CHAPTER 11

Real Application Clusters
In Chapter 4, “Physical Database Layouts and Storage Management,” we presented an overview of Automatic Storage Management (ASM) and Oracle Managed Files (OMF) and how they can ease administration, enhance performance, and improve availability. You can add one or more disk volumes to a rapidly growing VLDB without bringing down the instance.

In Chapter 6, “Monitoring Space Usage,” we talked about bigfile tablespaces and how they not only allow the total size of the database to be much larger than in previous versions of Oracle, but also ease administration by moving the maintenance point from the datafile to the tablespace. Chapter 16 will focus on Oracle Net, providing you with the basics for ensuring that your clients can reach the database servers in an efficient and prompt manner. Chapter 17 will expand our coverage of bigfile tablespaces in addition to presenting other tools to make large database management easier, such as partitioned table support, transportable tablespaces, and new to Oracle 10g, Oracle Data Pump.

As your databases get larger, and the number of users increases, the need for availability becomes even more critical. Real Application Clusters (RAC) will tie together OMF, bigfile tablespaces, a robust network infrastructure, and ASM into key elements of the RAC architecture. In this chapter, we will revisit many of these database features, but with an emphasis on how they can be leveraged in a RAC environment.

This chapter focuses on a number of RAC topics, including how to set up your operating system environment—kernel parameters, network configuration, and user accounts. You will perform the requisite installations to support RAC, such as Cluster Ready Services (CRS) for creating a clustered environment, along with the various installation options within the Oracle Universal Installer (OUI) to configure your network, shared storage, and database software installation for both CRS and the Oracle 10g database itself.

During the installation of a RAC, you can configure the Enterprise Manager agent and Enterprise Manager Database Control to manage your cluster. EM Database Control extends the functionality available to manage a single instance by providing a cluster-aware layer; you can manage both the Oracle instances and the underlying cluster configuration from a single web interface.

In subsequent chapters, we will present other ways to ensure high database availability and recoverability: Chapter 14 will give a detailed look at Oracle Data Guard for near-realtime failover capabilities, and Chapter 18 will cover Oracle Streams for advanced replication. In Chapter 15, we’ll finish up our discussion on Flashback options started in Chapter 7 by showing you how to perform Flashback Drop and Flashback Database as well as how to use LogMiner to undo individual transactions.

Overview of Real Application Clusters

A Real Application Cluster is highly available and scalable. The failure of one node in the cluster does not affect client sessions or the availability of the cluster itself until the last node in the cluster fails; the only impact a lost node has on the cluster is a slight degradation in response time, depending on the total number of nodes in the cluster.

A RAC has a few disadvantages. Licensing costs are higher, because each node in the cluster has its own Oracle license. The close physical proximity of the nodes in the cluster due to the high-speed requirements of the cluster interconnect means that a natural disaster can take out the entire cluster; using a remote standby database can help alleviate some of these concerns.

SAMPLE CHAPTER FOR REVIEW ONLY.

You will have to weigh the cost of high availability (or the lack thereof) compared to the increased cost and slight increase in maintenance of a RAC.

In the next few sections, we’ll cover some of the hardware and software requirements for a RAC as well as detail the network configuration and disk storage requirements to build a successful RAC.

Hardware Configuration
A complete discussion of all possible RAC hardware configurations is beyond the scope of this book. You want to have at least two and preferably three nodes for a RAC, each with redundant power supplies, network cards, dual CPUs, and error-correcting memory; these are desirable characteristics for any type of server, not just an Oracle server! The higher the number of nodes configured in the cluster, the lower the performance hit you will take when one of the cluster’s nodes fails.

The shared disk subsystem should also have hardware redundancy built in—multiple power supplies, RAID-enabled disks, and so forth. You will balance the redundancy built into the shared disk with the types of disk groups you will create for the RAC. The higher redundancy built into the disk subsystem hardware can potentially reduce the amount of software redundancy you specify when you create the database’s disk groups.

Software Configuration
Although Oracle clustering solutions have been available since version 6, not until version 10g has there been a native clusterware solution that more tightly couples the database to the volume management solution. Cluster Ready Services (CRS) is the clustering solution that can be used on all major platforms instead of an OS vendor or third-party clusterware.

CRS is installed before the RDBMS and must be in its own home directory, referred to as the CRS_HOME. If you are only using a single instance in the near future, but plan to cluster at a later date, it is useful to install CRS first so that the components of CRS that are needed for ASM and RAC are in the RDBMS directory structure. If you do not install CRS first, you will have to perform some extra steps later to remove the CRS-related process executables from the RDBMS home directory.

After CRS is installed, you install the database software in the home directory, referred to as the ORACLE_HOME. On some platforms, such as Microsoft Windows, this directory can be a directory common to all nodes, whereas other platforms, such as Linux, require OCFS version 2.x or later. Otherwise, each node will have its own copy of the binary executables.

Network Configuration
Each node in a RAC has a minimum of three IP addresses: one for the public network, one for the private network interconnect, and a virtual IP address to support faster failover in the event of a node failure. As a result, a minimum of two physical network cards are required to support RAC; additional network cards are used to provide redundancy on the public network and thus an alternate network path for incoming connections. For the private network, additional network cards can boost performance by providing more total bandwidth for interconnect traffic. Figure 11-1 shows a two-node RAC with one network card on each node for the private interconnect and one network card on each node to connect to the public network.
The public network is used for all routine connections to and from the server; the interconnect network, or private network, supports communication between the nodes in the cluster, such as node status information and the actual data blocks shared between the nodes. This interface should be as fast as possible, and no other types of communication between the nodes should occur on this interface, otherwise, the performance of the RAC may suffer.

The virtual IP address is the address assigned to the Oracle listener process and supports rapid connect-time failover, which is able to switch the network traffic and Oracle connection to a different instance in the RAC much faster than a third-party high-availability solution.

## Disk Storage

The shared disk drive may or may not be a RAID device to support redundancy; more importantly, the disk controllers and connections to the shared storage should be multiplexed to ensure high availability. If the disks in the shared drive are not mirrored, you can use the mirroring capabilities of ASM to provide performance and availability benefits.

For the purposes of the examples in this chapter, we will use a Linux server with the raw device configuration listed in Table 11-1. These raw disks reside on a shared SCSI storage device and have the same device name on each node in the cluster.

The two raw disks that are 100MB in size are reserved for the voting disk and the OCR disk; we will present the uses for these disks in the section "Cluster Ready Services."

![RAC Network Configuration](image-url)
Installation and Setup

For the examples in this chapter, we will use Red Hat Linux to install the RAC and demonstrate its features. However, most, if not all, the installation tips, techniques and methods presented in this chapter will be applicable to other Unix-like platforms and even Windows-based installations.

We will show you how to set up a three-node RAC; although a two-node RAC can demonstrate most the features of a RAC, you will need a three-node RAC to see how the remaining nodes in the cluster can still operate as a RAC and recover from the loss of a single node in the cluster. In practice, the more nodes in the cluster, the less impact there is to throughput when one node in the cluster fails.

On each node, the Oracle software will reside in a local ORACLE_HOME; the database and recovery files will use ASM, and the OCR and voting disks will use raw devices.

NOTE
As an alternative to ASM and raw disks, Oracle Cluster File System (OCFS) version 2.x, available at http://oss.oracle.com, can be used to store both database files and Oracle executables on a common, shared file system.

Finally, we will assume that the shared disk is accessible via the same node name in the /dev directory and that each node in the cluster can access the shared disk simultaneously; the ASM instance on each node will automatically coordinate access to the shared disk.

Operating System Configuration

The first step is to prepare the operating system. Install Red Hat 3.0 ES or AS, and install every option! The small amount of disk space you might save is quickly offset when you are missing a component later and must find the installation CDs to obtain the missing component. Once everything is installed, be sure to apply all patches from Red Hat Network to take advantage of all security and performance enhancements, although Oracle 10g will run as advertised on an unpatched Red Hat 3.0 ES or AS Service Pack 2 installation.

---

### Table 11-1. Raw Disks for ASM Disk Groups, the Voting Disk, and the OCR Disk

<table>
<thead>
<tr>
<th>Raw Device Name</th>
<th>Physical Device Name</th>
<th>Capacity</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/raw/raw1</td>
<td>/dev/sda1</td>
<td>10GB</td>
<td>ASM Disk #1: +DATA1</td>
</tr>
<tr>
<td>/dev/raw/raw2</td>
<td>/dev/sdb1</td>
<td>10GB</td>
<td>ASM Disk #1: +DATA1</td>
</tr>
<tr>
<td>/dev/raw/raw3</td>
<td>/dev/sdc1</td>
<td>10GB</td>
<td>ASM Disk #2: +RECOV1</td>
</tr>
<tr>
<td>/dev/raw/raw4</td>
<td>/dev/sdd1</td>
<td>10GB</td>
<td>ASM Disk #2: +RECOV1</td>
</tr>
<tr>
<td>/dev/raw/raw5</td>
<td>/dev/sde1</td>
<td>100MB</td>
<td>OCR Disk</td>
</tr>
<tr>
<td>/dev/raw/raw6</td>
<td>/dev/sdf1</td>
<td>100MB</td>
<td>Voting Disk</td>
</tr>
</tbody>
</table>

---
Memory and Disk Requirements
For each node in the cluster, a minimum of 512MB is recommended. The swap space should be at least twice this value, or 1GB. For a successful installation, there should be 400MB free in the /tmp file system.

The Oracle software itself requires approximately 2.5GB of disk space, and the default database files require another 1.2GB; the growth of your database depends, of course, on the application.

On your shared disk subsystem, you need two special partitions: one for a voting disk and one for the Oracle Cluster Registry (OCR). The voting disk is used by Oracle’s clustering software, Cluster Ready Services (CRS), to arbitrate ownership of the cluster in case of a private network failure. The OCR disk is used to maintain all metadata about the cluster: the cluster configuration and the cluster database configuration.

Kernel Parameters
Most of the “out of the box” kernel parameters are set correctly for Oracle except a few; ensure that the kernel parameters in Table 11-2 are set to the values provided in the table.

You can confirm that these values are in effect using the following command:

```
[root@rhes30]# /sbin/sysctl -a | egrep 'sem|shm|file-max|ip_local'
net.ipv4.ip_local_port_range = 1024 65000
kernel.sem = 250 32000 100 128
kernel.shmmni = 4096
kernel.shmall = 2097152
kernel.shmmax = 2147483648
fs.file-max = 65536
[root@rhes30]#
```

In a default Red Hat Linux 3.0 installation, some of these parameters are already set. For those values that vary from Table 11-2, simply append the parameter name and the value from the previous sample output to the file /etc/sysctl.conf and then run the /sbin/sysctl -p command to change the values immediately. After the next reboot, the values specified in /etc/sysctl.conf will be set automatically.

### TABLE 11-2. Oracle Database 10g Minimum Kernel Parameter Values

<table>
<thead>
<tr>
<th>Kernel Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel.shmall</td>
<td>2097152</td>
</tr>
<tr>
<td>kernel.shmmax</td>
<td>2147483648</td>
</tr>
<tr>
<td>kernel.shmmni</td>
<td>4096</td>
</tr>
<tr>
<td>kernel.sem</td>
<td>250 32000 100 128</td>
</tr>
<tr>
<td>fs.file-max</td>
<td>65536</td>
</tr>
<tr>
<td>net.ipv4.ip_local_port_range</td>
<td>1024 65000</td>
</tr>
</tbody>
</table>
Network Configuration

Each node in a RAC requires at least two network cards; one card is to connect to the public network for client communication, and the other is used for private network traffic between the nodes in the cluster. For the examples in this chapter, we will use the following /etc/hosts file:

```plaintext
# Do not remove the following line, or various programs
# that require network functionality will fail.
# 127.0.0.1   localhost.localdomain   localhost
192.168.2.99      rhes30
192.168.2.166     oltp
192.168.2.167     mw
192.168.2.101     oc1     #public1
192.168.1.101     poc1    #private1
192.168.2.176     voc1    #virtual1
192.168.2.102     oc2     #public2
192.168.1.102     poc2    #private2
192.168.2.177     voc2    #virtual2
192.168.2.103     oc3     #public3
192.168.1.103     poc3    #private3
192.168.2.178     voc3    #virtual3
```

If you are using a DNS server, these names and addresses should be hosted there also.

A couple things are worth noting at this point. We only need two network cards on each server; why is there a “virtual” address for each node? The virtual addresses support rapid connect-time failover, a concept we will explore in more detail later in this chapter. All client connections use these virtual addresses for their connections, and each RAC node’s listener will be listening on the virtual nodes instead of the public node names. Note also that each virtual address must be on the same subnet as the public address; the private interconnect network, however, is on its own private subnet.

TIP

Before proceeding with any Oracle software installations, be sure that you can connect from each node in the cluster to every other node using `ssh` without prompts for a password for both the root user and the oracle user.

User Accounts

Other than the root account, the only other account needed on your Linux server is the oracle account; in fact, in a production environment, you may not want any other user accounts on the server to prevent any inadvertent or intentional access of critical database files, control files, executables, or password files.
The groups oinstall and dba must exist on each node in the cluster, in addition to the oracle user. Use the following commands to create these if they do not already exist, and assign the oracle user to both groups, with oinstall as the primary group:

```
[root@rhes30 root]# /usr/sbin/groupadd oinstall
[root@rhes30 root]# /usr/sbin/groupadd dba
[root@rhes30 root]# /usr/sbin/useradd -g oinstall -G dba oracle
[root@rhes30 root]# passwd oracle
Changing password for user oracle.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[root@rhes30 root]#
```

For the oracle user, set up the default environment in the logon script; this sample logon script assumes the bash shell (Bourne Again Shell) on Red Hat:

```
# .bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
   . ~/.bashrc
fi

# User specific environment and startup programs
PATH=$PATH:$HOME/bin
export PATH
unset USERNAME
umask 022
ORACLE_BASE=/u01/app/oracle
# ORACLE_HOME is set after installation with OUI.
# ORACLE_HOME=$ORACLE_BASE/product/10.1.0/DB10gHome
# ORACLE_SID different on each node;
# same database, different instance.
ORACLE_SID=rac1
LD_ASSUME_KERNEL=2.4.19
PATH=$ORACLE_HOME/bin:$PATH
export ORACLE_BASE ORACLE_HOME ORACLE_SID PATH LD_ASSUME_KERNEL
```

**NOTE**

`LD_ASSUME_KERNEL` does not need to be set in the login profile if you are using Oracle 10g update 1 or later.

Make sure that the value for `ORACLE_SID` is unique on each node! As we install additional products such as CRS and create the RAC instances, we will make changes to this logon script as appropriate. Also, ensure that `LD_ASSUME_KERNEL` is set to 2.4.19 in root’s `.bash_profile`; this setting is required for Oracle installations on Red Hat 3.0.

**SAMPLE CHAPTER FOR REVIEW ONLY.**

To set up user equivalence between nodes in the cluster, use either the .rhosts or /etc/hosts.equiv file to support the rsh and rcp commands; better yet, and more secure, ensure that ssh and scp are configured for all nodes in the cluster. Starting with Oracle 10g, the OUI will use ssh and scp if possible, and fall back to rsh and rcp if necessary. Configuring ssh using the ssh-keygen utility is beyond the scope of this book; consult with your Unix or Linux system administrator to configure ssh and scp.

Software Directories
Because we are using ASM in these examples for RAC storage, only one directory, /u01/app/oracle, needs to be created on the local storage to hold the Oracle Database and the CRS software. The disk volume on which this directory resides must have at least 2.5GB of space for the database and CRS software. Use these commands to create this directory and assign the correct permissions:

```
[root@rhes30 root]# mkdir -p /u01/app/oracle
[root@rhes30 root]# chown -R oracle:oinstall /u01/app/oracle
[root@rhes30 root]# chmod -R 775 /u01/app/oracle
```

Software Installation
Whether you are creating a two-node or a 16-node cluster, the procedure is the same; if you have configured your servers as detailed in the previous sections, the installations you will perform in the following sections will automatically push to each node you specify in the configuration screens for Cluster Ready Services and the Oracle Database software itself.

Therefore, the following discussion is divided into three parts: CRS to prepare the clustering environment, Oracle Database 10g software installation, and creating an instance of the database on each node in the cluster. As we review each screen of each installation, we will provide background and explanation as required so that you’re prepared to make adjustments to your environment once the installations are complete.

In many cases, the installation steps for the database software, the database itself, and ASM are similar or identical to that you’ve already seen in Chapters 1 and 4; in the following sections, we’ll focus on the differences you’ll see for a RAC installation.

Cluster Ready Services
As mentioned earlier in this chapter, CRS should be installed in its own home directory called CRS_HOME. As part of the CRS installation, you will have to configure two particular locations that are not specific to any instance, but are used by the cluster itself: the Oracle Cluster Registry and the voting disk. The Oracle Cluster Registry (OCR) is the location where the cluster stores its metadata, and 100MB is more than enough disk space to hold the metadata for a very large cluster. The voting disk is used by the cluster to resolve situations where one or more nodes in the cluster lose contact with other nodes in the cluster over the private interconnect. In this way, you have a way to shut off one node or one group of nodes from writing to the shared disk files because it assumes it is in control of the shared storage. As with the OCR disk, the voting disk requires no more than 100MB.
Oracle Database 10g DBA Handbook

The locations of the OCR disk and the voting disk must be on separate raw devices, even when you are using ASM for your other database files; however, if you are using OCFS, the OCR disk and the voting disk can exist as files on an OCFS volume. In the examples that follow, we will use raw devices for the OCR and voting disks; this is the Oracle-recommended method.

The CRS software is provided on a single CD-ROM. After you mount the CD, run the script ./runInstaller as the oracle user. The first screen you will see after the welcome screen and the Oracle inventory location screen is shown in Figure 11-2.

The installation for CRS is similar to that for a database install; you specify a home directory for the executables. In this case, you will use a home directory called CRS10gHome, which must be different from the Oracle Database home.

After selecting the language for the installation, you provide a name for the cluster along with the public and private node names. As you can see in Figure 11-3, you specify the cluster name as lec1 and provide the public and private node names as defined in the /etc/hosts file you defined earlier in this chapter.

On the next installation screen, shown in Figure 11-4, you specify which of your network devices is to be used as the public interface (to connect to the public network) and which will be used for the private interconnect to support cache fusion and the cluster heartbeat. As noted earlier, you can have more than one public and more than one private interface; on this screen you can also mark an interface to not be used at all by CRS.

FIGURE 11-2. Executable file locations
FIGURE 11-3. Cluster configuration

FIGURE 11-4. Private interconnect enforcement

SAMPLE CHAPTER FOR REVIEW ONLY.
In Figure 11-5, you specify /dev/raw/raw5 as the raw disk for the Oracle Cluster Registry. The OCR is a metadata repository for the cluster configuration, keeping track of things like where a particular service is running, if it is running, and so forth.

In a similar fashion, you specify /dev/raw/raw6 as the raw disk for the voting disk. The processes known as Cluster Synchronization Services (CSS) use the voting disk to arbitrate cluster ownership and interprocess communications in a cluster environment. In a single-instance environment, CSS facilitates communications between the ASM instance and the RDBMS instance.

The next screen, shown in Figure 11-7, directs you to run the script orainstRoot.sh on each node in the cluster to create directories and set permissions on those directories; this script must be run as root.
Chapter 11: Real Application Clusters

FIGURE 11-6. Voting disk location

FIGURE 11-7. Script to run as root
After the pre-installation summary screen shown in Figure 11-8 appears, you click the Install button and the installation begins. In addition to installing the software on the node where you initiated the installation, the installed directory structure is copied to every node on the cluster.

After installation is complete, you are once again prompted to run a script as root on each node of the cluster. Here is the output from running the command on the first node:

```
L 11-6
[root@oc1 mnt]# /u01/app/oracle/product/10.1.0/CRS10gHome/root.sh
Checking to see if Oracle CRS stack is already up...
/etc/oracle does not exist. Creating it now.
Setting the permissions on OCR backup directory
Oracle Cluster Registry configuration upgraded successfully
WARNING: directory '/u01/app/oracle/product/10.1.0' is not owned by root
WARNING: directory '/u01/app/oracle/product' is not owned by root
assigning default hostname oc1 for node 1.
assigning default hostname oc2 for node 2.
assigning default hostname oc3 for node 3.
Successfully accumulated necessary OCR keys.
Using ports: CSS=49895 CRS=49896 EVMC=49898 and EVMR=49897.
node <nodenumber>: <nodename> <private interconnect name> <hostname>
node 1: oc1 poc1 oc1
node 2: oc2 poc2 oc2
node 3: oc3 poc3 oc3
```

FIGURE 11-8. Pre-installation summary

After the pre-installation summary screen shown in Figure 11-8 appears, you click the Install button and the installation begins. In addition to installing the software on the node where you initiated the installation, the installed directory structure is copied to every node on the cluster.

After installation is complete, you are once again prompted to run a script as root on each node of the cluster. Here is the output from running the command on the first node:
Creating OCR keys for user 'root', privgrp 'root'...
Operation successful.
Now formatting voting device: /dev/raw/raw6
Successful in setting block0 for voting disk.
Format complete.
Adding daemons to inittab
Preparing Oracle Cluster Ready Services (CRS):
Expecting the CRS daemons to be up within 600 seconds.
CSS is active on these nodes.
oc1
CSS is inactive on these nodes.
oc2
oc3
Local node checking complete.
Run root.sh on remaining nodes to start CRS daemons.
[root@oc1 mnt]#

The voting and OCR disks are initialized the first time this script is run. When you run the script on the other two nodes, you see similar results, except for the voting disk and OCR disk initialization; here is the output for the second node, oc2:

[root@oc1 etc]# ssh oc2
Last login: Thu Oct 14 19:32:44 2004 from oc1
[root@oc2 root]# /u01/app/oracle/product/10.1.0/CRS10gHome/root.sh
Checking to see if Oracle CRS stack is already up...

clscfg: EXISTING configuration version 2 detected.
clscfg: version 2 is 10G Release 1.
assigning default hostname oc1 for node 1.
assigning default hostname oc2 for node 2.
assigning default hostname oc3 for node 3.
Successfully accumulated necessary OCR keys.
Using ports: CSS=49895 CRS=49896 EVMC=49898 and EVMR=49897.
node <nodenumber>: <nodename> <private interconnect name> <hostname>
node 1: oc1 poc1 oc1
node 2: oc2 poc2 oc2
node 3: oc3 poc3 oc3
clscfg: Arguments check out successfully.

NO KEYS WERE WRITTEN. Supply -force parameter to override.
-force is destructive and will destroy any previous cluster configuration.
Oracle Cluster Registry for cluster has already been initialized
Adding daemons to inittab
Preparing Oracle Cluster Ready Services (CRS):
Expecting the CRS daemons to be up within 600 seconds.
CSS is active on these nodes.
oc1
oc2
CSS is inactive on these nodes.

oc3

Local node checking complete.

Run root.sh on remaining nodes to start CRS daemons.

[root@oc2 root]#

On the third and final node of our three-node cluster, we see similar messages with a confirmation that the CRSD and EVMD processes have started:

[root@oc2 root]# ssh oc3

Last login: Thu Oct 14 20:40:15 2004 from oc2

[root@oc3 root]# /u01/app/oracle/product/10.1.0/CRS10gHome/root.sh

Checking to see if Oracle CRS stack is already up...

/etc/oracle does not exist. Creating it now.

Adding daemons to inittab

Preparing Oracle Cluster Ready Services (CRS):

Expecting the CRS daemons to be up within 600 seconds.

CSS is active on these nodes.

oc1

oc2

oc3

CSS is active on all nodes.

Waiting for the Oracle CRSD and EVMD to start

Oracle CRS stack installed and running under init(1M)

[root@oc3 root]#

NOTE

A discussion of CRSD and EVMD is beyond the scope of this book; see the Oracle Press book Oracle Database 10g High Availability with RAC, Flashback & Data Guard, by Hart and Jesse.

You can verify that the CRS installation completed successfully by changing to the executable directory for CRS (in CRS10gHome) on any node and then running this command:

[oracle@oc1 oracle]$ cd /u01/app/oracle/product/10.1.0/CRS10gHome/bin
[oracle@oc1 bin]$ olsnodes -n

oc1 1
oc2 2
oc3 3

[oracle@oc1 bin]$

The output from olsnodes shows the node names and node numbers that are running CRS.

Database Software Install

Once you have the cluster software running successfully on each node, you are ready to install the database software into the same directory on each node. In this section, we’ll primarily focus on the parts of the database software install that differ from the single-instance installation you performed in Chapter 1.

SAMPLE CHAPTER FOR REVIEW ONLY.

Although we can create a database at the same time we install the Oracle software, we will only install the software now and run the Database Configuration Assistant later to create the database. From the root directory of the database installation CD, run the script ./runInstaller as the oracle user, just as you did for the CRS install. The first screen you will see after the welcome screen and the Oracle inventory location screen is shown in Figure 11-9.

Although you can install the software in any directory, make sure that this directory is available to the user oracle on all nodes in the cluster. In addition, make sure that this directory is not the same as the CRS installation directory.

As shown in Figure 11-10, if the installer detects clustering software running on the node, it gives you the option to install the software on the entire cluster or to perform a single-instance install. In this case, you select all the nodes you configured earlier as part of the cluster.

As with a single-instance installation, you can select either the Enterprise Edition or the Standard Edition. Choose the edition that is compatible with your site’s licensing options. After selecting the edition, the installer confirms that the environment for the Oracle Database software is configured correctly, as you can see in Figure 11-11.

You may get some errors or warnings about modules or products that are missing or the wrong version; in Figure 11-11, the installer did not find the openmotif package at all. You can verify that the package is installed by using a command similar to the following:

```
rpm -q openmotif
```

```
openmotif-2.2.3-3.RHEL3
```

[root@oc1 root]#

![Figure 11-9. Oracle database file locations](image-url)
FIGURE 11-10.  Hardware cluster node locations

FIGURE 11-11.  Platform configuration checks
Because you have a later version of openmotif on your system, you can safely specify that this package does exist and click Next. On the next screen, you do not specify a starter database; you will create the database with DBCA later.

The summary screen you see in Figure 11-12 is nearly identical to the one you see in a single-instance installation, except that you are installing the software on more than one node in the cluster.

The subsequent screens detail the progress of the installation. Upon completion, you are prompted to run a new root.sh script on each node in the cluster. Here are the results of running the script on the first node (you must be logged on as root to run this script):

```
L11-11 [root@oc1 root]# /u01/app/oracle/product/10.1.0/DB10gHome/root.sh
Running Oracle10 root.sh script...

The following environment variables are set as:

  ORACLE_OWNER= oracle
  ORACLE_HOME= /u01/app/oracle/product/10.1.0/DB10gHome

Enter the full pathname of the local bin directory: [/usr/local/bin]:

  Copying dbhome to /usr/local/bin ...
  Copying oraenv to /usr/local/bin ...
  Copying coraenv to /usr/local/bin ...

Creating /etc/oratab file...

Adding entry to /etc/oratab file...

Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created
```

FIGURE 11-12. Database pre-installation summary
Finished running generic part of root.sh script.
Now product-specific root actions will be performed.
[root@oc1 root]#

For the first node in the cluster only, this script spawns the Virtual IP Configuration Assistant (VIPCA) to configure the virtual IP addresses you will use for application failover and the Enterprise Manager agent. After the VIPCA welcome screen, the Network Interfaces screen, shown in Figure 11-13, provides you with a list of the physical network interfaces available on this server. Choose the public interface and click Next.

In Figure 11-14, you enter one of the virtual IP node names you specified earlier in the /etc/hosts file (in this case, voc1). The installer automatically fills in the other alias names and addresses from /etc/hosts.

After you see the pre-installation summary screen, click Next; Figure 11-15 shows you the installation progress screen for all three nodes.

After the installation is complete, you open a session on each of the other nodes in the cluster and run root.sh on other nodes. The script produces similar results to those you saw on the first node, but because you already configured with VIPCA, you get the message

FIGURE 11-13. VIPCA network interfaces
FIGURE 11-14. VIPCA addresses for cluster nodes

FIGURE 11-15. VIPCA installation progress
Now that the software is installed, you can proceed with creating the database along with one instance per node in the next section. As mentioned earlier, you could have created a starter database with the software installation, but launching the Database Configuration Assistant gives you more options and more flexibility when creating a database and a new instance.

Creating the RAC with the Database Creation Assistant

Launching the Database Creation Assistant (DBCA) for installing a RAC is much the same as launching DBCA for a single instance; if DBCA detects cluster software installed, it gives you the option to install a RAC or a single instance, as you can see in Figure 11-16, after DBCA is launched:

```
[oracle@oc1 oracle] dbca &
```

After selecting the option to create a database, you see the dialog shown in Figure 11-17; select the nodes that will participate in the cluster. In this case, you select all nodes.

On the next screen, choose the type of database: data warehouse, general purpose, transaction processing, or custom. For the purposes of creating a RAC, the type of database you select will not change the configuration of the cluster.

---

**FIGURE 11-16.  DBCA instance type options**
Chapter 11: Real Application Clusters

On step 4 of DBCA, you give the cluster database a name and a SID prefix, just as you would with a standalone database installation. Step 5 asks you if you want to configure your RAC to use EM Database Control, and whether to configure the database with Enterprise Manager or Grid Control. Specify your mail server and e-mail notification address. In step 6, you specify the password for the privileged accounts in the database: SYS, SYSTEM, DBSNMP, and SYSMAN. In step 7, you specify ASM as your database file storage method. Finally, in step 8 you specify the parameters for the ASM instance, as you did in Chapter 4.

Automatic Storage Management (ASM) instances, although available for storage management with standalone Oracle instances, are ideal for use with RAC. ASM eliminates both the need for configuring raw devices (raw devices are mapped once within an ASM instance and subsequently are available for all nodes in the cluster) and the need for a cluster file system for database files. Cluster file systems such as Oracle Cluster File System (OCFS) are still available if you want your ORACLE_HOME on a cluster file system instead of a copy on each node in the cluster; Oracle best practices recommends that each node have its own local copy of the Oracle software. More details on how to configure and use ASM can be found in Chapter 4. If you use ASM, it only needs to be configured once, during these steps.
NOTE

OCFS version 2.x supports a shared Oracle Home.

The next few screens track the progress of the creation of the ASM instance. After this process is completed, you are prompted to create the first ASM disk group, as you can see in Figure 11-18. You choose two of the raw devices available to be the DATA1 disk group using normal redundancy. Additionally, you create the RECOV1 disk group using the remaining two disk groups; this disk group will be used to mirror the control file and redo log files as well as to host the Flash Recovery Area. In Figure 11-19, you specify DATA1 as the disk group for database storage. In Figure 11-20, you specify RECOV1 for the Flash Recovery Area.
Chapter 11: Real Application Clusters

FIGURE 11-19. Selecting the ASM disk group for storage

FIGURE 11-20. Selecting the ASM disk for Flash Recovery Area
In the next step, shown in Figure 11-21, you will see another option that you did not see in a single-instance install: creating a custom RAC service. The service you create in Figure 11-21 adds another entry for racsvc in tnsnames.ora, in addition to the standard entry created (rac) when the tnsnames.ora file is created, as you can see in this listing:

```
rac =
  (description =
    (address = (protocol = tcp)(host = voc1)(port = 1521))
    (address = (protocol = tcp)(host = voc2)(port = 1521))
    (address = (protocol = tcp)(host = voc3)(port = 1521))
    (load_balance = yes)
    (connect_data =
      (server = dedicated)
      (service_name = rac.world)
    )
  )

listeners_rac =
  (address_list =
    (address = (protocol = tcp)(host = voc1)(port = 1521))
    (address = (protocol = tcp)(host = voc2)(port = 1521))
    (address = (protocol = tcp)(host = voc3)(port = 1521))
  )
```

**FIGURE 11-21.** Creating a RAC service

In the next step, shown in Figure 11-21, you will see another option that you did not see in a single-instance install: creating a custom RAC service.

The service you create in Figure 11-21 adds another entry for racsvc in tnsnames.ora, in addition to the standard entry created (rac) when the tnsnames.ora file is created, as you can see in this listing:
Chapter 11: Real Application Clusters

```plaintext

racsvc =
(description =
(address = (protocol = tcp)(host = voc1)(port = 1521))
(address = (protocol = tcp)(host = voc2)(port = 1521))
(address = (protocol = tcp)(host = voc3)(port = 1521))
(load_balance = yes)
(connect_data =
(server = dedicated)
(service_name = racsvc.world)
(failover_mode =
(type = select)
(method = basic)
(retries = 180)
(delay = 5)
)
)
)

rac3 =
(description =
(address = (protocol = tcp)(host = voc3)(port = 1521))
(connect_data =
(server = dedicated)
(service_name = rac.world)
(instance_name = rac3)
)
)

rac2 =
(description =
(address = (protocol = tcp)(host = voc2)(port = 1521))
(connect_data =
(server = dedicated)
(service_name = rac.world)
(instance_name = rac2)
)
)

rac1 =
(description =
(address = (protocol = tcp)(host = voc1)(port = 1521))
(connect_data =
(server = dedicated)
(service_name = rac.world)
(instance_name = rac1)
)
)
```

SAMPLE CHAPTER FOR REVIEW ONLY.
Notice also that each node in the RAC has its own entry so that you may connect to a specific node when necessary. Note also that the hostnames for each node are using the virtual node names instead of the physical node names.

The entry for racsvc has a few additional parameters for FAILOVER_MODE; these modes and their values are defined in the following list:

- **type**  The type of failover. Specifying *session* creates a new session for the client, but this does not preserve the position in a cursor when you are running a SELECT statement. Specifying *select* preserves the state of the cursor during a SELECT, but involves extra overhead on the client side. The default, *none*, disables failover functionality.

- **method**  How fast failover occurs. Using a value of *basic* establishes connections when the failover occurs, and this incurs no overhead on the backup server(s). A value of *preconnect* provides faster failover, but as the name implies, it uses resources on the backup server(s) even when no failover scenario is active.

- **retries**  The number of times to attempt to connection after a failover.

- **delay**  The amount of time, in seconds, to wait between connection attempts when a failover scenario is active.

Later in this chapter, we will show you how connecting to the racsvc service ensures high availability for client connections when the client connects to a node and the node fails.

The next few screens are the same as for a single-instance database installation; see Chapter 1 for the options available on these screens. Figure 11-22 summarizes the cluster database installation, which includes the location of the database’s SPFILE on the DATA1 disk group.
Once the installation is complete, EM Database Control is automatically configured and started, just as it is with a single-instance installation; however, you can manage the entire cluster and not just individual nodes.

NOTE

*Installing and configuring Enterprise Manager Grid Control 10g is beyond the scope of this book; see the book *Oracle Database 10g High Availability with RAC, Flashback & Data Guard* for more information.*

In Figure 11-23, you can see that the EM Database Control login connects you to the entire cluster instead of just a single node.

As with a single-instance database, you can see the status of the database on the database home page; again, the difference is that EM Database Control is cluster-aware. In Figure 11-24, you can see the status of the Oracle instances in the cluster, and in Figure 11-25, you see the status of the cluster itself; remember that CRS is managing the cluster as a separate product from the database.

---

**FIGURE 11-23.** *EM DB Control cluster login*
FIGURE 11-24. **EM DB Control cluster database status**

FIGURE 11-25. **EM DB Control cluster hardware status**
In Figure 11-26, you see the characteristics of one of the redo log members for the cluster; note that we placed one of the group’s members on the DATA1 disk group and the other on the RECOV1 disk group. Because each disk group is mirrored on two different raw devices, you have the equivalent of four-way redundancy for the members of your redo log group.

RAC Characteristics
A RAC instance is different in many ways from a standalone instance; in this section, we will show you the initialization parameters that are specific to a RAC. In addition, we’ll show you some of the data dictionary views and dynamic performance views that are either unique to a RAC or have columns that are only populated when the instance is part of a RAC.

Server Parameter File Characteristics
As you saw previously in the section “Creating the RAC with the Database Configuration Assistant,” the server parameter file (SPFILE) resides on the DATA1 disk group and therefore is shared by each node in the cluster. Within the SPFILE, you can assign different values for given parameters on an instance-by-instance basis; in other words, the value for an initialization parameter can differ between instances. If an initialization parameter is the same for all nodes in the cluster, it is prefixed with “*.”; otherwise, it is prefixed with the node name.
In this example, the physical memory on the cluster server oc2 is temporarily reduced. Therefore, to reduce the demands of the instance on the server, you will change the value of SHARED_POOL_SIZE for the instance rac2 the next time this instance is restarted:

```sql
SQL> select sid, name, value
2  from v$spparameter where name = 'shared_pool_size';
SID   NAME                 VALUE
---------- -------------------- ---------------
*       shared_pool_size     65614720

SQL> alter system set shared_pool_size=48M
2    scope=spfile sid='rac2';
System altered.

SQL> select sid, name, value
2  from v$spparameter where name = 'shared_pool_size';
SID   NAME                 VALUE
---------- -------------------- ---------------
*       shared_pool_size     65614720
  rac2  shared_pool_size     50331648
```

Once the memory issue has been resolved, you can restore the size of the shared pool on the rac2 instance as follows:

```sql
SQL> alter system set shared_pool_size=64M
2    scope=spfile sid='rac2';
System altered.
SQL>
```

RAC-Related Initialization Parameters

A number of initialization parameters are used only in a RAC environment. Although these initialization parameters exist in any instance, in a single-instance environment they are either null or have a value of 1 (for example, INSTANCE_NUMBER). In Table 11-3, we give you an overview of some of the key RAC-related initialization parameters.

Dynamic Performance Views

In a single-instance environment, all dynamic performance views that begin with V$ have a corresponding view beginning with GV$, with the additional column INST_ID always set to 1. For a RAC environment with two nodes, the GV$ views have twice as many rows as the corresponding V$ views; for a three-node RAC, there are three times as many rows, and so forth. In the sections that follow, we’ll review some of the V$ dynamic performance views that show the same contents regardless of the node you are connected to, along with some of the GV$ views that can show you the contents of the V$ views on each node without connecting to each node explicitly.
Common Database File Views

Some dynamic performance views are the same whether you’re in a RAC environment or a single-instance environment; the ASM configuration is a perfect example of this. In this query, you want to verify that all your database files are stored in one of the two ASM disk groups, +DATA1 or +RECOV1:

```sql
SQL> select name from v$datafile union
2  select name from v$tempfile union
3  select member from v$logfile union
4  select name from v$controlfile union
5  select name from v$flashback_database_logfile;
```

<table>
<thead>
<tr>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>+DATA1/rac/controlfile/current.260.3</td>
</tr>
<tr>
<td>+DATA1/rac/datafile/example.264.1</td>
</tr>
<tr>
<td>+DATA1/rac/datafile/sysaux.257.1</td>
</tr>
<tr>
<td>+DATA1/rac/datafile/system.256.1</td>
</tr>
<tr>
<td>+DATA1/rac/datafile/undotbs1.258.1</td>
</tr>
<tr>
<td>+DATA1/rac/datafile/undotbs2.265.1</td>
</tr>
<tr>
<td>+DATA1/rac/datafile/undotbs3.266.1</td>
</tr>
<tr>
<td>+DATA1/rac/datafile/users.259.1</td>
</tr>
<tr>
<td>+DATA1/rac/onlinelog/group_1.261.1</td>
</tr>
<tr>
<td>+DATA1/rac/onlinelog/group_2.262.1</td>
</tr>
<tr>
<td>+DATA1/rac/onlinelog/group_3.269.1</td>
</tr>
</tbody>
</table>

 Initialization Parameter | Description
--------------------------|---------------------------------------------------
 INSTANCE_NUMBER           | Unique number identifying this instance in the cluster. 
 INSTANCE_NAME            | The unique name of this instance within the cluster; typically the cluster name with a numeric suffix. 
 CLUSTER_DATABASE         | This parameter is TRUE if this instance is participating in a RAC environment. 
 CLUSTER_DATABASE_INSTANCES | The number of instances configured for this cluster, whether each instance is active or not. 
 ACTIVE_INSTANCE_COUNT    | Specifies the primary instance in a two-node cluster; otherwise it is the number of instances in the cluster. 
 CLUSTER_INTERCONNECTS    | Specifies the network used for the cluster’s IPC traffic. 
 MAX_COMMIT_PROPAGATION_DELAY | Controls how fast committed transactions are propagated to other nodes. 

**TABLE 11.3. RAC-Related Initialization Parameters**
Cluster-Aware Dynamic Performance Views
The GV$ views make it easy to view each instance’s characteristics in a single SELECT statement, while at the same time filtering out nodes that you do not want to see; these views also make it easier to aggregate totals from some or all of the nodes in the cluster, as in this example:

```
L 11-18
SQL> select nvl(to_char(inst_id),'TOTAL') INST#, 
       2       count(inst_id) SESSIONS from gv$session 
       3      group by rollup(inst_id) 
       4      order by inst_id;
```

```
INST#  SESSIONS
------- --------
 1       29
 2       39
 3       26
TOTAL   94
```

4 rows selected.

From this query, you can see the number of sessions per instance and the total number of instances for the cluster using the view GV$SESSION.

RAC Maintenance
Most of the maintenance operations you perform on a single-node instance apply directly to a multiple-node RAC environment. In this section, we will review the basics for maintaining a RAC—including starting up a RAC and discussing how redo logs and undo tablespaces work—and then work through an example of an instance failure scenario using Transparent Application Failover (TAF) as well as rebuilding a failed node and adding it back to the cluster.
Starting Up a RAC
Starting up a RAC is not much different from starting up a standalone instance; the nodes in a RAC can start up in any order, and they can be shut down and started up at any time with minimal impact to the rest of the cluster. During database startup, first the ASM instance starts and mounts the shared disk groups; next, the RDBMS instance starts and joins the cluster.

On Unix, the file /etc/oratab can be modified to auto-start the instances (both the ASM instance and the RDBMS instance) on each cluster:

L11-19
# This file is used by ORACLE utilities. It is created by root.sh
# and updated by the Database Configuration Assistant when creating
# a database.
#
# A colon, ':', is used as the field terminator. A new line terminates
# the entry. Lines beginning with a pound sign, '#', are comments.
#
# Entries are of the form:
# $ORACLE_SID:$ORACLE_HOME:<N|Y>:
#
# The first and second fields are the system identifier and home
# directory of the database respectively. The third field indicates
# to the dbstart utility that the database should, "Y", or should not,
# "N", be brought up at system boot time.
#
# Multiple entries with the same $ORACLE_SID are not allowed.
#
*:/u01/app/oracle/product/10.1.0/DB10gHome:N
+ASM1:/u01/app/oracle/product/10.1.0/DB10gHome:Y
rac:/u01/app/oracle/product/10.1.0/DB10gHome:Y

Redo Logs in a RAC Environment
As with a single-node instance, online redo logs are used for instance recovery in a RAC environment; each instance in a RAC environment has its own set of online redo log files that are used to roll forward all information in the redo logs and then roll back any uncommitted transactions initiated on that node using the undo tablespace.

Even before the failed instance has restarted, one of the surviving instances detects the instance failure and uses the online redo log files to ensure that no committed transactions are lost; if this process completes before the failed instance restarts, the restarted instance does not need instance recovery. Even if more than one instance fails, all that is required for instance recovery is one remaining node. If all instances in a RAC fail, the first instance that starts up will perform instance recovery for the database using the online redo log files from all instances in the cluster.

If media recovery is required and the entire database must be recovered, all instances except for one must be shut down and media recovery is performed from a single instance. If you are recovering noncritical database files, all nodes may be up as long as the tablespaces containing the files to be recovered are marked as OFFLINE.
Undo Tablespaces in a RAC Environment

As with redo logs, each instance in a RAC environment must have its own undo tablespace on a shared drive or disk group. This undo tablespace is used for rolling back transactions during normal transactional operations or during instance recovery. In addition, the undo tablespace is used by other nodes in the cluster to support read consistency for transactions that are reading rows from a table on node rac2 while a data-entry process on node rac1 makes updates to the same table and has not yet committed the transaction. The user on rac2 needs to see the before-image data stored in rac1’s undo tablespace. This is why all undo tablespaces must be visible to all nodes in the cluster.

Failover Scenarios and TAF

If you have configured your client correctly and the instance to which the client is connected to fails, the client connection is rapidly switched to another instance in the cluster and processing can continue with only a slight delay in response time.

Here is the tnsnames entry for the service racsvc we created earlier:

```sql
L 11-20 racsvc =
  (description =
    (address = (protocol = tcp)(host = voc1)(port = 1521))
    (address = (protocol = tcp)(host = voc2)(port = 1521))
    (address = (protocol = tcp)(host = voc3)(port = 1521))
    (load_balance = yes)
    (connect_data =
      (server = dedicated)
      (service_name = racsvc.world)
      (failover_mode =
        (type = select)
        (method = basic)
        (retries = 180)
        (delay = 5)
      )
    )
  )
)

We will show you what happens and how you will know if a session is connected to the cluster and its instance fails. First, you connect to the cluster via racsvc and find out the node and instance that you are connected to:

```sql
L 11-21 SQL> connect rjb/rjb@racsvc;
Connected.
SQL> select instance_name, host_name, failover_type,
  2     failover_method, failed_over
  3 from v$instance
  4 cross join
  5 (select failover_type, failover_method, failed_over
```
You are using the columns from V$INSTANCE to give you the instance name and host name that you are connected to and then joining this to V$SESSION and retrieving the columns related to failover, which are only populated in a RAC environment. In this case, the session has not yet failed over, and the failover type is BASIC, as we specified when we created the service.

Next, you will shut down instance rac1 from another session while you are still connected to the first session:

```
L 11-22
SQL> connect system/manager@rac1
Connected.
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
```

Back at your user session, you rerun the query to find out what node you are connected to:

```
L 11-23
SQL> select instance_name, host_name, failover_type,
    2   failover_method, failed_over
    3 from v$instance
    4 cross join
    5 (select failover_type, failover_method, failed_over
    6    from v$session
    7 where username = 'RJB');

INSTANCE_NAME HOST_NAME FAILOVER_TYPE FAILOVER_METHOD FAILED_OVER
------------- --------- ------------- --------------- -----------
rac3          oc3       SELECT        BASIC           YES

SQL>
```

If you were running a query at the time the instance was shut down, your query would pause for a second or two and then continue as if nothing happened.

**RAC Node Failure Scenario**

One of the benefits of a RAC environment is your ability to add or remove nodes to meet changing resource demands. One server that is underutilized in one business unit may be needed in another business unit that is entering its peak processing period. Adding or removing a node in a RAC
Oracle Database 10g DBA Handbook

An environment may also be driven by a failure of a node; while the remaining nodes in the cluster service ongoing requests, you will have to repair or replace the missing node and add it back to the cluster without bringing down the rest of the cluster.

In this section, we’ll show you the steps required to remove a node’s metadata from the cluster registry and then rebuild a node and add it back to the cluster. The assumption in this scenario is that the local hard disk of the third cluster node is damaged beyond repair; therefore, you will rebuild the node from scratch and add it to the cluster registry. After this step, you will reinstall the Oracle software and create the instance as part of the database cluster.

Remove the Instance

Even if the instance on the failed server is not available, you still want to remove any traces of the instance from the remaining nodes in the cluster. You can use the `srvctl` command to remove the instance from the cluster, as in this example:

```
[oracle@oc1 oracle]$ srvctl remove instance -d rac -i rac3
Remove instance rac3 for the database rac? [y/[n]] y
```

The parameter `-d rac` specifies the RAC to be modified, and `-i rac3` specifies the instance to be removed from the RAC.

Remove the Node from the Cluster

To remove the server itself from the cluster, execute the `rootdeletenode.sh` command from the CRS10gHome directory, specifying both the node name and the CRS-assigned node number, as in the following example:

```
[root@oc1 root] # cd /u01/app/oracle/product/10.1.0/CRS10gHome/bin
[root@oc1 bin]# ./olsnodes -n
oc1 1
oc2 2
oc3 3
[root@oc1 bin]# cd ../install
[root@oc1 install]# ./rootdeletenode.sh oc3,3
clscfg: EXISTING configuration version 2 detected.
clscfg: version 2 is 10G Release 1.
Successfully deleted 13 values from OCR.
Key SYSTEM.css.interfaces.nodeoc3 marked for deletion is not there.
Ignoring.
Successfully deleted 5 keys from OCR.
Node deletion operation successful.
'oc3,3' deleted successfully
[root@oc1 install]# cd .. /bin
```

SAMPLE CHAPTER FOR REVIEW ONLY.
You also need to remove the node from the list of node locations maintained by the Oracle Universal Installer (OUI); in the directory $ORACLE_BASE/oraInventory/ContentsXML, identify any files that reference the deleted node, such as this example in the file inventory.xml:

```
<HOME NAME="CRS10gHome"
     LOC="/u01/app/oracle/product/10.1.0/CRS10gHome"
     TYPE="O" IDX="1" CRS="true">
  <NODE_LIST>
    <NODE NAME="oc1"/>
    <NODE NAME="oc3"/>
    <NODE NAME="oc2"/>
  </NODE_LIST>
</HOME>
```

**NOTE**

See MetaLink Note 269320.1 for other procedures specific to your environment that may need to be performed to remove a node from a cluster.

Note that you have specified the node name of the server that hosts the instance. There are now only two nodes in your CRS clusterware environment.

**Install Operating System Software**

The next step is to reinstall the server software and prepare the environment as you did in the examples earlier in this chapter in the section “Operating System Configuration.” At the end of this process, you will have the Oracle directories created along with the oracle user account, but without the CRS and database software installed. You will also assign the public, private, and virtual IP addresses using the same addresses you used when this node was first created. As a result, you will not have to change the /etc/hosts file on the remaining nodes in the cluster.

**Add the Node to the Cluster with CRS**

The node is ready to add to the cluster at the clusterware layer so that the other nodes in the cluster consider it to be a part of the cluster again. From one of the remaining nodes in the cluster, change to $CRS_HOME/oui/bin and run the `addNode.sh` command, which launches OUI and prompts you for the new node, as you can see in Figure 11-27.
After presenting a summary of the existing nodes and the node to be added, you click Next and the CRS files are copied to the new node. To start the services on the new node, you are prompted to run rootaddnode.sh on the active node and root.sh on the new node, as you can see in this example:

```
L11-27 [root@oc1 oraInventory]#
    /u01/app/oracle/product/10.1.0/CRS10gHome/rootaddnode.sh
clscfg: EXISTING configuration version 2 detected.
clscfg: version 2 is 10G Release 1.
Attempting to add 1 new nodes to the configuration
Using ports: CSS=49895 CRS=49896 EVMC=49898 and EVMR=49897.
node <nodenumber>: <nodename> <private interconnect name> <hostname>
node 3: oc3 poc3 oc3
Creating OCR keys for user 'root', privgrp 'root'..
Operation successful.
[root@oc1 oraInventory]# ssh oc3
Last login: Sun Oct 31 21:12:08 2004 from oc2
[root@oc3 root]# /u01/app/oracle/product/10.1.0/CRS10gHome/root.sh
Checking to see if Oracle CRS stack is already up...
/etc/oracle does not exist. Creating it now.
Setting the permissions on OCR backup directory
```

FIGURE 11-27. Specifying the node to add using OUI
Oracle Cluster Registry configuration upgraded successfully
WARNING: directory '/u01/app/oracle/product/10.1.0' is not owned by root
WARNING: directory '/u01/app/oracle/product' is not owned by root
clscfg: EXISTING configuration version 2 detected.
clscfg: version 2 is 10G Release 1.
assigning default hostname oc1 for node 1.
assigning default hostname oc2 for node 2.
assigning default hostname oc3 for node 3.
Successfully accumulated necessary OCR keys.
Using ports: CSS=49895 CRS=49896 EVMC=49898 and EVMR=49897.
node <nodenumber>: <nodename> <private interconnect name> <hostname>
node 1: oc1 poc1 oc1
node 2: oc2 poc2 cc2
node 3: oc3 poc3 oc3
clscfg: Arguments check out successfully.

NO KEYS WERE WRITTEN. Supply -force parameter to override.
-force is destructive and will destroy any previous cluster configuration.
Oracle Cluster Registry for cluster has already been initialized
Adding daemons to initstab
Preparing Oracle Cluster Ready Services (CRS):
Expecting the CRS daemons to be up within 600 seconds.
CSS is active on these nodes.
   oc1
   oc2
   oc3
CSS is active on all nodes.
Waiting for the Oracle CRSD and EVMD to start
Waiting for the Oracle CRSD and EVMD to start
Oracle CRS stack installed and running under init(1M)
[root@oc3 root]#

For release 10.1.0.2.0, you may have to use the racgons utility in the CRS_HOME directory
to configure the Oracle Notification Services (ONS) port number, as follows:

[root@oc3 root]# $CRS_HOME/bin/racgons add_config oc3:4948

If the port is already configured, you will receive this message:


Install Oracle Software on the New Node
In this step, you will copy the Oracle software from one of the existing nodes in the cluster to
the new node. From $ORACLE_HOME/oui/bin run the addNode.sh script. Make sure you are
in $ORACLE_HOME and not $CRS_HOME.
The OUI will start in Add Node mode, and after the startup screens, you will see the Specify Cluster Nodes screen, as shown in Figure 11-28. On this screen, you add the new node oc3. 

After you see the summary screen, similar to the screen you saw for the CRS install, click Next to copy the Oracle software to the new node. After this step completes, you will be prompted to run the root.sh script on the new node. In the final step of the procedure, the updated cluster information is saved to the OCR disk.

Create a New Oracle Instance
To create the Oracle instance on the new node, run DBCA from an existing node and choose a RAC database. On a subsequent screen, choose Instance Management and then add an instance to the existing cluster, as you can see in Figure 11-29.

On the next screen, you are prompted for the new instance name; to be consistent with your previous naming convention, choose rac3 and click Next. In step 6, you will see the existing cluster services; update the services with the new node name as appropriate. On the last step, step 7, you specify the tablespaces, datafiles, and redo log groups that will be added for this instance; in this case, an undo tablespace, the datafile for the tablespace, and two redo log groups, as you can see in Figure 11-30.
FIGURE 11-29. Adding an instance to an existing cluster.

FIGURE 11-30. Adding an instance to an existing cluster.
A confirmation screen appears when the instance is up and running; the cluster once again has three nodes:

```
SQL> select inst_id from gv$instance;

<table>
<thead>
<tr>
<th>INST_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>
```

Tuning a RAC Node

The first step in tuning a RAC is to tune the instance first. If an individual instance is not tuned correctly, the performance of the entire RAC will not be optimal. You can use the Automatic Workload Repository (AWR) to tune an instance as if it was not part of a cluster.

Using EM Database Control, you can further leverage the statistics from the AWR to produce reports on a RAC-wide basis. In Figure 11-31, you can see how EM Database Control makes it easy to analyze the performance of the shared global cache as well as the cache performance on an instance-by-instance basis.
Tablespace Management

In a RAC environment, tablespace management is much the same as in a single-instance environment. There is still only one database and one set of tablespaces to manage; it’s just that there is more than one instance accessing the tablespaces.

Automatic Segment Space Management (ASSM), introduced in Oracle9i, enhances the usability of tablespaces in a RAC environment. Because you no longer have to worry about more freelists and freelist groups to support multiple instances, and therefore more concurrent writers to a table, adding more instances to the cluster does not necessarily require table reorganizations.