Oracle Database Failover Cluster with Grid Infrastructure 11g Release 2

Robert Bialek
Principal Consultant, MU-IMS
Oracle Certified Master
robert.bialek@trivadis.com
DOAG Regional Meeting
Nürnberg, 31.03.2011
Agenda

- Introduction
- Cluster Resources
- Configuration
- Summary

Data are always part of the game.
Introduction

- **Failover Cluster** is still one of the most popular database service HA solution
  - Cheep, easy to implement
  - Single instance database (administration)

- **Oracle Clusterware** (Grid Infrastructure) can be used to implement it. But:
  - What about the functionality, stability, experiences?
  - Which pros/cons and limitations do we need to consider?

- 11.2 version introduced RAC One Node (failover + live migration)
  - Option for Enterprise Edition
Introduction – Licensing (1)

According to Oracle Database Licensing Information 11gR2:

**Oracle Clusterware can be used to protect any application** (restarting or failing over the application in the event of a failure), **free of charge**, if **one or more** of the following conditions are met:

1. **The server OS is supported by a valid Oracle Unbreakable Linux support contract.**

2. **The product to be protected is either:**
   - **Any Oracle product** (e.g. Oracle Applications, Siebel, Hyperion, Oracle Database EE, Oracle Database XE)
   - **Any third-party product** that directly or indirectly stores data in an Oracle database

3. **At least one of the servers in the cluster is licensed for Oracle Database (SE or EE)**
Introduction – Licensing (2)

- For active/passive Failover Cluster environment you can benefit from the “10-day rule usage”

  *In this type of environment, Oracle permits its licensed Technology customers to run the Technology Programs (listed on the Technology Price List) on an unlicensed spare computer for up to a total of ten separate days in any given calendar year.*

  ...

- Only one failover node per clustered environment is at no charge for up to ten separate days even if multiple nodes are configured as failover nodes.

  ...

Failover Database with Clusterware – OTN


Oracle Clusterware 11g Release 1

Using Oracle Clusterware to protect any kind of application
- Using Oracle Clusterware to Protect 3rd Party Applications (PDF) February 2008
- Using Oracle Clusterware to Protect Oracle Application Server (PDF) November 2005
- Using Oracle Clusterware to Protect an Oracle Database 10g with Oracle Enterprise Manager Grid Control Integration (PDF) February 2008

Oracle application protected by Oracle Clusterware
- Using Oracle Clusterware to Protect A Single Instance Oracle Database 11g (PDF) February 2008
- Siebel CRM Applications protected by Oracle Clusterware (PDF) Providing High Availability for Siebel CRM Applications - January 2008
- Oracle GoldenGate High Availability Using Oracle Clusterware Providing High Availability for Oracle GoldenGate (PDF) - March 2010

Oracle Clusterware 11g Release 2

Using Oracle Clusterware to protect any kind of application
- Not available for Oracle Clusterware 11g Release 2 yet

Oracle application protected by Oracle Clusterware
- Oracle GoldenGate High Availability Using Oracle Clusterware Providing High Availability for Oracle GoldenGate (PDF) - March 2010

Pre-configured agents for Oracle Clusterware
- Providing High Availability for SAP Resources using Oracle Clusterware 11g Release 2 (PDF) August 2010 - NEW

- As of now, no white paper with description for 11.2 cluster
  - 11.1 white paper uses tools/methods which are deprecated (available for backward compatibility) in 11.2

- In the next weeks we will publish an article about it. Stay tuned!
Agenda

- Introduction
- Cluster Resources
- Configuration
- Summary

Data are always part of the game.
Cluster Resources – Introduction

- Every component managed by Oracle Clusterware is registered as a resource

<table>
<thead>
<tr>
<th>HA Resource</th>
<th>Target</th>
<th>State</th>
<th>Node(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ora.maa_site2.db</td>
<td>ON</td>
<td>ON</td>
<td>green red white</td>
</tr>
<tr>
<td>ora.red.vip</td>
<td>ON</td>
<td>ON</td>
<td>red</td>
</tr>
</tbody>
</table>

- Resource defines how to manage application with resource attributes, e.g.:
  - Resource placement, frequency check, start/stop dependencies, etc.

- Every registered resource must have a resource type, which describes its attributes. Only attributes defined in a resource type can be used!

- If you need additional attributes → create your own types, e.g.:

```bash
  crsctl add type FO.type -basetype cluster_resource \
  -attr "ATTRIBUTE=TNS_ADMIN,TYPE=string,FLAGS=REQUIRED" \
  -attr "ATTRIBUTE=ORACLE_HOME,TYPE=string, FLAGS=REQUIRED, \n  DEFAULT_VALUE=/u00/app/oracle/product/10.2.0"
```
Cluster Resources – Resource Type

- As already mentioned ➔ every registered resource base on a resource type

- There are three generic predefined resource types:
  - application – exists only for backward compatibility
  - cluster_resource – for cluster aware resources (subject to switchover/failover, resource cardinality, etc.)
  - local_resource – for resources which should run on each server in a cluster. Local resource instances are managed automatically

  ```bash
  crsctl add resource FO102.lsnr -type cluster_resource \\
  -attr "ACTION_SCRIPT=/u00/app/oracle/local/dba/bin/crs_listener.ksh,\\
  CARDINALITY=1,\\
  ...
  ```

- Other resource types are used for specific Oracle components like listener, VIP, database instance, service, etc.
CRS Resource Management

- **AGENT_FILENAME** – manages resource directly or calls an ACTION_SCRIPT. There are two built-in generic agents:
  - `scriptagent` and `appagent`. Default depends on the resource type
  
  ```
  AGENT_FILENAME=%CRS_HOME%/bin/scriptagent
  ```

- **ACTION_SCRIPT** – script called to start/stop/check/clean a resource by an AGENT_FILENAME
  
  ```
  ACTION_SCRIPT=/u00/app/oracle/local/dba/bin/crs_db.ksh
  ```

  Every resource attribute can be accessed by an ACTION_SCRIPT as a variable with `_CRS_` or `_CAA_` prefix (depends on resource type)

  ```
  ${_CRS_NAME} #Resource Name Attribute
  ${_CRS_RESTART_ATTEMPTS} # RESTART_ATTEMPTS Resource Attribute
  ```
Resource Dependencies

- **START_DEPENDENCIES** – set of relationships considered during resource startup/switchover/failover
  - Dependency types: **hard**, **weak**, **pullup**, ...
  - Modifiers: **intermediate**, **global**, **concurrent**, **always**, **type** ...

  ```
  START_DEPENDENCIES='hard(F0111.vip) pullup(F0111.vip)
  START_DEPENDENCIES='hard(ora.DATA.dg) pullup(ora.DATA.dg)
  weak(type:ora.listener.type, global:type:ora.scan_listener.type)'
  ```

- REQUIRED_RESOURCES, OPTIONAL_RESOURCES are deprecated in 11.2. Available only for resources of **application** type

- **STOP_DEPENDENCIES** – set of relationships considered during resource shutdown/crash
  - only **hard** dependency type, modifiers: **intermediate**, **global**, **shutdown**

  ```
  STOP_DEPENDENCIES=hard(ora.net1.network)
  ```
Agenda

- Introduction
- Cluster Resources
- Configuration
- Summary

Data are always part of the game.
Registering Database Resources – Overview

- Up to three additional cluster resources need to be created
  - VIP, Listener, Oracle database instance resource

- Resource HA assumptions
  - VIP resources ➔ no restart, always failover including dependences
  - database instance resource
    - try to restart locally, if not possible ➔ failover including dependences
    - not possible to start after failover ➔ DBA intervention required, resource remains OFFLINE
  - listener resource ➔ try to restart only locally, no failover
Registering Database Resources – Dependencies

- Resource dependencies graph

- Resource dependencies graph

```
ora.net1.network

ASM DG

FO VIP

FO LISTENER

FO DATABASE

START_DEPENDENCY ➔ HARD, PULLUP

STOP_DEPENDENCY ➔ HARD
```
Registering Database Resources (1)

- VIP resource (or with $GRID_HOME/bin/appvipcfg$)

```bash
sudo crsctl add resource FO111.vip -type app.appvip.type -attr "USR_ORA_VIP=192.168.122.21,
DESCRIPTION=VIP resource for FO111,
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"
```

- Listener resource

```bash
crsctl add resource FO111.lsnr -type cluster_resource -attr "ACTION_SCRIPT=/u00/app/oracle/local/dba/bin/crs_listener.ksh,
CARDINALITY=1,
DEGREE=1,
PLACEMENT=balanced,
CHECK_INTERVAL=15,
RESTART_ATTEMPTS=5,
FAILURE_THRESHOLD=1,
FAILURE_INTERVAL=3600,
UPTIME_THRESHOLD=8h,
DESCRIPTION=Oracle database listener resource for FO111,
START_DEPENDENCIES=hard(FO111.vip) pullup(FO111.vip),
STOP_DEPENDENCIES=hard(FO111.vip)"
```
Registering Database Resources (2)

- Database instance resource

```sh
crsctl add resource FO111.inst -type cluster_resource
-attr "ACTION_SCRIPT=/u00/app/oracle/local/dba/bin/crs_db.ksh,
CARDINALITY=1,
DEGREE=1,
PLACEMENT=balanced,
CHECK_INTERVAL=15,
RESTART_ATTEMPTS=2,
FAILURE_THRESHOLD=2,
FAILURE_INTERVAL=3600,
UPTIME_THRESHOLD=8h,
DESCRIPTION=Oracle database instance resource,
START_DEPENDENCIES='hard(ora.DATA.dg,ora.FRA.dg,FO111.lsnr)
pullup(ora.DATA.dg,ora.FRA.dg,FO111.lsnr)',
STOP_DEPENDENCIES='hard(intermediate:ora.asm,shutdown:ora.DATA.dg,ora.FR
A.dg,FO111.vip)''
```
Registering Database Resources

DEMO
Resource Monitoring (1)

- Resource restart/failover behavior can be controlled with several attributes

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

- Max. 2 resource restarts per server in 8h interval
- Max. 1 resource failover in 60 min. interval
- To sum it up ➔ max. 5 restarts, subsequently resource remains OFFLINE (admin intervention required)

- **RESTART_ATTEMPTS=0** ➔ no attempt to restart, always failover

- **FAILURE_THRESHOLD=1** ➔ no automatic failover
Resource Monitoring (2)

- Cluster does not manage/monitors a disabled resources (ENABLED=0, either directly or because of a dependency)
  - Before maintenance tasks disable them

```bash
crslctl modify resource FO111.inst –attr “ENABLED=0”
```

- Do not shutdown database instance with SQL*Plus for which a resource is enabled

```
FO111.inst 1 1 state changed from: ONLINE to: OFFLINE
Agent sending message to PE: RESOURCE_STATS[Proxy] ID 20481:778
Agent received the message: RESOURCE_START[FO111.inst 1 1] ID 4098:2980
Preparing START command for: FO111.inst 1 1
FO111.inst 1 1 state changed from: OFFLINE to: STARTING
[start] Executing action script:
/u00/app/oracle/local/da/bin/crs_db.ksh[start]
```
Resource Monitoring (3)

- Restart/failover operations will be logged to CRS alert log, **ONLY** on the CRSD master node!

  ```
  [crsd(29841)] CRS-2765: Resource 'FO111.inst' has failed on server 'rac1'.
  [crsd(29841)] CRS-2765: Resource 'FO111.inst' has failed on server 'rac1'.
  [crsd(29841)] CRS-2771: Maximum restart attempts reached for resource 'FO111.inst'; will not restart.
  [crsd(29841)] CRS-2765: Resource 'FO111.inst' has failed on server 'rac2'.
  [crsd(29841)] CRS-2765: Resource 'FO111.inst' has failed on server 'rac2'.
  [crsd(29841)] CRS-2771: Maximum restart attempts reached for resource 'FO111.inst'; will not restart.
  [crsd(29841)] CRS-2768: Failure threshold exhausted by resource 'FO111.inst'.
  ```

- Resource runtime attributes (monitoring)

  ```bash
  crsctl status resource FO111.inst -v | grep -E '
  > ^RESTART_COUNT|^LAST_RESTART|^FAILURE_COUNT|^FAILURE_HISTORY'
  RESTART_COUNT=1
  FAILURE_COUNT=0
  FAILURE_HISTORY=
  LAST_RESTART=11/19/2010 19:25:06
  ```
Clusterware Resource Monitoring

DEMO
Resource Placement

- Resource placement behavior can be controlled with several attributes

- **PLACEMENT** – determines how to select a server
  - **balanced** ➔ less loaded servers are preferred to servers with greater loads (**LOAD** attribute)
  - **favored** ➔ preferred are servers assigned to SERVER_POOLS (preferred/available server configuration)
  - **restricted** ➔ considers only servers from SERVER_POOLS. May be used for “manual failover” configuration

- **SERVER_POOLS** – affinity between a resource and one or more server pools regarding placement
Resource Placement

DEMO
Production / Test Cluster Configuration

- To better utilize hardware resources
  - Production databases ➔ server 1
  - Test databases ➔ server 2

- In case of server 1 crash
  - Stop automatically test databases
  - Relocate production databases to server 2

- How to achieve this?
  - Use Server Pools with different IMPORTANCE attribute
    
    ```
    crsctl add srvpool Prod.sp -attr "IMPORTANCE=1, MIN_SIZE=1, MAX_SIZE=1"
    crsctl add srvpool Test.sp -attr "IMPORTANCE=0, MIN_SIZE=1, MAX_SIZE=1"
    ```

  - Control the resource placement behavior with SERVER_POOLS
    
    ```
    crsctl add resource FO111.inst -type cluster_resource \ 
    ... \ 
    SERVER_POOLS=Prod.sp,\ 
    ```

© 2011 Oracle Database Failover Cluster with Grid Infrastructure 11g Release 2
Production / Test Cluster Configuration

DEMO
Cluster Resources & EM

- Some tasks can be performed with EM ...
Agenda

- Introduction
- Cluster Resources
- Configuration
- Summary

Data are always part of the game.
Core Messages

- Oracle Clusterware is a stable and proved cluster stack, with sufficient functionality to implement a Failover Database Cluster.

- Carefully design the system, think about cluster node evictions, etc.

- For pre 11.2 databases some additional changes are necessary.

- More and more companies decide to use it (free of charge, support, etc.)

- Very good CLI tools
Thank you!