



# Oracle 10g on Linux Cluster

*By Bryan Thomas*

*Senior Performance Consultant, Performance Tuning Corporation*

7/07/2005

## Introduction

High Availability (HA) is in high demand today. Corporations lose thousands if not millions of dollars every time there is an outage of their IT systems. This white paper will cover the basic steps for creating a Linux Cluster. Specifically, it will cover installing Oracle 10g on a Linux Cluster in active/passive mode utilizing the Red Hat Cluster Manager.

Red Hat Cluster Manager is an HA cluster solution specifically suited for database applications, network file servers, and World Wide Web (Web) servers with dynamic content. A Red Hat Cluster Manager system features data integrity and application availability using redundant hardware, shared disk storage, power management, and robust cluster communication and application failover mechanisms.

This paper is broken into sections. In the first section, we will define a cluster. In the second section we will plan your cluster installation. In the third section we will cover the details of the cluster implementation. In the final section we will cover how to test the implementation. As with any white paper I write, your suggestions and comments are always welcome. Please send them to [bthomas@perftuning.com](mailto:bthomas@perftuning.com).

## Definition of Cluster

In the most simple of terms, a cluster is a group of independent servers that function as a single system. It can also be defined as multiple systems performing a single function; a black box.

Let's go over some cluster terminology. A server that is connected to the cluster is known as a *node* in the cluster. If a node in the cluster is currently processing data for the cluster it is *active*. If a node in the cluster is waiting for a failover to start working, it is a *passive* node in the cluster. A resource or a node is placed in a *failed* state after an unsuccessful attempt has been made to bring it online.

When you create a cluster, you create an *alias* that users connect to. From the cluster's standpoint, the alias refers to the server that owns the service at a particular time. From the users' standpoint, the alias refers to the service they connect to (e.g., Oracle 10g Database); regardless of which server the service resides on. When a failover occurs, users reconnect (unless the application or client operating system handles reconnection for them) to the alias (not to a server) and continue working. If users connect directly to the primary server rather than to the alias, they cannot reestablish their connections when the primary server fails.

Most clustering solutions employ cluster *objects* and *groups* (services). An application, a disk drive (for database volumes, files, or directories), an IP address, and so forth are examples of cluster objects. A cluster group is a collection of related objects that make up a cluster resource such as an Oracle instance.

*Failback* switches the service or object from the secondary node back to the primary node after a failover has occurred, and resets ordinary operating conditions. Failback is usually a manual process, but can be automatic.

*Failover* occurs when a node in the cluster or a component of the cluster configuration fails. Cluster services or resources from the faulty node relocate to the operational node. You can also trigger failover manually (e.g., by turning off a system to upgrade it), taking one server offline and shifting the work to the secondary server. System failover capability means that only the entire system can fail over; individual objects (such as applications, disk volumes, etc.) cannot. The secondary system assumes the identity and workload of the primary system (only on system failure) in addition to its identity and workload. Application failover capability means that the systems administrator can fail over one application or object at a time, keeping all other services and users intact on the nodes.

A *heartbeat* is the signal that the nodes in a cluster send each other to verify they are alive and functioning. The nodes transmit the heartbeat over direct LAN connections, through a hub or switch, or even via the SCSI bus. If the heartbeat ceases, one of the nodes has failed and the clustering software instructs the other node to take over. Employing more than one method to generate a heartbeat eliminates the problem of a minor failure triggering an unwanted failover.

The *interconnect* provides a communications link between the nodes in a cluster. Status information, heartbeat, and even lock information can travel over the interconnect. This connection can be over your LAN or directly from node to node, using Ethernet, 100Base-T, Fibre Channel, serial, SCSI, and so forth. Fault-tolerant clustering solutions typically use more than one interconnect path simultaneously to prevent unwanted failovers.

Now back to the definition of Clustering. On a more complex level clustering is taking a set of servers and running one to N number of applications on those servers. A cluster

can be as simple as two servers clustered together in a simple failover configuration. If one server fails, the second server takes over processing the application. Clusters can also be used to run very complex, distributed systems. One example of a complex cluster is the SETI@home project. The application takes the data and splits it into manageable chunks and independent servers process these manageable chunks. If one server fails, the others are not affected and the failed node's processes are redistributed to other nodes in the cluster.

There are several types of cluster: Failover Cluster, Distributed Database System, and Shared Disk System. A Failover Cluster is a set of servers where one server acts as a backup of the other server in case of a system failure. A distributed database system contains multiple servers running separate instances. The data from the instances are kept in sync somehow. A Shared Disk System is a set of servers that share a disk system. The data that resides on the shared disk can be accessed by all servers in the cluster.

### **Failover Cluster**

There are two types of Failover Clusters. One type is the active/passive cluster and the other type is the Active/Active cluster. With an active/passive cluster only one server has a load at any given time. The second server is running in a passive mode, waiting for the other server to go down, but keeping the two servers in sync.

With active/passive, only one of the servers is using its processing power for transaction processing. Only one server is actually running an Oracle database. The applications connect to the Oracle database via a virtual IP address. This IP address will failover to the other cluster if the Oracle database and database files fail over. No user intervention is needed if the active node in the cluster fails.

The second type of failover cluster is the active/active cluster. In an active/active cluster configuration, both servers perform meaningful work all the time. Active/active clusters let you fully utilize your server resources and can support load balancing. Each server will run one instance and will be the failover database for the instance running on the other server.

With an active/active cluster, both servers are being used for processing. This type of cluster will generally run two different applications, one on each of the servers in the cluster. One of the disadvantages of this type of cluster is you really need to size both servers in the cluster to be able to temporarily handle the processing of both nodes in the cluster in case of a failover. Both nodes in the cluster could be failed over to the other node at any time.

### **Distributed Database Systems**

In a distributed database system several database servers are somehow kept in sync. One type of distributed database system is a replicated database. Another type is a standby database.

With a replicated database there will be multiple copies of the same database. Some replicated databases will be read-only and some will be read/write. Advanced Replication is the Oracle implementation of the replicated database.

Another type of distributed database system is the standby database. A standby database is a copy of the active database. A standby database in Oracle is kept up to date using the Archive Logs. This standby database can be located in the same server room, in the same rack; or it can be located in another state/country. The standby database is as current as the logs that are applied. The logs can be applied each time there is a new archive log or they can be batched up and applied at a later time.

There are pluses and minuses for each kind of distributed database as well. A true replicated database can consume a large amount of available resources keeping multiple copies of the database in sync. If two-phase commit is enabled, a change to the database is not fully committed until all the distributed databases have updated the database. This can cause problems in a heavily changing environment. With a standby database, the data is only as current as the log files that have been applied. If a failure were to occur with a standby database, all the unapplied log files would have to be applied before the database was current.

Distributed databases are mostly used in systems with very specific requirements. The most common type of distributed database would be the standby.

### **Shared Disk System**

With Shared Disk Clustering, there is a shared file system. Multiple systems access the same set of disks. All nodes in the cluster must have access to the shared disks. With Oracle in a shared disk system, there is only one database running. Oracle Real Application Clusters (RAC) is a true shared disk system.

With a shared disk system, multiple nodes in the cluster will have a common set of disks. These disks are usually configured in some type of a RAID configuration. Normally there is group of disks for the cluster information. This is often called the quorum.

The main problem with a shared disk system has been keeping the nodes from interacting with each other in a negative way. Without proper control each node could cancel out the other's updates. If there is too tight of control, application processing would slow down.

## **Planning the Implementation**

The first step in any Oracle implementation is to plan your work. The steps involved in planning an implementation involve gathering requirements, analyzing requirements, create implementation plan, execute plan, and test that the plan was executed to the specifications.

## **Gather Customer Requirements**

This is arguably one of the most important steps. You must be proactive in obtaining the input you require. Don't expect accurate customer requirements to be delivered to you. One of the common ways to collect customer requirements is to have a Joint Application Design (JAD) session. This is normally a time consuming process. For short/under-funded projects you may only need to interview the customers. Most of your customers will not be able to understand technical jargon. Translating customers input into technical specification can be a daunting task.

Make sure to include the time sensitive requirements. On a recent project, the customer was constrained by their insurance company. The requirement from the insurance company was to have a working Disaster Recovery (DR) system in place by a certain date. This was an arbitrary, but mandatory date. These kinds of requirements can make or break a project.

Often the customer requirements are just – Make it run better than our current production. If there are existing systems that you wish to move to a cluster, you can collect CPU, memory and storage requirements from those systems. It is always best to utilize the experience you already have in house. You should always gather as much technical information as possible from these legacy or comparison systems. This is true even if the hardware platform will be different than your target system or even if you expect your new system to have significantly improved performance characteristics. In particular, gather statistics for CPU utilization, memory utilization, and I/Os per second (IOPS).

The deliverables from this phase are:

- Concrete customer requirements
- Legacy System metrics

## **Analyze Requirements**

After the requirements have been gathered it is time to analyze the data. Take the requirements from the customer and the Legacy system data and use them to determine your hardware and software requirements. See *Sizing Oracle on Microsoft Windows and Dell PowerEdge Servers* for details on Sizing an Oracle System. As this a paper on Linux Clustering, we will assume the requirements dictate a Cluster for High Availability (HA).

One of the most important steps is to convert the customer requirements to technical specifications. A future white paper will be written on this subject.

One item that must come out of this phase is a list of testing conditions. You should be able to measure whether your implemented system meets the requirements based on a set

of tests. It is very important that we list the testing criteria before we start the actual work.

The deliverables from this phase are:

- Technical Specifications from customer requirements
- Hardware requirements
- Software requirements
- Testing conditions

### **Create Implementation Plan**

The next phase is to create an implementation plan based on the requirements and specifications from the Analysis Phase. You should take into account the hardware and software requirements. If you need lead time to order hardware, make sure this is included in the plan.

It is very important to create this plan, so you can make sure the project is on track. I have been on several projects where there was not a clear plan created. This customer was very unhappy with the outcome because he/she expected completion by certain dates and this was not communicated up front.

Always include intermediate milestones so you can track your progress. Make sure you include some time in each phase for unexpected problems. This float time could really save you. Make sure you plan enough time for anyone on your team to complete the milestone, not just your senior person. That person may be called away to a more important project.

The key parts of the implementation of a Linux Cluster for Oracle 10g include:

- Hardware and Software purchases
- Cluster configuration
- Oracle installation
- Cluster testing
- Data Migration
- Production cut over

See the next section for details on how to implement Oracle 10g on a Linux Cluster.

The deliverables from this phase are:

- Implementation Plan

### **Execute Implementation Plan**

Although this step is never purposely overlooked, you should always plan to execute your plan. Plan your work and work your plan.

Milestones should be tracked closely. If you are consistently missing your milestones, look into why this is happening. Perhaps you underestimated the time it actually takes to complete each task. If a major milestone is missed, such as hardware delivery, you will have to rework the entire plan.

Here are some items to consider during the implementation phase:

- Gather the IP addresses for the members and for the bonded Ethernet ports, before installing Red Hat Enterprise Linux. Note that the IP addresses for the bonded Ethernet ports can be private IP addresses, (for example, 10.x.x.x).
- Do not place local file systems (such as /, /etc, /tmp, and /var) on shared disks or on the same SCSI bus as shared disks. This helps prevent the other cluster members from accidentally mounting these file systems, and also reserves the limited number of SCSI identification numbers on a bus for cluster disks.
- Place /tmp and /var on different file systems. This may improve member performance.
- When a member boots, be sure that the member detects the disk devices in the same order in which they were detected during the Red Hat Enterprise Linux installation. If the devices are not detected in the same order, the member may not boot.
- When using RAID storage configured with Logical Unit Numbers (LUNs) greater than zero, it is necessary to enable LUN support by adding the following to /etc/modules.conf:
  - `options scsi_mod max_scsi_luns=255`
  - After modifying modules.conf, it is necessary to rebuild the initial ram disk using mkinitrd. Refer to the Red Hat Enterprise Linux System Administration Guide for more information about creating ramdisks using mkinitrd.

## **Validation of requirements**

When possible, you should include time and budget for a validation phase. The earlier this is done in the sizing process the better. A “proof of concept” approach can be used with a system that closely approximates your target system. Even more useful is a “benchmark” approach, where multiple configurations are tested to see which most closely matches the requirements. Remember, it is better to find out that you cannot meet your requirements before you go into production, than after you have fully implemented the system.

## Cluster Installation

This section will cover the details on implementing an Oracle 10g database on a Linux Cluster. Specifically this will cover Red Hat Cluster Suite and Cluster Manager.

From the Red Hat Cluster Suite, Configuring and Managing a Cluster Guide:

To set up a failover cluster, you must connect the *member systems* (often referred to simply as *members* or *nodes*) to the cluster hardware, and configure the members into the cluster environment. The foundation of a cluster is an advanced host membership algorithm. This algorithm ensures that the cluster maintains complete data integrity at all times by using the following methods of inter-member communication:

- Network connections between the cluster systems for *heartbeat*
- *Shared state* on shared disk storage to hold cluster status

The following steps are necessary to getting an Oracle database clustered using Red Hat Cluster Suite. These are the steps an administrator/DBA would take to implement Oracle 10g on a Linux Cluster.

1. Install Red Hat Linux OS software on both nodes.
2. Configure and test network interfaces.
3. Perform storage deployment and configuration.
4. Install Oracle software on both nodes and create an Oracle database.
5. Prepare quorum devices on shared storage.
6. Install the Red Hat Cluster Manger Suite software.
7. Setup the Linux Cluster.
8. Define the Oracle Service.
9. Setup the second node.
10. Start Cluster Services on both nodes.
11. Customize Oracle Service scripts.

### Install Red Hat Linux Operating System

You will want to install the same OS version on both servers in your cluster. Please verify that the version of Oracle 10g you wish to install is certified on that release of Red Hat.

See Oracle 10g installation guide for required Linux packages, kernel settings and user requirements.

### Configure and test network interfaces

The next step is to configure the network interfaces. Most companies will want to set up network bonding of dual public interface ports. This is an HA system and you should think about network redundancy.

A “public” interface will need to be configured for communication from outside the cluster. The cluster will need to have another interface configured for the “private,” heartbeat communications.

Make sure the IP/Host names are configured in both systems `/etc/hosts` files. If you are using EMC/Navisphere you will want to put the public interface first in the `/etc/hosts` file.

It is also important that you test the network interfaces now. If you wait until the Oracle installation has started, you may end up with problems.

You should test with `ping`, and `ssh`. I also verify the speed of the interfaces using the `ttcp` rpm package.

### **Perform storage deployment and configuration**

The next item to configure for the Cluster deployment is the Shared Storage. Shared storage is at the heart of the Linux Cluster. There are several types of shared storage you can use for your Cluster. We will be using an EMC SAN for this document.

1. Install EMC PowerPath and Naviagent on each server in the cluster.
2. Upgrade the HBA driver to the certified driver for your particular SAN/OS release.
3. Create the necessary LUNS for your Oracle Database and the cluster Quorum.
4. Present the LUNS to the servers
5. Create the required partitions on the LUNS.
6. Lay a file system down on the Oracle partitions. This can be EXT2 or EXT3 for the Oracle files, i.e. `mke2fs -j -b 4096 /dev/sde3`
7. Add the partitions to the `/etc/fstab`
8. Verify that all of the `emcpower` devices point to the same LUNS on both servers

### **Install Oracle software on both nodes and create an Oracle database**

Install the Oracle 10g binaries on both servers. *Do not create a database at this time.* We will create the database after we have installed the binaries. Make sure that your Oracle home is set to a local, non-shared, directory. It is not advisable to install the Oracle Binaries on a shared partition at this time.

After the Oracle binaries have been installed, run `netca` to create a basic listener. This should be run on both servers in the cluster.

To create your database you will need to have the shared storage LUN mounted to both nodes in the cluster. Run dbca to create the database. Choose the mount point for the shared storage as the location for the Oracle files.

### **Prepare quorum devices on shared storage**

Red Hat Cluster Suite requires two quorum partitions: a primary quorum and a shadow quorum. Both partitions should have a minimum size of 10 MB. These partitions are setup as raw devices. The raw devices must be added to the `/etc/sysconfig/rawdevices` file. After this file has been modified restart the rawdevices service:

```
service rawdevices restart
```

Query all the raw devices by using the command `raw -aq`. The output should be similar to the following:

```
/dev/raw/raw1 bound to major 8, minor 17  
/dev/raw/raw2 bound to major 8, minor 18
```

### **Install the Red Hat Cluster Manger Suite software**

Install the Red Hat Cluster Manger Suite software with the Package Manager on both hosts. You made install the rpm packages or you can use the installation GUI tool.

The cluster manager suite is required on both nodes in the cluster. This can be installed for the Oracle binaries or after you have installed the Oracle binaries.

### **Setup the Linux Cluster**

Use the Cluster Configuration Tool to define Cluster Members, the Failover Domain, and the Cluster Daemon Properties. This will be done on the first node only. *Do not setup the cluster using the Cluster Configuration Tool on the second node in the cluster.*

Add cluster member definitions for both nodes in your cluster.

If you setup a failover domain, you will have the option to put only one node in the domain. This will provide you with a primary server. There is a drawback to defining a failover domain with only one server in it. If the service fails over to the secondary node and then the primary is booted up, the service will failback to the primary server. This could cause service interruption for customer. It is our recommendation that you put both servers in the failover domain and manually failback to the primary server during a convenient time.

If you do not have dedicated hardware Power Switch controls, you will need to enable the option for the Software Watchdog.

As noted above, use the Cluster Configuration Tool to configure Cluster Daemons. These daemons include cluquorumd (the Quorum daemon), clusrvcmgrd (the Service manager daemon), clurmtabd (NFS synchronization daemon), clulockd (the Global lock manager), and clumembd (the Membership daemon).

For clumembd we recommend leaving the failover speed at the default of 10 seconds. Broadcast Heartbeating can be enabled as opposed to Multicast Heartbeating.

For cluquorumd, the Tiebreaker IP can be set to the network gateway IP, per Red Hat recommendations for two node clusters. You can alternately enable a disk-based heartbeat of 2 seconds on both nodes. Both options could not be set simultaneously within the GUI. If you wish to enable both use the redhat-config-cluster-cmd command line option to set the disk ping interval in addition to the Tiebreaker IP.

For clurmtabd, we recommend using the default poll interval of 4 seconds for NFS synchronization. Change this setting based upon your use of NFS. If your NFS keeps timing out, change this setting.

For all daemons, we recommend setting the Log Level to “Debug” for the duration of testing. Once you have finished any testing activities, the Log Level for all daemons should be set to “Warn”.

### **Define the Oracle Service.**

From the Red Hat Cluster Suite, Configuring and Managing a Cluster Guide:

*A database service can serve highly-available data to a database application. The application can then provide network access to database client systems, such as Web servers. If the service fails over, the application accesses the shared database data through the new cluster system. A network-accessible database service is usually assigned an IP address, which is failed over along with the service to maintain transparent access for clients.*

The following are the steps you need to perform to define the Oracle Service on the first node only.

- Create the shared storage mount points
  - Database Storage
  -
- Create a consistent user/group configuration that can properly access Oracle service for each cluster system. For example:
  - mkdir /users
  - groupadd -g 500 dba
  - groupadd -g 501 oinstall
  - useradd -u 500 -g 501 -d /users/oracle -m oracle
  - usermod -G 500 oracle

- Create Oracle Service shell scripts
  - Stopdb
  - Startdb
  - Test
  - Driver script
- Run redhat-config-cluster and setup Oracle service
  - Enter a failover domain if one is required
  - Enter the driver script location
  - Create a virtual or floating IP address for the service. The tnsnames.ora and listener.ora files on each server should be configured to use the virtual IP address for client connections.
  - Add a child
    - Enter the virtual IP address information
  - Add a child
    - Enter the mount point for the Oracle files
  - Add a child
    - Enter administrative mount points
  - Save the service

### Setup the second node

All of the Linux cluster information is contained in one file. To setup clustering on the second node, copy the cluster configuration file, /etc/cluster.xml, from the first node to the second node.

### Start Cluster Services on both nodes

The next step is a key one. You must start the cluster services on both nodes. This is key because if you have made a mistake in any of the previous steps, the cluster services will not start properly on both nodes.

Use the **Cluster Status Tool** to start and stop the cluster service on both nodes. The **Cluster Configuration Tool** can be displayed by choosing **Cluster => Configure** within the **Cluster Status Tool**.

If it does not start on both nodes go back through the previous steps and look for mistakes and typos. If it starts fine on the first node, but not on the second node; check to make sure the cluster.xml file was copied over properly. Also check to see if the LUNs are mapped the same way on both nodes in the cluster.

### Customize Oracle Service scripts

The next step is to customize your Oracle scripts. From the Red Hat Cluster Suite, *Configuring and Managing a Cluster Guide*:

*The Oracle service example uses three scripts that must be placed in /users/oracle and owned by the Oracle administration account. The oracle script is used to start and stop the Oracle service. Specify this script when you add the service. This script calls the other Oracle example scripts. The startdb and stopdb scripts start and stop the database. Note that there are many ways for an application to interact with an Oracle database.*

The following is an example of the oracle script, which is used to start, stop, and check the status of the Oracle service:

```
#!/bin/sh
#
# Cluster service script to start, stop, and check status of oracle
#
cd /users/oracle
case $1 in
start)
su - oracle -c ./startdb
;;
stop)
su - oracle -c ./stopdb
;;
status)
status oracle
;;
esac
```

The following is an example of the startdb script, which is used to start the Oracle Database Server instance:

```
#!/bin/sh
#
#
# Script to start the Oracle Database Server instance.
#
#####
#
# ORACLE_RELEASE
#
# Specifies the Oracle product release.
#
#####
ORACLE_RELEASE=9.2.0
#####
#
# ORACLE_SID
#
# Specifies the Oracle system identifier or "sid", which is the name of
# the Oracle Server instance.
#
#####
export ORACLE_SID=TEST
#####
#
# ORACLE_BASE
#
# Specifies the directory at the top of the Oracle software product and
# administrative file structure.
#
#####
export ORACLE_BASE=/u01/app/oracle
#####
#
# ORACLE_HOME
#
# Specifies the directory containing the software for a given release.
# The Oracle recommended value is $ORACLE_BASE/product/release
#
#####
export ORACLE_HOME=/u01/app/oracle/product/${ORACLE_RELEASE}
#####
```

```

#
# LD_LIBRARY_PATH
#
# Required when using Oracle products that use shared libraries.
#
#####
export LD_LIBRARY_PATH=${ORACLE_HOME}/lib:$LD_LIBRARY_PATH
#####
#
# PATH
#
# Verify that the users search path includes $ORACLE_HOME/bin
#
#####
export PATH=$PATH:${ORACLE_HOME}/bin
#####
#
# This does the actual work.
#
# Start the Oracle Server instance based on the initSID.ora
# initialization parameters file specified.
#
#####
/u01/app/oracle/product/9.2.0/bin/sqlplus << EOF
sys as sysdba
spool /home/oracle/startdb.log
startup pfile = /u01/app/oracle/product/9.2.0/admin/test/scripts/init.ora open;
spool off
quit;
EOF
exit

```

The following is an example of the stopdb script, which is used to stop the Oracle Database Server instance:

```

#!/bin/sh
#
#
# Script to STOP the Oracle Database Server instance.
#
#####
#
# ORACLE_RELEASE
#
# Specifies the Oracle product release.
#
#####
ORACLE_RELEASE=9.2.0
#####
#
# ORACLE_SID
#
# Specifies the Oracle system identifier or "sid", which is the name
# of the Oracle Server instance.
#
#####
export ORACLE_SID=TEST
#####
#
# ORACLE_BASE
#
# Specifies the directory at the top of the Oracle software product
# and administrative file structure.
#
#####
export ORACLE_BASE=/u01/app/oracle
#####
#
# ORACLE_HOME
#
# Specifies the directory containing the software for a given release.
# The Oracle recommended value is $ORACLE_BASE/product/release
#
#####
export ORACLE_HOME=/u01/app/oracle/product/${ORACLE_RELEASE}
#####
#
# LD_LIBRARY_PATH
#
# Required when using Oracle products that use shared libraries.
#
#####
export LD_LIBRARY_PATH=${ORACLE_HOME}/lib:$LD_LIBRARY_PATH

```

```
#####
#
# PATH
#
# Verify that the users search path includes $ORACLE_HOME/bin
#
#####
export PATH=$PATH:${ORACLE_HOME}/bin
#####
#
# This does the actual work.
#
# STOP the Oracle Server instance in a tidy fashion.
#
#####
/u01/app/oracle/product/9.2.0/bin/sqlplus << EOF
sys as sysdba
spool /home/oracle/stopdb.log
shutdown abort;
spool off
quit;
EOF
exit
```

We recommend configuring the Oracle Service as recommended by Red Hat. This included the use of a “floating” or virtual IP address for the service. The \$ORACLE\_HOME/network/admin/tnsnames.ora file and the \$ORACLE\_HOME/network/admin/listener.ora file on each server were configured to use the virtual IP address for client connections.

To support the Oracle Service, the text from the example Oracle scripts in the Red Hat Cluster Suite manual was adapted to create Oracle Service shell scripts. There are two additional items we would recommend doing:

- 1) Automatic startup of the Oracle Listener process when the database is started.
- 2) Check the status of the public Ethernet interface. Relocate the service if the interface has died.

## Test the installation

Always follow a test plan. It is very important that you create and follow the plan. You may want to create a check list and check it off as you test each item. This will save you time and effort when things are not working right. You can go back and check that you tested the important items. A test plan can verify the functionality and reliability of the cluster.

## **Network Setup Test**

### **Ping**

- Verify that each host in the RAC configuration can ping all other host in the cluster on the public network
- Verify that each host in the RAC configuration can ping all other host in the cluster on the private network

### **SSH (Linux)**

- Verify that ssh is operational between all nodes in the cluster

## **Loss of One Public Network Cable Test**

- Remove one network cable from the primary host
  - Network bonding should fail over at the network level without affecting the cluster

## **Loss of Disk Test**

- Remove one disk from the Storage Array
  - The hot spare should take over. When the disk is replaced, it should eventually be added back in the RAID configuration.

## **Loss of Network Test**

- Remove all network cables from the primary host
  - The host should reboot
- Run clustat on the other host to verify cluster status

## **Loss of Access to One Quorum Partition Test**

- Run dd to write zeros to the shared partition
  - dd if=/dev/zero of=/dev/raw/raw1 bs=512 count=1
  - shutil -p /cluster/header
- An event is logged
  - The data from the good shared partition is copied to the partition which returned errors

## **Loss of Access to Both Quorum Partitions Test**

- Remove the Fibre Channel cable from the primary host
  - The host should reboot (assuming the target topology is used)
- Run clustat on the other host to verify cluster status

## **clumembd Crash Test**

- On the primary host, “killall -KILL clumembd”
  - The host should reboot
- Run clustat on the other host to verify cluster status

### **clumembd Hang Test**

- On the primary host, “killall –STOP clumembd”
  - The host should reboot after (failover time -1 second) expires
- Run clustat on the other host to verify cluster status

### **cluquorumd Crash Test**

- On the primary host, “killall –KILL cluquorumd”
  - The host should reboot
- Run clustat on the other host to verify cluster status

### **clusvcmgrd Crash Test**

- On the primary host, “killall –KILL clusvcmgrd”
  - cluquorumd re-spawns clusvcmgrd, which runs the stop phase of all services
  - Services which are stopped are started
- Consult system logs for a warning message from cluquorumd

### **Unexpected System Reboot Test**

- On the primary host, “reboot –fn”
- Run clustat on the other node to verify cluster status

### **Unexpected System Reboot of Both Systems Test**

- On both hosts, “reboot –fn” or power down
- Run clustat on the other node to verify cluster status

## **References**

“Dell and Oracle”; 2004, Dell Corporation; <http://www.dell.com/oracle>

“Sizing Oracle on Microsoft Windows and Dell PowerEdge Servers”, 2005, Performance Tuning Corp.; [www.perftuning.com](http://www.perftuning.com)

**Bryan Thomas** ([bthomas@perftuningl.com](mailto:bthomas@perftuningl.com)) is a Senior Performance Consultant for Performance Tuning Corporation. He has been involved with Oracle deployment and configuration on Linux for the past two years. He has over 10 years experience with Oracle. He is currently working on Oracle 10g RAC and Clustering deployments, as well as performance optimization on all versions of Oracle. Bryan has a Bachelors degree in Computer Science with a minor in Physics from Texas A&M University, and over 18 years experience in the IT field.