Using Oracle Clusterware to Protect 3rd Party Applications

An Oracle White Paper
February 2008

ORACLE

Using Oracle Clusterware to Protect 3rd Party Applications

Using Oracle Clusterware to Protect 3rd Party Applications

Oracle Database 10*g* Release 2 delivers a High Availability Framework to enable Oracle Clusterware to provide protection for 3rd party applications .

INTRODUCTION

Have you ever wondered why on a Oracle Real Application Clusters 10*g* node if you 'kill -9' a database LGWR process in Linux the database gets restarted automatically? Or when using the Task Manager in Microsoft Windows, kill the oracle.exe or tnslsnr.exe process, they restart again automatically? Or how, when a node dies in an Oracle RAC 10g cluster, the Virtual IP (VIP) fails over to a different node?

The service that provides this is a feature of Oracle Clusterware. One of the significant new features in Oracle Database 10*g* Release 2 is the extension of this framework to protect 3rd party applications.

The advent of Oracle RAC 10*g* brought customers the opportunity to simplify their software stack. Oracle Clusterware gave them the option of dispensing with proprietary, expensive vendor clusterware and using an entire stack from Oracle, from disk management with Oracle ASM to data management with Oracle Database 10*g* and RAC.

The ability to protect non-Oracle applications from failure was not part of Oracle Database 10*g* Release 1. With the advent of Oracle Database 10*g* Release 2, non-Oracle applications running on a cluster can benefit from the same level of protection as existing Oracle Database processes.

Oracle Clusterware was used in Oracle Database 10*g* Release 1 to allow automatic restart of failed Oracle processes and the restart of the Oracle Virtual IP (VIP) on another node in the cluster on node failure. VIP failover allows Oracle database clients to rapidly recognize that a node has died, improving the reconnect time. Oracle Database 10*g* Release 2 extends this VIP capability to user applications

Customers and Oracle partners can now place their own applications under the management and protection of Oracle Clusterware, such that they restart on process failure or failover to another node on node failure.

The scripts provided as part of this paper are sample code, which can be used to base your own scripts on. These scripts have been tested on an OEL 4 2 node cluster. **Oracle Support cannot provide any direct support for these scripts.** You should thoroughly test the scripts – in particular the start/stop/check action of each script to ensure compatibility with your environment.

Please do not call Oracle Support to discuss the scripts in this paper, this is an un-supported example.

CONSIDERATIONS FOR SUCCESSFULLY MANAGED APPLICATIONS

An important concept for the success of applications managed by Oracle Clusterware is 'node independence'. For an application to be managed well by Oracle Clusterware it should not care on which physical node the application is actually running on. As such the application binaries should be made visible to all nodes in the cluster either via local installs or the Oracle Cluster File System (OCFS).

There are two other considerations:

- Application configuration files.

Some applications store information in configuration files on disk. For the application to run successfully it must be able to access that stored information. Oracle solution for this is Clustered file systems. So on Linux & Windows the recommendation is that applications that require access to such files should store them on the Oracle Clustered File System.

- Client connectivity

Clients should be able to access the application irrespective of the physical node it is actually running on. The Oracle solution to this is 'Application Virtual IP's'. Note this is similar concept to the VIP that is used by Oracle RAC 10*g* but is used in a different way. The VIP used by Oracle RAC 10*g* is used as a method of providing a NAK reply to the Oracle client network layer when the VIP fails over on a node down condition. Application VIP's are used as method of consistently accessing an application from a network. The Application is made dependent on the VIP. The 2 fail over together as a combined resource. The Application mode VIP script is provided as part of Oracle Clusterware. If the server application has configuration files they must specify the VIP hostname and not the local hostname.

ORACLE CLUSTERWARE HA FRAMEWORK & HA API

Oracle Clusterware includes two new publicly available components that can be used to help protect an application.

- A high availability framework
- A 'C' Application Programming Interface

This paper covers the first of these two options

The High Availability Framework

The High Availability Framework provides an infrastructure to manage any application. The Oracle Clusterware ensures that the application it manages start when the system starts. The Oracle Clusterware also monitors the applications to make sure that they are always available. For example, if a process fails, then Oracle Clusterware attempts to restart the process based on scripts that you customize. If a node in the cluster fails, then you can configure Oracle Clusterware to start the program processes, that normally run on the failed node, on another node. The monitoring frequency, starting and stopping of the applications and the application dependencies are configurable.

To make applications highly available, first create an action program that describes how the Oracle Clusterware should monitor your application and how the Oracle Clusterware should respond to changes in your application's status. Then create an application profile that identifies your application. Then, using the information contained in the application profile, register the application with Oracle Clusterware. Oracle Clusterware then uses the action program to start, stop and check the status of the application based on the registered information.

The 'C' API

The C API can be used to directly manipulate the Oracle Cluster Registry, which defines how Oracle Clusterware protects applications. It is expected that most users who use Oracle Clusterware will use the Framework and not the API. The API can be used to modify, at run time, how Oracle Clusterware manages an application.

This paper discusses the first of these two capabilities: The Framework .For more information on the API see the Oracle Database Documentation detailed in the References section of this paper.

RULES FOR CLUSTERS USING ORACLE CLUSTERWARE

This section details the requirements for running Oracle Clusterware on a Cluster

Oracle Clusterware can be used to protect 3rd party (non-Oracle) applications on a cluster.

A Cluster is a collection of 2 or more nodes where the nodes share:

- A common disk: used by the Oracle Clusterware system files:
  - OCR & VOTE
- A common network interconnect
- Are configured with the same Operating System

There is a basic technical requirement that Oracle Database 10*g* Release 2 Clusterware needs to be installed and configured, Oracle Clusterware uses the shared disk for it's configuration files and decisions on node membership. The dedicated network interconnect is used to maintain the integrity of the cluster.

The following figure is a typical 4 node configuration

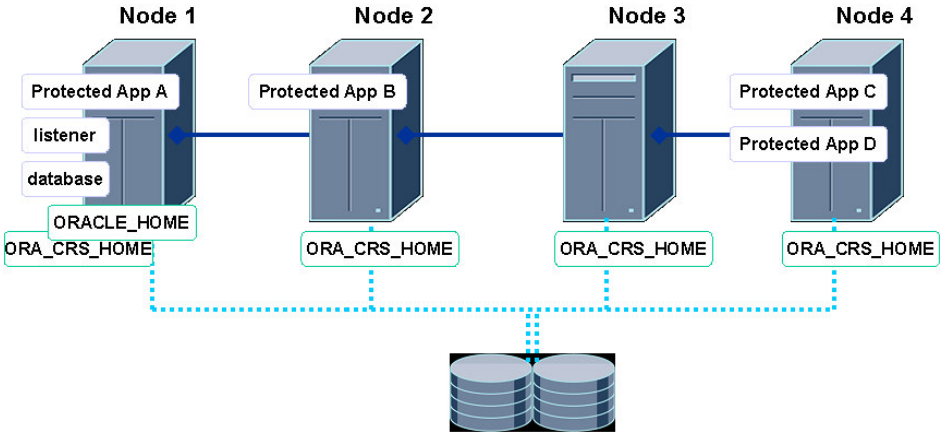

Figure 1

Node 1 is running an Oracle database. It also has Oracle Clusterware installed, which is protecting Application A

Node 2 is running Oracle Clusterware protecting Application B

Node 3 currently is only running Oracle Clusterware

Node 4 is running Oracle Clusterware protecting Applications C & D

Oracle Clusterware will monitor all applications A-D and will restart on the same node or relocate to another node based on the failover policy.

USING THE APPLICATION FRAMEWORK

The Application Framework makes it easy to add Oracle Clusterware Protection for your Application. There are 3 steps that need to be completed.

Enabling protection for an existing application is quite straightforward. Oracle Clusterware needs to know how to start, stop & check the status of the application. It also needs to know how you would like Oracle Clusterware to manage failure of your application. It needs to be told under what circumstances it can restart the application on the node it was running on and when to decide to fail the application over to another node in the cluster.

There are four steps that need to be completed to successfully place an application under the protection of Oracle Clusterware

- Create an application VIP – required if the application is access via network clients.
- Create an action program - This program is used by Oracle Clusterware to start, stop & query the status of the protected application. This program can be written in C, java or almost any scripting language
- Create an application Profile - A text file that describes the existing process and the limits covering how it is to be protected.
- Register the application - Register the Application Profile with Oracle Clusterware using a provided command

From now on Oracle Clusterware will manage the application. It will be started, stopped and made highly available according to the information and rules contained in the Application Profile used to register with the Oracle Cluster Registry (OCR). The OCR is used to provide consistent configuration information of applications to Oracle Clusterware. Oracle Clusterware is responsible for making this information available to all nodes in the cluster.

WHO IS EXPECTED TO USE THE ORACLE CLUSTERWARE FRAMEWORK ?

It is expected that Developers, Systems Administrators and Database Administrators will use this feature to protect applications by creating an action program to allow Oracle Clusterware to protect their application on failure.

It is also expect that independent software vendors will use Oracle Clusterware to help protect their applications. They would provide an 'action program' together with recommendations for the 'application profile'.

WHERE CAN I GO FOR MORE INFORMATION?

The following sections of this paper describe a simple application and how you use Oracle Clusterware to protect it.

A chapter and appendix are available in the Oracle Database 10*g* Release 2 documentation set covering this material in great detail. See 'Oracle® Real Application Clusters Administration and Deployment Guide'. For more information on specific relevant sections of this documentation see the 'references' section of this paper.

USING THE ORACLE CLUSTERWARE FRAMEWORK - A WORKED EXAMPLE

This worked example starts from scratch with a new application. It shows how you create an action program . How you create an application profile How you register with Oracle Clusterware

It then goes on to show you how to test your application with common failure scenarios

A simple example is used to show how Oracle Clusterware can be used to protect an application.

Description of Service

A sample 'C' program 'myapp' is included in the appendix of this documents. 'myapp' is a simple application that listens on a network socket. When it receives data from a network client connection it returns the date and time from the machine as a string. This code was compiled and run on Linux (RedHat AS3 Update 4) and, with very little change, will run on any of Oracle's supported 10*g*R2 platforms. The source code is available as Appendix A

The Program name when compiled is 'myapp'. The Oracle Clusterware name for the application will be 'Oracle Date Time Service'. Note the code samples provided in Appendices A and B are only a brief example. They do not include error checking or buffer overrun bounds checking etc

When executed this program sits in a blocking read on a network socket. To test this program using telnet from a client open a connection to port 8087 using the application virtual IP (VIP) as the hostname.

After hitting [enter] the application will return the current date and time from the server to the telnet session. It will then close the telnet session.

The telnet request can be repeated as many times as required.

After killing this process using the linux 'kill –9' command repeated telnet requests will fail. The application providing the service has gone away

The application is not yet under the protection of Oracle Clusterware.

The environment used for this example is a 2 node Linux cluster with Oracle 10*g* Release 2 Clusterware installed. gcc was used to compile the C source code. The 2 nodes used are mangrid2-vm and mangrid3-vm.

Placing An application under the protection of Oracle Clusterware

Now the four-step procedure needs to be followed to protect the application. The following steps include the use of the commands crs_register and crs_profile. These can be found in the $ORA_CRS_HOME/bin directory.

Create an Application VIP

The VIP is used by clients to locate the application irrespective of the node it is running on.

Oracle provides a script than can be used to create application VIPs. They exists as protected resources. The following commands must be used to create the application VIP.

In this VIP section replace the values as indicated

value	Description
$ORA_CRS_HOME	The location of the Oracle Clusterware home
eth0	The name of the public network adapter
138.3.83.78	The new IP Address to be created
255.255.240.0	The subnet mask for the IP (should be the same as the base eth0 subnet mask)

As oracle operating system user create a profile for the VIP.

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_profile  -create
          myvip \
          -t application \
          -a $ORA_CRS_HOME/bin/usrvip \
          -o oi=eth0,ov=138.3.83.78,on=255.255.240.0
```

In this example 138.3.83.78 resolves via DNS to mangrid-vm.

As oracle operating system user register the VIP with Oracle Clusterware.

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_register myvip
```

The Application VIP script has to run as root. As the root operating system user change the owner of the resource.

```
[root@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_setperm myvip
                                -o root
```

As root operating system user allow oracle to execute this script

```
[root@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_setperm myvip
                        -u user:oracle:r-x
```

As the oracle OS user start the VIP

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_start myvip
```

Create an Action Program

An action program is how the Oracle Clusterware Framework interacts with 3rd party applications. It provides a method for starting, stopping and checking the status of an application

The action program needs to accept 1 of 3 different parameters: start, stop or check.

If the start/stop commands are successful they should return 0.
If they are unsuccessful they should return 1.
If the check command discovers the process is running it should return 0.
If the process is not running it should return 1.

The source code for the 'Action Program' used to check Oracle Time Date Service is in Appendix B. Remember this program could be written in any scripting language capable of starting, stopping, checking the status and return error codes to the calling process, which is actually the Oracle Clusterware. Compile this program and call it myappcheck, saved in /tmp.

Create an Application Profile

The profile defines how the Application is registered inside the Oracle Cluster Registry

The Application Profile is a simple text file with some name-value keypairs. Instead of users crafting this file by hand Oracle provides an executable 'crs_profile' which accepts command line parameters for profile creation or manipulation. This executable then generates the necessary Application Profile file.

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_profile \
         -create myapp \
         -t application \
         -d "Oracle Date Time Service" \
         -r myvip \
         -a /tmp/myappcheck \
         -o ci=5,ra=60
```

The parameters used here are:

Name	Value	Description
-create	myapp	Name of the application as stored inside the OCR.
-t	application	Type of OCR entry (must be Application).
-d	"Oracle Date Time Service"	'Long' name of the application.
-r	myvip	Name of the Oracle Clusterware managed resource that must be in status ONLINE for our application to start. The VIP
-a	/tmp/myappcheck	Name of the action program used to start, stop and check the application.
-o		See following entries in table.
	ci=5	Check Interval
	ra=60	Restart Attempts

Many of the parameters are optional and have default values applied if not specified on the command line. These are detailed in Appendix E of the Oracle® Real Application Clusters Administration and Deployment Guide 10*g* Release 2 (10.2).

Remember that although the generated 'Application Profile' file is a simple text file the supported method of generating this file is by using the crs_profile command to generate this file rather than a text editor. Using crs_profile guarantees the correct syntax of the Application Profile file. The default name for this profile file is <profilename>.cap and, when run by a non root user, will be generated in the $ORA_CRS_HOME\crs\public directory. Scripts generated by commands run as a root user get saved in the $ORA_CRS_HOME\crs\profile directory. The Application Profile file generated by the above crs_profile command can be found in Appendix C.

Register the Application Profile with Oracle Clusterware

Registering inserts information from the Application Profile into the Oracle Cluster Registry

Use the supplied command crs_register to register the application with Oracle Clusterware. When run by a non-root user the crs_register command will look in the $ORA_CRS_HOME/crs/public directory for a file called <passed parameter>.cap so in this case the file is $ORA_CRS_HOME/crs/public/myapp.cap. It uses this file to register the application with Oracle Clusterware.

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_register myapp
```

After successfully competing the above step the application will be registered with Oracle Clusterware, entries are made in the Oracle Cluster Registry (OCR). Oracle Clusterware can now control the availability of this service, restarting on process failure and relocating on node failure.

Once registered the profile stored in the OCR can be changed dynamically using the crs_register –update command e.g. The following command alters the restart attempts value in the OCR.

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_register myapp \
                                                           -update \
                                                           -o ra=3
```

Managing the Application when protected by Oracle Clusterware

Oracle provides a series of commands that can be used to query the state of the application as managed by Oracle Clusterware. Other commands start , stop and relocate the application.

A series of crs command are used to manage and check the state of the protected application. The next section details various examples of controlling resources using these commands.

Querying the state of the application

To query the status of the newly registered application use the **crs_stat** command. This causes Oracle Clusterware to call the action program **myappcheck** with the **check** parameter. Initially, after registration, the application is offline.

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_stat myapp
NAME=myapp
TYPE=application
TARGET=OFFLINE
STATE=OFFLINE
```

Starting your application

To start the application we use the **crs_start** command. Oracle Clusterware calls the action program passing the **start** parameter.

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_start myapp
Attempting to start `myapp` on member `mangrid2-vm`
Start of `myapp` on member `mangrid2-vm` succeeded.
```

If we then query the status of the application we see that it is online.

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_stat myapp
NAME=myapp
TYPE=application
TARGET=ONLINE
STATE=ONLINE on mangrid2-vm
```

We can query the state of all Oracle Clusterware registered applications using the –t –v options to crs_stat

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_stat -t -v
Name           Type           R/RA   F/FT   Target    State     Host
----------------------------------------------------------------------
ora....-vm.gsd application    0/5    0/0    ONLINE    ONLINE    mangrid2-vm
ora....-vm.ons application    0/3    0/0    ONLINE    ONLINE    mangrid2-vm
ora....-vm.vip application    0/0    0/0    ONLINE    ONLINE    mangrid2-vm
ora....-vm.gsd application    0/5    0/0    ONLINE    ONLINE    mangrid3-vm
ora....-vm.ons application    0/3    0/0    ONLINE    ONLINE    mangrid3-vm
ora....-vm.vip application    0/0    0/0    ONLINE    ONLINE    mangrid3-vm
myapp          application    0/3    0/0    ONLINE    ONLINE    mangrid2-vm
myvip          application    0/1    0/0    ONLINE    ONLINE    mangrid2-vm
```

Relocating the application

The application can be relocated to another node in the cluster using the **`crs_relocate -f`** command.

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_relocate -f myapp
Attempting to stop `myapp` on member `mangrid2-vm`
Stop of `myapp` on member `mangrid2-vm` succeeded.
Attempting to stop `myvip` on member `mangrid2-vm`
Stop of `myvip` on member `mangrid2-vm` succeeded.
Attempting to start `myvip` on member `mangrid3-vm`
Start of `myvip` on member `mangrid3-vm` succeeded.
Attempting to start `myapp` on member `mangrid3-vm`
Start of `myapp` on member `mangrid3-vm` succeeded.
```

Querying the state of the application shows that it is still online but is running on the other node in our cluster.

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_stat myapp
NAME=myapp
TYPE=application
TARGET=ONLINE
STATE=ONLINE on mangrid3-vm
```

Obtaining further information about the status of the application

The crs_stat –v command can be used to view additional information about the status of the application. For example: The number of times the application has failed over and the number times the application has failed. For a full list see Chapter 15 of the 'Oracle® Real Application Clusters Administration and Deployment Guide 10*g* Release 2 (10.2)'

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_stat  -v myapp
NAME=myapp
TYPE=application
RESTART_ATTEMPTS=3
RESTART_COUNT=0
FAILURE_THRESHOLD=0
FAILURE_COUNT=0
TARGET=ONLINE
STATE=ONLINE on mangrid3-vm
```

Stopping the application

To stop the application from running use the **`crs_stop`** command. The action program gets called with the **`stop`** parameter

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_stop myapp
Attempting to stop `myapp` on member `mangrid3-vm`
Stop of `myapp` on member `mangrid3-vm` succeeded.
```

Removing the application from Oracle Clusterware protection

Finally the application can remove from the protection of Oracle Clusterware using the **crs_unregister** command. Oracle Clusterware removes the configuration details relating to myapp from the Oracle Cluster Registry.

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_unregister myapp
```

Subsequent queries of the status of the application return an error, as Oracle Clusterware is not protecting the application.

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_stat myapp
CRS-0210: Could not find resource `myapp`.
```

TESTING ORACLE CLUSTERWARE PROTECTION OF THE APPLICATION WHEN FAILURE OCCURS

Now we can test both: Application failure and node failure. In each case Oracle Clusterware will restart the application

The following sections details some common failure scenarios and how Oracle Clusterware helps to manage failure.

If the application was 'unregistered' then it must be re-registered and started using the relevant CRS commands.

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_register myapp
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_start myapp
Attempting to start `myapp` on member `mangrid2-vm`
Start of `myapp` on member `mangrid2-vm` succeeded.
```

Testing Application Failure

Application failure will be tested from a third machine 'mangrid1-vm'. Once again a telnet session is opened and connected to port 8087 on mangrid-vip. Hit the [ENTER] key to return

```
[oracle@mangrid1-vm oracle]$ telnet mangrid-vip 8087
Trying 138.3.83.78...
Connected to mangrid-vip.uk.oracle.com (138.3.83.78).
Escape character is '^]'.

Fri Mar 11 13:17:52 2005
Connection closed by foreign host.
```

As before the date time should be returned from the server.

On the server, using an OS command, kill the application.

```
[oracle@mangrid2-vm oracle]$ ps -aef | grep myapp
oracle    1392  3076  0 13:17 pts/4    00:00:00 /tmp/myapp
[oracle@mangrid2-vm oracle]$ kill -9 1392
```

Now try requesting the datetime again using telnet. This time the service will once again return the server date and time.

```
[oracle@mangrid1-vm oracle]$ telnet mangrid-vip 8087
Trying 138.3.83.78...
Connected to mangrid-vip.uk.oracle.com (138.3.83.78).
Escape character is '^]'.

Fri Mar 11 13:18:03 2005
Connection closed by foreign host.
```

Remember, as configured, the action program only checks the state of the protected service every 5 seconds) The Oracle Clusterware detected the application failure and restarted it.

Testing Node Failure

Killing the node (a power off) will force Oracle Clusterware to relocate and start the application on another node in the cluster. The VIP should failover with the application so the same request (to mangrid-vip) should return the date/time from the node: mangrid3-vm.

First check which node is running the application

```
[oracle@mangrid2-vm oracle]$ $ORA_CRS_HOME/bin/crs_stat myapp
NAME=myapp
TYPE=application
TARGET=ONLINE
STATE=ONLINE on mangrid2-vm
```

Then request the date time from the application

```
[oracle@mangrid1-vm oracle]$ telnet mangrid-vip 8087
Trying 138.3.83.78...
Connected to mangrid-vip.uk.oracle.com (138.3.83.78).
Escape character is '^]'.

Fri Mar 11 13:21:03 2005
Connection closed by foreign host.
```

Now power off the node hosting the application.

The status of the application can be checked from the surviving node

```
[oracle@mangrid3-vm oracle]$ $ORA_CRS_HOME/bin/crs_stat myapp
NAME=myapp
TYPE=application
TARGET=ONLINE
STATE=ONLINE on mangrid3-vm
```

and then request the date time

```
[oracle@mangrid1-vm oracle]$ telnet mangrid-vip 8087
Trying 138.3.83.78...
Connected to mangrid-vip.uk.oracle.com (138.3.83.78).
Escape character is '^]'.

Fri Mar 11 13:21:13 2005
Connection closed by foreign host.
```

Oracle Clusterware has detected the node failure and has relocated services and applications, as defined in the Oracle Cluster Registry to the surviving nodes.

SUMMARY

Oracle Database 10*g* Release 2 brings your applications the same level of protection as the Oracle Database processes benefited from with Oracle Database 10*g* Release 1.

The architecture is open and flexible. A simple action script is all that is required to enable this protection. Oracle provides the underlying infrastructure to facilitate this.

- The Oracle Cluster Registry enables visibility of configuration information across all nodes of a cluster.
- The Virtual. IP infrastructure has been extended to enable you to create VIP's that can be assigned to your applications.
- Applications that required locally stored configuration data can use Oracle's Clustered File System to make that data visible to the nodes in a cluster.

This framework is an integral part of the Oracle Clusterware offering.

As Oracle Database 10*g* Release 2 begins to be adopted more and more by application vendors it is expected that they will provide action scripts together with recommendations for profile parameters to allow Oracle Clusterware to successfully manage the protection and availability of their application.

REFERENCES

Oracle® Real Application Clusters Administration and Deployment Guide
10*g* Release 2 (10.2)
Part Number B14197-01
- Chapter 15 Making Applications Highly Available Using the Oracle Clusterware
- Appendix E High Availability Oracle Clusterware Command-Line Reference and C API

APPENDIX A SOURCE CODE FOR ORACLE DATE TIME SERVICE

This source code is the program used as the Oracle Date Time Service. It should be compiled into an executable called oracledatetime and located in the /tmp directory on all nodes in the cluster.
The source file is named: myapp.c.

Use the following command to create the application.
gcc myapp.c -o /tmp/myapp

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <time.h>
#define PORT 8087

int main() {
int insock, outsock, addrlen;
struct sockaddr_in inaddr, from;
time_t timeval;
char buff[100];
int nallowreuse = 1;

  insock = socket( PF_INET, SOCK_STREAM, 0);
  inaddr.sin_family = AF_INET;
  inaddr.sin_addr.s_addr = htonl(INADDR_ANY);
  inaddr.sin_port = htons(PORT);
  setsockopt(insock, SOL_SOCKET, SO_REUSEADDR, (char *)&nallowreuse, sizeof(nallowreuse));
  bind(insock, (struct sockaddr *) &inaddr, sizeof(inaddr));
  listen(insock,5);
  while (strncmp(buff, "EXIT", 4) != 0) {
    addrlen = sizeof(inaddr);
    outsock = accept( insock, (struct sockaddr*) &from, &addrlen);
    read(outsock, buff, sizeof(buff));
    if (strncmp(buff, "EXIT",4) == 0)
       close(insock);
    else
       {
       time(&timeval);
       strcpy(buff, ctime(&timeval));
       write(outsock,buff,strlen(buff));
       }
    close(outsock);
  }
}
```

APPENDIX B SOURCE CODE FOR THE ACTION PROGRAM

This source code is used as the action program to start, stop and check the status of the Oracle Date Time Service. It should be compiled into an executable called myappcheck and located in the /tmp directory on all nodes.
The source file is named: myappcheck.c.

Use the following command to create the application.
gcc myappcheck.c -o /tmp/myappcheck

```
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define SUCCESS 0
#define FAILURE 1
#define PORT 8087
#define MYPROCESS "//tmp//myapp &"

int main(int argc, char ** argv[])
{
char command[128];
int ret;
int sockfd;
int len;
struct sockaddr_in address;

    strcpy(command, (char *)argv[1]);
    ret = FAILURE;
    if (strcasecmp(command, "start") == 0)
        {
        system(MYPROCESS);
        ret = SUCCESS;
        }
    else if (strcasecmp(command, "stop") == 0)
        {
        sockfd = socket(PF_INET, SOCK_STREAM, 0);
        address.sin_family = AF_INET;
        address.sin_addr.s_addr = inet_addr("127.0.0.1");
        address.sin_port = htons(PORT);
        if (connect(sockfd, (struct sockaddr *)&address, sizeof(address)) != -1)
            {
            write(sockfd, "EXIT", 4);
            sleep(2);
            close(sockfd);
            }
        ret = SUCCESS;
        }
    else if (strcasecmp(command, "check") == 0)
        {
        sockfd = socket(PF_INET, SOCK_STREAM, 0);
        address.sin_family = AF_INET;
        address.sin_addr.s_addr = inet_addr("127.0.0.1");
        address.sin_port = htons(PORT);
        if (connect(sockfd, (struct sockaddr *)&address, sizeof(address)) != -1)
            ret = SUCCESS;
        }
    return(ret);
}
```

APPENDIX C THE APPLICATION PROFILE FILE

This is the Application Profile File generated by the crs_profile command.

```
NAME=myapp
TYPE=application
ACTION_SCRIPT=/tmp/myappcheck
ACTIVE_PLACEMENT=0
AUTO_START=restore
CHECK_INTERVAL=5
DESCRIPTION=Oracle Date Time Service
FAILOVER_DELAY=0
FAILURE_INTERVAL=0
FAILURE_THRESHOLD=0
HOSTING_MEMBERS=
OPTIONAL_RESOURCES=
PLACEMENT=balanced
REQUIRED_RESOURCES=myvip
RESTART_ATTEMPTS=60
SCRIPT_TIMEOUT=60
START_TIMEOUT=0
STOP_TIMEOUT=0
UPTIME_THRESHOLD=7d
USR_ORA_ALERT_NAME=
USR_ORA_CHECK_TIMEOUT=0
USR_ORA_CONNECT_STR=/ as sysdba
USR_ORA_DEBUG=0
USR_ORA_DISCONNECT=false
USR_ORA_FLAGS=
USR_ORA_IF=
USR_ORA_INST_NOT_SHUTDOWN=
USR_ORA_LANG=
USR_ORA_NETMASK=
USR_ORA_OPEN_MODE=
USR_ORA_OPI=false
USR_ORA_PFILE=
USR_ORA_PRECONNECT=none
USR_ORA_SRV=
USR_ORA_START_TIMEOUT=0
USR_ORA_STOP_MODE=immediate
USR_ORA_STOP_TIMEOUT=0
USR_ORA_VIP=
```

Using Oracle Clusterware to Protect 3rd Party Applications
February 2008
Author: Philip Newlan
Version 1.4

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com