Oracle Database Failover Cluster with Grid Infrastructure 11g R2

Robert Bialek
Trivadis GmbH
Munich, DE

Keywords

Oracle Grid Infrastructure, Oracle Clusterware, Failover Cluster, Oracle Database High Availability, Cluster Resource Monitoring, Cluster Resource Placement, Cluster Resource Management, Resource Action Scripts.

Introduction

Oracle database high availability is not a new trend in the IT world. There are many technologies or build-in database features which are designed to increase the uptime of the database service, such as Real Application Cluster, Active/Passive (Failover) Cluster, Data Guard, etc. Database high availability solution should always be selected after careful business requirements analysis. Every technology has its pros and cons and there is no "best solution for everyone".
This paper provides an overview of how to implement an Oracle database failover cluster based on Oracle Grid Infrastructure (Oracle Clusterware) 11g Release 2 and how the Oracle Clusterware failover resource management works. Oracle Clusterware is a general purpose, sophisticated cluster software, not constrained to high available database resources; it can be used for almost every high available software component. Furthermore, the presented failover database solution can be used with every database edition and release from 10g Release 2 onwards, without additional license fees (RDBMS license is assumed; for details consult *Oracle Database Licensing Information 11g Release 2*).

Cluster Resources

Every software component managed by Oracle Clusterware (e.g. VIP, database, listener, etc.) is registered in the cluster repository (OCR) as a **resource**. Every resource contains a set of resource **attributes** which generally describe how to manage it within the cluster. To achieve the desired resource behavior, especially of user-defined failover resources, it is very important to carefully set up their attributes to appropriate values. The following are the most important attributes used for failover resources (for a full list and description see *Oracle Clusterware Administration and Deployment Guide*):

To define the binary and additionally an action script able to manage a cluster resource (start, stop, check and clean functionality)

- AGENT_FILENAME
- ACTION_SCRIPT

To define the resource placement in the cluster

- PLACEMENT
- SERVER_POOLS

Used by cluster resource monitoring (restart and failover operations)

- CHECK_INTERVAL
- RESTART_ATTEMPTS
- UPTIME_THRESHOLD
- FAILURE_THRESHOLD
- FAILURE_INTERVAL

Attributes which define relationships between cluster resources (e.g. listener is dependent on the VIP) used during startup, switchover, failover, resource shutdown and crash:

- START_DEPENDENCIES
- STOP_DEPENDENCIES

Generally, resource attributes must be predefined in a **resource type**. Oracle Clusterware is supplied with three **generic** predefined resource types (useful for user defined resources; there are more types used by Oracle internally for VIPs, database services, etc.):

- application – exists only for backward compatibility, shouldn't be used anymore
- cluster_resource – new in 11g Release 2; should be used for cluster aware failover resources
- local_resource – new in 11g Release 2; cluster resources available only on a specific server; no failover functionality

It is possible, and in some cases very useful to create your own resource types with new attributes not predefined by the mentioned generic resource types (for example one resource type per database version with default values for ORACLE_HOME and some other specific to your environment variables). Every newly created resource type must be based on one of the Oracle predefined resource types (in the following example the FO10g.type base on cluster_resource; FO10g.type will also inherit all attributes from cluster_resource type):

```
crsctl add type FO10g.type -basetype cluster_resource \
>   -attr "ATTRIBUTE=TNS_ADMIN,TYPE=string,FLAGS=REQUIRED" \
>   -attr "ATTRIBUTE=ORACLE_HOME,TYPE=string, FLAGS=REQUIRED, \
>          DEFAULT_VALUE=/u00/app/oracle/product/11.2.0.2" \
```

Action Script

Oracle Clusterware can manage any kind of application providing, an application-specific action script or agent able to start/stop/check and clean the application exists. They need to follow a certain structure and must return an exit code 0 in case of a success and other in case of a failure. Example structure for a database action script:

```
case ${Action} in
          START)
            #code to start the database instance
            lerr=$?
            ;;
          STOP)
            #code to stop the database instance
            lerr=$?
```

```
            ;;
          CHECK)
            #code to check the database instance
            lerr=$?
            ;;
          CLEAN)
            #code to perform a clean operation
            lerr=$?
            ;;
          *)
            echo "ERROR: ${Action} is not a valid parameter."
            exit 1
            ;;
esac
...
if [ $lerr -eq 0 ] ; then
  exit 0
else
  exit 1
fi
```

CLEAN is a new and necessary action script parameter in the version 11g Release 2. This script code will be called by the Oracle Clusterware agent, amongst others in the following cases:

- Resource START failed, check routine returns code <> 0
- Resource STOP failed, action script timed out (default 180 sec.)

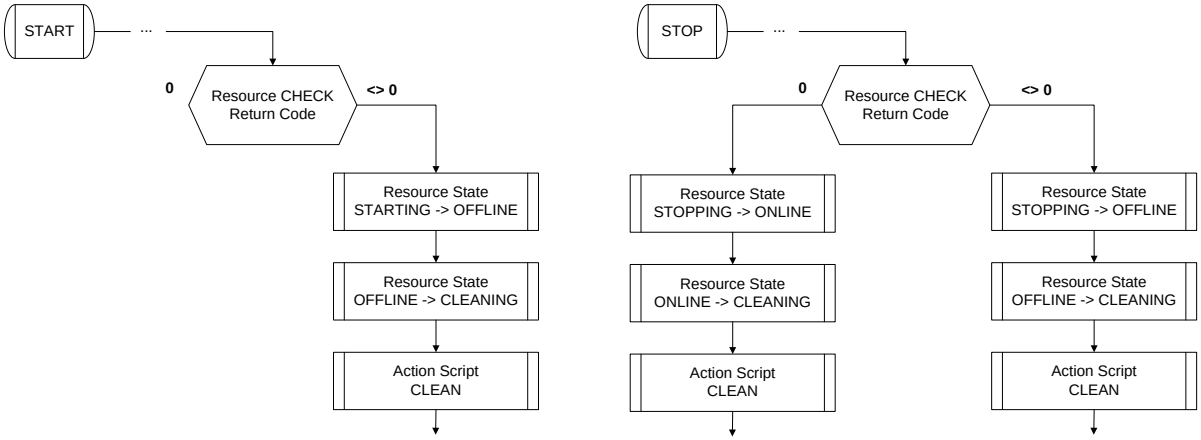

Figure 1 Agent execution flow during failed resource START/STOP

The user-defined resource action script will be called by an Oracle Clusterware supplied **generic** agent. The goal of the generic agent is to call an application specific action script with an appropriate input parameter (depending on the resource state). There are two generic agents:

- appagent – exists for backward compatibility, shouldn't be used anymore
- scriptagent – new in the version 11g Release 2 (will be automatically selected if cluster_resource or local_resource type has been used)

In the following example scriptagent will call the crs_db.ksh action script to manage the failover database resource FO111.inst:

```
crsctl add resource FO111.inst -type cluster_resource \
> -attr "ACTION_SCRIPT=/u00/app/oracle/local/dba/bin/crs_db.ksh, \
```

Resource Placement

Oracle Clusterware considers during resource startup the PLACEMENT attribute, to decide on which node it should be started. This attribute can have the following values:

1. PLACEMENT=balanced

In this case less loaded cluster node will be preferred to servers with greater load. Load does not mean the cluster will check the actual server load (CPU usage, etc.), but rather check additional resource attribute called LOAD (which defaults to 1). The value of the LOAD attribute quantitatively represents how much server capacity a resource consumes, relative to other resources. Oracle Clusterware will try to place a resource on a server with the least total load (sum of the LOAD attribute) of all running resources. Example creation of a resource with LOAD value 5:

```
crsctl add resource FO111.inst -type cluster_resource \
> -attr "PLACEMENT=balanced, \
>        LOAD=5, \
```

2. PLACEMENT=favored

This value can be used to set up a preferred/available server configuration. Oracle Clusterware considers first servers assigned to the SERVER_POOL resource attribute. Only if no servers assigned to the SERVER_POOL attribute are available, the resource will be placed on any other available server. Example with one node assigned to a server pool called FO.sp:

```
crsctl add srvpool FO.sp -attr "PARENT_POOLS=Generic, SERVER_NAMES=white"

crsctl add resource FO111.inst -type cluster_resource \
> -attr "PLACEMENT=favored,\
>        ACTIVE_PLACEMENT=0,\
>        SERVER_POOLS=FO.sp,\
```

In the example above, there is additional attribute called ACTIVE_PLACEMENT. For resources with PLACEMENT=favored and ACTIVE_PLACEMENT=1 Oracle Clusterware reevaluate the placement of the resource if it currently runs on a non-favored server and a favored one has joined the cluster (stop on the current server and restart on the favored one). ACTIVE_PLACEMENT=0 means no placement reevaluation.

3. PLACEMENT=restricted

Oracle Clusterware considers only servers assigned to the SERVER_POOL resource attribute, other servers will not be considered, also if no servers are currently assigned to the server pool:

```
crsctl add resource FO111.inst -type cluster_resource \
> -attr "PLACEMENT=restricted,\
>        SERVER_POOLS=FO.sp,\
```

Resource Monitoring

One of the most important jobs of every cluster stack, and Oracle Clusterware is not an exception to this rule, is the monitoring of all managed cluster resources, i.e. restarting or failing them over to other

cluster nodes in case of a failure. The cluster monitoring behavior can be controlled by a set of the following resource attributes:

- CHECK_INTERVAL – time in seconds, between repeated executions of the resource CHECK action. Resource check will be executed only for resources in an intended ONLINE resource state (unless OFFLINE_CHECK_INTERVAL attribute has been set)

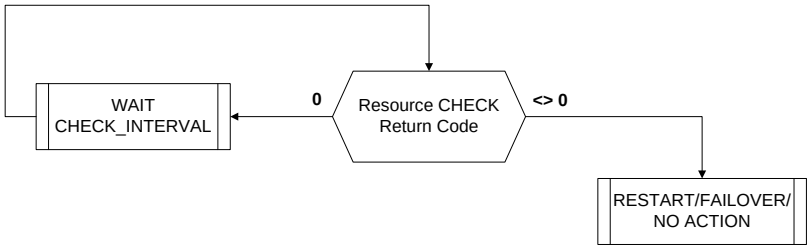

Figure 2 Resource check flow

- UPTIME_THRESHOLD – attribute used to define the stability of a resource, i.e. after a time period defined by this attribute elapses, Oracle Clusterware will reset internally the value of the read only attribute RESTART_COUNT to 0
- RESTART_ATTEMPTS – number of desired resource restarts on the current cluster node within UPTIME_THRESHOLD. In case the RESTART_COUNT value reaches RESTART_ATTEMPTS within the UPTIME_THRESHOLD value, the cluster will not consider further restarts on the current node and depending on the configuration failover it to other node or let the resource in an OFFLINE state. RESTART_ATTEMPTS=0 means no local restarts; always failover (if configured)

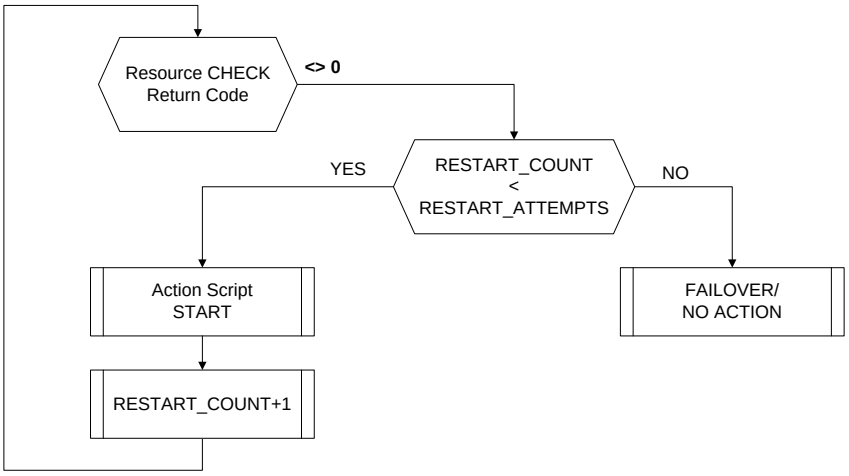

Figure 3 Resource restart flow

- FAILURE_INTERVAL – time in seconds, during which the FAILURE_THRESHOLD attribute will be applied
- FAILURE_THRESHOLD – the number of cluster resource failovers minus one within FAILURE_INTERVAL value. FAILURE_THRESHOLD=2 means 1 resource failover; FAILURE_THRESHOLD=1 means no failover at all.

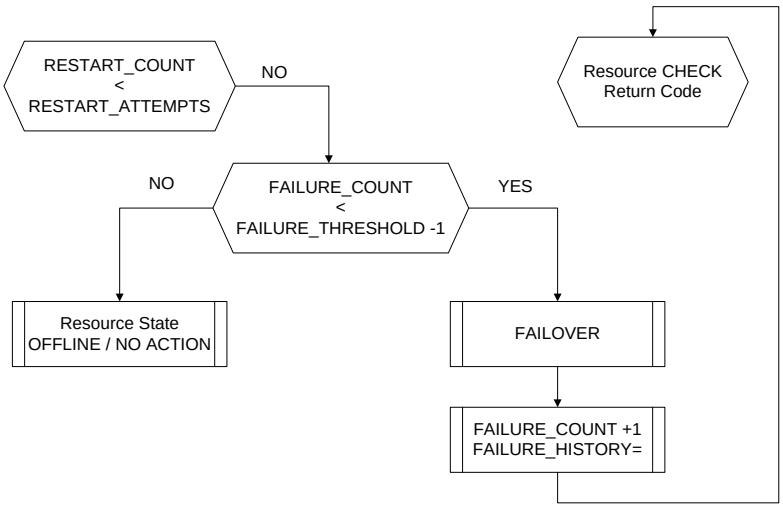

Figure 4 Resource failover flow

Example resource monitoring attribute values:

```
CHECK_INTERVAL=15
RESTART_ATTEMPTS=2
UPTIME_THRESHOLD=8h
FAILURE_THRESHOLD=2
FAILURE_INTERVAL=3600
```

It means:

- maximum two resource restarts per server in 8h interval
- maximum one resource failover in 60 minutes interval
- Summing it up – maximum five resource startup operations after a failure, subsequently resource remains in status OFFLINE (administrator intervention required)

Every restart and failover will be logged to the cluster alert log file on the OCR Master server (one node in cluster):

```
[crsd(4058)]CRS-2765:Resource 'FO111.inst' has failed on server 'white'.
[crsd(4058)]CRS-2765:Resource 'FO111.inst' has failed on server 'white'.
[crsd(4058)]CRS-2765:Resource 'FO111.inst' has failed on server 'white'.
[crsd(4058)]CRS-2771:Maximum restart attempts reached for resource
'FO111.inst'; will not restart.
[crsd(4058)]CRS-2765:Resource 'FO111.inst' has failed on server 'red'.
[crsd(4058)]CRS-2765:Resource 'FO111.inst' has failed on server 'red'.
[crsd(4058)]CRS-2765:Resource 'FO111.inst' has failed on server 'red'.
[crsd(4058)]CRS-2771:Maximum restart attempts reached for resource
'FO111.inst'; will not restart.
[crsd(4058)]CRS-2768:Failure threshold exhausted by resource 'FO111.inst'.
```

Registering Database Resources in Cluster

Depending on the configuration and project requirements, up to three additional cluster resources will be registered in the Oracle Clusterware repository. Some example configurations:

1. VIP, listener and database resource with an action script. In this configuration all three resources will be treated as a one “logical resource”. Failover of the database instance, will also cause failover of the VIP and listener (correctly set up dependencies assumed). Database clients will use the dedicated VIP and listener port for the connectivity.

First we create the dedicated VIP resource (possible also with an Oracle supplied script <GRID_HOME>/bin/appvipcfg). The VIP has a START and STOP dependencies on the first (default) public network. ACL (Access Control List) has been set up to allow also the oracle user start/stop of the VIP resource:

```
sudo crsctl add resource FO111.vip -type app.appvip_net1.type \
> -attr "USR_ORA_VIP=172.20.30.31,\
>       DESCRIPTION=VIP resource for FO111 database,\
>       START_DEPENDENCIES=hard(ora.net1.network) pullup(ora.net1.network),\
>       STOP_DEPENDENCIES=hard(ora.net1.network),\
>       ACL='owner:root:rwx,pgrp:root:r-x,other::r--,user:oracle:r-x'"
```

In the second step we create the listener cluster resource. The listener resource has a START and STOP dependencies on the FO111.vip (VIP failure will lead to stop of the listener resource; automatic failover of the listener has been deactivated):

```
crsctl add resource FO111.lsnr -type cluster_resource \
> -attr "ACTION_SCRIPT=/u00/app/oracle/local/dba/bin/crs_listener.ksh,\
>        CHECK_INTERVAL=15,\
>        RESTART_ATTEMPTS=5,\
>        FAILURE_THRESHOLD=1,\
>        FAILURE_INTERVAL=3600,\
>        UPTIME_THRESHOLD=8h,\
>        DESCRIPTION= Listener resource for FO111 database,\
>        START_DEPENDENCIES=hard(FO111.vip) pullup(FO111.vip),\
>        STOP_DEPENDENCIES=hard(FO111.vip)"
```

In the last step we create the database cluster resource. It has appropriate dependencies on the required ASM disk groups, VIP and the listener resources. The START dependency does not include the VIP directly (will be internally resolved by the listener START dependency: VIP -> listener -> database). VIP resource crash will cause the database resource to go into OFFLINE state; crash of the listener resource itself will not lead to stop or relocate of the database resource:

```
crsctl add resource FO111.inst -type cluster_resource \
> -attr "ACTION_SCRIPT=/u00/app/oracle/local/dba/bin/crs_db.ksh,\
>        PLACEMENT=balanced,\
>        CHECK_INTERVAL=15,\
>        RESTART_ATTEMPTS=2,\
>        FAILURE_THRESHOLD=2,\
>        FAILURE_INTERVAL=3600,\
>        UPTIME_THRESHOLD=8h,\
>        DESCRIPTION=Oracle database FO111 resource,\
>        START_DEPENDENCIES='hard(ora.U01.dg,ora.U02.dg,FO111.lsnr)
                           pullup(ora.U01.dg,ora.U02.dg,FO111.lsnr)',\
>        STOP_DEPENDENCIES='hard(intermediate:ora.asm,shutdown:ora.U01.dg,
                               shutdown:ora.U02.dg,FO111.vip)'"
```

Database resource startup will cause recursively automatic startup of all dependent resources:

```
oracle@white:~/ [+ASM1] crsctl start resource FO111.inst
CRS-2672: Attempting to start 'FO111.vip' on 'red'
CRS-2676: Start of 'FO111.vip' on 'rac2' succeeded
CRS-2672: Attempting to start 'FO111.lsnr' on 'red'
CRS-2676: Start of 'FO111.lsnr' on 'rac2' succeeded
CRS-2672: Attempting to start 'FO111.inst' on 'red'
CRS-2676: Start of 'FO111.inst' on 'rac2' succeeded
```

2. One VIP, listener and many database resources with an action script. In this configuration failover of one database will cause cascading failover of other databases which depend on the same VIP and listener resources. Database clients will use the VIP and dedicated listener for the connectivity. Configuration steps are basically the same as in example 1 with the exception that now more than one database resource depends on one VIP and listener:

```
crsctl add resource P312.inst -type cluster_resource \
> -attr "ACTION_SCRIPT=/u00/app/oracle/local/dba/bin/crs_db.ksh,\
>        START_DEPENDENCIES='hard(ora.U01.dg,ora.U02.dg,Prod300.lsnr)
                            pullup(ora.U01.dg,ora.U02.dg,Prod300.lsnr)',\
>        STOP_DEPENDENCIES='hard(intermediate:ora.asm,shutdown:ora.U01.dg,
                                 shutdown:ora.U02.dg,Prod300.vip)',\

crsctl add resource P313.inst -type cluster_resource \
> -attr "ACTION_SCRIPT=/u00/app/oracle/local/dba/bin/crs_db.ksh,\
>        START_DEPENDENCIES='hard(ora.U01.dg,ora.U02.dg,Prod300.lsnr)
                            pullup(ora.U01.dg,ora.U02.dg,Prod300.lsnr)',\
>        STOP_DEPENDENCIES='hard(intermediate:ora.asm,shutdown:ora.U01.dg,
                                 shutdown:ora.U02.dg,Prod300.vip)',\
```

3. Only database resource with an action script. In this configuration we don't use additional dedicated VIPs or listener resources. This configuration uses the new 11.2 SCAN (Single Client Access Name) feature and the default local listeners running out of the Grid Infrastructure home. REMOTE_LISTENER is set statically to the SCAN endpoints (method depends on the database version), LOCAL_LISTENER will be set dynamically to the local host VIP by the action script during resource START. Database clients will use the SCAN IPs and port for the connectivity.

Integration with Enterprise Manager Grid Control

Database failover resources can also be integrated with Enterprise Manager Grid Control. In the first step one registers the database resource, as in the following example:

```
emcli add_target \
> -name="FO111.trivadis.com" \
> -type="oracle_database" \
> -host="white.trivadis.com" \
> -credentials="UserName:dbsnmp;password:dbsnmp;Role:Normal" \
> -properties="SID:FO111;Port:1530;
               OracleHome:/u00/app/oracle/product/11.2.0.2;
               MachineName:italy.trivadis.com "
```

The START section of the database resource action script should be extended by code to automatically relocate the database target to the new agent host during every switchover or failover operations. Example for manual target relocation between cluster nodes:

```
emcli relocate_targets \
> -src_agent=white.trivadis.com:3872 \
> -dest_agent=red.trivadis.com:3872 \
> -target_name=FO111.trivadis.com \
> -target_type=oracle_database \
> -copy_from_src \
> -force=yes
```

Conclusion

Grid Infrastructure (Oracle Clusterware) is a stable, sophisticated and proved cluster stack, with sufficient functionality to implement a database failover cluster. With appropriate resource action scripts, it can be used to manage any high available cluster resources and is not constrained to a specific database edition or version. The stability of this solution depends mainly on the quality of the action script, in particular the CHECK routine.

References

Oracle Clusterware Administration and Deployment Guide 11g Release 2

Contact Address

Robert Bialek
Trivadis GmbH
Lehrer-Wirth-Strasse, 4
D-81829 Munich

Telefon:	+49 (0) 89-99 27 59 30
Fax:	+49 (0) 89-99 27 59 59
E-Mail	robert.bialek@trivadis.com
Internet:	www.trivadis.com