Oracle 10g Real Application Clusters for Administrators

Student Workbook

itcourseware
Training Materials For IT Professionals

Oracle 10g RAC for Administrators

Published by ITCourseware, LLC., 7245 South Havana Street, Suite 100, Centennial, CO 80112

Contributing Authors: John Mullins and Rob Roselius

Technical Editor: Danielle Hopkins

Editors: Danielle North and Jan Waleri

Special thanks to: Many instructors whose ideas and careful review have contributed to the quality of this workbook, including Jef Barnhart and John McAlister, and the many students who have offered comments, suggestions, criticisms, and insights.

Contents

CHAPTER 1 - COURSE INTRODUCTION

Course Objectives

- Explain the architecture of a RAC system.
- Install and configure Oracle Clusterware and Oracle Database Software in a RAC Environment.
- Add, remove, backup, and recover voting disks and the OCR.
- Correctly set RAC initialization parameters.
- Start and stop individual RAC instances and an entire RAC database.
- Manage redo log files and undo tablespaces for a RAC database.
- Configure and Manage Services in a RAC Environment.
- Configure Transparent Application Failover for client load balancing, connect-time, and application failover.
- Configure a RAC database for archiving redo logs and flashback database mode.
- Backup and recover a RAC database and its components.
- Prepare, configure, remove, and add nodes to an existing cluster.
- Set up and manage an ASM instance.
- Migrate a RAC database to ASM.
- Locate and interpret the contents of critical RAC alert and RAC component log files.
- Monitor and diagnose performance issues with built-in utilities, views, reports, and the OEM.

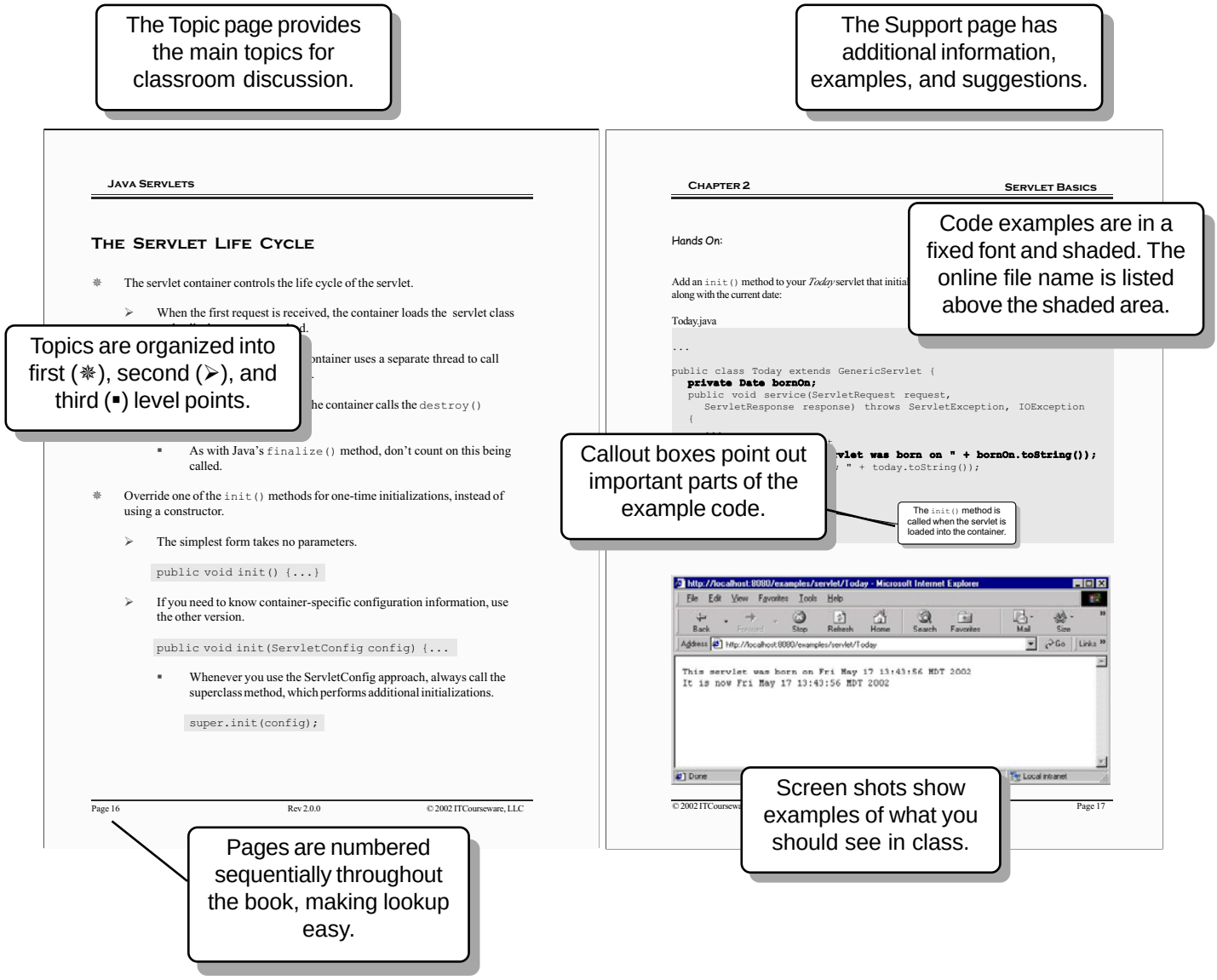
COURSE OVERVIEW

- **Audience**: Database administrators new to the RAC environment.
- **Prerequisites**: *Oracle 10g Database Administration* and at least 6 months of administration experience recommended.
- **Classroom Environment:**
  - A workstation per student.

USING THE WORKBOOK

This workbook design is based on a page-pair, consisting of a Topic page and a Support page. When you lay the workbook open flat, the Topic page is on the left and the Support page is on the right. The Topic page contains the points to be discussed in class. The Support page has code examples, diagrams, screen shots and additional information. *Hands On* sections provide opportunities for practical application of key concepts. Try It and Investigate sections help direct individual discovery.

In addition, there is an index for quick lookup. Printed lab solutions are in the back of the book as well as online if you need a little help.



SUGGESTED REFERENCES

Dyke, Julian and Shaw, Steve. 2006. ***Pro Oracle Database 10g RAC on Linux: Installation, Administration, and Performance***. Apress, Berkeley, CA. ISBN 1590595246.

Gopalakrishnan, K. 2006. ***Oracle Database 10g Real Application Clusters Handbook***. McGraw Hill Osborne Media, Emeryville, CA. ISBN 007146509X.

Vallath, Murali. 2006. ***Oracle 10g RAC Grid, Services & Clustering***. Elsevier Digital Press, Burlington, MA. ISBN 1555583210.

The following books are available online at ***http://www.oracle.com/pls/db102/homepage*** :

- ***Oracle Database Administrator's Guide 10g Release 2 (10.2)***
- ***Oracle Database 2 Day + Real Application Clusters Guide 10g Release 2 (10.2)***
- ***Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment Guide***
- ***Oracle Clusterware and Oracle Real Application Clusters Installation Guide for Linux***

CHAPTER 2 - CLUSTER ARCHITECTURE

OBJECTIVES

- Explain the general architecture of a cluster.
- Identify the locations of the various file types within a cluster.
- List the different shared storage file system options.

CLUSTER ARCHITECTURE

- A *cluster* consists of multiple interconnected hosts (*nodes*), that appear to users and applications as if they are one host.
  - The nodes maintain close communication with one another, typically over high-speed LAN connections.
  - Typically, the nodes use shared storage, such as a Storage Area Network (SAN), Network Attached Storage (NAS), or other devices.
- Software, often called *clusterware*, running with operating system privileges on each node, coordinates activities in the cluster.
  - Tasks of the clusterware include:
    - Identifying which nodes are currently members, and which might need to be removed (*evicted*) from cluster activities.
    - Monitoring application software (database servers, application servers, other services), and stopping them, starting them, or moving their work to another node (*failing over*) as necessary.
  - The clustering software may use shared storage to record its own state and other information.
- Clustering can provide one or more of several benefits:
  - High availability — Failure of one or more nodes, and the bringing up of their replacements, is hidden from the user and the overall cluster remains available.
  - Scalability — More nodes can be brought online to meet growing user demand.
  - Load balancing — New requests are distributed equally (or at least intelligently) among available nodes.
  - High performance — Portions of large tasks can be parcelled out to separate nodes to be executed in parallel.

In a cluster:
- Each node is in communication with all other nodes.
- All nodes agree on which nodes are valid and working.

Clusterware on each node provides *heartbeat* information, visible to all other nodes, at frequent intervals. When other members don't see a node's heartbeat, it can mean:
- The node has crashed
- The node's communication link has failed.
- The node is hung.

In this case, the remaining nodes may vote to evict the node from the cluster. Remaining nodes will then take over the failed node's provided services as well as the shared resources it controlled.

If multiple nodes lose contact with one another, but remain running, each may think it is the only remaining node. Applications on the separated nodes could write to shared resources simultaneously, each assuming it's safe to do so. Either the clusterware or the application software must therefore be designed to quickly detect and resolve this *split-brain* situation.

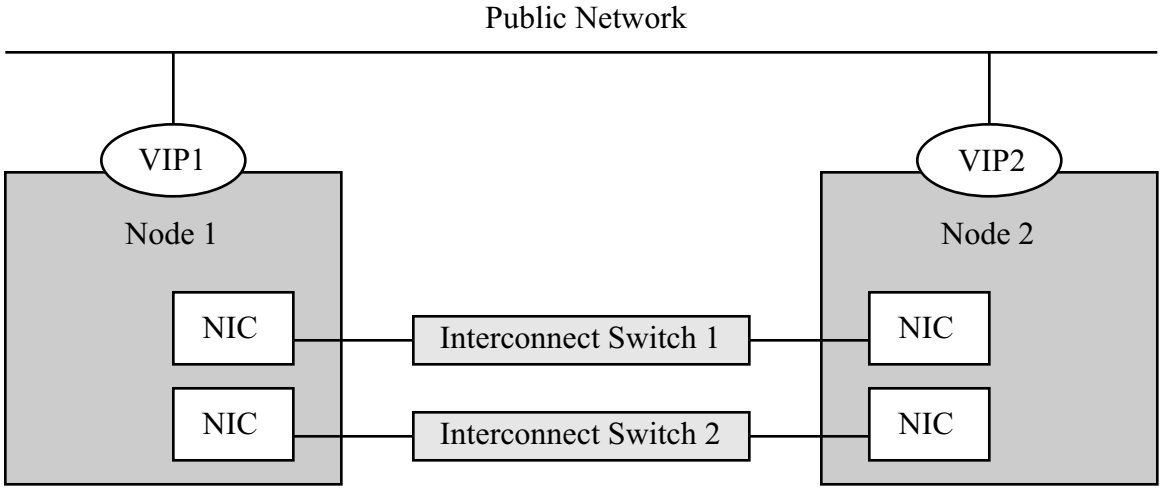
SHARED STORAGE

- The nodes in a cluster need shared access to data in various files.
  - For example, in a RAC Database, at a minimum, shared storage is needed for:
    - The Oracle Cluster Registry (OCR)
    - The voting disk
    - The database files
  - The Oracle RDBMS software can be installed on shared storage or locally on each node.
- Following are shared storage file system options and the types of files that are valid on each:
  - Oracle Cluster File System (OCFS V1, V2)
    - Oracle RDBMS software (required for OCFS V2 only)
    - The Oracle Cluster Registry (OCR)
    - The voting disk
    - Database files
    - **SPFILE**
  - Automatic Storage Management (ASM)
    - Database files
  - Raw Devices
    - The Oracle Cluster Registry (OCR)
    - The voting disk
    - Database files

NODES AND INTERCONNECTS

- To use the cluster, a client application connects to one of the nodes using a public IP address for the node, on a public network.
- The clusterware instances on the nodes, and applications running on the cluster, communicate with each other using a *private interconnect*, sometimes called the *cluster interconnect*.
- The private interconnect is a very high-speed link that allows processes on different nodes to interact as one system.
- Many cluster architectures use gigabit (or faster) ethernet for the private interconnect.
  - Other implementations include, among others:
    - HP Hyperfabric
    - HP Memory Channel
    - Sun Scalable Coherent Interface (SCI)
    - Veritas LLT
- If a node's private interconnect fails, it can no longer function as part of the cluster.
  - Clusterware on it and the other nodes will fence it out (prevent output from the node onto shared storage), evict the node from the cluster, and possibly initiate a reboot of the node.
- If the entire private network fails (say, the gigabit switch goes down), the clusterware on the nodes must detect and immediately resolve the potential split-brain situation, where several nodes each think they're the only surviving member of the cluster.
- Fully redundant private interconnects are a wise choice in any production cluster, and can be accomplished through:
  - Multiple network interface cards for the private network on each node.
  - Plugging each node in to multiple network switches, with switches on different circuits and power supplies.

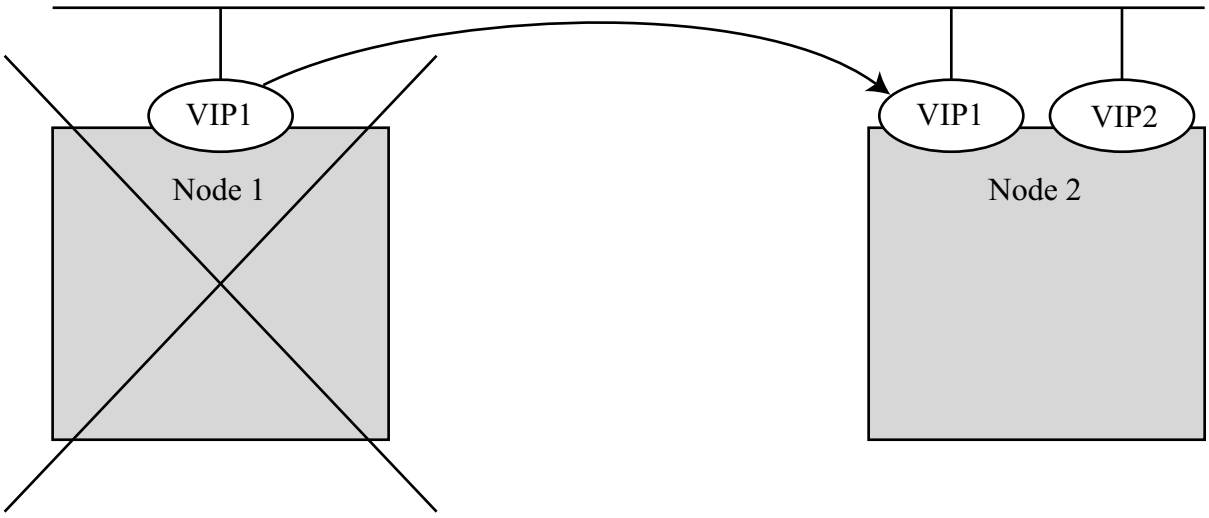
Two Node Cluster with Redundant Interconnects



In an Oracle RAC installation, the private interconnect is used both by the clusterware itself (for cluster synchronization, membership services, etc.), and by the RAC database instances (for exchanging data blocks, managing locks, etc.). High bandwidth and low latency are important for the best performance of the RAC database. On many systems, multiple network interface cards (NIC) can be bonded, creating a single logical interface of higher bandwidth. Some of these solutions provide load balancing across the NICs, and may include automatic failover if one of the NICs fails.

VIRTUAL IP ADDRESSES

- To use a service on a cluster, a client application connects to one of the nodes.
  - The client may first connect to a dispatcher program, which can determine (on the basis of load balancing needs, or attributes of the client, for example) the node to which the client should connect.
    - Oracle's **TNSLISTENER** serves this function in an Oracle RAC cluster.
- Once "connected," the client sends TCP/IP packets to the node's IP address.
  - The lower-level ARP protocol associates this IP address with the physical MAC address of the node's network interface card.
  - If the node is down (crashed, hung, or disconnected from the public network), a client normally waits a specified amount of time (possibly several minutes) before giving up and generating an error.
- Virtual IP addresses allow for quicker failover when a node goes down.
  - Each node has a second, "virtual" IP address (VIP) and hostname configured for it.
  - Clients connect to the virtual IP address, rather than the direct IP address.
  - When a node goes down, the clusterware detects the failure almost immediately — much more quickly than the TCP/IP timeout.
    - A different node is assigned the failed node's VIP, and broadcasts its hardware MAC address and the VIP to the network.
    - The client will receive a TCP/IP **RESET** message, indicating immediately that the old node is gone, and can connect to the new node.
- Applications can be written in such a way as to gracefully handle the TCP/IP **RESET**, and continue working.



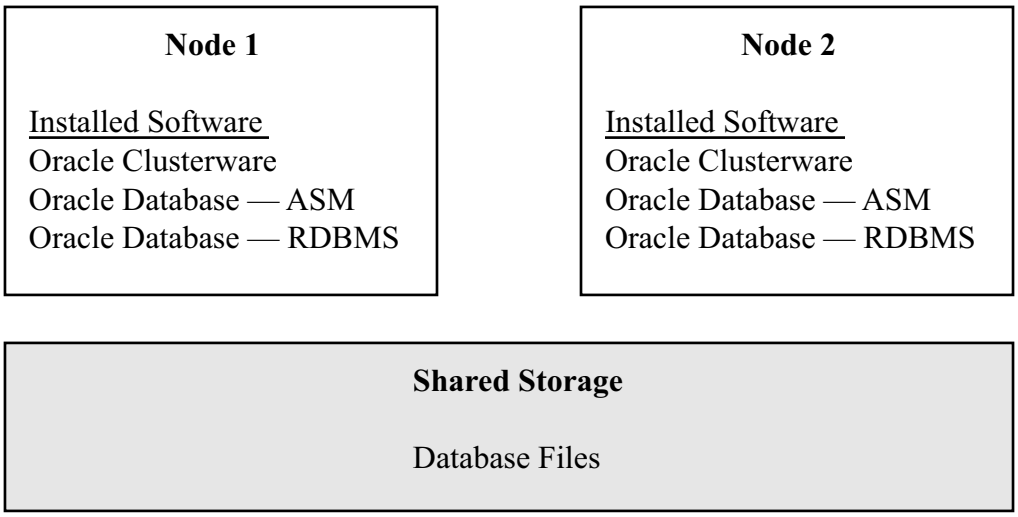
Node 1 has experienced a failure, causing its VIP address to transfer over to Node 2.

Users attempting to connect to Node 1 will be redirected to Node 2.

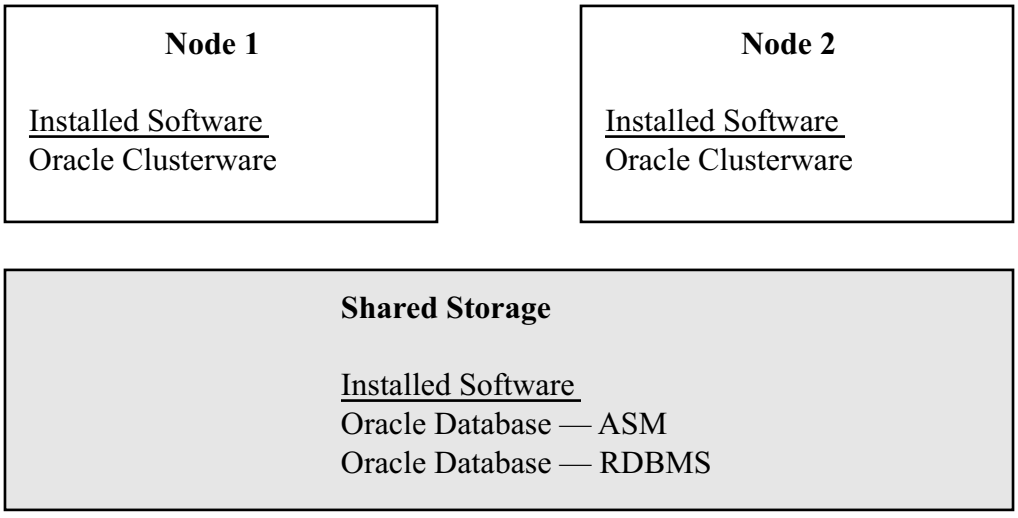
ORACLE SOFTWARE

- There are two, and optionally a third, software components involved in configuring a RAC environment.
  - The three software components are:
    - Oracle Clusterware (Required)
    - Oracle Database Software for the RDBMS (Required)
    - Oracle Database Software for ASM (Optional)
- In Oracle 10*g* R2 and later, the Oracle Clusterware software must be installed locally on each node in the cluster.
- The Oracle Database software for ASM or the RDBMS can either be installed on each node or on the shared storage device.
  - The database software for ASM and RDBMS is actually the same software.
- There are pros and cons for installing the Oracle Database Software on each node or on a shared storage device.
  - Installing on each node increases maintenance because files must be copied to each node.
  - Installing on each node allows for increased flexibility when applying patches or upgrades, as one node at a time can be patched while the other nodes continue to function.
    - However, if installed on a shared storage device, all nodes would need to be stopped at once to apply the patch.

Oracle Software File Locations — Option 1

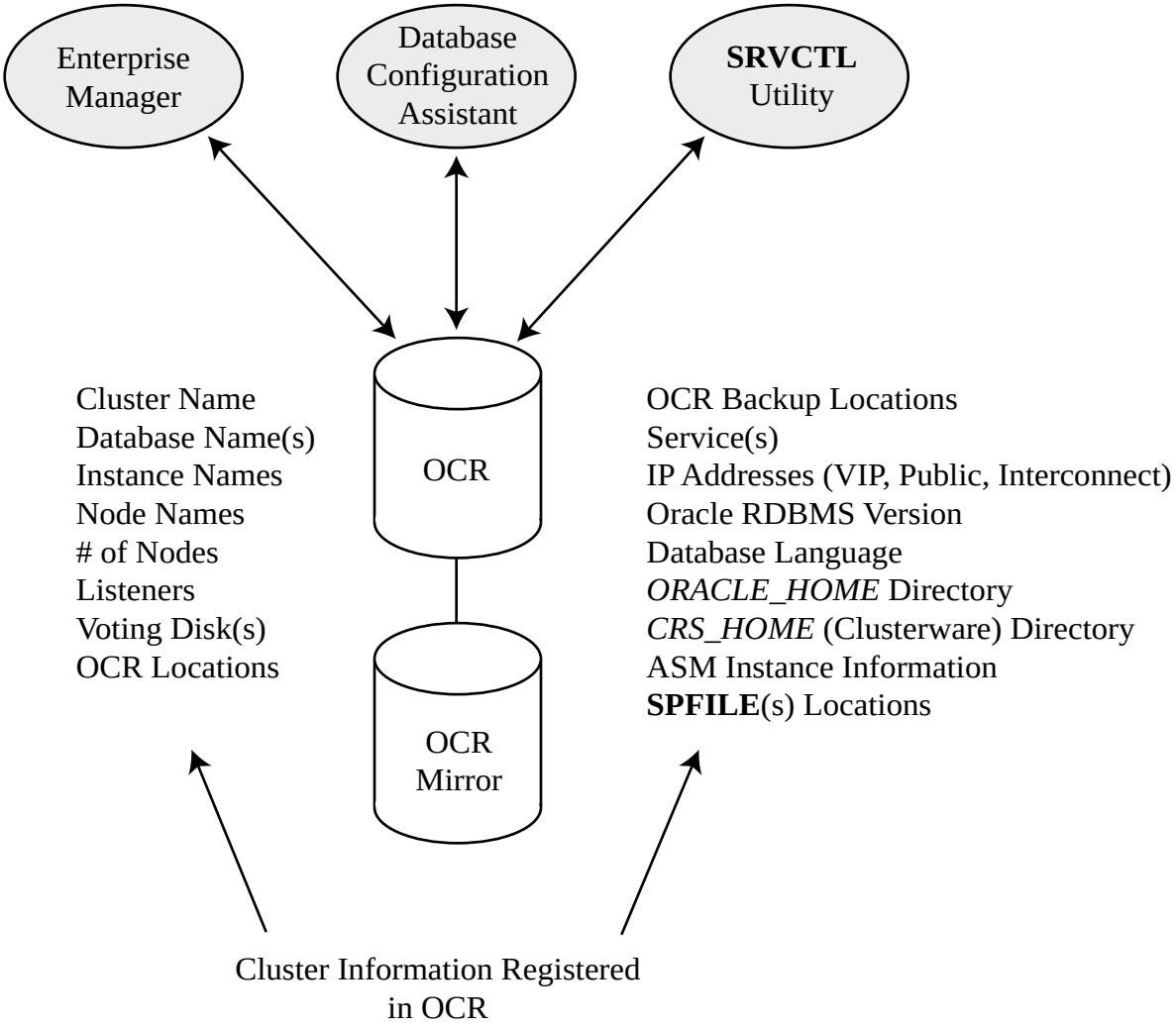


Oracle Software File Locations — Option 2



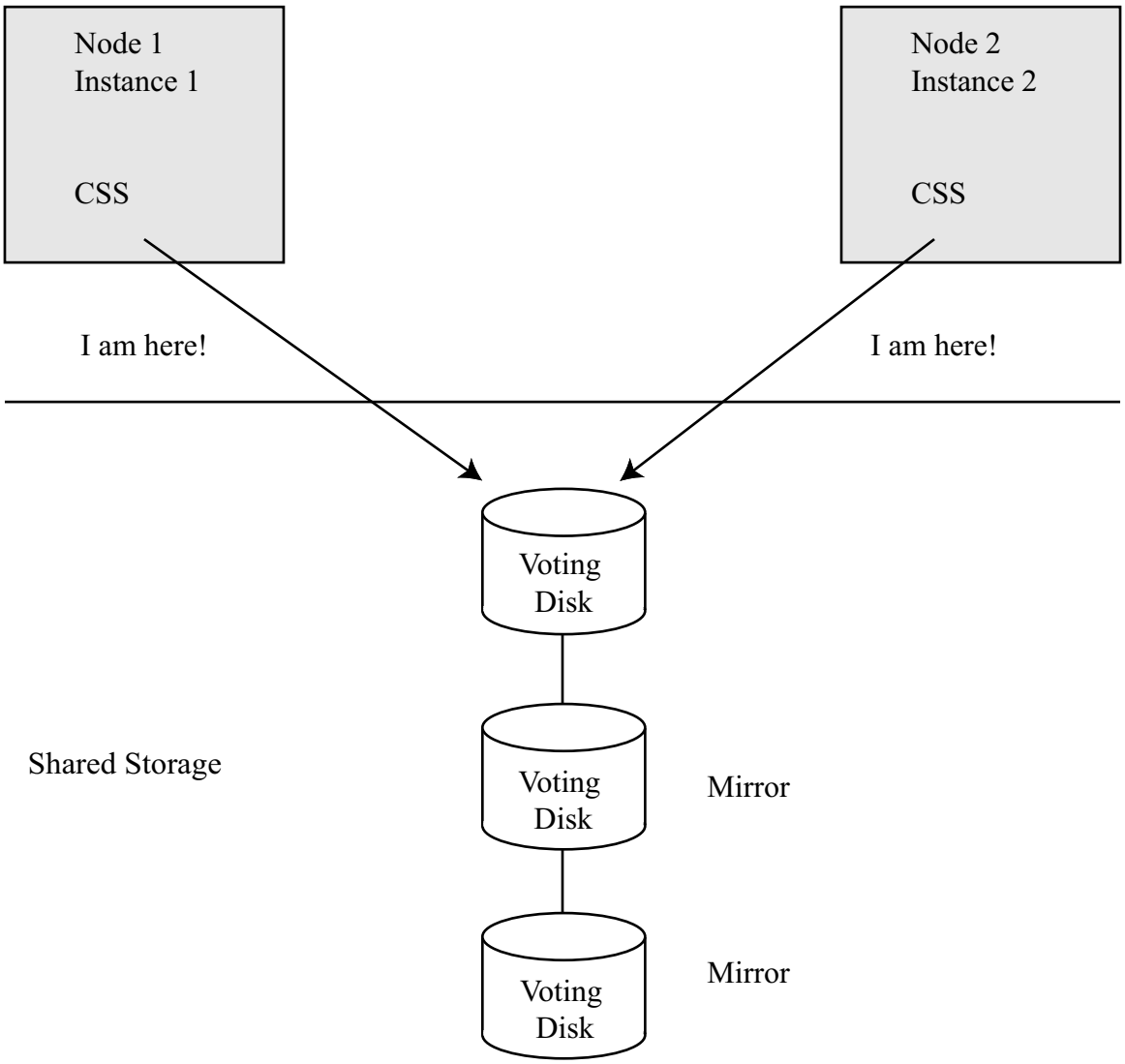
Oracle Cluster Registry (OCR)

- The OCR is a file that maintains cluster and database configuration information, such as:
  - Cluster Name
  - Database Name(s)
  - Instance Names
  - Node Names
  - Number of Nodes
  - Listeners
  - Voting Disk(s)
  - OCR Locations
  - OCR Backup Locations
  - Services
  - IP Addresses (VIP, Public, Interconnect)
  - Oracle RDBMS Version
  - Database Language
  - *ORACLE_HOME* Directory
  - *CRS_HOME* (Clusterware) Directory
  - ASM Instance Information
  - **SPFILE**(s) Locations
- The OCR must be located on shared storage.
- The OCR file is created during the installation of the Oracle Clusterware software.
- The OCR can be read and updated using Enterprise Manager (EM), the **SRVCTL** utility, the Database Configuration Assistant (DBCA), and a number of other utilities and tools, including SQL*Plus.
- In Oracle 10*g* R2 and later, the OCR file can be mirrored.


Enterprise Manager
Database Configuration Assistant
SRVCTL Utility
OCR
OCR Mirror
Cluster Name
Database Name(s)
Instance Names
Node Names
# of Nodes
Listeners
Voting Disk(s)
OCR Locations
OCR Backup Locations
Service(s)
IP Addresses (VIP, Public, Interconnect)
Oracle RDBMS Version
Database Language
ORACLE_HOME Directory
CRS_HOME (Clusterware) Directory
ASM Instance Information
SPFILE(s) Locations
Cluster Information Registered in OCR

THE RAC VOTING DISK

- The *voting disk* is a file on shared storage that is used by the Cluster Synchronization Services (CSS) daemon OCSSD to manage node membership in the cluster.
  - Nodes are notified any time a node joins or leaves the cluster.
- The voting disk file is created during the installation of the Oracle Clusterware software.
- In Oracle 10*g* R2 and later, the voting disk file can be mirrored.
  - By default, three copies of the voting disk file will be created.


Node 1
Instance 1
CSS
Node 2
Instance 2
CSS
I am here!
I am here!
Voting
Disk
Voting
Disk
Mirror
Voting
Disk
Mirror
Shared Storage

LABS

❶ Name the cluster component that is responsible for maintaining cluster and database configuration information.

❷ The Oracle database files can be created onto what type(s) of file systems?

❸ What role do virtual IP addresses play in the event of a failed node?

❹ Which cluster component maintains a cluster membership list and provides timely notification of membership changes?

❺ What is the main purpose of the private (cluster) interconnect?

❻ From the following list of files, list the file under its correct designation on the shared storage of a RAC system. The Control Files item has been done for you.

OCR
Redo Log Groups and Members
SPFILE
Archived Redo Logs
Data Files
Password File
Flash Recovery Area
Voting Disk
Control Files
Temp Files
Undo Tablespace

Instance-Specific Files	Shared Files
	Control Files

CHAPTER 7 - RAC INSTANCE MANAGEMENT

OBJECTIVES

- Correctly set RAC initialization parameters.
- Start and stop an individual RAC instance.
- Start and stop an entire RAC database.
- Add, remove, and monitor redo log files for a RAC database.
- Manage undo tablespaces in a RAC database.

OVERVIEW OF RAC INSTANCE MANAGEMENT

- You can manage, start up, and shut down instances with OEM, SQL*Plus, or **SRVCTL**.
  - Both OEM and **SRVCTL** provide options to start up and shut down all of the instances in an Oracle RAC database with a single step.
  - Using SQL*Plus requires mulitple steps to start up and shut down all of the instances in an Oracle RAC database.
- The OEM is a browser-based tool that enables you to start up, shut down, and monitor databases, instances, etc., and is available with two different controls.
  - Database Control — Manages a single RAC instance/database.
  - Grid Control — Manages multiple RAC instances/databases.
  - The OEM, however, allows you to manage other items beyond **SRVCTL**.
    - Schemas, Tablespaces, Storage, Segments, Performance Monitoring, and more.
- SQL*Plus allows you to operate only the current instance.
  - With the appropriate privileges you can shut down and start up instances, but only one instance at a time.
- **SRVCTL** is a command-line utility that can perform many of the same instance/ database functions as OEM.
- **SRVCTL** stores configuration data in the OCR.
  - This configuration data can also be used by the OEM.

Oracle Instance Management Tools Summary

Tool	Format	Advantages
Oracle Enterprise Manager	Browser-Based	Easy to use. Can be used to manage the complete instance and database. With Grid Control can be used to manage multiple instances. Integrated with the OCR.
SRVCTL	Command Line	Can be used to manage multiple instances such as services or listeners. Integrated with the OCR.
SQL*Plus	Browser-Based Command Line GUI-Based	Easy to use.

STARTING AND STOPPING A RAC DATABASE

- In a RAC environment, starting and stopping a RAC database means starting and stopping all of the RAC instances for that database.

- The OEM and **SRVCTL** have the ability to shut down and start up a RAC database with a single step.

- SQL*Plus must shut down each RAC instance one at time in multiple steps.

  - In the case where you want to shut down every RAC instance in the cluster, OEM and **SRVCTL** offer a big advantage over SQL*Plus, especially in large clusters.

- Starting a RAC database with **SRVCTL**:

```
svrctl start database -d <databasename>
[-o <startupoptions>] [-c <connectstring> | -q]
```

- Stopping a RAC database with **SRVCTL**:

```
svrctl stop database -d <databasename> [-o <stopoptions>]
[-c <connectstring> | -q]
```


Valid startup options:

- **NOMOUNT**
- **MOUNT**
- **OPEN** (default)

```
# svrctl start database -d xyz -o OPEN
```

Valid stop options:

- **NORMAL**
- **IMMEDIATE** (default)
- **TRANSACTIONAL**
- **ABORT**

```
# svrctl stop database -d xyz -o IMMEDIATE
```

STARTING AND STOPPING A RAC INSTANCE

- In a RAC environment, multiple instances can have the same database open at the same time.
- The procedures for starting and stopping a RAC instance are the same as starting and stopping a single instance.
- A RAC instance can be started or stopped with either OEM, SQL*Plus, or with **SRVCTL**.
  - When using ASM, the ASM instances must be started before the database instances.
- Starting a RAC instance options with **SRVCTL**:

```
srvctl start instance –d <databasename> –i
<instancenamelist> [–o <startupoptions>]
[–c <connectstring> | –q]
```

  - The startup options for an instance are the same as the start options for a RAC database.
- Stopping a RAC instance options with **SRVCTL**:

```
srvctl stop instance –d <databasename> –i <instancenamelist>
[–o <stopoptions>] [–c <connectstring> | –q]
```

  - The stop options for an instance are the same as the stop options for a RAC database.
- As with single instance databases, details about each instance startup and shutdown can be found in that instance's alert log.

Stop RAC instances **xyz1** and **xyz2** that are part of RAC database **xyz** with the **IMMEDIATE** option:

```
srvctl stop instance -d xyz -i xyz1,xyz2 -o immediate
```

Stop RAC instances **xyz1** and **xyz2** that are part of RAC database **xyz** with the **OPEN** option:

```
srvctl start instance -d xyz -i xyz1,xyz2 -o open
```

Start all RAC instances that are registered with the RAC database **xyz**:

```
srvctl start database -d xyz -o open
```

RAC Database Identical Parameters

- Using a single **SPFILE** for all your RAC instances is preferred.
  - Set the **SPFILE** parameter to the location, on shared storage, of your **SPFILE**.
- Set **CLUSTER_DATABASE=true**.
- Setting **CLUSTER_DATABASE_INSTANCES** to the correct number allows Oracle to tune memory use appropriately.
- If there is more than one cluster interconnect, list them in **CLUSTER_INTERCONNECTS**.
  - You do not need to set this parameter if:
    - Your interconnects are bonded at the OS level.
    - You have only one interconnect interface.

The following parameters must be set identically for all RAC database instances:

- **ACTIVE_INSTANCE_COUNT**
- **ARCHIVE_LAG_TARGET**
- **CLUSTER_DATABASE**
- **CONTROL_FILES**
- **DB_BLOCK_SIZE**
- **DB_FILES**
- **DB_DOMAIN**
- **DB_RECOVERY_FILE_DEST**
- **DB_RECOVERY_FILE_DEST_SIZE**
- **DB_UNIQUE_NAME**
- **MAX_COMMIT_PROPAGATION_DELAY**
- **TRACE_ENABLED**
- **UNDO_MANAGEMENT**

RAC DATABASE UNIQUE PARAMETERS

- Each instance must have its own unique setting for the following parameters:
  - **THREAD**
    - Used in archived redo log file names.
    - Use the same number as **INSTANCE_NUMBER**.
  - **ROLLBACK_SEGMENTS**
    - Not needed, and in fact ignored, if you set **UNDO_MANAGEMENT** to **AUTO** as recommended.
  - **INSTANCE_NAME**
    - Normally, the **INSTANCE_NUMBER** concatenated to **DB_NAME**.
  - **INSTANCE_NUMBER**
  - **UNDO_TABLESPACE**
    - Each instance uses its own undo tablespace, which is visible (on shared storage) and used by the other instances.

CHANGING PARAMETER VALUES

- How you change your initialization parameter values depends on whether you are using a **PFILE** or an **SPFILE** for your parameters.
  - The **PFILE** is a text file and its contents can be changed with your operating system editor.
  - The **SPFILE** is a binary file and its contents can be changed with the **ALTER SYSTEM** command.
    - Editing your **SPFILE** with an operating system editor can result in a corrupted **SPFILE**.
- The **ALTER SYSTEM** command can be used to change the current value for a parameter or to change the value of a parameter in an **SPFILE** or both.
  - It cannot be used to change the value in a **PFILE**.
- The **SPFILE** has several advantages over the **PFILE** and is recommended to set values for your initialization parameters.
  - Simplifies administration
  - Maintains parameter setting consistency
  - Guarantees parameter setting persistence across database / instance shutdown and startup events
- All instances in the cluster database use the same **SPFILE** at startup.
  - The **SPFILE** needs to reside on the shared file system in a RAC environment.
- Parameter values can also be changed from Enterprise Manager.

When changing the values of your **SPFILE** parameters there is a very important option to consider:

```
ALTER SYSTEM SET <parametername> = <value> [SIDoption] [SCOPEoption] ;
```

The **SID*option*** updates the specified parameter in the **SPFILE** and associates the parameter and its value to a particular instance.

```
ALTER SYSTEM SET db_cache_size=500M SID='xyz2' ;

Resulting SPFILE Entry
xyz2.db_cache_size=500M
```

If you do not specify the **SID*option***, the default is *****, which means the parameter and its value apply to all of the instances in the RAC environment.

```
ALTER SYSTEM SET db_cache_size=500M ;

Resulting SPFILE Entry
*.db_cache_size=500M
```

Administering Undo Tablespaces in RAC

- Each RAC instance has its own undo tablespace.
  - All undo tablespaces reside on shared storage.
- All instances can always read all undo blocks throughout the cluster environment for consistent read purposes.
- Automatic Undo is the recommended procedure.
  - It eases the management of undo segments.
  - Oracle automatically manages undo segments (how many and their space usage) among the various active sessions.
  - To enable automatic undo management, set the **UNDO_MANAGEMENT** initialization parameter to **AUTO**.
- Oracle automatically manages undo segments within a specific undo tablespace that is assigned to an instance.
  - Only the instance assigned to the undo tablespace can modify the contents of that tablespace.
  - Undo tablespaces in your Oracle RAC database are assigned by specifying a different value for the **UNDO_TABLESPACE** parameter for each instance.
  - Each instance in your RAC environment should have its own undo tablespace.
- All instances of an Oracle RAC database must operate in the same undo mode.

To verify your undo management mode and the undo tablespaces being used, check the **UNDO_MANAGEMENT** and **UNDO_TABLESPACE** parameters. Remember, all instances of an Oracle RAC database must operate in the same undo mode.

```
  SELECT inst_id, instance_name, name, value
    FROM gv$instance NATURAL JOIN gv$spparameter
   WHERE name = 'undo_management' OR
         name = 'undo_tablespace'
ORDER BY instance_name, name;

INSTANCE_NAME    NAME             VALUE
---------------  ---------------  ------------
xyz1              undo_management AUTO
xyz1              undo_tablespace UNDOTBS2
xyz1              undo_tablespace UNDOTBS1
xyz2              undo_management AUTO
xyz2              undo_tablespace UNDOTBS1
xyz2              undo_tablespace UNDOTBS2
```

Both instances are operating in Automatic Undo Management mode.

If you need to change the assignment of an undo tablespace to an instance, use the **ALTER SYSTEM** command. Remember to use the **SID*option*** to assign the undo tablespace to the appropriate instance.

```
ALTER SYSTEM SET undo_tablespace = 'UNDOTBS3' SCOPE = BOTH SID='xyz2';
```

Note:
The **SCOPE*option*** must occur before the **SID*option*** in the **ALTER SYSTEM** statement.

After the last **ALTER SYSTEM** command, your parameter file contents will be:

```
*.undo_management='AUTO'
xyz1.undo_tablespace='UNDOTBS1'
xyz2.undo_tablespace='UNDOTBS3'
```

ADMINISTERING REDO LOGS IN RAC

- In a single-instance Oracle database environment, redo logs are stored in two or more redo log file groups.
  - Each of these groups contains a redo log file, and possibly one or more mirrored copies of that file.
- In a RAC database, each instance requires its own set of redo log groups.
  - All redo log groups reside on shared storage.
  - Each set is known as a *thread* of redo logs.
    - Each redo log thread must contain at least two redo log groups.
  - Each instance writes and archives the redo log groups in its own thread.
    - This behavior is the same as with single-instance Oracle databases.
  - In recovery mode, the instance performing the recovery is able to read and process all of the redo threads for the database, regardless of which instance generated the redo threads.
  - If your database is in **ARCHIVELOG** mode, then each instance must save filled log files into its own archive log thread.

To view your current redo log groups and the instances/threads they belong to:

```
  SELECT DISTINCT thread#, instance, group#
    FROM gv$log JOIN gv$thread USING (thread#)
ORDER BY thread#;

   THREAD# INSTANCE        GROUP#
---------- ----------  ----------
         1 xyz1                 1
         1 xyz1                 2
         2 xyz2                 3
         2 xyz2                 4
```

To add a redo log group to a specific thread/instance:

```
ALTER DATABASE
ADD LOGFILE THREAD 1 GROUP 5   ('/u02/oradata/xyz/redo05.log')
SIZE 50M;
```

Note:
If you do not specify the thread, the redo log group gets added to the current instance.

LABS

❶ For each node in your RAC cluster, use SQL*Plus to add a new redo log group to each instance.

❷ Use SQL*Plus commands to increase the **db_cache_size** by 10% for each instance in your cluster. Make sure the change is persistent across instance restarts.

CHAPTER 15 - RAC TROUBLESHOOTING

OBJECTIVES

- Locate and interpret the contents of critical RAC alert and RAC component log files.
- Execute the **crsctl** command to diagnose cluster issues.
- Execute the **cluvfy** utility and use its various commands to diagnose cluster problems.
- Execute various Oracle-supplied scripts to diagnose RAC problems.

THE ORACLE CLUSTERWARE ALERT LOG

- The clusterware alert log is used to track clusterware startup, shutdown, and operational messages.
  - The log resides in *$CRS_HOME/log/node*.
  - It's commonly used to determine if there has been a loss of either voting or OCR disk and clusterware membership.
  - It only applies to cluster specific resources and can't be used for database and **nodeapps** related resources.
- The log contains the following information:
  - OCR file availability
  - OCR version number
  - Voting disk availability
  - Number of active voting
  - Cluster membership (nodes in the cluster)
  - Cluster daemon startup

The following example shows a problem with the */dev/raw/raw2* OCR file operating system permissions:

```
[client(15193)]CRS-1006:The OCR location /dev/raw/raw2 is inaccessible.
Details in /u01/app/oracle/product/10.2.0/crs/log/rst-consult05/client/
ocrconfig_15193.log.

# more /u01/app/oracle/product/10.2.0/crs/log/rst-consult05/client/
ocrconfig_15193.log
/u01/app/oracle/product/10.2.0/crs/log/rst-consult05/client/
ocrconfig_15193.log: Permission denied
```

The following is an example of a node leaving the cluster (graceful shutdown) and rejoining the cluster:

```
[cssd(15818)]CRS-1601:CSSD Reconfiguration complete. Active nodes are
rst-consult05 .
2007-09-22 09:27:52.490
[crsd(15407)]CRS-1204:Recovering CRS resources for node rst-consult06.
[cssd(15818)]CRS-1601:CSSD Reconfiguration complete. Active nodes are
rst-consult05 rst-consult06 .
```

CLUSTERWARE COMPONENT LOG FILES

- The three clusterware daemons (**crsd**, **cssd**, **evmd**) all have log files that contain log and trace information; however, the **crsd** and **cssd** logs are the most useful when debugging clusterware operations.

- The **crsd** log shows the clusterware starting and stopping resources.

  - The **crsd** log files can be found in the *$CRS_HOME/log/node-name/crs/crsd.log* file.

  - Each cluster node generates and maintains a separate **crsd** log, which contains the following information:
    - Clusterware startup
    - Number of OCR found at startup
    - OCR version
    - Nodeapps, ASM instance and database instance startup
    - VIP failover over and failback
    - Cluster membership

- The **cssd** log shows the network and disk heartbeat status and any messages from third party clusterware; it's essential to track down the cause of node evictions.

  - The log files are stored in the *$CRS_HOME/log/node-name/cssd/ocssd.log* file and contain the following information:
    - css daemon startup
    - Node membership
    - Network heartbeat timeout message
    - Voting disk timeout messages
    - Node fencing messages
    - Routine trace messages

- The **evmd** daemon is used for messaging between clusterware nodes.

  - The log files are stored in the *$CRS_HOME/log/node-name/evmd* directory.

  - Not documented - refer to Oracle support with regard to these logs.

The following **crsd** log shows the startup of **nodeapps**, ASM, and database resources:

```
2007-09-27 17:45:03.728: [  CRSRES][2723687344]0Attempting to start 'ora.rst-
consult02.vip' on member 'rst-consult02'
2007-09-27 17:45:07.663: [  CRSRES][2723687344]0Start of 'ora.rst-consult02.vip'
on member 'rst-consult02' succeeded.
2007-09-27 17:45:07.709: [  CRSRES][2723687344]0startRunnable: setting CLI values
2007-09-27 17:45:07.712: [  CRSRES][2723687344]0Attempting to start 'ora.rst-
consult02.LISTENER_RST-CONSULT02.lsnr' on member 'rst-consult02'
2007-09-27 17:45:09.759: [  CRSRES][2723687344]0Start of 'ora.rst-
consult02.LISTENER_RST-CONSULT02.lsnr' on member 'rst-consult02' succeeded.
2007-09-27 17:45:13.095: [  CRSRES][2681052080]0Attempting to start 'ora.rst-
consult02.ons' on member 'rst-consult02'
2007-09-27 17:45:14.634: [  CRSRES][2681052080]0Start of 'ora.rst-consult02.ons'
on member 'rst-consult02' succeeded.
2007-09-27 17:45:14.665: [  CRSRES][2713197488]0Start of 'ora.rst-
consult02.ASM2.asm' on member 'rst-consult02' succeeded.
2007-09-27 17:45:14.748: [  CRSRES][2713197488]0Attempting to start
'ora.orcl.orcl2.inst' on member 'rst-consult02'
2007-09-27 17:45:35.419: [  CRSRES][2713197488]0Start of 'ora.orcl.orcl2.inst' on
member 'rst-consult02' succeeded.
```

The primary use of the **cssd** log is to identify network and voting disk access problems. It's advisable to **grep** this log for the literal "timeout" to look for network and voting disk access errors.
The following example shows the start of the **css** daemon

```
$ more $CRS_HOME/log/node/cssd/ocssd.log
Oracle Database 10g CRS Release 10.2.0.1.0 Production Copyright 1996, 2005
Oracle.  All rights reserved.
[    CSSD]2007-09-26 16:20:12.311 >USER:    Oracle Database 10g CSS Release
10.2.0.1.0 Production Copyright 1996, 2004 Oracle.  All rights reserved.
[    CSSD]2007-09-26 16:20:12.311 >USER:    CSS daemon log for node rst-
consult01, number 1, in cluster cluster01
[  clsdmt]Listening to (ADDRESS=(PROTOCOL=ipc)(KEY=rst-consult01DBG_CSSD))
[    CSSD]2007-09-26 16:20:12.319 [3086882496] >TRACE:   clssscmain: local-only
set to false
[    CSSD]2007-09-26 16:20:12.337 [3086882496] >TRACE:   clssnmReadNodeInfo:
added node 1 (rst-consult01) to cluster
[    CSSD]2007-09-26 16:20:12.350 [3086882496] >TRACE:   clssnmReadNodeInfo:
added node 2 (rst-consult02) to cluster
[    CSSD]2007-09-26 16:20:12.356 [61107120] >TRACE:   clssnm_skgxnmon: skgxn
init failed, rc 1
[    CSSD]2007-09-26 16:20:12.356 [3086882496] >TRACE:   clssnm_skgxnonline:
Using vacuous skgxn monitor
[    CSSD]2007-09-26 16:20:12.365 [3086882496] >TRACE:   clssnmDiskStateChange:
state from 1 to 2 disk (0//dev/raw/raw2)
```

USING CRSCTL TO DIAGNOSE CLUSTER ISSUES

- **crsctl** is the clusterware command-line interface available to systems and database administrators.
- **crsctl stop crs** is the supported method for stopping the Oracle Clusterware.
  - Stopping the clusterware will stop all dependent resources (instances, ASM, **nodeapps**).
- **crsctl start crs** is the supported method for starting the Oracle Clusterware.
- The **stop** and **start crs** options require the user to have **root** privileges.
- **crsctl check crs** lists the status of the Oracle clusterware.
  - Can be run as user **oracle** or user **root**.
- **crsctl disable crs** disables the automatic startup of the clusterware.
- **crsctl enable crs** enables automatic startup of the clusterware at boot time.
  - These are frequently required when debugging clusterware, rebooting, or when there is maintenance scheduled and you want to ensure that the clusterware doesn't accidentally start up.


```
# crsctl stop crs
Stopping resources. This could take several minutes
Stopping resources. This could take several minutes.
Successfully stopped CRS resources.
Stopping CSSD.
Shutting down CSS daemon.
Shutdown request successfully issued.

# crsctl start crs
Attempting to start CRS stack
The CRS stack will be started shortly

# crsctl check crs
CSS appears healthy
CRS appears healthy
EVM appears healthy
```

USING DIAGCOLLECTION.PL

- *diagcollection.pl* is a Perl script that collects diagnostics, including cluster logs and OS core files, and creates zipped *tar* files.
  - Oracle Support frequently requires this collection be done while logging RAC related *tar*'s.
  - The script resides in the *$CRS_HOME/bin* directory.
  - It only gathers logs from the node on which it's executed, so you must execute it on each RAC node for a complete collection.
  - Note that the **ORA_CRS_HOME** environment variable must be set for **crs** diagnostics and the **ORACLE_HOME** environment variable must be set for Oracle Home diagnostic.
  - The script must be executed with **root** privileges when gathering **crs** diagnostics.

```
$ $CRS_HOME/bin/diagcollection.pl –collect – clean –coreanalyze
```

  - **collect --option**
    - **crs** — Collects **crs diag** information.
    - **oh** — Collects Oracle Home diagnostic information.
    - **-all** — Collects all information. (Default)
    - **nocore** — Unix only. Does not package core files.
  - **clean** — Cleans up previous executions of the script.
  - **coreanalyze** — Unix only. Extracts core file information.

The following example will collect **crs** and Oracle home diagnostics:

```
$  export  ORA_CRS_HOME=/u01/app/oracle/product/10.2.0/crs
$  export  ORACLE_HOME=/u01/app/oracle/product/10.2.0/DB
$ su
#  $ORA_CRS_HOME/diagcollection.pl   –collect –all

$ ls *.gz

-rw-r–r–  1 root root 36129 Sep 27 12:28 crsData_rst-consult02.tar.gz
-rw-r–r–  1 root root  7297 Sep 27 12:28 ocrData_rst-consult02.tar.gz
-rw-r–r–  1 root root  3331 Sep 27 12:29 oraData_rst-consult02.tar.gz
```

CHECKING INTERCONNECT SETTINGS

- Operating system and Oracle commands can be used to determine whether the interconnect is configured correctly.
- **$CRS_HOME/bin/oifcfg** command will display the subnet and interfaces being used for the public and private interconnect.

```
oifcfg getif –global
```

  - **oifcfg** is also used to change the subnet/interface being used for the public or private network.
  - The change will not take place until the clusterware is restarted on all nodes.

```
oifcfg setif –node node-name | –global interface/subnet/type
```

    - **-global** — All nodes
    - **subnet** —The network address with 0 as the end number (10.1.1.0)
    - **interface** — The hardware interface (**eth1**, **bond0**,etc)
    - **type** — Public or private

- **ifconfig** can be used to detect packets falling off of the UPD buffer.
  - Cache Fusion uses multiple block requests when sending blocks across the interconnect, and a heavy Cache Fusion load can flood the UDP send and receive buffers.
  - RX or TX errors indicate that UDP packets are not being sent or received correctly.

The following will change the interconnect to be on the **bond0** interface on all RAC nodes:

```
# oifcfg setif -global bond0/10.1.1.0/cluster_interconnect
```

The following displays the status of the **eth1** interface:

```
# ifconfig eth1
eth1       Link encap:Ethernet  HWaddr 00:14:22:22:0C:E6
           inet addr:192.168.2.2  Bcast:192.168.2.255  Mask:255.255.255.0
           inet6 addr: fe80::214:22ff:fe22:ce6/64 Scope:Link
           UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
           RX packets:82467 errors:0 dropped:0 overruns:0 frame:0
           TX packets:85621 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:1000
           RX bytes:46412399 (44.2 MiB)  TX bytes:54983934 (52.4 MiB)
           Base address:0xccc0 Memory:df8e0000-df900000
```

There are two ways to resolve UDP buffer errors. You can either increase the UDP buffer size or lower **DB_FILE_MULTIBLOCK_READ_CNT** *init.ora* parameter. Lowering this parameter has a major impact on the use of indexes versus full table scans. Lower values are favored in OLTP based databases (index access) whereas Data Warehouse databases favor higher values (full table scans).

CLUVFY – VERIFYING CLUSTERWARE COMPONENT INTEGRITY

- **cluvfy** verifies that the clusterware components and node operating systems are configured correctly.
  - In addition, it checks user equivalence to see if the **nodeapps** are running, unlike **crsctl**.
    - For user equivalence errors, verify that the **ssh** keys for user **oracle** haven't changed.
    - For cluster manager integrity errors, check the clusterware log files to determine why the applicable daemons are not running.

```
$ cluvfy stage -post crsinst -n node-name,node-name …
```



```
$ /u01/app/oracle/product/10.2.0/crs/bin/cluvfy stage -post crsinst -n rst-
consult01,rst-consult02

Performing post-checks for cluster services setup

Checking node reachability...
Node reachability check passed from node "rst-consult01".

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

Checking Cluster manager integrity...

Checking CSS daemon...
Daemon status check passed for "CSS daemon".

Cluster manager integrity check passed.

Checking cluster integrity...
Cluster integrity check passed

Checking OCR integrity...

Checking the absence of a non-clustered configuration...
All nodes free of non-clustered, local-only configurations.

Uniqueness check for OCR device passed.

Checking the version of OCR...
OCR of correct Version "2" exists.

Checking data integrity of OCR...
Data integrity check for OCR passed.

OCR integrity 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.

Checking node application existence...

Checking existence of VIP node application (required)
Check passed.

Checking existence of ONS node application (optional)
Check passed.

Checking existence of GSD node application (optional)
Check passed.

Post-check for cluster services setup was successful
```

CLUVFY –VERIFYING CLUSTER REGISTRY INTEGRITY

- **cluvfy** command also checks the OCR files/devices to ensure that they're consistent.
  - Any inconsistent OCR should be replaced.
    - **OCRCONFIG –RESTORE**
    - **dd** of a binary backup of a consistent OCR

```
$ cluvfy comp ocr
```



```
$ cluvfy comp ocr

Verifying OCR integrity

Checking OCR integrity...

Checking the absence of a non-clustered configuration...
All nodes free of non-clustered, local-only configurations.

Uniqueness check for OCR device passed.

Checking the version of OCR...
OCR of correct Version "2" exists.

Checking data integrity of OCR...
Data integrity check for OCR passed.

OCR integrity check passed.

Verification of OCR integrity was successful.
```

CLUVFY – VERIFYING CLUSTER INTEGRITY

- **cluvfy** verifies that the clusterware daemons are running and performs a health check.
  - Any daemons that are not running should be diagnosed and fixed.
  - Use the appropriate log files to diagnose the problem.
  - Once the problem is fixed, use **crsctl start crs** to restart the clusterware.

```
$ cluvfy comp crs
```



```
$./cluvfy comp crs

Verifying CRS integrity

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.

Verification of CRS integrity was successful.
```

ocrcheck – Verifying the Oracle Cluster Repository

- Use **ocrcheck** to check the Oracle Cluster Repository to ensure that the primary and mirrored repositories are consistent.

- An inconsistent OCR can be repaired by copying a consistent OCR or restoring from an OCR backup.

  - Oracle backs up the OCR every four hours.

  - Oracle keeps three backups: the backups for the last four hours, the last day, and the last week.


```
$ ocrcheck
Status of Oracle Cluster Registry is as follows :
         Version                  :          2
         Total space (kbytes)     :     263928
         Used space (kbytes)      :       3840
         Available space (kbytes) :     260088
         ID                       :  451284365
         Device/File Name
   : /dev/raw/raw1
                                  Device/File integrity check succeeded
         Device/File Name         : /dev/raw/raw2
                                  Device/File integrity check succeeded

         Cluster  registry  integrity  check  succeeded
```

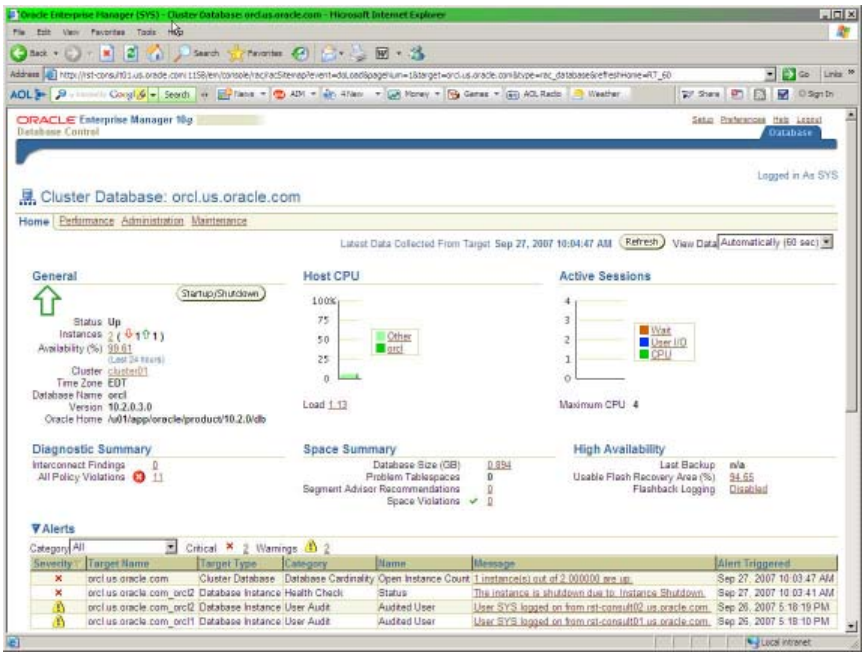
The following shows the **ocrcheck** output when one of the OCR files is not consistent

```
[oracle@rst-consult01 bin]$ ./ocrcheck
Status of Oracle Cluster Registry is as follows :
         Version                  :          2
         Total space (kbytes)     :     263928
         Used space (kbytes)      :       3840
         Available space (kbytes) :     260088
         ID                       :  451284365
         Device/File Name         : /dev/raw/raw1
                                  Device/File integrity check succeeded
         Device/File Name         : /dev/raw/raw2
                                  Device/File integrity check failed
n
         Cluster registry integrity check succeeded
```

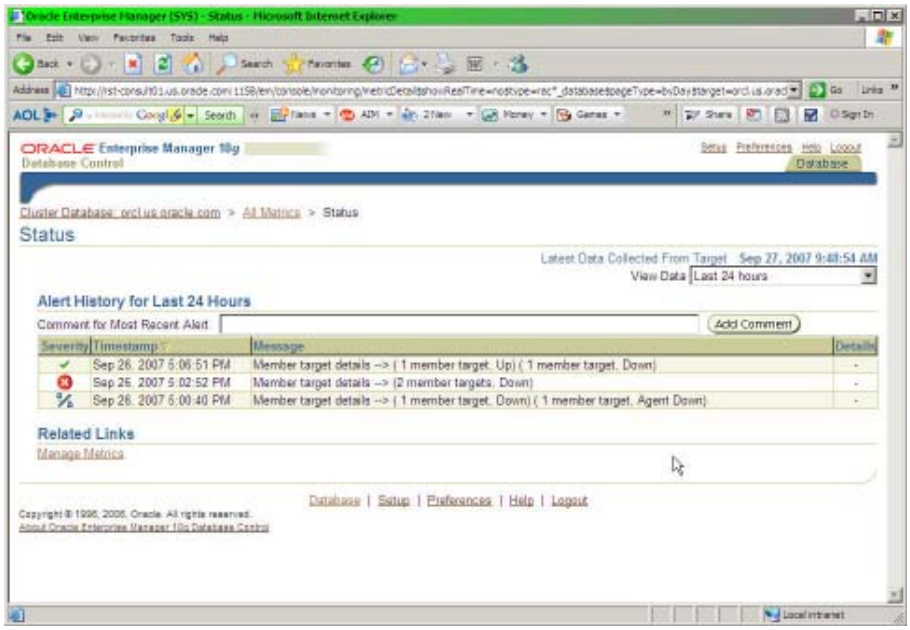
RAC Database Alerts

- Use either the Grid Control or the Database Console to look at RAC Database alerts or have notifications automatically sent via email.
  - RAC Database alerts are in the Cluster Database Home page.
- Alerts are triggered by metrics that exceed a given threshold.
- The Oracle Enterprise Manager Database and Database Related Metric Reference Manual contains a detailed list of all alerts.
  - Recommended alerts for RAC monitoring are:
    - Response Status — Shows when a cluster node leaves the cluster.
    - Open Instance Count — Shows when instances are shutdown.
- Current alerts are displayed on the Cluster Database page.
  - For alert history, go to the Cluster Database page, select All Metrics, select the desired metric, and the alert history will be displayed.
- The **MGMT$ALERT_HISTORY** view contains historical information for any alerts logged in the Management Repository.
  - Some useful columns are:
    - **target_name** — Name of the target.
    - **metric_name** — The formal metric name.
    - **metric_label** — The metric name as an intuitive display name.
    - **collection_timestamp** — The date and time the alert was generated.
    - **message** — Suggested action messages.

This example shows the current alerts when an instance is aborted.



The following shows the alert history for Response Status generated when clusterware is shutdown (**crsctl stop crs**).



THE RACDIAG.SQL SCRIPT

- The *racdiag* script gathers common RAC related statistics.
  - The text for the script is in metalink note 1357141.1 – copy the text into a file to use it.
  - When prompted, supply the name of the instance to diagnose.
  - The script generates a *racdiag* output file with name "racdiag_" followed by the database name, an underscore, the name of the instance, and the literal "?.amp.out".

The script gathers:

- Instance start and up times
- Session level wait events
- GES Lock blockers
- GES Lock waiters
- Local enqueues
- Latch statistics
- Global cache CR performance
- Resource usage
- Generates and hangcheck analyze dump
- DLM (Distributed Lock Manger) traffic
- Lock conversations
- Top 10 write pinging/Cache Fusion objects
- Top 10 false pinging
- *Init.ora* parameters
- Session/Process reference
- System statistics
- SQL for waiting sessions

THE ORADEBUG UTILITY

- The **oradebug** utility is frequently used by Oracle Support to debug RAC and non-RAC related issues, such as:
  - Verifying the Cache Fusion interconnect.
    - Even if the clusterware is using the correct interconnect, there are cases when the database will use the public instead of the private network for Cache Fusion traffic.
    - The **oradebug ipc** command will verify which interconnect is being used for Cache Fusion by looking at the IP address listed in the **SSKGXPT** portion of the trace file.

      ```
      SQL> oradebug ipc
      ```
  - Generating a hangcheck analyze.
    - The **oradebug** command can generate a hangcheck analyze report to confirm if the database is really hung or if performance is so slow that it appears to be hung.
    - Frequently requested by Oracle support.

      ```
      oradebug –g all hanganalyze level
      ```
  - Generating **SYSTEMSTATE** dumps.
    - Frequently requested by support when dealing with performance related issues.
    - Shows what each process in the system is doing at the time the state is taken.

      ```
      SQL> oradebug dump system state 266
      ```


This example verifies that the interconnect on IP 192.168.1.2 using UDP protocol is being used for Cache Fusion.

```
SQL> oradebug setmypid
Statement processed.
SQL> oradebug ipc
Information written to trace file.

$ vi *.trc

SSKGXPT 0xcd969c0 flags SSKGXPT_READPENDING    socket no 7    IP 192.168.2.1
UDP 21700
context timestamp 0x5
       no ports
   sconno     accono   ertt  state   seq#   sent  async   sync rtrans   acks
0x09268a10 0x6fcc91a0     32      3  32764      1      1      0      0      1
```

LABS

❶ Run the **crsctl check crs** command to check to see if the clusterware is running on both RAC nodes.

❷ Run the **cluvfy** command and execute a post **crs** installation check. Validate the current state of the OCR files.

❸ Bring up the DB Console. Set the Open Instance count metric to generate a warning when instance count falls below 2.

Use SQL*Plus to abort one of the Database Instances

Go to the Cluster Database page and manually refresh the screen. Look at the Alerts page.

View the alert history for the Open Instance Metric.

❹ Use SQL*Plus to query the alter history for your RAC database. Use Enterprise Manager to view the Alert history for the Response Status alert.

❺ The *racdiag* script is commonly used by Oracle support to help resolve RAC performance issues. The script is embedded in Metalink note 135714.1. To implement this script, bring up the metalink document using a browser. Using the copy function of your browser, highlight the code portion of the node, open an editor, and paste the code portion. Connect to your RAC database as user **sys** and execute the script using SQL*Plus.

❻ Generate a hangcheck analyze and **SYSTEMSTATE** dumps of your RAC database.

