



PART VII

Clustered Oracle— The Grid



CHAPTER 40

Oracle Real Application Clusters

In an Oracle Real Application Clusters (RAC) environment, multiple instances access a single database. The instances are commonly created on separate servers that are connected via a high-speed interconnect with access to a shared disk area. The database files reside on a shared set of disks, and each instance has its own control files and online redo log files that are also stored on the shared disks.

A user can be directed to connect to the database through one of the instances attached to it; if an instance goes down, the connection may be dynamically reconnected to the database via another instance in the cluster. RAC thus provides a high-availability solution (accounting for server and instance failure) and a scalable high-performance solution (by increasing the memory and processors available to the connecting processes).

In this chapter, you will see an overview of the installation and management of RAC. Many of the details are operating-system-specific, but the configuration process flow will be the same across platforms. The focus of this chapter will be the common processes and the features introduced in Oracle Database 10g to enhance RAC management and implementation.

Preinstallation Steps

Before starting the Oracle RAC installation, you need to verify that your environment and cluster are properly configured to support it. The servers that will host the instances are referred to as nodes in the cluster. Each of the nodes should have external shared disks. Each node should have redundant switches to support failover requirements. Each node should have one private Internet Protocol (IP) address for private connections and one public IP address for client connections and connection failovers from other nodes. Oracle Database 10g provides a tool called VIPCA (Virtual IP Configuration Assistant) to support the configuration of your IP addresses.

Each node must support TCP/IP (Transmission Control Protocol/Internet Protocol) and a supported interconnect software protocol.

NOTE

You must choose an interconnect that is supported by Oracle. See the Metalink site (<http://metalink.oracle.com>) for the latest certification matrix.

Oracle provides clusterware, simplifying the installation and configuration process. Installation steps related to the clusterware configuration are described in Chapter 41. For the shared disk area, Oracle recommends the use of its Automatic Storage Management feature (see Chapter 46).

NOTE

Be sure to install all required patches on the operating system prior to starting the installation process. As part of your standard Oracle software installation procedures, you should review all required operating system patch levels.

Installing RAC

The installation of RAC in Oracle Database 10g is a two-step process. In the first step, use the Oracle Universal Installer (OUI) to install CRS (Cluster Ready Services). CRS, available as of Oracle

Database 10g, provides an integrated set of solutions for cluster management. You can use CRS to manage cluster membership, group services, group resource management, and high-availability configuration. You can use CRS to define services that support distributing workload across servers. The Automatic Workload Repository (AWR) collects statistics on the services; see Chapter 46 for further details on AWR.

NOTE

RAC works with vendor-supported clusterware as well as Oracle's clusterware. While users of earlier versions of RAC may already have a third-party solution in place, this chapter focuses on Oracle's offering.

NOTE

RAC is available with both the Standard Edition and the Enterprise Edition of Oracle Database 10g.

Both steps of the installation process use the OUI; see Chapter 2 for details on OUI. After completing the first step (installation of the CRS), you can begin the second step of the installation: installing the Oracle Database 10g software with RAC in a different Oracle home directory than you used in the first step. You should also use the installer to create and configure services for your RAC environment. After your software installation completes, OUI will start the Database Configuration Assistant to complete environment configuration and create the RAC database.

NOTE

If you have installed a previous Oracle clustered database version, the Database Upgrade Assistant (DBUA) will upgrade your database to the Release 10g version of RAC.

Storage

As of Oracle Database 10g, you can use Automatic Storage Management (ASM) to create disk groups and simplify their management. As described in Chapter 46, database administrators can add new disks to disk groups; Oracle will manage the contents and the distribution of the files among the disks involved.

You can use ASM to provide redundancy from within Oracle if there is no hardware redundancy available. When you define the disk group, you can specify a failure group; Oracle will use the failure group as a redundant version of the primary group of disks in the disk group. Oracle's installation programs refer to the software-based redundancy as "normal" redundancy. You can specify multiple failure groups for additional redundant protection (at the cost of additional space). If hardware mirroring is available, you can use external (file-system-based) redundancy.

Many RAC environments use a combination of raw devices and logical volume management software. Logical volume management (LVM) provides a file-system management interface to raw devices.

You can use ASM disk groups to further reduce the management complexity of the RAC database—Oracle manages the files and the file systems. You can administer the files and disk groups manually or via Oracle Enterprise Manager (OEM)'s Grid Control options (see Chapter 41).

Initialization Parameters

As part of configuring the instances, you must set values for the database initialization parameters maintained in the server parameter file. RAC installations should use the server parameter file to manage the set of parameters shared by the instances.

The initialization parameters listed in Table 40-1 must have the same value for all instances in the RAC cluster, along with their descriptions from Oracle.

Parameter	Description
ACTIVE_INSTANCE_COUNT	Enables you to designate one instance in a two-instance cluster as the primary instance and the other instance as the secondary instance. This parameter has no functionality in a cluster with more than two instances. When you set this parameter to 1, the first instance you start up becomes the primary instance and accepts client connections. The second instance starts up as a secondary instance and can accept client connections only if the first instance fails. In such an event, the secondary instance becomes the primary instance.
CLUSTER_DATABASE	True/false flag that specifies whether or not Real Application Clusters is enabled. Set to TRUE.
CONTROL_FILES	Specifies one or more names of control files, separated by commas.
DB_BLOCK_SIZE	Specifies (in bytes) the size of Oracle database blocks.
DB_DOMAIN	Specifies the logical location of the database within the network structure. The value consists of the extension components of a global database name, consisting of valid identifiers, separated by periods.
DB_FILES	Specifies the maximum number of database files that can be opened for this database. The maximum valid value is the maximum number of files, subject to operating-system constraint, that will ever be specified for the database, including files to be added by add datafile commands. If you increase the value of DB_FILES, then you must shut down and restart all instances accessing the database before the new value can take effect. If you have a primary and standby database, then they should have the same value for this parameter.
DB_NAME	Specifies a database identifier of up to eight characters. This parameter must be specified and must correspond to the name specified in the create database command.
DB_RECOVERY_FILE_DEST	Specifies the default location for the flash recovery area. The flash recovery area contains multiplexed copies of current control files and online redo logs, as well as archived redo logs, flashback logs, and RMAN backups. Specifying this parameter without also specifying the DB_RECOVERY_FILE_DEST_SIZE initialization parameter is not allowed.

TABLE 40-1. *Initialization Parameters with the Same Values Across RAC Instances*

Parameter	Description
DB_RECOVERY_FILE_DEST_SIZE	Specifies (in bytes) the hard limit on the total space to be used by target database recovery files created in the flash recovery area.
DB_UNIQUE_NAME	Specifies a globally unique name for the database. Every database's DB_UNIQUE_NAME must be unique within the enterprise.
MAX_COMMIT_PROPAGATION_DELAY	This initialization parameter should not be changed except under a limited set of circumstances specific to the cluster database. This parameter specifies the maximum amount of time allowed before the system change number (SCN) held in the SGA of an instance is refreshed by the log writer process (LGWR). It determines whether the local SCN should be refreshed from the lock value when getting the snapshot SCN for a query. Units are in hundredths of seconds. Under very unusual circumstances involving rapid updates and queries of the same data from different instances, the SCN might not be refreshed in a timely manner. Setting the parameter to zero causes the SCN to be refreshed immediately after a commit. The default value (700 hundredths of a second, or seven seconds) is an upper bound that allows the preferred existing high-performance mechanism to remain in place.
SPFILE	Names the current server parameter file (SPFILE) in use. This parameter can be defined in a client-side PFILE to indicate the name of the server parameter file to use. When the default server parameter file is used by the server, the value of SPFILE is internally set by the server.
TRACE_ENABLED	True/false flag that controls tracing of the execution history, or code path, of Oracle. When this parameter is set to TRUE, Oracle records information in log files when errors occur. Oracle records this information for all instances, even if only one instance terminates.
UNDO_MANAGEMENT	Specifies which undo space management mode the system should use. When set to AUTO, the instance starts in automatic undo management mode. In manual undo management mode, undo space is allocated externally as rollback segments.

TABLE 40-1. *Initialization Parameters with the Same Values Across RAC Instances (continued)*

NOTE

The setting for DML_LOCKS must be identical for all instances only if set to 0.

Table 40-2 lists the initialization parameters that must have unique values for each instance in the RAC cluster, along with their descriptions from Oracle.

Parameter	Description
INSTANCE_NUMBER	Specifies a unique number that maps the instance to one free list group for each database object created with storage parameter FREELIST GROUPS.
ROLLBACK_SEGMENTS	Allocates one or more rollback segments by name to this instance. When UNDO_MANAGEMENT is set to AUTO, ROLLBACK_SEGMENTS is ignored.
THREAD	If specified, this parameter must have unique values on all instances. The THREAD parameter specifies the number of the redo thread to be used by an instance. You can specify any available redo thread number as long as that thread number is enabled and is not used.
UNDO_TABLESPACE	Specifies the undo tablespace to be used when an instance starts up. If this parameter is specified when the instance is in manual undo management mode, then an error will occur and startup will fail.

TABLE 40-2. Initialization Parameters with Distinct Values Across RAC Instances

NOTE

Within an instance, set `INSTANCE_NUMBER` and `THREAD` to the same value.

In addition to the parameters shown in the preceding tables, you should specify values for the following initialization parameters:

Parameter	Description
CLUSTER_DATABASE_INSTANCES	Sets the number of instances in the RAC environment.
DISPATCHERS	Sets DISPATCHERS to enable a shared server configuration (in which multiple user processes connect to a dispatcher).

Starting and Stopping RAC Instances

You can use Grid Control to manage all of the instances associated with a RAC database. You can stop, start, and monitor databases, instances, and their listeners via Grid Control. As with any non-RAC database, you can use SQL*Plus to issue **shutdown** and **startup** commands. For RAC databases you can also use the SRVCTL utility, as shown in the examples in this section.

Regardless of the tool used to shut down or start the instance, the following rules apply to RAC clusters:

- Shutting down an instance does not affect the operation of other running instances.
- Shutting down a database mounted in shared mode requires that you shut down every instance in the RAC cluster.

- After a **shutdown normal** or a **shutdown immediate**, no instance recovery is needed. If you use **shutdown abort** on a RAC instance, a still-running instance performs instance recovery for the shutdown instance. If the aborted instance was the last one running, the next instance to start will perform the recovery.
- If you use **shutdown transactional** with the **local** option, all transactions in the instance will have to be either committed or rolled back prior to the shutdown. Transactions on other instances do not impact the local **shutdown transactional**. If you do not specify the **local** clause, then the **shutdown transactional** will wait for all transactions on all instances that started prior to the **shutdown** command to complete.

You can use SQL*Plus commands to shut down and start up RAC databases one at a time until the whole environment has started. See Chapter 46 for details on **startup** and **shutdown** commands. You can start up an instance on your local node via the **startup mount** command.

You can use the SRVCTL utility to manage multiple instances in a single command. The syntax to start instances is

```
srvctl start instance -d <db_name> -i <inst_name_list>
                    [-o <start_options>] [-c <connect_str> | -q]
```

The keyword following **srvctl start** specifies the service being started—instance, database, service, nodeapps, or asm. Note that multiple instances can be passed as part of the *inst_name_list* parameter value (via a comma-separated list). The *start_options* are **open**, **mount**, and **nomount**. The *connect_str* parameter allows you to pass a connect string when connecting to the database. The *q* parameter, if specified, tells SRVCTL to prompt for user connection information when SRVCTL is run; by default SRVCTL will connect to the local database as SYSDBA.

The following command will start and open a local database named MYDB:

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

The following command will start two instances for the MYDB database:

```
srvctl start instance -d mydb -i mydb1,mydb2
```

For ASM instances, specify the node name (-n), the instance name (-i), and the startup options (-o), as in the following example:

```
srvctl start asm -n mynode1 -i asm1 -o open
```

NOTE

*To see the instances that are part of your cluster, you can either use OEM or SQL*Plus. From within SQL*Plus, query the V\$ACTIVE_INSTANCES view. The view will display one row for each instance—showing the instance number, the host name, and the instance name.*

You can use the

```
srvctl stop {database | instance | service | nodeapps | asm }
```

command to stop databases, instances, and other services.

SRVCTL Command	Description
srvctl add {database instance service nodeapp asm}	Adds a configuration to your cluster database configuration. srvctl add service adds services to a database and assigns them to instances. You can use the srvctl add service command to configure the Transparent Application Failover (TAF) policy for a service.
srvctl config {database service nodeapp asm}	Displays the configuration information stored in the Oracle Cluster Registry.
srvctl disable {database instance service asm}	Disables the named object and prevents it from being started automatically.
srvctl enable {database instance service asm}	Enables the named object to be started automatically.
srvctl getenv {database instance service nodeapp}	Displays environment status.
srvctl modify {database instance service nodeapp}	Modifies the object configuration without removing and adding the CRS resources.
srvctl relocate service	Relocates the named service names from one named instance to another named instance.
srvctl remove {database instance service nodeapp asm}	Removes the named object from the cluster environment.
srvctl setenv {database instance service nodeapp}	Sets environment values in the configuration file.
srvctl start {database instance service nodeapp asm}	Starts CRS-enabled, nonrunning applications for the database, all or named instances, all or named service names, or node-level applications.
srvctl status {database instance service nodeapp asm}	Displays the current state of the named object.
srvctl stop {database instance service nodeapp asm}	Stops the CRS applications for the database, all or named instances, all or named service names, or node level applications.
srvctl unsetenv {database instance service nodeapp}	Unsets environment values in the configuration file.

TABLE 40-3. *SRVCTL Command Options*

Other available SRVCTL commands are listed in Table 40-3.

Transparent Application Failover

RAC provides support for transparent application failover (TAF) in the event of an instance failure. To guard against hardware failure, the database file storage must be failure tolerant (usually through the use of RAID arrays, mirroring, or ASM-enforced software mirroring). When a user connects to the database through an instance, any transactions executed by the user are recorded in the redo log thread for that instance.

If an instance fails (or is shut down via a **shutdown abort**), one of the surviving instances will perform the needed recoveries. With TAF, the user's connection will be reestablished via another instance in the same cluster, connecting to the same database.

When you install and configure the RAC environment, Oracle configures the network services for you. For example, if you create a service called db.us.acme.com, the service name entry in the tnsnames.ora file may resemble this:

```
db.us.acme.com=
(description= (load_balance=on) (failover=on)
(address_list=
  (address=(protocol=tcp) (host=db1-server) (port=1521))
  (address=(protocol=tcp) (host=db2-server) (port=1521)))
(connect_data= (service_name=db.us.acme.com)))
```

The user can connect to the instance via the db.us.acme.com service name (for example, via a remote SQL*Plus session) and execute queries. While the user is connected, you can shut down the db1-server host. If the user—still in the same session—attempts to execute another query, the session will be transparently reestablished via the db2-server host and the query will be executed.

The listener.ora file entry for the database will contain entries similar to the following listing:

```
listeners_db.us.acme.com=
(address=(protocol=tcp) (host=db1-server) (port=1521)) (address=(protocol=
tcp) (host=db2-server) (port=1521))
```

If the user has changed environment variables within the session (via **alter session**), or has set session-specific variables, those values may need to be reestablished in the newly connected session.

Services

As described earlier in this chapter, you can create and manage services within your RAC cluster. You can assign services to run on one or more instances; services integrate cluster database resources to simplify cluster manageability.

As shown in Table 40-2, you can use SVRCTL to create and add services. You can assign services to instances for preferred (normal) and available (recovery) processing. You can identify other instances that are available to support the service levels change or for planned outages.

NOTE

You can use the DBCA to manage service assignments and service preferences.

Adding Nodes and Instances to the Cluster

You can add nodes to an existing cluster and you can add instances to an existing RAC environment. The nodes must meet the criteria listed earlier in this chapter—they must be configured with the proper networking protocol configurations and they must support high-speed data transmissions with the other nodes in the cluster. It is generally simplest to make the operating system on the new node a copy of one of the existing nodes in the cluster. As described in Chapter 41, the nodes must be able to execute remote commands via equivalence files.

The specific installation steps for the new node vary by operating system. In general, a UNIX cluster administrator will add the new node to the clusterware configuration (based on directions from its vendor). You can then start the Oracle Universal Installer (OUI) in addNode mode (in UNIX, via `<CRS_HOME>/OUI/bin/addNode.sh` and in Windows via `<CRS_HOME>\oui\bin\addNode.bat`). OUI will then present you with a series of prompts to add the new node to the cluster as it verifies the configuration.

OUI will then begin the installation process, instantiating root scripts if needed, copying the CRS files to the new nodes if needed, and then prompting you to run several scripts at the operating

system level. OUI then saves the cluster inventory. You can then execute the CRSETUP utility to add CRS information for the new nodes. You can then run one final utility, RACGONS, to configure the Oracle Notification Services port for the new server. See the Oracle documentation for the syntax of the CRSETUP and RACGONS utilities, as they involve detailed configuration of node names, node numbers, and ports.

Once the node has been added to the cluster, you can add an instance. Start the Database Configuration Assistant (see Chapter 2) and choose “Real Application Clusters database”. You can then select “Add Instance” and the name of the database the instance will access. Enter the instance parameters where prompted and accept the summary information to initiate the instance creation. Oracle will create the instance and its network services.

To remove an existing instance from a RAC cluster, start the DBCA on a node other than the node hosting the instance that will be removed. The DBCA will display a list of the current instances in the configuration; select the one to be removed from the list. DBCA will display a list of services assigned to that instance so you can relocate them to other instances in the cluster. The selected instance will be deregistered from the RAC cluster.

Managing the Cluster Registry and Services

The Oracle Cluster Registry (OCR) contains configuration details for the database, services, and other cluster components. As shown in Table 40-2, you can use the **srvctl config** command to display the configuration details stored in the OCR. You should make copies of the registry on a regular basis to support any recovery needs you may have later.

One of the CRS-registered instances in the cluster will automatically back up the OCR every four hours, retaining three backup copies. Additional backups are taken once per day and once per week. You cannot modify the backup frequency or the number of backups retained. You can take additional backups via the ocrconfig tool, allowing you to make backups before and after significant changes to the cluster database environment. The OCRCONFIG options are shown in Table 40-4.

Option	Purpose
-export	To export the contents of the OCR into a target file.
-import	To import OCR contents from a previously <i>exported</i> OCR file.
-restore	To restore the OCR from an automatically created OCR backup file.
-backuploc	To change OCR backup file location. For this entry, use a full path that is accessible by all nodes.
-showbackup	To display the location, timestamp, and the node name of origin for the last three automatically created backup files.
-upgrade	To upgrade the OCR to a later version.
-downgrade	To downgrade the OCR to an earlier version.
-help	To display help for the ocrconfig commands.

TABLE 40-4. *OCRCONFIG Options*