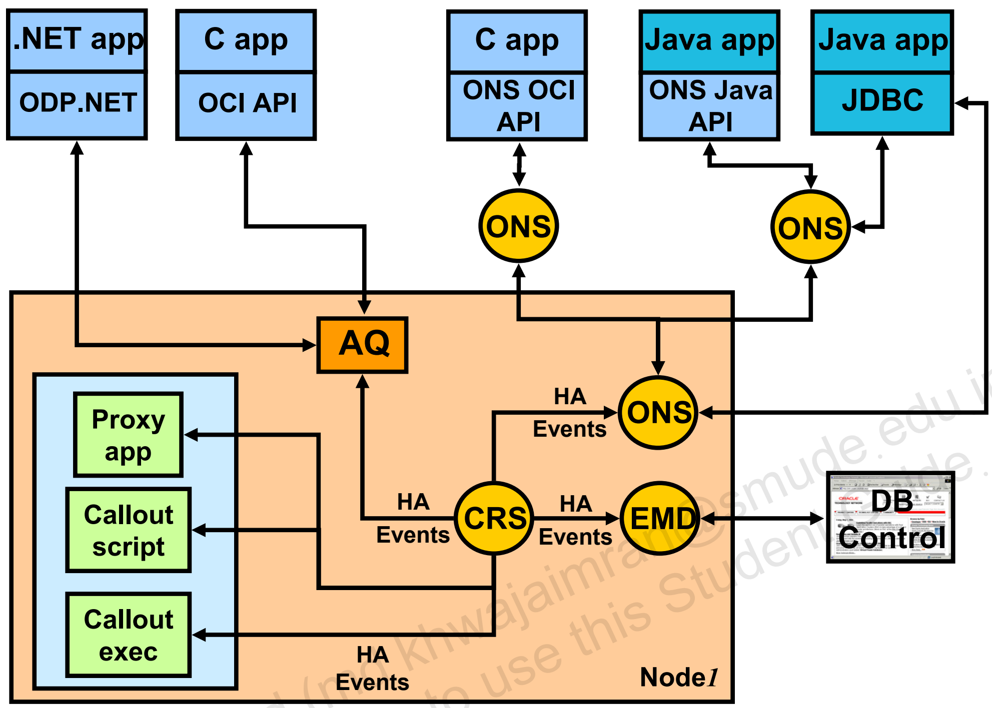
You configure this feature by setting the `REMOTE_LISTENER` initialization parameter of each instance to a TNS name that describes the list of all available listeners. The slide shows the shared entry in the SPFILE as well as its corresponding server-side TNS entry.

Depending on the load information, as computed by the Load Balancing Advisory, and sent by each PMON process, a listener redirects the incoming connection request (2) to the listener of the node where the corresponding service is performing the best (3).

In the example, the listener on `NODE2` is tried first. Based on workload information dynamically updated by PMON processes, the listener determines that the best instance is the one residing on `NODE1`. The listener redirects the connection request to the listener on `NODE1` (4). That listener then starts a dedicated server process (5), and the connection is made to that process (6).

Note: For more information, refer to the *Net Services Administrator's Guide*.

Fast Application Notification: Overview



Fast Application Notification: Overview

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. With FAN, the continuous service built into Oracle Real Application Clusters 11*g* is extended to applications and mid-tier servers. When the state of a database service changes, (for example, up, down, or not restarting), the new status is posted to interested subscribers through FAN events. Applications use these events to achieve very fast detection of failures, and rebalancing of connection pools following failures and recovery.

The easiest way to receive all the benefits of FAN, with no effort, is to use a client that is integrated with FAN:

- JDBC Implicit Connection Cache
- User extensible callouts
- Connection Manager (CMAN)
- Listeners
- Oracle Notification Service (ONS) API
- OCI Connection Pool or Session Pool
- Transparent Application Failover (TAF)
- ODP.NET Connection Pool

Note: Not all the above applications can receive all types of FAN events.

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
  - Enterprise Manager
  - Data Guard Broker

ORACLE

Fast Application Notification: Benefits

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 e-mail/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 Data Guard Broker, Oracle JDBC implicit connection cache, ODP.NET, TAF, and Enterprise Manager. For example, Oracle Database 11*g* 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.

Note: For more information about FAN and Data Guard integration, refer to the lesson titled “Design for High Availability” in this course.

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
`SERVICE_METRICS`	Load Balancing Advisory

ORACLE

FAN-Supported Event Types

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 `SERVICE_METRICS` are discussed later in this lesson.

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.
`restart_failed`	Managed resource fails to start locally after a discrete number of retries.
`Unknown`	Status is unrecognized.

FAN Event Status

This table describes the event status for each of the managed cluster resources seen previously.

FAN Event Reasons

Event Reason	Description
`user`	User-initiated commands, such as `srvctl` and `sqlplus`
`failure`	Managed resource polling checks detecting a failure
`dependency`	Dependency of another managed resource that triggered a failure condition
`unknown`	Unknown or internal application state when event is triggered
`autostart`	Initial cluster boot: Managed resource has profile attribute `AUTO_START=1`, and was offline before the last Oracle Clusterware shutdown.
`Boot`	Initial cluster boot: Managed resource was running before the last Oracle Clusterware shutdown.

ORACLE

FAN Event Reasons

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 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.oracle.com
database=RACDB status=up reason=user card=4
timestamp=16-Mar-2004 19:08:15
```

```
NODE VERSION=1.0 host=strac-1
status=nodedown timestamp=16-Mar-2004 17:35:53
```

ORACLE

FAN Event Format

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, which is currently 1.0
- The name of the primary or shadow application service. This name is excluded from `NODE` events.
- The name of the RAC database, which is also excluded from `NODE` events
- The name of the RAC instance, which is excluded from `SERVICE`, `DATABASE`, and `NODE` events
- The name of the cluster host machine, which is excluded from `SERVICE` and `DATABASE` events
- The service cardinality, which is excluded from all events except for `SERVICE status=up` events
- The server-side date and time when the event is detected

The general FAN event format is described in the slide along with possible FAN event examples. Note the differences in event payload for each FAN event type.

Load Balancing Advisory: FAN Event

Parameter	Description
Version	Version of the event payload
Event type	SERVICE_METRICS
Service	Matches DBA_SERVICES
Database unique name	Unique DB name supporting the service
Time stamp	Date and time stamp (local time zone)
Repeated	
Instance	Instance name supporting the service
Percent	Percentage of work to send to this database and instance
Flag	GOOD, VIOLATING, NO DATA, UNKNOWN

Load Balancing Advisory: FAN Event

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.

Server-Side Callouts Implementation

- The callout directory:
  - <*CRS Home*>`/racg/usrco`
  - Can store more than one callout
  - Grants execution on callout directory and callouts only to the Oracle Clusterware user
- Callouts execution order is nondeterministic.
- Writing callouts involves:
  1. Parsing callout arguments: The event payload
  2. Filtering incoming FAN events
  3. Executing event-handling programs

Server-Side Callouts Implementation

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 `$ORA_CRS_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 that 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
```


Server-Side Callout Parse: Example

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` string. 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
```


Server-Side Callout Filter: Example

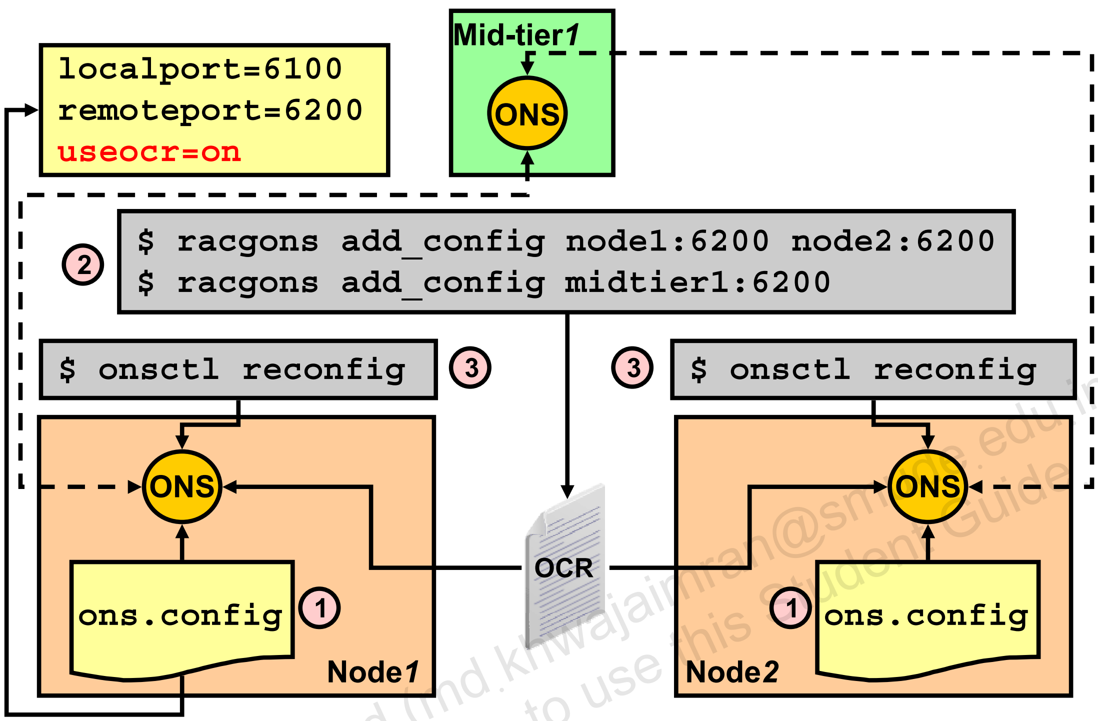
The example in the slide shows you a way to filter FAN events from a callout script. This example is based on the example in the previous slide.

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.

Configuring the Server-Side ONS



Configuring the Server-Side ONS

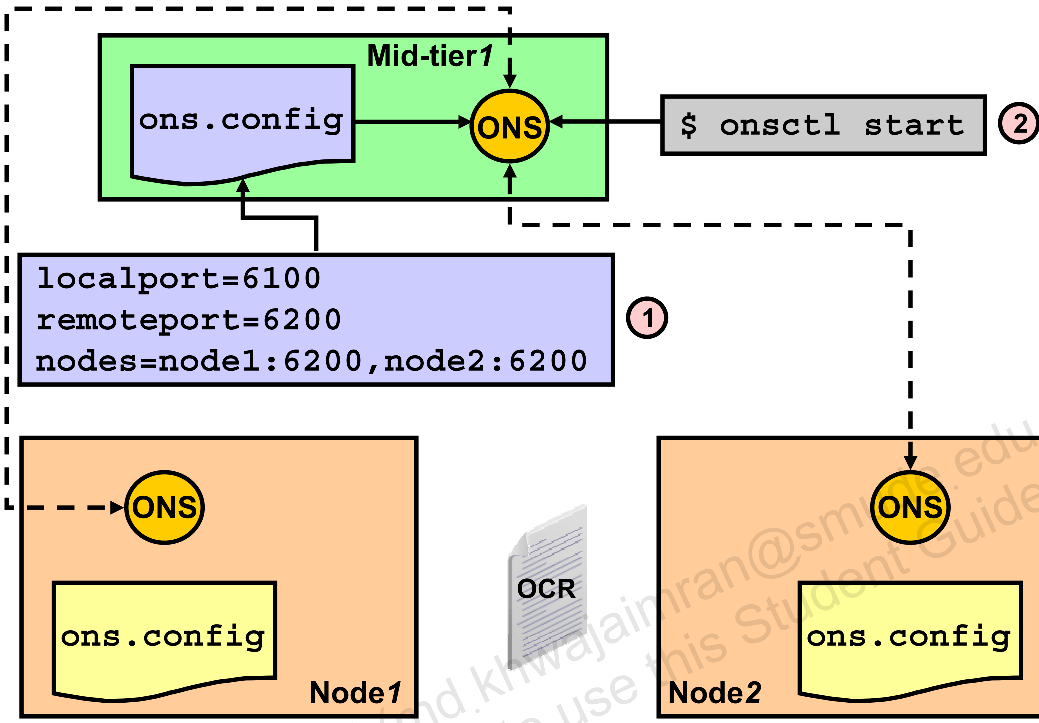
The ONS configuration is controlled by the *<CRS HOME>*`/opmn/conf/ons.config` configuration file. This file is automatically created during installation. There are three important parameters that should always be configured for each ONS:

- 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 should include all RAC ONS daemons, and all mid-tier ONS daemons. Node values are given as either host names or IP addresses followed by its `remoteport`. Instead, you can store this data in Oracle Cluster Registry (OCR) using the `racgons add_config` command and having the `useocr` parameter set to `on` in the `ons.config` file. By storing nodes information in OCR, you do not need to edit a file on every node to change the configuration. Instead, you need to run only a single command on one of the cluster nodes.

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. However, if you want to use OCR, you should edit the `ons.config` file on each node, and then add the configuration to OCR before reloading it on each cluster node. This is illustrated in the slide.

Note: You should run `racgons` whenever you add or remove a node that runs an ONS daemon. To remove a node from OCR, you can use the `racgons remove_config` command.

Optionally Configure the Client-Side ONS



Optionally Configure the Client-Side ONS

Oracle Database 10*g* Release 1 FAN uses Oracle Notification Service (ONS) on the mid-tier to receive FAN events when you are using the Java Database Connectivity (JDBC) Implicit Connection Cache (ICC). To use ONS on the mid-tier, you need to install ONS on each host where you have client applications that need to be integrated with FAN. Most of the time, these hosts play the role of a mid-tier application server. Therefore, 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.

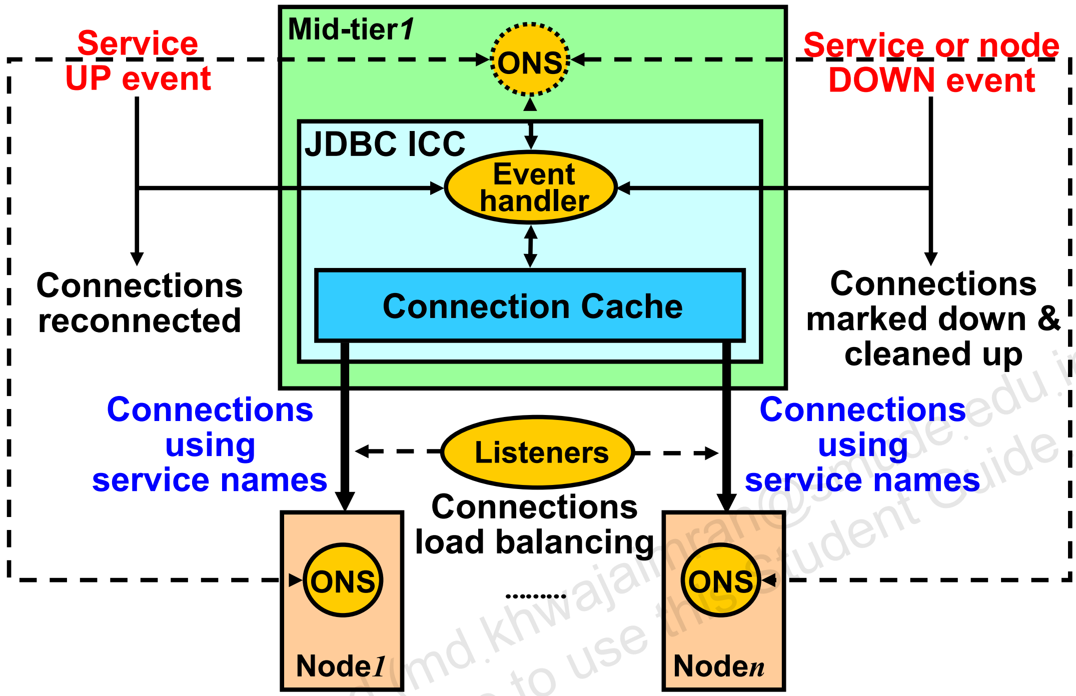
After configuring ONS, you start the ONS daemon with the `onsctl start` command. It is your responsibility to make sure that an ONS daemon is running at all times. You can check that the ONS daemon is active by executing the `onsctl ping` command.

Note: With Oracle Database 10*g* Release 2 and later, there is no requirement to use ONS daemons on the mid-tier when using the JDBC Implicit Connection Cache. 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. For example,

`setONSConfiguration("nodes=host1:port1,host2:port2 ");`

The `ons.jar` file must be on the client's `CLASSPATH`. There are no daemons to start or manage.

JDBC Fast Connection Failover: Overview



JDBC Fast Connection Failover: Overview

Oracle Application Server 10*g* integrates JDBC ICC with the ONS API by having application developers enable Fast Connection Failover (FCF). FCF works in conjunction with the JDBC ICC to quickly and automatically recover 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 JDBC ICC 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 the connections across the preferred instances of the service without waiting for application connection requests or retries.

For more information, refer to the *Oracle Database JDBC Developer's Guide and Reference*.

Note: Similarly, ODP.NET also allows you to use FCF using AQ for FAN notifications.

Using Oracle Streams Advanced Queuing for FAN

- Use AQ to publish FAN to ODP.NET and OCI.
- Turn on FAN notification to alert queue.

```
exec DBMS_SERVICE.MODIFY_SERVICE (
  service_name => 'SELF-SERVICE', aq_ha_notification => TRUE);
```

- View published FAN events:

```
SQL> select object_name,reason
  2  from dba_outstanding_alerts;

OBJECT_NAME REASON
----------- ----------------------------------------
xwkE        Database xwkE (domain ) up as of time
            2005-12-30 11:57:29.000000000 -05:00;
            reason code: user
JFSERV      Composite service xwkE up as of time
            2006-01-02 05:27:46.000000000 -05:00;
            reason code: user
```


Using Oracle Streams Advanced Queuing for FAN

Real Application Clusters publish FAN events to a system alert queue in the database using Oracle Streams Advanced Queuing (AQ). ODP.NET and OCI client integration uses this method to subscribe to FAN events.

To have FAN events for a service posted to that alert queue, the notification must be turned on for the service using either the DBMS_SERVICE PL/SQL package as shown in the slide, or by using the Enterprise Manager interface.

To view FAN events that are published, you can use the DBA_OUTSTANDING_ALERTS or DBA_ALERT_HISTORY views. An example using DBA_OUTSTANDING_ALERTS is shown in the slide.

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.

ORACLE

JDBC/ODP.NET FCF Benefits

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, changing demand
  - Stops sending work to slow, hung, or failed nodes

Load Balancing Advisory

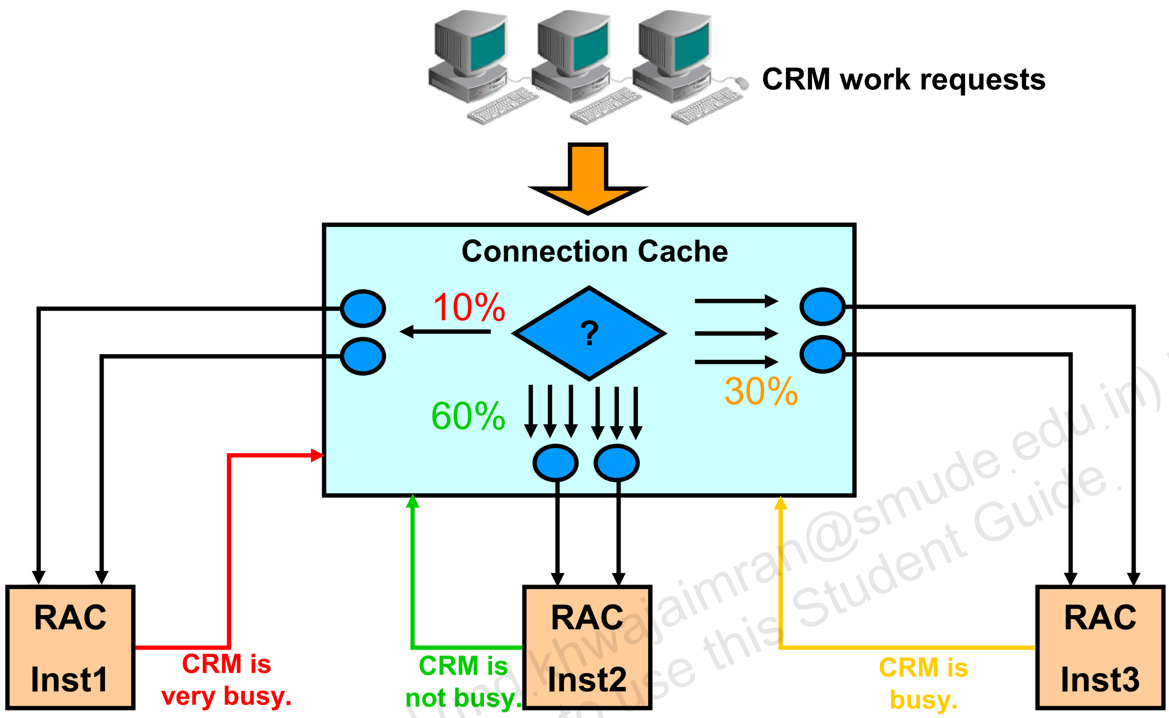
Well-written applications use persistent connections that span the instances of RAC offering a service. Connections are created infrequently and exist for a long duration. Work comes into the system with high frequency, borrows these connections, and exists for a relatively short duration.

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 Oracle's key clients, such as Connection Load Balancing, JDBC Implicit Connection Cache, OCI Session Pool, Oracle Data Provider (ODP) Connection Pool for .NET, and is open for third-party subscription via ONS.

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.

JDBC/ODP.NET Runtime Connection Load Balancing: Overview



JDBC/ODP.NET Runtime Connection Load Balancing: Overview

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 feedback 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.

Connection Load Balancing in RAC

- Connection load balancing allows listeners to distribute connection requests to the best instances.
- Three metrics are available for listeners to decide:
  - Session count by instance
  - Run queue length of the node
  - Service goodness
- The metric used depends on the connection load-balancing goal defined for the service:
  - `LONG`
  - `NONE`
  - `SHORT`

Connection Load Balancing in RAC

Balancing connection requests is referred to as connection load balancing. Connections are routed to different instances in the cluster on the basis of load information available to the listener. Three metrics are available for the listeners to use when selecting the best instance, as follows:

- **Session count by instance:** For services that span RAC instances uniformly and similar capacity nodes, the session count evenly distributes the sessions across RAC. This method is used when the service's connection load-balancing goal is set to `LONG`.
- **Run queue length of the node:** For services that use a subset of RAC instances and different capacity nodes, the run queue length places more sessions on the node with least load at the time of connection creation.
- **Goodness by service:** The goodness of the service is a ranking of the quality of service that the service is experiencing at an instance. It also considers states such as restricted access to an instance. This method is used when the service's connection load-balancing goal is set to `SHORT`. To prevent a listener from routing all connections to the same instance between updates to the goodness values, each listener adjusts its local ratings by a delta as connections are distributed. The delta value used represents an average of resource time that connections consume by using a service. To further reduce login storms, the listener uses a threshold delta when the computed delta is too low because no work was sent over the connections yet.

Load Balancing Advisory: Summary

- Uses `DBMS_SERVICE.GOAL`
  - Service time: weighted moving average of elapsed time
  - Throughput: weighted moving average of throughput
- AWR
  - Calculates goodness locally (MMNL), forwards to master MMON
  - Master MMON builds advisory for distribution of work across RAC, and posts load balancing advice to AQ
  - IMON retrieves advice and send it to ONS
  - EMON retrieves advice and send it to OCI
  - Local MMNL post goodness to PMON
- Listeners use `DBMS_SERVICE.CLB_GOAL=SHORT`
  - Use goodness from PMON to distribute connections.
- Load Balancing Advisory users (inside the pools)
  - Use percentages and flags to send work.

Load Balancing Advisory: Summary

You enable the Load Balancing Advisory when setting the service's goal to `DBMS_SERVICE.GOAL_SERVICE_TIME` or to `DBMS_SERVIVCE.GOAL_THROUGHPUT`.

MMNL (Manageability MoNitor Light) calculates the service metrics for service goal and resource consumption every five seconds. MMNL derives the service goodness from these data.

MMON computes and posts the LBA FAN event to a system queue, and MMNL forwards the service goodness and delta to PMON.

IMON (Instance Monitor) and EMON (Event MONitor) retrieve the event from the queue, and PMON forwards the goodness and delta values to the listeners.

IMON posts the LBA FAN event to the local ONS daemon, and EMON posts it to AQ subscribers.

The server ONS sends the event to the mid-tier ONS (if used).

The mid-tier receives the event and forwards them to subscribers. Each connection pool subscribes to receive events for its own services. On receipt of each event, the Connection Pool Manager refreshes the ratio of work to forward to each RAC instance connection part of the pool. It also ranks the instances to use when aging out connections.

Work requests are routed to RAC instances according to the ratios calculated previously.

Monitor 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$RLBTYP('JFSERV', 'VERSION=1.0 database=xwkE
         service=JFSERV { {instance=xwkE2 percent=50
         flag=UNKNOWN}{instance=xwkE1 percent=50 flag=UNKNOWN}
         } timestamp=2006-01-02 06:19:46')
04:20:16 SYS$RLBTYP('JFSERV', 'VERSION=1.0 database=xwkE
         service=JFSERV { {instance=xwkE2 percent=80
         flag=UNKNOWN}{instance=xwkE1 percent=20 flag=UNKNOWN}
         } timestamp=2006-01-02 06:20:16')
SQL>
```


Monitor LBA FAN Events

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

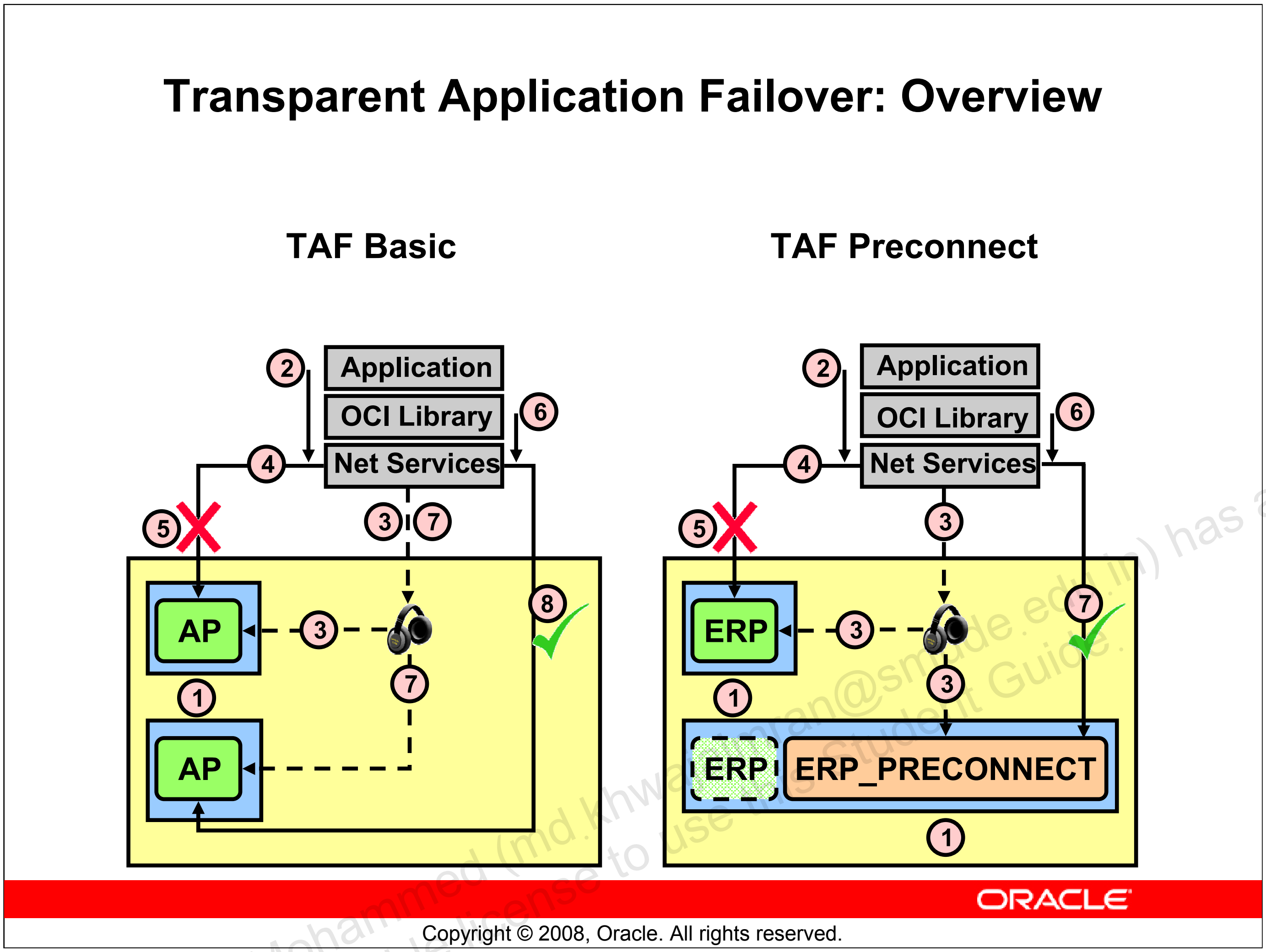
FAN Release Map

Oracle Release	FAN Product Integration	Event System	FAN Event Received and Used
10.1.0.2	JDBC ICC&FCF Server-side callouts	ONS RAC	Up/down/Load Balancing Advisory Up/down
10.1.0.3	CMAN Listeners	ONS PMON	Down Up/down/LBA
10.2	OCI connection pool OCI session pool TAF ODP.NET Custom OCI DG Broker (10.2.0.3)	AQ AQ AQ AQ AQ AQ	Down Down Down Up/down/LBA All Down
11.1	OCI session pool	AQ	LBA

FAN Release Map

The release map for FAN is shown in the slide.

Note: LBA stands for *Load Balancing Advisory*.



Transparent Application Failover (TAF): Overview

TAF is a run-time 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 Without FAN: Example

```
$ srvctl add service -d RACDB -s AP -r I1,I2 \
> -P BASIC
$ srvctl start service -d RACDB -s 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=SESSION)
        (METHOD=BASIC)
        (RETRIES=180)
        (DELAY=5))))
```


TAF Basic Configuration Without FAN: Example

Before using TAF, it is recommended that you create and start a service that is used during connections. By doing so, you benefit from the integration of TAF and services. When you want to use `BASIC` TAF with a service, you should have the `-P BASIC` option when creating the service. After the service is created, you simply start it on your database.

Then, your application needs to connect to the service by using a connection descriptor similar to the one shown in the slide. The `FAILOVER_MODE` parameter must be included in the `CONNECT_DATA` section of your connection descriptor:

- `TYPE` specifies the type of failover. The `SESSION` value means that only the user session is reauthenticated on the server side, whereas open cursors in the OCI application need to be reexecuted. The `SELECT` value means that not only the user session is reauthenticated on the server side, but also the open cursors in the OCI can continue fetching. This implies that the client-side logic maintains fetch-state of each open cursor. A `SELECT` statement is reexecuted by using the same snapshot, discarding those rows already fetched, and retrieving those rows that were not fetched initially. TAF verifies that the discarded rows are those that were returned initially, or it returns an error message.
- `METHOD=BASIC` is used to reconnect at failover time.
- `RETRIES` specifies the number of times to attempt to connect after a failover.
- `DELAY` specifies the amount of time in seconds to wait between connect attempts.

Note: If using TAF, do not set the `GLOBAL_DBNAME` parameter in your `listener.ora` file.

TAF Basic Configuration with FAN: Example

```
$ srvctl add service -d RACDB -s AP -r I1,I2
```

```
$ srvctl start service -d RACDB -s AP
```

```
execute dbms_service.modify_service (                         ,-
 service_name => 'AP'                                         ,-
 aq_ha_notifications => true                                  ,-
 failover_method  => dbms_service.failover_method_basic ,-
 failover_type    => dbms_service.failover_type_session ,-
 failover_retries => 180, failover_delay => 5                 ,-
 clb_goal => dbms_service.clb_goal_long);
```

```
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)))
```


TAF Basic Configuration with FAN: Example

Oracle Database 10*g* Release 2 supports server-side TAF with FAN. To use server-side TAF, create and start your service using `SRVCTL`, then configure TAF in the RDBMS by using the `DBMS_SERVICE` package as shown in the slide. When done, make sure that you define a TNS entry for it in your `tnsnames.ora` file. Note that this TNS name does not need to specify TAF parameters as with the previous slide.

TAF Preconnect Configuration: Example

```
$ srvctl add service -d RACDB -s ERP -r I1 -a I2 \
> -P PRECONNECT
$ srvctl start service -d RACDB -s 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)))
```

ORACLE

TAF Preconnect Configuration: Example

In order to use PRECONNECT TAF, it is recommended 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 –P PRECONNECT option. The shadow service is always named using the format <service_name>_PRECONNECT.

Like with the BASIC method without FAN, you need to use a special connection descriptor to use the PRECONNECT method while connecting to the service. 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.

Note: In all cases where TAF cannot use the PRECONNECT method, TAF falls back to the BASIC method automatically.

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
```


TAF Verification

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.
- Do not use FCF when working with TAF, and vice versa.
- 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.

ORACLE

FAN Connection Pools and TAF Considerations

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. Moreover, FCF and TAF cannot work together.

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.

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

Practice 8: Overview

This practice covers the following topics:

- **Monitoring high availability of connections**
- **Creating and using callout scripts**
- **Using the Load Balancing Advisory**
- **Using the Transparent Application Failover feature**

ORACLE

9

Oracle Clusterware Administration

Objectives

After completing this lesson, you should be able to:

- Manually control the Oracle Clusterware stack
- Change voting disk configuration
- Back up or recover your voting disks
- Manually back up OCR
- Recover OCR
- Replace an OCR mirror
- Repair the OCR configuration
- Change VIP addresses
- Use the CRS framework
- Prevent automatic instance restarts

ORACLE

Objectives

The goal of this lesson is to make sure you understand the various administration tasks you can operate at the Oracle Clusterware level. Although some important procedures are clearly detailed in the lesson, the complete syntax for each command-line tool used is not systematically explained. In this lesson you are using the following tools to administer Oracle Clusterware:

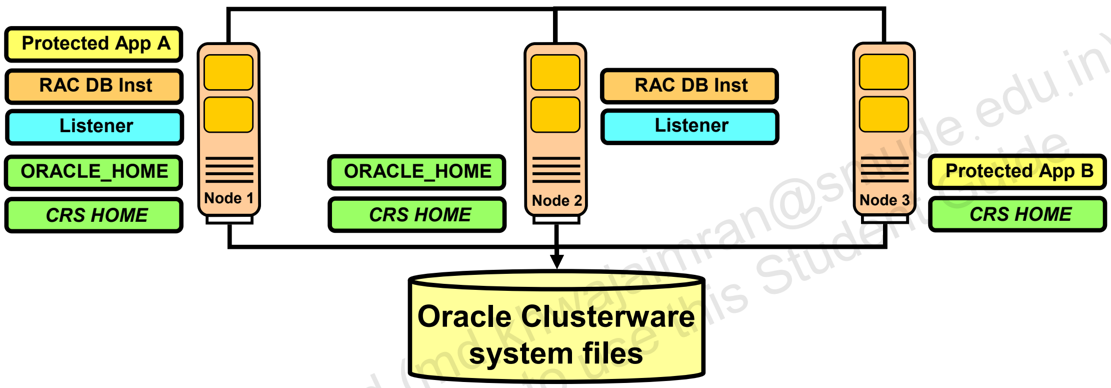
- crsctl
- crs_stat
- ocrconfig
- ocrcheck
- ocrdump
- srvctl
- oifcfg
- crs_profile, crs_register, crs_setperm, crs_start, crs_relocate, crs_stop, crs_unregister

For more information about the various options of the commands you can see in this lesson, refer to the *Oracle Database Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide*.

Oracle Clusterware: Overview

Portable cluster infrastructure that provides HA to RAC databases and/or other applications:

- **Monitors applications' health**
- **Restarts applications on failure**
- **Can fail over applications on node failure**



ORACLE

Oracle Clusterware: Overview

Oracle Clusterware is a portable cluster infrastructure that provides High Availability (HA) to RAC databases and other applications. Oracle Clusterware makes applications highly available by monitoring the health of the applications, by restarting applications on failure, by relocating applications to another cluster node when the currently used node fails or when the application can no longer run in the current node. In the case of node failure, certain type of protected applications, such as a database instance, are not failed over surviving nodes.

Here, a cluster is a collection of two or more nodes where the nodes share a common pool of storage used by the Oracle Clusterware system files (OCR and voting disk), a common network interconnect, and a common operating system.

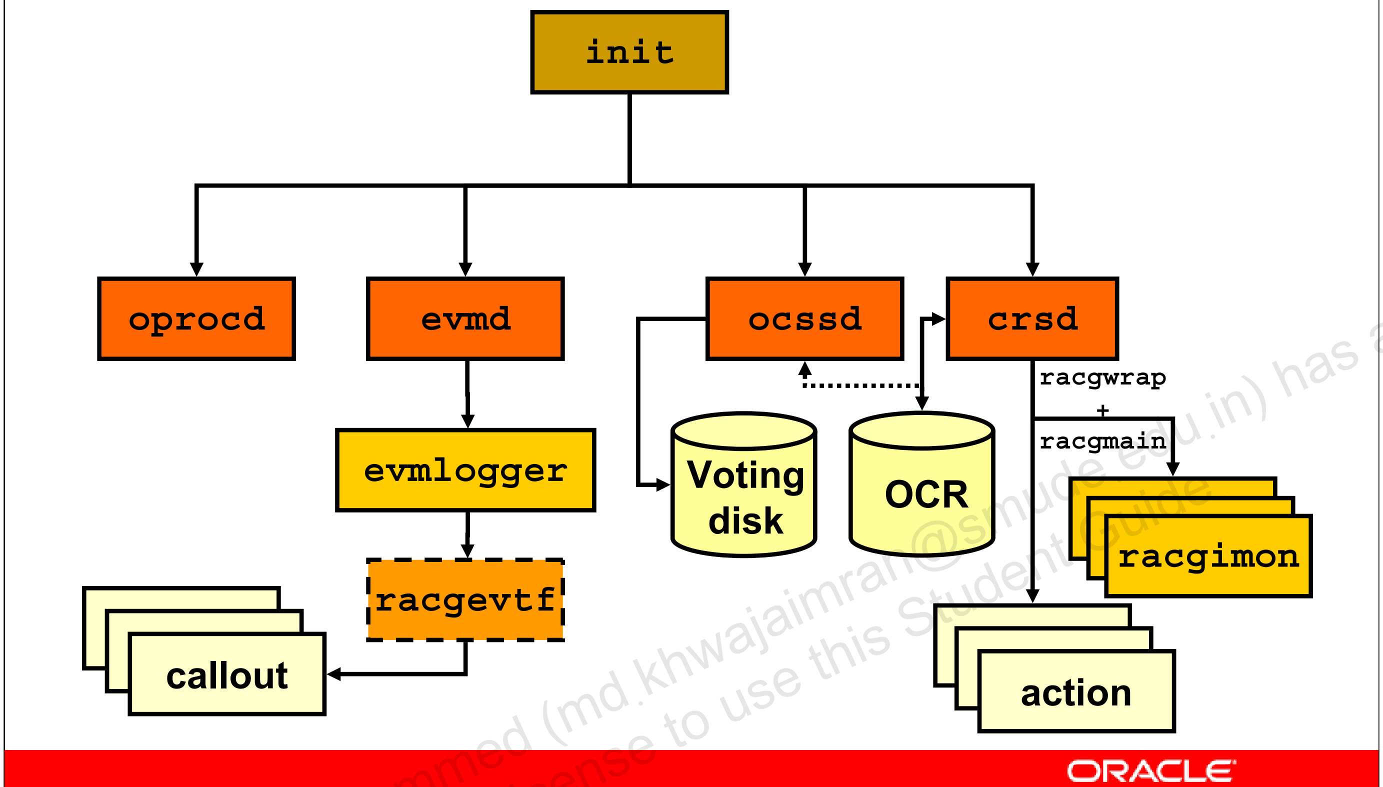
The graphic in the slide describes a possible three-node configuration where Node1 runs a RAC database instance, a listener, and application A, all protected by Oracle Clusterware.

On Node2, only one RAC database instance and a listener are protected by Oracle Clusterware.

On Node3, one application B is protected by Oracle Clusterware.

Oracle Clusterware monitors all protected applications periodically, and based on the defined failover policy, it can restart them either on the same node or relocate them to another node, or it can decide to not restart them at all.

Note: Although Oracle Clusterware is a required component for using RAC, it does not require a RAC license when used only to protect applications other than RAC databases.



Oracle Clusterware Run-Time View

On UNIX, the Oracle Clusterware stack is run from entries in `/etc/inittab` with respawn. On Windows, it is run using the services controller. Here is a basic description of each process:

- **Cluster Synchronization Services Daemon (OCSSD):** This process runs in both vendor clusterware and nonvendor clusterware environments. It integrates with existing vendor clusterware, when present. OCSSD's primary job is internode health monitoring, primarily using the network interconnect as well as voting disks, and database/ASM instance endpoint discovery via group services. OCSSD runs as user `oracle`, and failure exit causes machine reboot to prevent data corruption in the event of a split brain.
- **Process Monitor Daemon (OPROCD):** This process is spawned in any nonvendor clusterware environment. If OPROCD detects problems, it kills a node. It runs as `root`. This daemon is used to detect hardware and driver freezes on the machine. If a machine was frozen long enough for the other nodes to evict it from the cluster, it needs to kill itself to prevent any I/O from being reissued to the disk after the rest of the cluster has remastered locks.

Oracle Clusterware Run-Time View (continued)

- **Cluster Ready Services Daemon (CRSD):** This process is the engine for High Availability operations. It manages Oracle Clusterware registered applications and starts, stops, check, and fails them over via special action scripts. CRSD spawns dedicated processes called RACGIMON that monitor the health of the database and ASM instances and host various feature threads such as Fast Application Notification (FAN). One RACGIMON process is spawned for each instance. CRSD maintains configuration profiles as well as resource statuses in OCR (Oracle Cluster Registry). It runs as `root` and is restarted automatically on failure. In addition, CRSD can spawn temporary children to execute particular actions such as:
  - `racgeut` (Execute Under Timer), to kill actions that do not complete after a certain amount of time
  - `racgmdb` (Manage Database), to start/stop/check instances
  - `racgchsn` (Change Service Name), to add/delete/check service names for instances
  - `racgons`, to add/remove ONS configuration to OCR
  - `racgvip`, to start/stop/check instance virtual IP
- **Event Management Daemon (EVMD):** This process forwards cluster events when things happen. It spawns a permanent child `evmlogger` that, on demand, spawns children such as `racgevtf` to invoke callouts. It runs as `oracle`, and is restarted automatically on failure.

Note: The RACG infrastructure is used to deploy the Oracle database in highly available clustered environment. This infrastructure is mainly implemented using the `racgwrap` script that invokes the `racgmain` program. It is used by CRS to execute actions for all node-centric resources as well as to proxy actions for all instance-centric resources to RACGIMON. Basically, this infrastructure is responsible for managing all `ora.*` resources.

Manually Control Oracle Clusterware Stack

Might be needed for planned outages:

```
# crsctl stop crs -wait
```

```
# crsctl start crs -wait
```

```
# crsctl disable crs
```

```
# crsctl enable crs
```


ORACLE

Manually Control Oracle Clusterware Stack

When a node of Oracle Clusterware comes up, the Oracle Clusterware processes start up automatically. You can control this by using `crsctl` commands. You may have to manually control the Oracle Clusterware stack while applying patches or during any planned outages. In addition, these commands can be used by third-party clusteware when used in combination with Oracle Clusterware.

You can stop the Oracle Clusterware stack by using the `crsctl stop crs` command.

You can also start the Oracle Clusterware stack by using the `crsctl start crs` command. The `-wait` option displays progress and status for each daemon. Without this option, the command returns immediately.

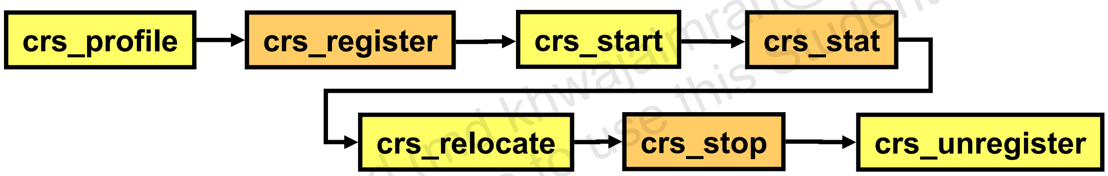
Use the `crsctl disable crs` command to disable Oracle Clusterware from being started in a subsequent reboot. This command does not stop the currently running Oracle Clusterware stack.

Use the `crsctl enable crs` command to enable Oracle Clusterware to be started in a subsequent reboot.

Note: You must run these commands as `root`.

CRS Resources

- A resource is a CRS-managed application.
- Application profile attributes are stored in OCR:
  - Check interval
  - Action script
  - Dependencies
  - Failure policies
  - Privileges
  - …
- An action script must be able to:
  - Start the application
  - Stop the application
  - Check the application
- Life cycle of a resource:



CRS Resources

CRS is the primary program for managing High Availability operations of applications within the cluster. Applications that CRS manages are called resources. By default, CRS can manage RAC resources such as database instance, ASM instances, listeners, instance VIPs, services, ONS, and GSD. However, CRS is also able to manage other type of application processes and application VIPs. CRS resources are managed according to their configuration parameters (resource profile) stored in OCR and an action script stored anywhere you want. The resource profile contains information such as the check interval, failure policies, the name of the action script, privileges that CRS should use to manage the application, and resource dependencies. The action script must be able to start, stop, and check the application.

CRS provides the following facilities to support the life cycle of a resource:

- `crs_profile` creates and edit a resource profile.
- `crs_register` adds the resource to the list of applications managed by CRS.
- `crs_start` starts the resource according to its profile. After a resource is started, its application process is continuously monitored by CRS using a check action at regular intervals. Also, when the application goes offline unexpectedly, it is restarted and/or failed over to another node according to its resource profile.
- `crs_stat` informs you about the current status of a list of resources.
- `crs_relocate` moves the resource to another node of the cluster.
- `crs_unregister` removes the resource from the monitoring scope of CRS.

RAC Resources

```
$ <CRS HOME>/bin/crs_stat -t
Name                            Type        Target State  Host
----------------------------------------------------------------
ora.atlhp8.ASM1.asm             application ONLINE ONLINE atlhp8
ora.atlhp8.LISTENER_ATLHP8.lsnr application ONLINE ONLINE atlhp8
ora.atlhp8.gsd                  application ONLINE ONLINE atlhp8
ora.atlhp8.ons                  application ONLINE ONLINE atlhp8
ora.atlhp8.vip                  application ONLINE ONLINE atlhp8
ora.atlhp9.ASM2.asm             application ONLINE ONLINE atlhp9
ora.atlhp9.LISTENER_ATLHP9.lsnr application ONLINE ONLINE atlhp9
ora.atlhp9.gsd                  application ONLINE ONLINE atlhp9
ora.atlhp9.ons                  application ONLINE ONLINE atlhp9
ora.atlhp9.vip                  application ONLINE ONLINE atlhp9
ora.xwkE.JF1.cs                 application ONLINE ONLINE atlhp8
ora.xwkE.JF1.xwkE1.srv          application ONLINE ONLINE atlhp8
ora.xwkE.JF1.xwkE2.srv          application ONLINE ONLINE atlhp9
ora.xwkE.db                     application ONLINE ONLINE atlhp9
ora.xwkE.xwkE1.inst             application ONLINE ONLINE atlhp8
ora.xwkE.xwkE2.inst             application ONLINE ONLINE atlhp9
```


RAC Resources

The `crs_stat -t` command shows you all the resources that are currently under Oracle Clusterware control. In the example shown in the slide, only resources starting with the prefix `ora.` exist. These are the resources that implement RAC high availability in a clustered environment.

You can see that, by default, Oracle Clusterware can control databases, database and ASM instances, VIP/ONS/GSD/Listener (also called `nodeapps`), services, and service members.

In the slide, the `Target` status for the resources is `ONLINE`, which means that at next node restart, Oracle Clusterware will try to start them up automatically.

`State` shows you the current status of the resource.

`Target` can be `ONLINE` or `OFFLINE`.

`State` can be `ONLINE`, `OFFLINE`, or `UNKNOWN`. `UNKNOWN` results from a failed start/stop action, and can be reset only by a `crs_stop -f` *`resourceName`* command. The combination of `Target` and `State` can be used to derive whether a resource is starting or stopping.

`Host` shows you the name of the host on which the resource is managed.

Note: Using the `crs_stat -t` command truncates the resource names for formatting reasons. The output example reestablishes entire names for clarity purposes.

Resource Attributes: Example

```
$ <CRS HOME>/bin/crs_stat -p ora.JFDB.JFDB1.inst
NAME=ora.JFDB.JFDB1.inst
TYPE=application
ACTION_SCRIPT=/u01/app/oracle/product/11g/bin/racgwrap
ACTIVE_PLACEMENT=0
AUTO_START=1
CHECK_INTERVAL=600
DESCRIPTION=CRS application for Instance
FAILOVER_DELAY=0
FAILURE_INTERVAL=0
FAILURE_THRESHOLD=0
HOSTING_MEMBERS=atlhp8
PLACEMENT=restricted
REQUIRED_RESOURCES=ora.atlhp8.ASM1.asm
RESTART_ATTEMPTS=5
…
$ <CRS HOME>/bin/crs_stat -t ora.xwkE.xwkE1.inst
Name           Type         Target    State     Host
-----------------------------------------------------
ora....E1.inst application  ONLINE    ONLINE    atlhp8
```


Resource Attributes: Example

You can use the `crs_stat -p` *`resource_name`* command to print the OCR contents for the named resource. The example in the slide shows you what you get for a RAC database instance. Not all attributes are mandatory for each resource. Here is a brief description of the most important attributes shown on the output above:

- `NAME` is the name of the application resource.
- `TYPE` must be `APPLICATION` for all CRS resources.
- `ACTION_SCRIPT` is the name and location of the action script used by CRS to start, check, and stop the application. The default path is `<CRS HOME>/crs/script`.
- `ACTIVE_PLACEMENT` defaults to `0`. When set to `1`, Oracle Clusterware reevaluates the placement of a resource during addition or restart of a cluster node.
- `AUTO_START` is a flag indicating whether Oracle Clusterware should automatically start a resource after a cluster restart, regardless of whether the resource was running before the cluster restart. When set to `0`, Oracle Clusterware starts the resource only if it was running before the restart. When set to `1`, Oracle Clusterware always starts the resource after a restart. When set to `2`, Oracle Clusterware never restarts the resource (regardless of the resource's state when the node stopped).
- `CHECK_INTERVAL` is the time interval, in seconds, between repeated executions of the check command for the application.

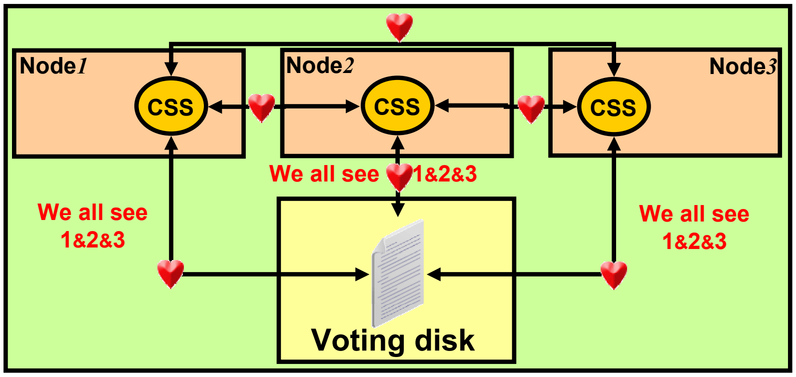
Resource Attributes: Example (continued)

- DESCRIPTION is a description of the resource.
- FAILOVER_DELAY is the amount of time, in seconds, that Oracle Clusterware waits before attempting to restart or fail over a resource.
- FAILURE_INTERVAL is the interval, in seconds, during which Oracle Clusterware applies the failure threshold. If the value is zero (0), then tracking of failures is disabled.
- FAILURE_THRESHOLD is the number of failures detected within a specified FAILURE_INTERVAL before Oracle Clusterware marks the resource as unavailable and no longer monitors it. If a resource's check script fails this several times, then the resource is stopped and set offline. If the value is zero (0), then tracking of failures is disabled. The maximum value is 20.
- HOSTING_MEMBERS is an ordered list of cluster nodes separated by blank spaces that can host the resource. Run the olsnodes commands to see your node names.
- PLACEMENT defines the placement policy (balanced, favored, or restricted) that specifies how Oracle Clusterware chooses the cluster node on which to start the resource:
  - balanced: Oracle Clusterware favors starting or restarting the application on the node that is currently running the fewest resources. The host with the fewest resources running is chosen. If no node is favored by these criteria, then any available node is chosen.
  - favored: Oracle Clusterware refers to the list of nodes in the HOSTING_MEMBERS attribute of the application profile. Only cluster nodes that are in this list and that satisfy the resource requirements are eligible for placement consideration. The order of the hosting nodes determines which node runs the application. If none of the nodes in the hosting node list are available, then Oracle Clusterware places the application on any available node. This node may or may not be included in the HOSTING_MEMBERS list.
  - restricted: Similar to the favored policy, except that if none of the nodes on the hosting list are available, then Oracle Clusterware does not start or restart the application. A restricted placement policy ensures that the application never runs on a node that is not on the list, even if you manually relocate it to that node.
- REQUIRED_RESOURCES is an ordered list of resource names separated by blank spaces that this resource depends on. Oracle Clusterware relocates or stops an application if a required resource becomes unavailable. Therefore, in the example on the previous page, it is clear that to start the JFDB1 instance, the ASM instance ASM1 must be started first.
- RESTART_ATTEMPTS is the number of times that Oracle Clusterware only attempts to restart a resource on a single cluster node before attempting to relocate the resource. After the time period that you have indicated by the setting for UPTIME_THRESHOLD has elapsed, Oracle Clusterware resets the value for the restart counter (RESTART_COUNTS) to 0. Basically, RESTART_COUNTS cannot exceed RESTART_ATTEMPTS for the UPTIME_THRESHOLD period.

The crs_stat -t *resource_name* command shows you the named resource's states. In the slide, the Target status for the resource is ONLINE meaning that at the next node restart, Oracle Clusterware will try to start up the instance. State shows you the current status of the instance.

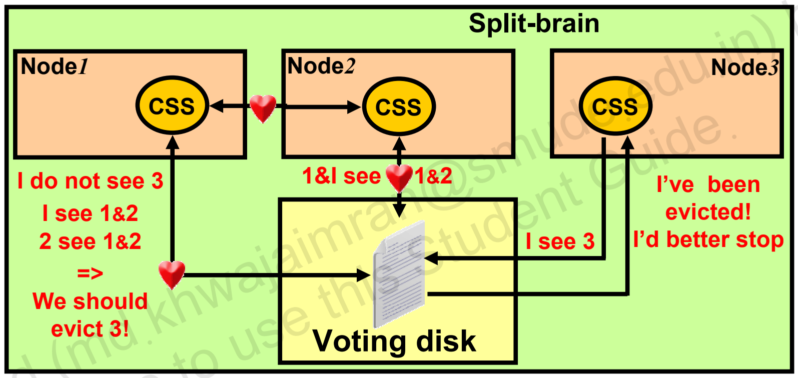
Note: The output shown in the slide is truncated for formatting reasons.

Main Voting Disk Function



Nodes can see each other.

Node*3* can no longer communicate through private interconnect. Others no longer see its heartbeats and evict that node by using the voting disk.



Main Voting Disk Function

CSS is the service that determines which nodes in the cluster are available, and provides cluster group membership and simple locking services to the other processes. CSS typically determines node availability via communication through a dedicated private network with a voting disk used as a secondary communication mechanism. Basically, this is done by sending heartbeat messages through the network and the voting disk as illustrated by the top graphic in the slide. The voting disk is a shared raw disk partition or file on a clustered file system that is accessible to all nodes in the cluster. Its primary purpose is to help in situations where the private network communication fails. When that happens, the cluster is unable to have all nodes remain available because they are no longer able to synchronize I/O to the shared disks. Therefore, some of the nodes must go offline. The voting disk is then used to communicate the node state information used to determine which nodes go offline. Without the voting disk, it can become impossible for an isolated node(s), to determine whether it is experiencing a network failure or whether the other nodes are no longer available. It would then be possible for the cluster to get into a state where multiple subclusters of nodes would have unsynchronized access to the same database files. This situation is commonly referred to as the cluster split-brain problem.

The graphic at the bottom of the slide illustrates what happens when node3 can no longer send heartbeats to other members of the cluster. When others can no longer see node3's heartbeats, they decide to evict that node by using the voting disk. When node3 reads the removal message, it generally reboots itself to make sure all outstanding write I/Os are lost.

Main Voting Disk Function (continued)

Note: In addition to the voting disk mechanism, a similar mechanism also exists for RAC database instances. At the instance level, the control file is used by all participating instances for voting. This is necessary because there can be cases where instances should be evicted, even if network connectivity between nodes is still in good shape.

For example, if LMON or LMD is stuck on one instance, it could then be possible to end up with a frozen cluster database. Therefore, instead of allowing a clusterwide hang to occur, RAC evicts the problematic instance(s) from the cluster.

When the problem is detected, the instances race to get a lock on the control file. The instance that obtains the lock tallies the votes of the instances to decide membership. This is called Instance Membership Reconfiguration (IMR).

Important CSS Parameters

- **MISSCOUNT:**
  - **Represents network heartbeat timeouts**
  - **Determines disk I/O timeouts during reconfiguration**
  - **Defaults to 30 seconds**
  - **Should not be changed**
- **DISKTIMEOUT:**
  - **Represents disk I/O timeouts outside reconfiguration**
  - **Defaults to 200 seconds**
  - **Can be *temporarily* changed when experiencing very long I/O latencies to voting disks:**
    1. **Shut down Oracle Clusterware on *all nodes but one.***
    2. **As `root` on available node, use:** `crsctl set css disktimeout M+1`
    3. **Reboot available node.**
    4. **Restart all other nodes.**
- **Can be changed ONLY under explicit guidance from Oracle Support**

Important CSS Parameters

The CSS `misscount` parameter represents the maximum time, in seconds, that a network heartbeat across the interconnect can be missed before entering into a cluster reconfiguration for node eviction purposes. The default value for the `misscount` parameter value is 30 seconds. The `misscount` parameter's value drives cluster membership reconfigurations and directly effects the availability of the cluster. Its default settings should be acceptable. Modifying this value not only can influence the timeout interval for the I/O to the voting disk, but also influences the tolerance for missed network heartbeats across the interconnect. This directly affects database and cluster availability. The CSS `misscount` default value, when using vendor (non-Oracle) clusterware, is also 30 seconds, and you should not change the default `misscount` value if you are using vendor clusterware.

The CSS `disktimeout` parameter represents the maximum time, in seconds, that a disk heartbeat can be missed (outside cluster reconfiguration events) before entering into a cluster reconfiguration for node eviction purposes. Its default value is 200 seconds. However, if I/O latencies to the voting disk are greater than the default internal I/O timeout, the cluster may experience CSS node evictions. The most common cause in these latencies relate to multipath I/O software drivers and the reconfiguration times resulting from a failure in the I/O path. Therefore, until the underlying storage I/O latency is resolved, `disktimeout` could be temporarily modified based only on "maximum I/O latency to the voting disk" including latencies resulting from I/O path reconfiguration plus one second (*M+1*).

Multiplexing Voting Disks

- Voting disk is a vital resource for your cluster availability.
- Use one voting disk if it is stored on a reliable disk.
- Otherwise, use multiplexed voting disks:
  - There is no need to rely on multipathing solutions.
  - Multiplexed copies should be stored on independent devices.
  - Make sure that there is no I/O starvation for your voting disks devices.
  - *Use at least three multiplexed copies.*
- CSS uses a simple majority rule to decide whether voting disk reads are consistent.

Multiplexing Voting Disks

CSS availability can be improved by configuring it with multiple voting disks. Using only one voting disk is adequate for clusters that are configured to use a single, highly available shared disk, where both the database files and the CSS voting disk reside. However, it is desirable to use multiple copies of the voting disk when using less reliable storage. Also, you can use multiple voting disks so that you do not have to rely on a multipathing solution.

The way voting disk multiplexing is implemented forces you to have at least three voting disks. To avoid a single point of failure, your multiplexed voting disk should be located on physically independent storage devices with a predictable load well below saturation.

When using multiplexed copies of the voting disk, CSS multiplexes voting data to all the voting disks. When CSS needs to read the voting disk, it reads all the information from all the voting disks. If strictly more than half of the voting disks are up and contain consistent information, CSS can use that consistent data in the same way as a single voting disk configuration. If less than half of the voting disks have readable consistent data, CSS will need to self-terminate like in the situation where a single voting disk cannot be read by CSS. This self-termination is to prevent disjoint subclusters from forming. You can have up to 32 voting disks, but use the following formula to determine the number of voting disks you should use: `v = f*2+1`, where `v` is the number of voting disks, and `f` is the number of disk failures you want to survive.

Note: A typical voting disk configuration comprises between three and five disks.

Change Voting Disk Configuration

- Voting disk configuration can be changed dynamically.
- To add a new voting disk:

```
# crsctl add css votedisk <new voting disk path>
```

- To remove a voting disk:

```
# crsctl delete css votedisk <old voting disk path>
```

- If Oracle Clusterware is down on all nodes, use the `-force` option:

```
# crsctl add css votedisk <new voting disk path> -force
```

```
# crsctl delete css votedisk <old voting disk path> -force
```


Change Voting Disk Configuration

During Oracle Clusterware installation, you can multiplex your voting disk by using the Specify Voting Disk Location screen of the Oracle Universal Installer. This screen allows you to specify three voting disk locations. However, you can dynamically add and remove voting disks after installing Oracle Clusterware by using the following commands as the `root` user:

- To add a voting disk: `crsctl add css votedisk` *path*
- To remove a voting disk: `crsctl delete css votedisk` *path*

Here, *path* is the fully qualified path.

If your cluster is down, you can use the `-force` option (at the very end of the `crsctl` command) to modify the voting disk configuration with either of these commands without interacting with active Oracle Clusterware daemons. However, using the `-force` option while any cluster node is active may corrupt your configuration.

Back Up and Recover Your Voting Disks

- Should not be needed. Instead, you should add/remove.
- Recommendation is to use symbolic links.
- Back up one voting disk by using the `dd` command.
  - After Oracle Clusterware installation
  - After node addition or deletion
  - Cannot be done online

```
$ crsctl query css votedisk
```

```
$ dd if=<voting disk path> of=<backup path> bs=4k
```

- Recover voting disks by restoring the first one using the `dd` command, and then multiplex it if necessary.
- If no voting disk backup is available, reinstall Oracle Clusterware.

Back Up and Recover Your Voting Disks

There should be no need to back up a voting disk. Simply add a new one and drop a bad one.

It is recommended to use symbolic links to specify your voting disk paths. This is because the voting disk paths are directly stored in OCR, and editing the OCR file directly is not supported. By using symbolic links to your voting disks, it becomes easier to restore your voting disks if their original locations can no longer be used as a restore location.

A new backup of one of your available voting disks should be taken any time a new node is added, or an existing node is removed. The recommended way to do that is to use the `dd` command (`ocopy` in Windows environments). As a general rule on most platforms, including Linux and Sun, the block size for the `dd` command should be at least 4 KB to ensure that the backup of the voting disk gets complete blocks.

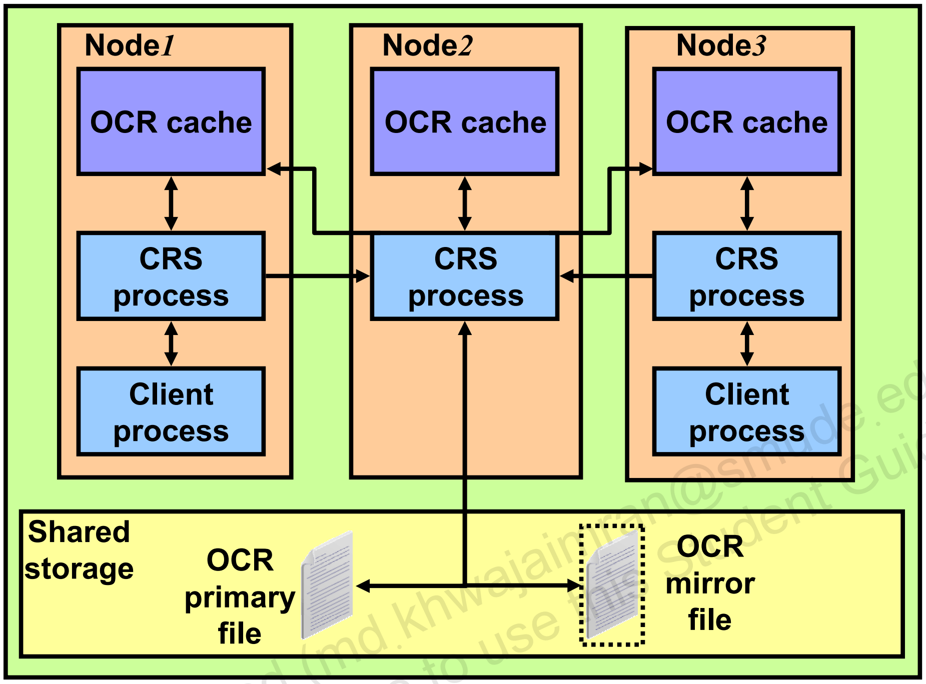
Before backing up your voting disk with the `dd` command, make sure that you stopped Oracle Clusterware on all nodes.

The `crsctl query css votedisk` command lists the voting disks currently used by CSS. This can help you to determine which voting disk to backup.

The slide shows you the procedure you can follow to back up and restore your voting disk.

Note: If you lose all your voting disks and you do not have any backup, you must reinstall Oracle Clusterware.

OCR Architecture



OCR Architecture

Cluster configuration information is maintained in Oracle Cluster Registry (OCR). OCR relies on a distributed shared-cache architecture for optimizing queries, and clusterwide atomic updates against the cluster repository. Each node in the cluster maintains an in-memory copy of OCR, along with the Cluster Ready Services Daemon (CRSD) that accesses its OCR cache. Only one of the CRS processes actually reads from and writes to the OCR file on shared storage. This process is responsible for refreshing its own local cache, as well as the OCR cache on other nodes in the cluster. For queries against the cluster repository, the OCR clients communicate directly with the local OCR process on the node from which they originate. When clients need to update OCR, they communicate through their local CRS process to the CRS process that is performing input/output (I/O) for writing to the repository on disk.

The main OCR client applications are the Oracle Universal Installer (OUI), `SRVCTL`, Enterprise Manager (EM), the Database Configuration Assistant (DBCA), the Database Upgrade Assistant (DBUA), `NetCA`, and the Virtual Internet Protocol Configuration Assistant (VIPCA). Furthermore, OCR maintains dependency and status information for application resources defined within Oracle Clusterware, specifically databases, instances, services, and node applications.

OCR Architecture (continued)

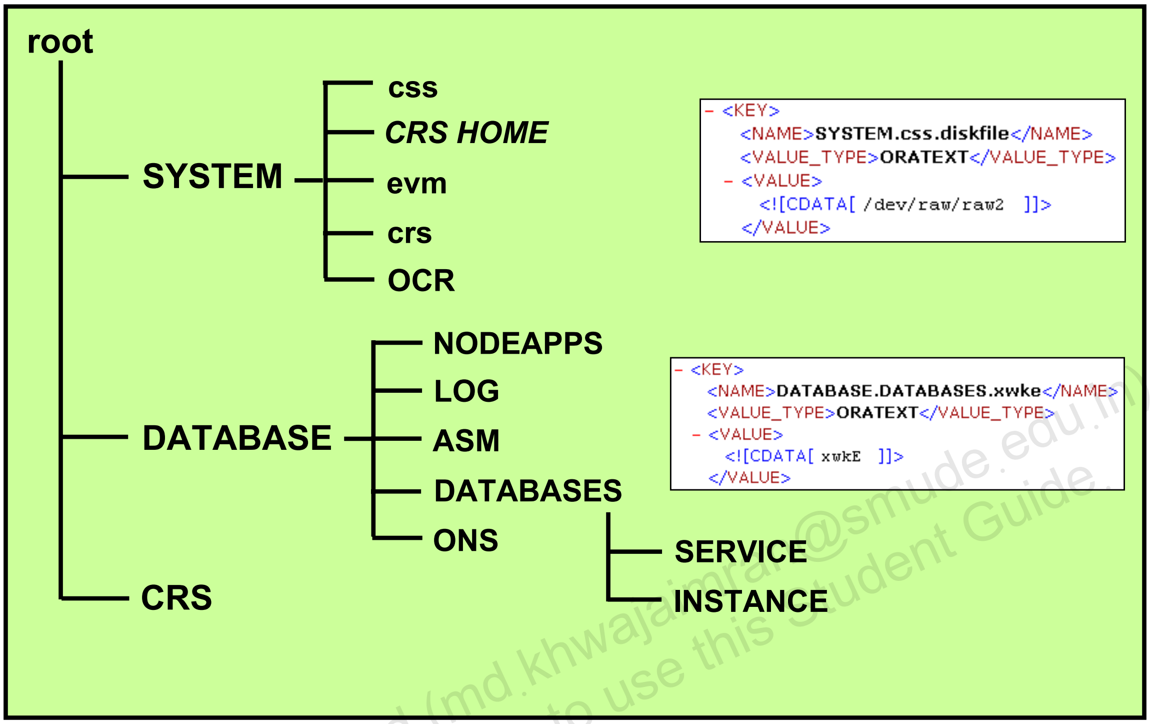
The installation process for Oracle Clusterware gives you the option of automatically mirroring OCR. This creates a second OCR file (the OCR *mirror* file) to duplicate the original OCR file (the *primary* OCR file). You can put the OCR mirror file on a cluster file system or on a shared raw device. Although it is recommended to mirror your OCR, you are not forced to do it during installation.

The name of the OCR configuration file on UNIX-based system is `ocr.loc`, and the OCR file location variables are `ocrconfig_loc` and `ocrmirrorconfig_loc`.

It is strongly recommended that you use mirrored OCR files if the underlying storage is not RAID. This prevents OCR from becoming a single point of failure.

Note: OCR also serves as a configuration file in a single instance with the ASM, where there is one OCR per node.

OCR Contents and Organization



ORACLE

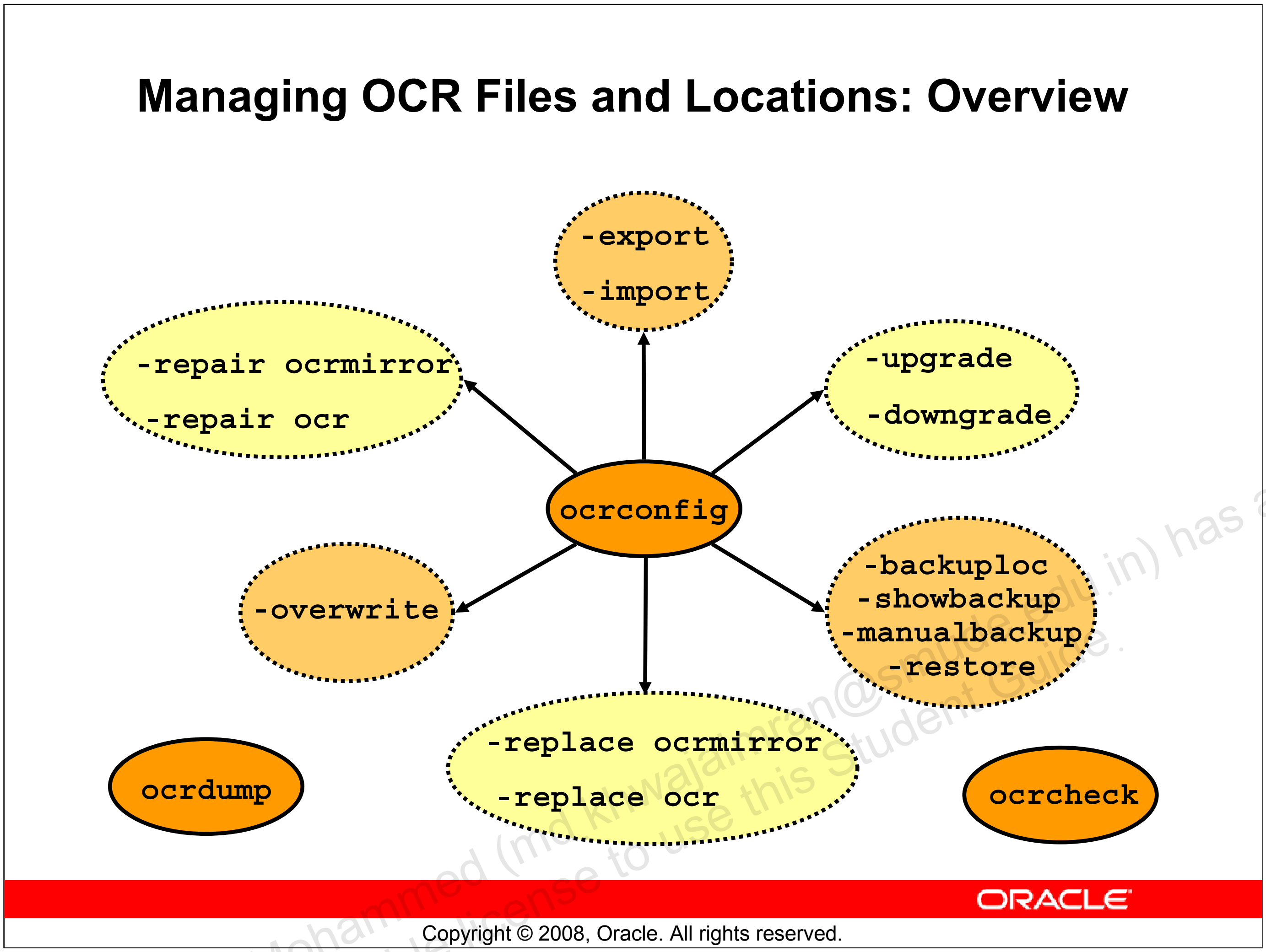
OCR Contents and Organization

Every clustering technology requires a repository through which the clustering software and other cluster-aware application processes can share information. Oracle Clusterware uses Oracle Cluster Registry to store information about resources it manages. This information is stored in a treelike structure using key–value pairs.

The slide shows you the main branches composing the OCR structure:

- The SYSTEM keys contain data related to the main Oracle Clusterware processes such as CSSD, CRSD, and EVMD. For example, CSSD keys contain information about the `misscount` parameter and voting disk paths.
- The DATABASE keys contain data related to the RAC databases that you registered with Oracle Clusterware. As shown, you have information about instances, `nodeapps`, services, and so on.
- The last category of keys that you can find in OCR relate to the resource profiles used by Oracle Clusterware to maintain availability of the additional application you registered. These resources include the additional application VIPs, the monitoring scripts, and the check interval values.

Note: The XML data on the right side of the slide were obtained by using the `ocrdump -xml` command.



Managing OCR Files and Locations: Overview

You use the `ocrconfig` tool (the main configuration tool for Oracle Cluster Registry) to:

- Generate logical backups of OCR using the `–export` option, and use them later to restore your OCR information using the `–import` option
- Upgrade or downgrade OCR
- Use the `–showbackup` option to view the generated backups (by default, OCR is backed up on a regular basis). These backups are generated in a default location that you can change using the `–backuploc` option. If need be, you can then restore physical copies of your OCR using the `–restore` option. You can also manually create OCR backups using the `-manualbackup` option.
- Use the `–replace ocr` or `-replace ocrmirror` options to add, remove, or replace the primary OCR files or the OCR mirror file
- Use the `-overwrite` option under the guidance of Support Services because it allows you to overwrite some OCR protection mechanisms when one or more nodes in your cluster cannot start because of an OCR corruption
- Use the `-repair` option to change the parameters listing the OCR and OCR mirror locations

The `ocrcheck` tool enables you to verify the OCR integrity of both OCR and its mirror. Use the `ocrdump` utility to write the OCR contents (or part of it) to a text or XML file.

Automatic OCR Backups

- The OCR content is critical to Oracle Clusterware.
- OCR is automatically backed up physically:
  - Every four hours: CRS keeps the last three copies.
  - At the end of every day: CRS keeps the last two copies.
  - At the end of every week: CRS keeps the last two copies.

```
$ cd $ORACLE_BASE/Crs/cdata/jfv_clus
$ ls -lt
-rw-r--r-- 1 root root 4784128 Jan  9 02:54 backup00.ocr
-rw-r--r-- 1 root root 4784128 Jan  9 02:54 day_.ocr
-rw-r--r-- 1 root root 4784128 Jan  8 22:54 backup01.ocr
-rw-r--r-- 1 root root 4784128 Jan  8 18:54 backup02.ocr
-rw-r--r-- 1 root root 4784128 Jan  8 02:54 day.ocr
-rw-r--r-- 1 root root 4784128 Jan  6 02:54 week_.ocr
-rw-r--r-- 1 root root 4005888 Dec 30 14:54 week.ocr
```

- Change the default automatic backup location:

```
# ocrconfig –backuploc /shared/bak
```

ORACLE

Automatic OCR Backups

OCR contains important cluster and database configuration information for RAC and Oracle Clusterware. One of the Oracle Clusterware instances (CRSD master) in the cluster automatically creates OCR backups every four hours, and CRS retains the last three copies. That CRSD process also creates an OCR backup at the beginning of each day and of each week, and retains the last two copies. This is illustrated in the slide where you can see the content of the default backup directory of the CRSD master.

Although you cannot customize the backup frequencies or the number of retained copies, you have the possibility to identify the name and location of the automatically retained copies by using the `ocrconfig -showbackup` command.

The default target location of each automatically generated OCR backup file is the `<CRS Home>/cdata/<cluster name>` directory. It is recommended to change this location to one that is shared by all nodes in the cluster by using the `ocrconfig -backuploc <new location>` command. This command takes one argument that is the full path directory name of the new location.

Back Up OCR Manually

- Daily backups of your automatic OCR backups to a different storage device:
  – Use your favorite backup tool.
- On demand physical backups:

```
# ocrconfig –manualbackup
```

- Logical backups of your OCR before and after making significant changes:

```
# ocrconfig –export file name
```

- Make sure that you restore OCR backups that match your current system configuration.

Back Up OCR Manually

Because of the importance of OCR information, it is also recommended to manually create copies of the automatically generated physical backups. You can use any backup software to copy the automatically generated backup files, and it is recommended to do that at least once daily to a different device from where the primary OCR resides.

You can performs an OCR backup on demand using the `–manualbackup` option. The backup is generated in the location that you specify with the `-backuploc` option. .

In addition, you should also export the OCR contents before and after making significant configuration changes such as adding or deleting nodes from your environment, modifying Oracle Clusterware resources, or creating a database. Use the `ocrconfig -export` command as the `root` user to generate OCR logical backups. You need to specify a file name as the argument of the command, and it generates a binary file that you should not try to edit.

Most configuration changes that you make not only change the OCR contents but also cause file and database object creation. Some of these changes are often not restored when you restore OCR. Do not perform an OCR restore as a correction to revert to previous configurations if some of these configuration changes fail. This may result in an OCR with contents that do not match the state of the rest of your system.

Note: If you try to export OCR while an OCR client is running, you get an error.

Recover OCR Using Physical Backups

1. **Locate a physical backup:** `$ ocrconfig -showbackup`
2. **Review its contents:** `# ocrdump -backupfile file_name`
3. **Stop Oracle Clusterware on all nodes:** `# crsctl stop crs`
4. **Restore the physical OCR backup:**

```
# ocrconfig -restore <CRS HOME>/cdata/jfv_clus/day.ocr
```

5. **Restart Oracle Clusterware on all nodes:** `# crsctl start crs`
6. **Check OCR integrity:** `$ cluvfy comp ocr -n all`

Recover OCR Using Physical Backups

Use the following procedure to restore OCR on UNIX-based systems:

1. Identify the OCR backups by using the `ocrconfig -showbackup` command. You can execute this command from any node as user `oracle`. The output tells you on which node and which path to retrieve both automatically and manually generated backups. Use the `auto` or `manual` argument to display only one category.
2. Review the contents of the backup by using `ocrdump -backupfile` *`file_name`*, where *`file_name`* is the name of the backup file.
3. Stop Oracle Clusterware on all the nodes of your cluster by executing the `crsctl stop crs` command on all the nodes as the `root` user.
4. Perform the restore by applying an OCR backup file that you identified in step one using the following command as the `root` user, where *`file_name`* is the name of the OCR file that you want to restore. Make sure that the OCR devices that you specify in the OCR configuration file (`/etc/oracle/ocr.loc`) exist and that these OCR devices are valid before running this command: `ocrconfig -restore` *`file_name`*
5. Restart Oracle Clusterware on all the nodes in your cluster by restarting each node or by running the `crsctl start crs` command as the `root` user.
6. Run the following command to verify OCR integrity, where the `-n all` argument retrieves a listing of all the cluster nodes that are configured as part of your cluster: `cluvfy comp ocr -n all`

Recover OCR Using Logical Backups

1. Locate a logical backup created using an OCR export.
2. Stop Oracle Clusterware on all nodes:

```
# crsctl stop crs
```

3. Restore the logical OCR backup:

```
# ocrconfig -import /shared/export/ocrback.dmp
```

4. Restart Oracle Clusterware on all nodes:

```
# crsctl start crs
```

5. Check OCR integrity:

```
$ cluvfy comp ocr -n all
```


Recover OCR Using Logical Backups

Use the following procedure to import OCR on UNIX-based systems:

1. Identify the OCR export file that you want to import by identifying the OCR export file that you previously created using the `ocrconfig -export` *`file_name`* command.
2. Stop Oracle Clusterware on all the nodes in your RAC database by executing the `crsctl stop crs` command on all the nodes as the `root` user.
3. Perform the import by applying an OCR export file that you identified in step one using the following command, where *`file_name`* is the name of the OCR file from which you want to import OCR information: *`ocrconfig -import file_name`*
4. Restart Oracle Clusterware on all the nodes in your cluster by restarting each node using the `crsctl start crs` command as the `root` user.
5. Run the following Cluster Verification Utility (CVU) command to verify OCR integrity, where the `-n all` argument retrieves a listing of all the cluster nodes that are configured as part of your cluster: `cluvfy comp ocr -n all`

Replace an OCR Mirror: Example

```
# ocrcheck
Status of Oracle Cluster Registry is as follows:
   Version                  :          2
   Total space (kbytes)     :     200692
   Used space (kbytes)      :       3752
   Available space (kbytes) :     196940
   ID                       :  495185602
   Device/File Name         : /oradata/OCR1
    Device/File integrity check succeeded
   Device/File Name         : /oradata/OCR2
    Device/File needs to be synchronized with the other device

# ocrconfig -replace ocrmirror /oradata/OCR2
```


Replace, Add, or Remove an OCR File

The code example in the slide shows you how to replace the existing OCR mirror file. It is assumed that you already have an OCR mirror, and that this mirror is no longer working as expected. Such a reorganization can be triggered because you received an OCR failure alert in Enterprise Manager, or because you saw an alert directly in the Oracle Clusterware alert log file.

Using the `ocrcheck` command, you clearly see that the OCR mirror is no longer in sync with the primary OCR. You then issue the `ocrconfig -replace ocrmirror` *`filename`* command to replace the existing mirror with a copy of your primary OCR. In the example, *`filename`* can be a new file name if you decide to also relocate your OCR mirror file.

If it is the primary OCR file that is failing, and if your OCR mirror is still in good health, you can use the `ocrconfig -replace ocr` *`filename`* command instead.

Note: The example in the slide shows you a replace scenario. However, you can also use a similar command to add or remove either the primary or the mirror OCR file:

- Executing `ocrconfig -replace ocr|ocrmirror` *`filename`* adds the primary or mirror OCR file to your environment if it does not already exist.
- Executing `ocrconfig -replace ocr|ocrmirror` removes the primary or the mirror OCR file.

Repair OCR Configuration: Example

1. **Stop Oracle Clusterware on Node2:**

```
# crsctl stop crs
```

2. **Add OCR mirror from Node1:**

```
# ocrconfig -replace ocrmirror /OCRMirror
```

3. **Repair OCR mirror location on Node2:**

```
# ocrconfig -repair ocrmirror /OCRMirror
```

4. **Start Oracle Clusterware on Node2:**

```
# crsctl start crs
```

ORACLE

Repair OCR Configuration: Example

Use the `ocrconfig -repair` command to repair inconsistent OCR configuration information. The OCR configuration information is stored in:

- `/etc/oracle/ocr.loc` on Linux and AIX
- `/var/opt/oracle/ocr.loc` on Solaris and HP-UX
- Registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Oracle\ocr` on Windows

You may need to repair an OCR configuration on a particular node if your OCR configuration changes while that node is stopped. For example, you may need to repair the OCR on a node that was not up while you were adding, replacing, or removing an OCR.

The example in the slide illustrates the case where the OCR mirror file is added on the first node of your cluster while the second node is not running Oracle Clusterware.

You cannot perform this operation on a node on which Oracle Clusterware is running.

Note: This repairs the OCR configuration information only; it does not repair OCR itself.

OCR Considerations

- If using raw devices to store OCR files, make sure they exist before add or replace operations.
- You must be the `root` user to be able to add, replace, or remove an OCR file while using `ocrconfig`.
- While adding or replacing an OCR file, its mirror needs to be online.
- If you remove a primary OCR file, the mirror OCR file becomes primary.
- Never remove the last remaining OCR file.

Replacing OCR Considerations

Here is a list of important considerations when you use the `ocrconfig -replace` command:

- If you are using raw devices, make sure that the file name exists before issuing an add or replace operation using `ocrconfig`.
- To be able to execute an add, replace, or remove operation using `ocrconfig`, you must be logged in as the `root` user.
- The OCR file that you are replacing can be either online or offline.
- If you remove a primary OCR file, then the mirrored OCR file becomes the primary OCR file.
- Do not perform an OCR removal operation unless there is at least one other active OCR file online.

Change VIP Addresses

1. **Determine the interface used to support your VIP:**

```
$ ifconfig -a
```

2. **Stop all resources depending on the VIP:**

```
$ srvctl stop instance -d DB -i DB1
```

```
$ srvctl stop asm -n node1
```

```
# srvctl stop nodeapps -n node1
```

3. **Verify that the VIP is no longer running:**

```
$ ifconfig -a
```
[+
```
$ crs_stat
```
]

4. **Change IP in `/etc/hosts` and DNS.**

ORACLE

Change VIP Addresses

The VIP address is a static IP address with a virtual host name defined and resolved through either the DNS or your hosts file. During Oracle Clusterware installation, you are prompted to enter a Virtual IP and virtual host name for each of the nodes in the cluster. These are stored in OCR, and different components within the Oracle Clusterware HA framework depend on these VIPs. If, for some reasons, you want to change the VIP address, use the following procedure on each node, one at a time:

1. Confirm the current IP address for the VIP by running the `ifconfig -a` command. On Windows, run the `ipconfig /all` command. This should show you the current VIP bound to one of the network interfaces.
2. Stop all resources that are dependent on the VIP on that node: First, stop the database instance, and then the ASM instance. When done, stop `nodeapps`.
3. Verify that the VIP is no longer running by executing the `ifconfig -a` command again, and confirm that its interface is no longer listed in the output. If the interface still shows as online, this is an indication that a resource which is dependent on the VIP is still running. The `crs_stat -t` command can help to show resources that are still online.
4. Make any changes necessary to all nodes' `/etc/hosts` files (on UNIX), or `\WINNT\System32\drivers\etc\hosts` files on Windows, and make the necessary DNS changes, to associate the new IP address with the old host name.

Change VIP Addresses

5. Modify your VIP address using `srvctl`:

```
# srvctl modify nodeapps -n node1 -A
192.168.2.125/255.255.255.0/eth0
```

6. Start `nodeapps` and all resources depending on it:

```
# srvctl start nodeapps -n node1
```

7. Repeat from step 1 for the next node.

Change VIP Addresses (continued)

5. Modify `nodeapps` and provide the new virtual IP address. Use the `srvctl modify nodeapps` command with the `-A` option. This command should be run as `root`. In the slide example, you specify the new IP address (`192.168.2.125`), then the corresponding netmask (`255.255.255.0`), and the interface that you want the VIP to use (`eth0`).
6. Start `nodeapps` again.
7. Repeat the same steps for all the nodes in the cluster. You can stay connected from the first node because `srvctl` is a clusterwide management tool.

Note: If only the IP address is changed, it is not necessary to make changes to the `listener.ora`, `tnsnames.ora` and initialization parameter files, provided they are using the virtual host names. If changing both the virtual host name and the VIP address for a node, it will be necessary to modify those files with the new virtual host name. For the `listener.ora` file, you can use `netca` to remove the old listener and create a new listener. In addition, changes will need to be made to the `tnsnames.ora` file of any clients connecting to the old virtual host name.

Change Public/Interconnect IP Subnet Configuration: Example

Use `oifcfg` to add or delete network interface information in OCR:

```
$ <CRS HOME>/bin/oifcfg getif                              1
eth0 139.2.156.0 global public
eth1 192.168.0.0 global cluster_interconnect
```

```
$ oifcfg delif -global eth0                                2
$ oifcfg setif -global eth0/139.2.166.0:public
```

```
$ oifcfg delif -global eth1                                3
$ oifcfg setif -global eth1/192.168.1.0:cluster_interconnect
```

```
$ oifcfg getif                                             4
eth0 139.2.166.0 global public
eth1 192.168.1.0 global cluster_interconnect
```

ORACLE

Change Public/Interconnect IP Subnet Configuration

When installing Oracle Clusterware and RAC, it is possible for you to specify wrong information during the OUI interview regarding the public and interconnect interfaces that Oracle Clusterware should use. If that happens, Oracle Clusterware will be able to start at the end of the installation process, but you might end up having trouble later to communicate with other nodes in your cluster. If either the interface, IP subnet, or IP address for both your public network and interconnect are incorrect or need to be changed, you should make the changes using the Oracle Interface Configuration Tool (`oifcfg`) because this will update the corresponding OCR information.

An example is shown in the slide, where both IP subnet for the public and private network are incorrect:

1. You get the current interfaces information by using the `getif` option.
2. You delete the entry corresponding to public interface first by using the `delif` option, and then enter the correct information by using the `setif` option.
3. You do the same for your private interconnect.
4. You check that the new information is correct.

Note: A network interface can be stored as a global interface or as a node-specific interface. An interface is stored as a global interface when all the nodes of a RAC cluster have the same interface connected to the same subnet (recommended). It is stored as a node-specific interface only when there are some nodes in the cluster that have a different set of interfaces and subnets.

Third-Party Application Protection: Overview

- **High Availability framework:**
  - **Command-line tools to register applications with CRS**
  - **Calls control application agents to manage applications**
  - **OCR used to describe CRS attributes for the applications**
- **High Availability C API:**
  - **Modify directly CRS attributes in OCR**
  - **Modify CRS attributes on the fly**
- **Application VIPs:**
  - **Used for applications accessed by network means**
  - **NIC redundancy**
  - **NIC failover**
- **OCFS:**
  - **Store application configuration files**
  - **Share files between cluster nodes**

ORACLE

Third-Party Application Protection: Overview

Oracle Clusterware provides two publicly available components that can be used to help protect any application on a cluster:

- The High Availability framework provides facilities to manage your applications under CRS protection via command-line tools such as `crs_register`, `crs_start`, and `crs_stop`. This framework is also used to automatically invoke control scripts that you created so that CRS can start, stop, and monitor your applications. OCR is used as a repository to define failover policies and other important parameters for CRS to control your applications.
- The C API can be used to directly manipulate OCR to define how CRS should protect an application. This API can be used to modify, at run time, how the application should be managed by CRS. Discussing the C API is out of the scope of this course.

If the application you want CRS to protect is accessed by way of a network, you have the possibility to create a Virtual Internet Protocol address for your application. This is referred to as an application VIP. Application VIPs created by Oracle Clusterware are able to fail over from one network interface card (NIC) to another on the same node as well as from one NIC to another one located on another node in case all public networks are down on a given node.

In addition, your application might need to store configuration files on disk. To share these files among nodes, Oracle Corporation also provides you with the Oracle Cluster File System (OCFS).

Application VIP and RAC VIP Differences

- RAC VIP is mainly used in case of node down events:
  - VIP is failed over to a surviving node.
  - From there it returns NAK to clients forcing them to reconnect.
  - There is no need to fail over resources associated to the VIP.
- Application VIP is mainly used in case of application down events:
  - VIP is failed over to another node together with the application(s).
  - From there, clients can still connect through the VIP.
  - Although not recommended, one VIP can serve many applications.

ORACLE

Application VIP and RAC VIP Differences

Most of the differences between resources attached to application VIPs and RAC VIPs reside in the fact that they are configured differently within Oracle Clusterware. For example, it makes no sense from a RAC perspective to fail over either a database instance or listener because there is already a listener and an instance waiting on another node. Therefore, the listener does not listen on any other VIPs than the one node-specific VIP. Looking at the CRS profile of those resources, you will see the differences. Also, most of the time, there are many applications attached to a RAC VIP such as listeners, database instances, and ASM instances. Although it is possible to associate an application VIP to multiple applications, this is not recommended because if one of the applications cannot be started or restarted on a node, it will be failed over to another node with the VIP, which in turn will force the other applications to be also relocated. This is especially true if the applications are independent. However, one noticeable difference between a RAC VIP and an application VIP is that after a RAC VIP is failed over to a surviving node, it no longer accepts connections (NAK), thus forcing clients that are trying to access that address, to reconnect using another address. If it accepts new connections, then if a failback occurs, after the node is back again, then current connections going through the VIP on the failed-over node are lost because the interface is gone. Application VIPs, on the other side, are fully functional after they are failed over, and continue to accept connections.

RAC VIPs are mainly used when there is a node failure because clients can use other nodes to connect. Application VIPs are mainly used when the application cannot be restarted on a node.

Use CRS Framework: Overview

1. Create an application VIP, *if necessary*:
   a. Create a profile: Network data + `usrvip` predefined script
   b. Register the application VIP.
   c. Set user permissions on the application VIP.
   d. Start the application VIP by using `crs_start`.
2. Write an application action script that accepts three parameters:
   - `start`: Script should start the application.
   - `check`: Script should confirm that the application is up.
   - `stop`: Script should stop the application.

Use CRS Framework: Overview

The slide presents you the basic steps you need to follow to register an application that is monitored by the CRS framework:

1. If your application is accessed via the network, and if you want your application to be still available after some network problems, it is recommended that you create an application VIP for your application.
   a. First, you should create an application profile to define the network information relating to this VIP—for example, the name of the public network adapter to use, the IP address, and the netmask. In the profile, you should also specify the `usrvip` action script provided by Oracle Clusterware. You can then use the default values for the failover policies.
   b. Use the `crs_register` command to add this application VIP to the list of managed applications.
   c. On UNIX-based operating systems, the application VIP script must run as the `root` user. So, using `crs_setperm`, you can change the owner of the VIP to `root`. Using the same command tool, you can also enable another user, such as `oracle`, to start the application VIP.
   d. When done, you can use the `crs_start` command to start the VIP application.
2. You can now create an action script to support the start, check, and stop actions on your application.

Use CRS Framework: Overview

3. Create an application profile:
   - Action script location
   - Check interval
   - Failover policies
   - Application VIP, if necessary
4. Set permissions on your application.
5. Register the profile with Oracle Clusterware.
6. Start your application by using `crs_start`.

Use CRS Framework: Overview (continued)

3. Create the profile for your application. You should use enough resource attributes to define at least the action script location and name, the check interval, the failover policies, and the required application VIP resource (if necessary). You can manage application availability as follows:
   - Specify starting resources during cluster or node startup.
   - Restart applications that fail.
   - Relocate applications to other nodes if they cannot run in their current location.
4. Like for the VIP application, you can define under which user your application should be running as well as which user can start your application. That is why on UNIX-based platforms, Oracle Clusterware must run as the `root` user, and on Windows-based platforms, Oracle Clusterware must run as `Administrator`.
5. When done, you can register your application by using the `crs_register` command.
6. You are then ready to start your application that is going to be monitored by Oracle Clusterware. Do this by executing the `crs_start` command.

Use CRS Framework: Example

```
# crs_profile -create AppVIP1 -t application \
-a <CRS HOME>/bin/usrvip \
-o oi=eth0,ov=144.25.214.49,on=255.255.252.0
```
1

```
# crs_register AppVIP1
```
2

```
# crs_setperm AppVIP1 -o root
```
3

```
# crs_setperm AppVIP1 -u user:oracle:r-x
```
4

```
$ crs_start AppVIP1
```
5

Use CRS Framework: Example

Following the previous overview slides, here is an example that protects the apache application using Oracle Clusterware:

1. You create the AppVP1 application VIP profile by using the crs_profile -create command. In order, here are the parameters specified in the example:
   - Name of the application VIP
   - The application type
   - The predefined action script usrvip located in <*CRS HOME*>/bin
   - The name of the public network adapter, the VIP address used to locate your application regardless of the node it is running on, and the netmask used for the VIP

   The result of this command is to create a text file called AppVIP1.cap in <*CRS HOME*>/crs/profile. This file contains the attributes and is read by crs_register. If your session is not running as the root user, the .cap file is created in <*CRS HOME*>/crs/public.
2. Use the crs_register command to register your application VIP with Oracle Clusterware.
3. On UNIX-based operating systems, the application VIP action script must run as the root user. As the root user, change the owner of the resource as shown using the crs_setperm -o command.
4. As the root user, enable the oracle user to manage your application VIP via CRS commands. Use the crs_setperm -u command.
5. As the oracle user, start the application VIP using the crs_start command.

Use CRS Framework: Example

```
#!/bin/sh

VIPADD=144.25.214.49
HTTDCONFLOC=/etc/httpd/conf/httpd.conf
WEBCHECK=http://$VIPADD:80/icons/apache_pb.gif
case $1 in
'start')
    /usr/bin/apachectl -k start -f $HTTDCONFLOC
    RET=$?
    ;;
'stop')
    /usr/bin/apachectl -k stop
    RET=$?
    ;;
'check')
    /usr/bin/wget -q -delete-after $WEBCHECK
    RET=$?
    ;;
*)
    RET=0
    ;;
esac
exit $RET
```

6

Use CRS Framework: Example (continued)

6. After the application VIP is functional, you can write the action script for your application. The example shown in the slide can be used by Oracle Clusterware as an action script to protect the apache application. It is a shell script that can parse one argument with three different values. It uses the `apachectl` command tool to start and stop the apache application on your node. It uses the `wget` command to check whether a Web page can be accessed. These are the three actions CRS will perform while protecting your application.

For the next steps, it is supposed that this script is called `myApp1.scr`.

Note: Make sure you distribute this script on all nodes of your cluster in the same location. The default location is assumed to be `<CRS HOME>/crs/script` in this case.

Use CRS Framework: Example

```
# crs_profile -create myApp1 -t application -r AppVIP1 \
-a myapp1.scr -o ci=5,ra=2
```
7

```
# crs_register myApp1
```
8

```
# crs_setperm myApp1 -o root
```
9

```
# crs_setperm myApp1 -u user:oracle:r-x
```
10

```
$ crs_start myApp1
```
11

ORACLE

Use CRS Framework: Example (continued)

7. You can now create a profile for your application. Here your resource is called myApp1. It uses `myApp1.scr` as its action script and depends on the AppVIP1 application. If AppVIP1 fails or if it is relocated to another node, then Oracle Clusterware stops or moves the myApp1 application. The example also defines its check interval to be five seconds, and the number of attempts to restart the application to 2. This means that Oracle Clusterware will fail over the application to another node after a second local failure happens.
8. The `crs_register` command registers myApp1 with Oracle Clusterware.
9. Because you want the apache server listening on the default port 80, you want the application to execute as the `root` user. As the `root` user, change the owner of the resource, as shown, using the `crs_setperm -o` command.
10. As the `root` user, enable the `oracle` user to manage your application VIP via CRS commands. Use the `crs_setperm -u` command.
11. As the `oracle` user, start myApp1 by using the `crs_start` command.

Summary

In this lesson, you should have learned how to:

- Manually control the Oracle Clusterware stack
- Change voting disk configuration
- Backup and recover your voting disks
- Manually back up OCR
- Recover OCR
- Replace an OCR mirror
- Repair the OCR configuration
- Change VIP addresses
- Use the CRS framework
- Prevent automatic instance restarts

Practice 9: Overview

This practice covers the following topics:

- Mirroring the OCR
- Backing up and restoring OCR
- Multiplexing the voting disk
- Using Oracle Clusterware to protect the *xclock* application

10

Diagnosing Oracle Clusterware and RAC Components

ORACLE

Objectives

After completing this lesson, you should be able to:

- Collect Oracle Clusterware diagnostic files
- Use Cluster Verify

ORACLE

One Golden Rule in RAC Debugging

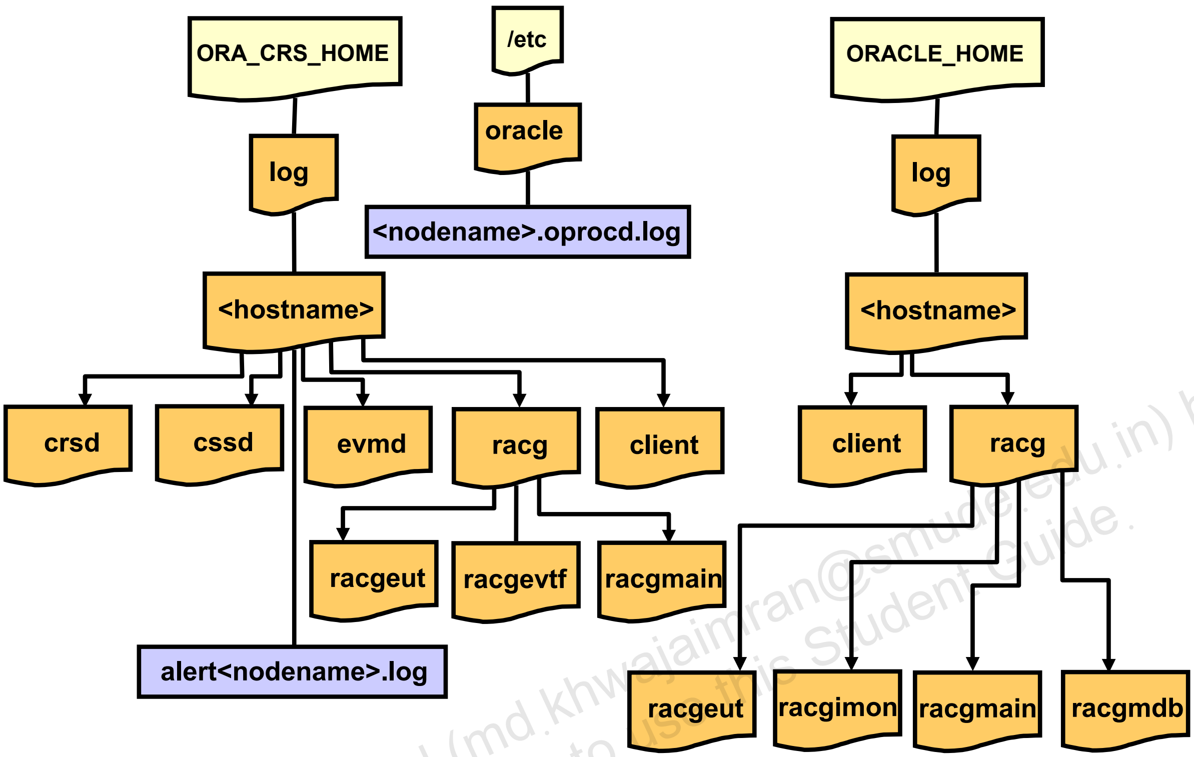
- Always make sure that your nodes have exactly the same system time to:
  - Facilitate log information analysis
  - Ensure accurate results when reading `GV$` views
  - Avoid untimely instance evictions
- The best recommendation is to synchronize nodes using Network Time Protocol.

One Golden Rule in RAC Debugging

It is strongly recommended to set up Network Time Protocol (NTP) on all cluster nodes, even before you install RAC. This will synchronize the clocks among all nodes, and facilitate analysis of tracing information based on time stamps as well as results from queries issued on `GV$` views.

Note: Adjusting clocks by more than 15 minutes can cause instance evictions. It is strongly advised to shut down all instances before date/time adjustments.

Oracle Clusterware Main Log Files



Oracle Clusterware Main Log Files

Oracle Clusterware uses a unified log directory structure to consolidate the Oracle Clusterware component log files. This consolidated structure simplifies diagnostic information collection and assists during data retrieval and problem analysis.

The slide shows you the main directories used by Oracle Clusterware to store its log files:

- CRS logs are in `$ORA_CRS_HOME/log/<hostname>/crsd/`. The `crsd.log` file is archived every 10 MB (`crsd.l01`, `crsd.l02`, …).
- CSS logs are in `$ORA_CRS_HOME/log/<hostname>/cssd/`. The `cssd.log` file is archived every 20 MB (`cssd.l01`, `cssd.l02`, …).
- EVM logs are in `$ORA_CRS_HOME/log/<hostname>/evmd`.
- Depending on the resource, specific logs are in `$ORA_CRS_HOME/log/<hostname>/racg` and in `$ORACLE_HOME/log/<hostname>/racg`. In the last directory, `imon_<service>.log` is archived every 10 MB for each service. Each RACG executable has a subdirectory assigned exclusively for that executable. The name of the RACG executable subdirectory is the same as the name of the executable.
- SRVM (`srvctl`) and OCR (`ocrdump`, `ocrconfig`, `ocrcheck`) logs are in `$ORA_CRS_HOME/log/<hostname>/client/` and in `$ORACLE_HOME/log/<hostname>/client/`.
- Important Oracle Clusterware alerts can be found in `alert<nodename>.log` in the `$ORA_CRS_HOME/log/<hostname>` directory.

Diagnostics Collection Script

- **Script to collect all important log files:**
  - **Must be executed as** `root`
  - **Is located in** `$ORA_CRS_HOME/bin/`
  - **Is called** `diagcollection.pl`
- **Generates the following files in the local directory:**
  - `basData_<hostname>.tar.gz`
  - `crsData _<hostname>. tar.gz`
  - `ocrData _<hostname>. tar.gz`
  - `oraData _<hostname>. tar.gz`

```
# export ORACLE_HOME=/u01/app/oracle/product/10.2.0/db_1
# export ORA_CRS_HOME=/u01/crs1020
# export ORACLE_BASE=/u01/app/oracle
# cd $ORA_CRS_HOME/bin
# ./diagcollection.pl -collect
```

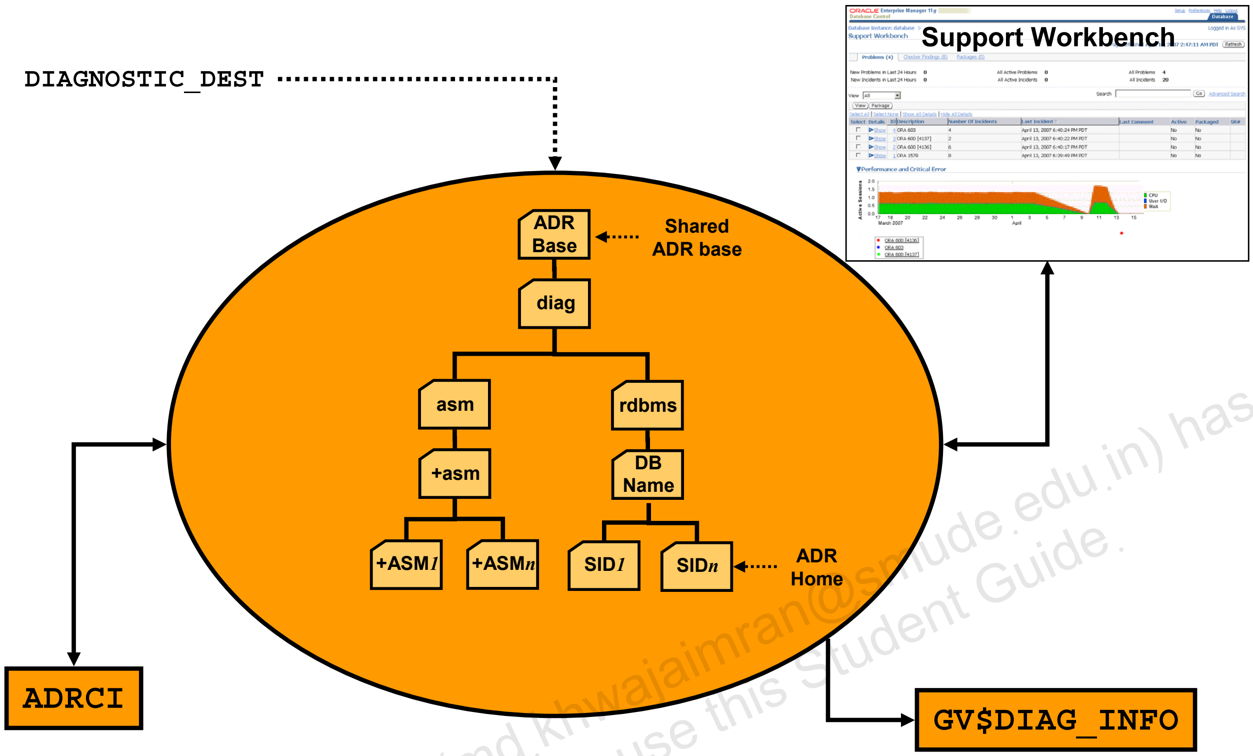
ORACLE

Diagnostics Collection Script

Use the `diagcollection.pl` script to collect diagnostic information from an Oracle Clusterware installation. The diagnostics provide additional information so that Oracle Support can resolve problems. This script is located in `$ORA_CRS_HOME/bin`. Before executing the script, you must be logged in as the `root` user, and you must set the following environment variables: `ORACLE_BASE`, `ORACLE_HOME`, `ORA_CRS_HOME`, `HOSTNAME`. The example in the slide shows you how to invoke the script to collect the diagnostic information. When invoked with the `-collect` option, the script generates, in the local directory, the four files mentioned in the slide. Mainly, `basData.tar.gz` contains log files from the `$ORACLE_BASE/admin` directory. `crsData.tar.gz` contains log files from `$ORA_CRS_HOME/log/<hostname>`. The `ocrData.tar.gz` files contain the results of an ocrdump, ocrcheck, and the list of ocr backups. `oraData.tar.gz` contains log files from `$ORACLE_HOME/log/<hostname>`.

If you invoke the script with the `-collect` option, and you already have the four files generated from a previous run in the local directory, the script asks you if you want to overwrite the existing files. You can also invoke the script with the `-clean` option to clean out the files generated from a previous run in your local directory. Alternatively, you can invoke the script to just capture a subset of the log files. You can do so by adding extra options after the `-collect` option: `-crs` for collecting Oracle Clusterware logs, `-oh` for collecting `ORACLE_HOME` logs, `-ob` for collecting `ORACLE_BASE` logs, or `-all` for collecting all logs. The `-all` option is the default. The `-coreanalyze` option enables you to extract to text files only core files found in the generated files.

Managing Diagnostic Data in RAC



Managing Diagnostic Data in RAC

Problems that span Oracle RAC instances can be the most difficult types of problems to diagnose. For example, you may need to correlate the trace files from across multiple instances, and merge the trace files. Oracle Database Release 11*g* includes an advanced fault diagnosibility infrastructure for collecting and managing diagnostic data, and uses the Automatic Diagnostic Repository (ADR) file-based repository for storing the database diagnostic data. When you create the ADR base on a shared disk, you can place ADR homes for all instances of the same Oracle RAC database and the all corresponding ASM instances under the same ADR Base. With shared storage, you can use the ADRCI command-line tool to correlate diagnostics across all instances because some ADRCI commands (such as `SHOW INCIDENT`) can work with multiple ADR homes simultaneously.

Note: Although not required, it is recommended that you share ADR base with your RAC databases. However, if you are using shared Oracle homes, you must share your ADR base.

Cluster Verify: Overview

- To verify that you have a well-formed cluster for Oracle Clusterware and RAC:
  – Installation
  – Configuration
  – Operation
- Full stack verification
- Nonintrusive verification
- Diagnostic mode seeks to establish a reason for the failure of any verification task.
- Easy-to-use interface:
  – Stage commands
  – Component commands

ORACLE

Cluster Verify: Overview

Cluster Verification Utility (CVU) is provided with Oracle Clusterware and Oracle Database 10*g* Release 2 (and later) with Real Application Clusters. The purpose of CVU is to enable you to verify during setup and configuration that all components required for a successful installation of Oracle Clusterware or Oracle Clusterware and a RAC database are installed and configured correctly, and to provide you with ongoing assistance any time you need to make changes to your RAC cluster.

There are two types of CVU commands:

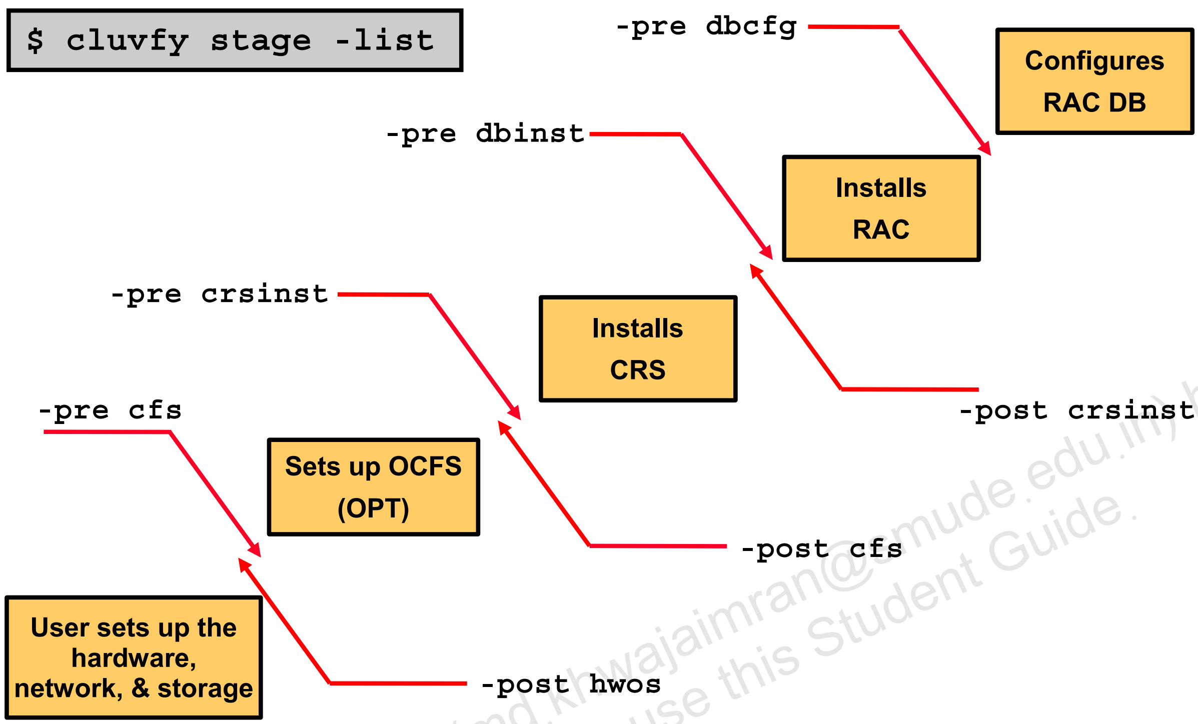
- Stage commands are CVU commands used to test system setup and readiness for successful software installation, database creation, or configuration change steps. These commands are also used to validate successful completion of specific cluster configuration steps.
- Component commands are CVU commands used to check individual cluster components, and determine their state.

It is recommended to use stage checks during the installation of Oracle Clusterware and RAC.

In addition, you can use CVU to verify a particular component while the stack is running or to isolate a cluster subsystem for diagnosis. During the diagnostic mode of operation, CVU tries to establish a reason for the failure of any verification task to help diagnose a problem.

Note: CVU is a nonintrusive tool in the sense that it does not try to fix any issues it finds.

Cluster Verify Stages



Cluster Verify Stages

A stage is a specific phase of an Oracle Clusterware or RAC deployment. Before performing any operations in a stage, a predefined set of checks must be performed to ensure the readiness of cluster for that stage. These checks are known as "pre" checks for that stage. Similarly, a predefined set of checks must be performed after completion of a stage to ensure the correct execution of operations within that stage. These checks are known as "post" checks for that stage. You can list verifiable stages with the `cluvfy stage -list` command. All stages have pre or post steps and some stages have both. Valid stage options and stage names are:

- **`-post hwos`:** Postcheck for hardware and operating system
- **`-pre cfs`:** Precheck for CFS setup
- **`-post cfs`:** Postcheck for CFS setup
- **`-pre crsinst`:** Precheck for CRS installation
- **`-post crsinst`:** Postcheck for CRS installation
- **`-pre dbinst`:** Precheck for database installation
- **`-pre dbcfg`:** Precheck for database configuration

Cluster Verify Components

- An individual subsystem or a module of the RAC cluster is known as a component in CVU.
- The availability and integrity of a cluster component can be verified.
- Components can be simple like a specific storage device, or complex like the Oracle Clusterware stack:
  - Space availability
  - Shared storage accessibility
  - Node connectivity
  - Cluster File System integrity
  - Oracle Clusterware integrity
  - Cluster integrity
  - Administrative privileges
  - Peer compatibility
  - System requirements

```
$ cluvfy comp -list
```

ORACLE

Cluster Verify Components

CVU supports the notion of component verification. The verifications in this category are not associated with any specific stage. A component can range from a basic one, such as free disk space, to a complex one (spanning over multiple subcomponents), such as the Oracle Clusterware stack. Availability, integrity, or any other specific behavior of a cluster component can be verified. You can list verifiable CVU components with the `cluvfy comp -list` command.

- **`nodereach:`** Checks reachability between nodes
- **`nodecon:`** Checks node connectivity
- **`cfs:`** Checks Oracle Cluster File System integrity (The sharedness check for the file system is supported for OCFS2 versions 1.2.1 and later.)
- **`ssa:`** Checks shared storage accessibility
- **`space:`** Checks space availability
- **`sys:`** Checks minimum system requirements
- **`clu:`** Checks cluster integrity
- **`clumgr:`** Checks cluster manager integrity
- **`ocr:`** Checks OCR integrity
- **`crs:`** Checks CRS integrity
- **`nodeapp:`** Checks node applications existence
- **`admprv:`** Checks administrative privileges
- **`peer:`** Compares properties with peers

Cluster Verify Locations

- **Download it from OTN:**
  - **Create a local directory.**
  - **Copy and extract `cvu_<OS>.zip`.**
- **Oracle software DVD:**
  - **`Disk1` directory**
  - **`runcluvfy.sh`**
- **Oracle Clusterware home:**
  - **`$ORA_CRS_HOME/bin/cluvfy`**
- **Oracle Home:**
  - **`$ORACLE_HOME/bin/cluvfy`**

ORACLE

Cluster Verify Locations

The Cluster Verification Utility (CVU) is first released in Oracle Clusterware release 10.2.0.1.0. CVU supports 11*g*R1, 10*g*R2 as well as 10*g*R1 for Oracle Clusterware and RAC products. CVU is available in three different forms:

- Available on Oracle Technology Network (OTN) at: http://www.oracle.com/technology/products/database/clustering/cvu/cvu_download_homepage.html
  From there, you need to download the package and unzip it to a local directory (`<cvhome>`). You can use the `cluvfy` command from the `<cvhome>/bin`. Optionally, you can set the `CV_DESTLOC` environment variable. This should point to a writable area on all nodes. CVU attempts to copy the necessary bits as required to this location. If this variable is not set, CVU uses `/tmp` as the default.
- Available in 11.1 Oracle software DVD as packaged version. Make use of `runcluvfy.sh`, which is needed when nothing is installed. You can find it in `Disk1`.
- Installed in both 11.1 Oracle Clusterware and RAC homes. Make use of `cluvfy` if the CRS software stack is installed. If the CRS software is installed, you can find `cluvfy` under `$ORA_CRS_HOME/bin`.

Note: For manual installation, you need to install CVU on only one node. CVU deploys itself on remote nodes during executions that require access to remote nodes.

Cluster Verify Configuration File

```
$ cat cvu_config
# Configuration file for CVU
# Version: 011405
#

#CV_ORACLE_RELEASE=11gR1

#CV_NODE_ALL=

CV_RAW_CHECK_ENABLED=TRUE

CV_ASSUME_DISTID=RHEL4

#CV_XCHK_FOR_SSH_ENABLED=TRUE

#ORACLE_SRVM_REMOTESHELL=/usr/bin/ssh

#ORACLE_SRVM_REMOTECOPY=/usr/bin/scp
```


Cluster Verify Configuration File

You can use the CVU's configuration file to define specific inputs for the execution of the CVU. The path for the configuration file is `$CV_HOME/cv/admin/cvu_config`. The following is the list of keys supported in `cvu_config`:

- **`CV_NODE_ALL`:** If set, it specifies the list of nodes that should be picked up when Oracle Clusterware is not installed and the `-n all` option has been used in the command line.
- **`CV_RAW_CHECK_ENABLED`:** If set to `TRUE`, it enables the check for accessibility of shared SCSI disks on Red Hat release 3.0 and higher. This shared disk accessibility check requires that you install a cvuqdisk rpm on all the nodes. By default, this key is set to `TRUE` and shared disk check is enabled.
- **`CV_ASSUME_DISTID`:** Specifies the distribution ID that CVU uses. For example, to make CVU working with SuSE 9 ES, set it to `Pensacola`.
- **`CV_XCHK_FOR_SSH_ENABLED`:** If set to `TRUE`, it enables the X-Windows check for verifying user equivalence with `ssh`. By default, this entry is commented out and X-Windows check is disabled.
- **`ORACLE_SRVM_REMOTESHELL`:** If set, it specifies the location for the `ssh/rsh` command to override CVU's default value. By default, this entry is commented out and the tool uses `/usr/sbin/ssh` and `/usr/sbin/rsh`.

Note: If CVU does not find a key entry defined in the configuration file, the CVU searches for the environment variable that matches the name of the key; otherwise, the CVU uses a default.

Cluster Verify Configuration File (continued)

- **ORACLE_SRVM_REMOTECOPY:** If set, it specifies the location for the scp or rcp command to override the CVU default value. By default, this entry is commented out and the CVU uses /usr/bin/scp and /usr/sbin/rcp.

If the CVU does not find a key entry defined in the configuration file, the CVU searches for the environment variable that matches the name of the key. If the environment variable is set, the CVU uses its value. Otherwise it uses a default value for that entity.

To provide the CVU with a list of all the nodes of a cluster, you can use the -n all option while executing a command. The CVU attempts to obtain the node list in the following sequence:

1. If vendor clusterware is available, the CVU selects all the configured nodes from the vendor clusterware using the lsnodes utility.
2. If Oracle Clusterware is installed, the CVU selects all the configured nodes from Oracle Clusterware using the olsnodes utility.
3. If neither the vendor nor Oracle Clusterware is installed, the CVU searches for a value for the CV_NODE_ALL key in the configuration file.

If the vendor and Oracle Clusterware are not installed and if no key named CV_NODE_ALL exists in the configuration file, the CVU searches for a value for the CV_NODE_ALL environmental variable. If you have not set this variable, the CVU reports an error.

Cluster Verify: Examples

```
$ cluvfy comp sys -n node1,node2 -p crs -verbose                1
$ cluvfy comp ssa -n all -s /dev/sda1                           2
$ cluvfy comp space -n all -l /home/product -z 5G               3
$ cluvfy comp nodereach -n node2 -srcnode node1                 4
$ cluvfy comp nodecon -n node1,node2 –i eth0 -verbose           5
$ cluvfy comp admprv -n all -o user_equiv -verbose              6
$ cluvfy comp nodeapp -n all -verbose                           7
$ cluvfy comp peer -n all -verbose | more                       8
```

ORACLE

Cluster Verify: Examples

The slide shows you some possible interesting examples:

1. To verify the minimal system requirements on the nodes before installing Oracle Clusterware or RAC, use the `sys` component verification command. To check the system requirements for installing RAC, use the `-p database` argument, and to check the system requirements for installing Oracle Clusterware, use the `-p crs` argument. To check the system requirements for installing Oracle Clusterware or RAC from Oracle Database 10*g* release 1 (10.1), use the `-r 10gR1` argument. The example verifies the system requirements for installing Oracle Clusterware on the cluster nodes known as `node1` and `node2`.
2. To verify whether storage is shared among the nodes in your cluster database or to identify all of the storage that is available on the system and can be shared across the cluster nodes, use the component verification command `ssa`. The example uses the `–s` option to specify the path to check.
3. You are planning to install more software on the local `/home/product` file system of each node in the cluster, and that software will take up 5 GB on each node. This command is successful if 5 GB is available in /home/product of every node; otherwise, it fails.

Note: The `-verbose` option can be used with any command. It basically gives you more information in the output.

Cluster Verify: Examples (continued)

4. To verify the reachability of the cluster nodes from the local node or from any other cluster node, use the component verification command `nodereach`. The example tries to check whether `node2` can be reached from `node1`.
5. To verify the connectivity between the cluster nodes through all of the available network interfaces or through specific network interfaces, use the component verification command `nodecon`. The example checks whether `node1` and `node2` can communicate through the `eth0` network interface. Without the `-i` option, the CVU discovers all the network interfaces that are available on the cluster nodes, reviews the interfaces' corresponding IP addresses and subnets, obtains the list of interfaces that are suitable for use as VIPs and the list of interfaces suitable for use as private interconnects, and verifies the connectivity between all the nodes through those interfaces.
6. To verify user accounts and administrative permissions–related issues for user equivalence, Oracle Clusterware installation, and RAC installation, use the component verification command `admprv`. On Linux and UNIX platforms, the example verifies user equivalence for all the nodes by first using `ssh` and then using `rsh` if the `ssh` check fails. To verify the equivalence only through `ssh`, use the `-sshonly` option. By default, the equivalence check does not verify X-Windows configurations, such as when you have disabled X-forwarding with the setting of the `DISPLAY` environment variable. To verify X-Windows aspects during user equivalence checks, set the `CV_XCHK_FOR_SSH_ENABLED` key to `TRUE` in the configuration file before you run the command. Use the `-o crs_inst` argument to verify whether you have permissions to install Oracle Clusterware. You can use the `-o db_inst` argument to verify the permissions that are required for installing RAC and the `-o db_config` argument to verify the permissions that are required for creating a RAC database or for modifying a RAC database's configuration.
7. The example verifies the existence of node applications, namely VIP, ONS, and GSD, on all the nodes. To verify the integrity of all the Oracle Clusterware components, use the component verification `crs` command. To verify the integrity of each individual Cluster Manager subcomponent (CSS), use the component verification command `clumgr`. To verify the integrity of Oracle Cluster Registry, use the component verification `ocr` command. To check the integrity of your entire cluster, which means to verify that all the nodes in the cluster have the same view of the cluster configuration, use the component verification `clu` command.
8. The example compares all the nodes and determines whether any differences exist between the values of preselected properties. This is successful if the same setup is found across all the nodes. You can also use the `comp peer` command with the `-refnode` option to compare the properties of other nodes against the reference node. This command allows you to specify the `–r 10gR1` option. Here is a truncated list of the preselected properties: `Total memory`, `Swap space`, `Kernel version`, `System architecture`, `Package existence` for various components (`glibc`, `make`, `binutils`, `gcc`, `compat-db`, ...), `Group existence for "oinstall"`, `Group existence for "dba"`, `User existence for "nobody"`.

Note: For stage examples, refer to the installation lessons in this course.

Cluster Verify Output: Example

```
$ cluvfy comp crs -n all -verbose

Verifying CRS integrity
Checking CRS integrity...
Checking daemon liveness...
…
Liveness of all the daemons
  Node Name     CRS daemon     CSS daemon       EVM daemon
  ------------  -------------  ---------------  ----------
  atlhp9        yes            yes              yes
  atlhp8        yes            yes              yes
Checking CRS health...
Check: Health of CRS
  Node Name                            CRS OK?
  ----------------------------------  ------------------------
  atlhp9                               yes
  atlhp8                               yes
Result: CRS health check passed.
CRS integrity check passed.
Verification of CRS integrity was successful.
```


Cluster Verify Output: Example

The slide shows you the output of the `cluvfy comp crs -n all -verbose` command. This command checks the complete Oracle Clusterware stack.

Note: The output is truncated for formatting reasons.

Summary

In this lesson, you should have learned how to:

- Collect Oracle Clusterware diagnostic files
- Use Cluster Verify

Practice 10: Overview

This practice covers the following topics:

- Identifying Oracle Clusterware log files
- Fixing voting disk corruptions