Why and How You Should Be Using Policy-Managed Oracle RAC Databases

Mark V. Scardina
Oracle Corporation
Redwood Shores, CA USA

Keywords: Database RAC Policy HA Services Manageability Cloud Multitenant PDB

Introduction

The database is no longer the center of the universe. Such an statement would have been heretical just a short time ago. However, the introduction of the Database Cloud and DBaaS has altered the deployment strategy and database management requirements. It is no longer sufficient to plan for simply performance, scalability and high availability. These new deployment models must also consider consolidation, provisioning and quality of service. To that end a new way to deploy and manage cluster databases was required that shifted the focus from database instances and servers to database services and server pools.

Oracle RAC databases whether multi-node or RAC One Node now may be deployed and managed in two different ways. The older database-centeric way is called an Administrator-managed deployment, also known as admin-managed, and is based upon the RAC deployment types that existed prior to Oracle Database 11g Release 2. It requires that each database instance be statically configured to run on a specific node and that database services are configured to run on specific instances using the preferred and available designation.

The new cloud-based service-centric mode is called a Policy-managed deployment and is based upon server pools where database services run within a server pool as a singleton service on a single node in the server pool, or a uniform service running across all of the nodes in the server pool. Databases are deployed in one or more server pools and the number of instances is governed by the size of the server pool(s) in the deployment. This is also the recommended default deployment for the new 12c Multitenant RAC databases where server pools host collections of PDBs and their associated services. This model not only provides the flexible resource allocation functionality required by a DBaaS Cloud environment but also solves a number of long standing management issues with Oracle RAC databases. In order to decide on whether to create or convert to a policy-managed deployment, it is necessary to first understand the functionality offered by server pools within a cluster.

Introducing Server Pools

The Oracle Grid Infrastructure 11g Release 2 introduced server pools as a method to logically partition a cluster into groups of servers offering database or application services. The scalability and availability of those databases and applications are controlled by server pool properties defined at the time the server pool is created or updated later. Each server pool can be configured with a minimum and maximum size to govern scalability of the services hosted in the pool. Availability can be managed between pools by configuring the importance value. Finally, servers are not assigned to server pools by name but simply by number or cardinality. Therefore any server must be configured to be able to run any database. If this cannot be configured due to, for example, heterogeneous servers or storage connectivity, then servers can be restricted by using server category definitions to determine sever pool membership eligibility.

To understand how server pools interact with policy-managed databases and services, we'll first examine several use cases that should be familiar to any experienced Oracle RAC DBA. Each of these illustrate a deployment issue or management difficulty which converting to server pools and policy-managed deployment solves. It should be noted that the databases are required to be at version 11g release 2 or above, in order to be created as or converted to policy-managed. Older versions can co-exist in the same cluster but will not share the same servers with policy-managed databases. Instead their servers will be restricted to a special server pool called Generic and deployed as admin-managed. Should a node failure occur in either the Generic pool or a policy-managed pool, database instances or services from the failed node will not failover nor servers move between these two types of pools.

Example Oracle RAC Deployment

For the following use cases and the rest of this paper, we will be using a example deployment where there is a four node cluster hosting an Oracle 12c Multitenant database with associated PDBs each with a single database service. *Figure 1* illustrates this deployment where the four nodes, dancer, dasher, comet and vixen, are currently running in two server pools, *backoffice* and *frontoffice*.

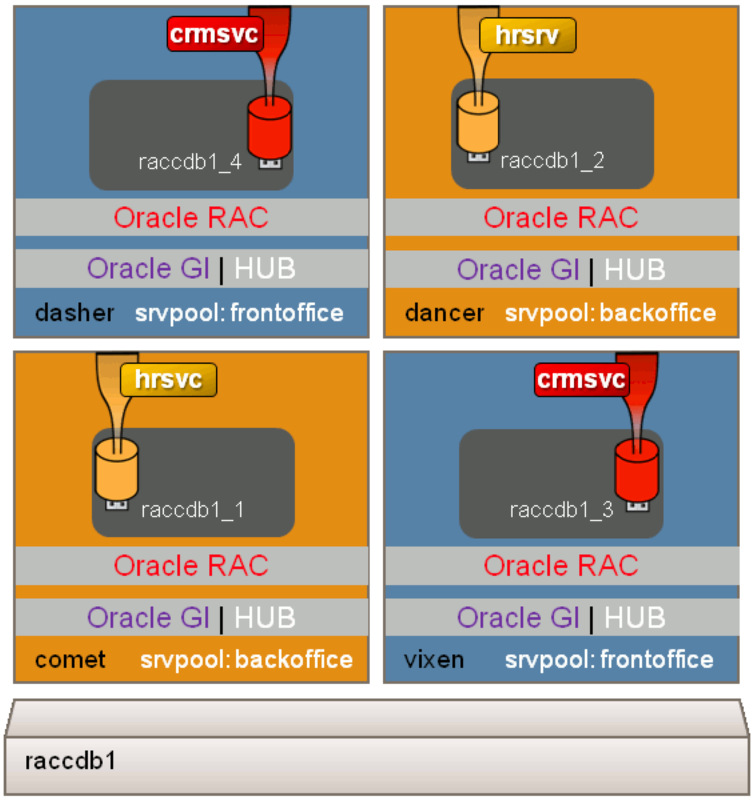


Figure 1: Example Cluster with Multitenant RAC database

Servers may only be in one server pool at a time. This cluster is hosting a single multitenant database, raccdb1, and has two PDBs each hosting a single service, *crmsvc* or *hrsvc*. While this is a 12c multitenant example, the following use cases are the same for an 11g Release 2 or 12c standard RAC or RAC One Node database. Additionally while this example has the database spanning both server pools, it will work the same if there were two databases, each wholly contained in its server pool. To keep the examples manageable, the use cases will consider a single RAC database with one service in each pool.

Use Case 1: Ensuring Database Service Start Order

Since database services were introduced, there has existed the problem of managing a start order that would eliminate the need to relocate services once the entire cluster is running. This was caused by all services assigned to the database(s) on the first node to join the cluster starting whether the instance was designated as preferred or available. This "clumping" and the resulting relocation meant extended start-up and settling time for a cluster along with manual intervention. Server pools hosting policy-managed RAC or RAC One Node databases can minimize or eliminate this issue.

Server pools have a collection of user set properties that govern the placement and failover process that Oracle Clusterware enforces. The Minimum (MIN) property specifies the minimum number of servers and should be set at a level that will meet the workload availability requirements. In *Figure 2*, the value is set to 2 servers for both the frontoffice and backoffice pools. A MIN value of 0 is allowed meaning that it is acceptable to have no servers and their associated services running in that pool.

The Maximum (MAX) property specifies the maximum number to be allocated and always must be equal to greater than the MIN. In *Figure 2*. that value is 2 servers. In special cases such as a cluster with only one server pool, it is useful to have the MAX be the current size of the cluster. This can be done by setting MAX=-1.

The Importance (IMP) value is less obvious and can have a multi-dimensional impact. On the surface it simply is a value between 0 and 1000 that specifies to Oracle Clusterware the relative availability importance between the pools with the higher number meaning higher importance. However, its impact governs both startup and failover behavior which is more easily explained in this and the following use cases. For this example in *Figure 2*, the frontoffice pool has IMP=10 and the backoffice pool has IMP=5. This means frontoffice has higher importance. Please note evaluation of Importance is a simple > or < check, therefore the magnitude of the numbers or their difference is not a factor.

```
[GRID]> srvctl config serverpool
Server pool name: frontoffice
Importance: 10, Min: 2, Max 2
Server pool name: backoffice
Importance: 5, Min: 2, Max 2
Server pool name: Free
Importance: 0, Min: 0, Max -1
```

Figure 2: SRVCTL Serverpool Configuration Properties

When this cluster starts up, servers join one at a time. Clusterware, when having to populate server pools, finds the highest importance pool not meeting its minimum number of servers. In *Figure 2*, this means the frontoffice pool. Therefore, the first server is guaranteed to be placed there and all autostart service(s) assigned to that pool, crmsvc, will start pulling up their associated database instance.

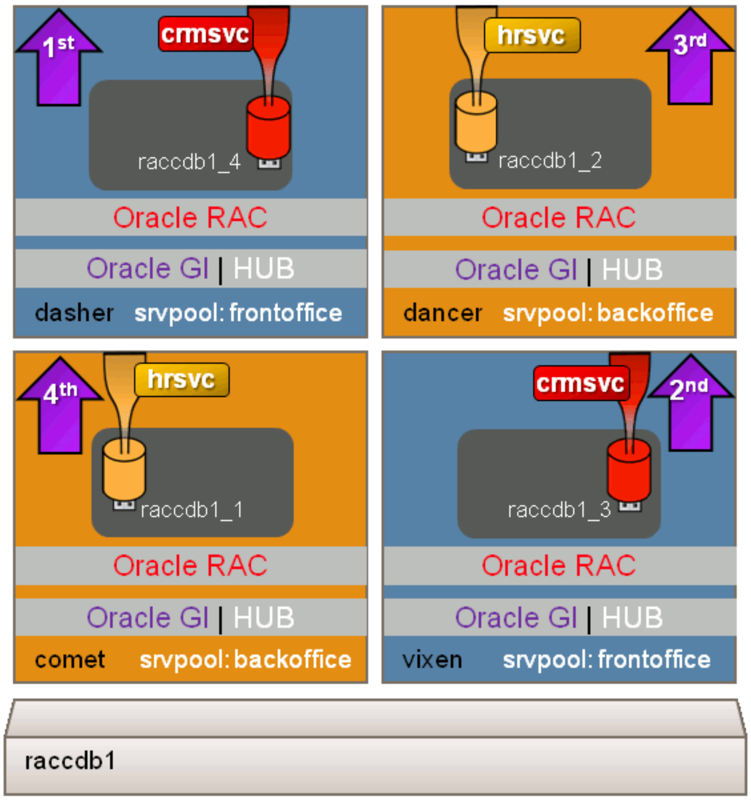


Figure 3: Service and Instance Start Order

Use Case 2: Improved HA Failover Strategy

A complimentary use case to start-up order is managing failover behavior. In RAC clusters prior to 11g Release 2 where databases were attached to servers and services specified to use preferred and available instances, any node failure would result in workloads being co-located or potentially over loading existing preferred instances resulting in a cascade failure. As a result workloads and their databases may be deployed in numerous two node clusters or significant headroom was left to absorb the failure. Policy-managed databases and server pools avoid this by never combining workloads outside their registered server pool but instead handle a failure by moving servers not workloads.

This new failover strategy also is implemented by using the MIN and IMP properties to specify the desired behavior. Returning to our example referenced in *Figure 3*, the failure of the dasher node would not result in all the work landing on vixen but instead one of the servers from the less important backoffice server pool would have its instances transactionally shut down draining off the hrsvc service workload and "moved" into the frontoffice pool starting up the crmsrv service and its instance. This is illustrated in *Figure 4* where dancer has moved from backoffice to frontoffice.

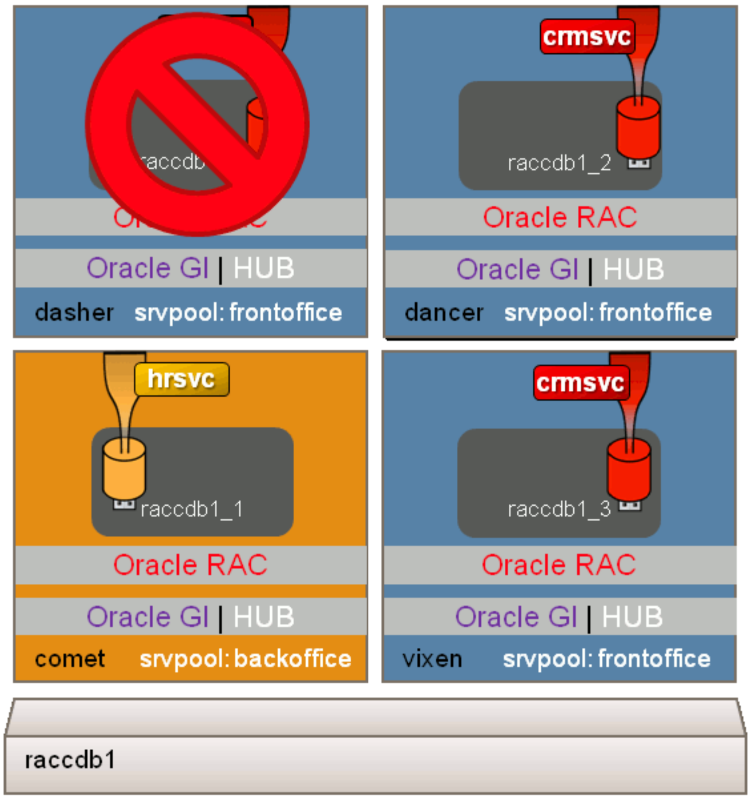

crmsvc
raccdb1_2
Oracle RAC
Oracle GI | HUB
dasher srvpool: frontoffice
dancer srvpool: frontoffice
hrsvc
raccdb1_1
crmsvc
raccdb1_3
Oracle RAC
Oracle GI | HUB
comet srvpool: backoffice
vixen srvpool: frontoffice
raccdb1

Figure 4: New Failover Policy-based Behavior

This failover behavior is determined by Oracle Clusterware re-evaluating the server pool properties as a result of the HA event. In this case it found frontoffice fell below its MIN of two servers and moved one from the least important pool to restore. On the other hand if frontoffice's MIN was only one server, no move would occur as the policy specified that one server was fine in this circumstance. Should dancer rejoin the cluster from, for example, a reboot, it would not return to the frontoffice pool but be placed in the backoffice pool restoring its MIN to two servers.

There are two advantages this new failover strategy provides. First, it allows multiple databases to share the same cluster when previously they needed to be in discrete ones to prevent their databases and workloads ever being co-located. Secondly, this improves high availability because the additional servers are available to support the more business critical workloads without needing to be idle or under-utilized.

Use Case 3: Managing the Last Service Standing

Taking Use Case 2 to its logical extreme brings us to the third use case which is to ensure one or more services are always available should there be a multi-node outage or split-brain cluster failure. If a business critical service was in an admin-managed database it would have to be available to run on all of the nodes of a cluster in order to ensure its survival. This has the disadvantage of compromising workload or database isolation. This is no longer the case for policy managed databases in server pools. Oracle Clusterware will ensure that even if there is only one server left, it is part of the server pool with the greatest importance and the registered services and instances are up and running.

This is illustrated in *Figure 5* where after dasher failed dancer did as well. Therfore the remaining of backoffice server, comet, had its instance shut down and moved to frontoffice where it joined vixen in running the business critical crmsvc service.

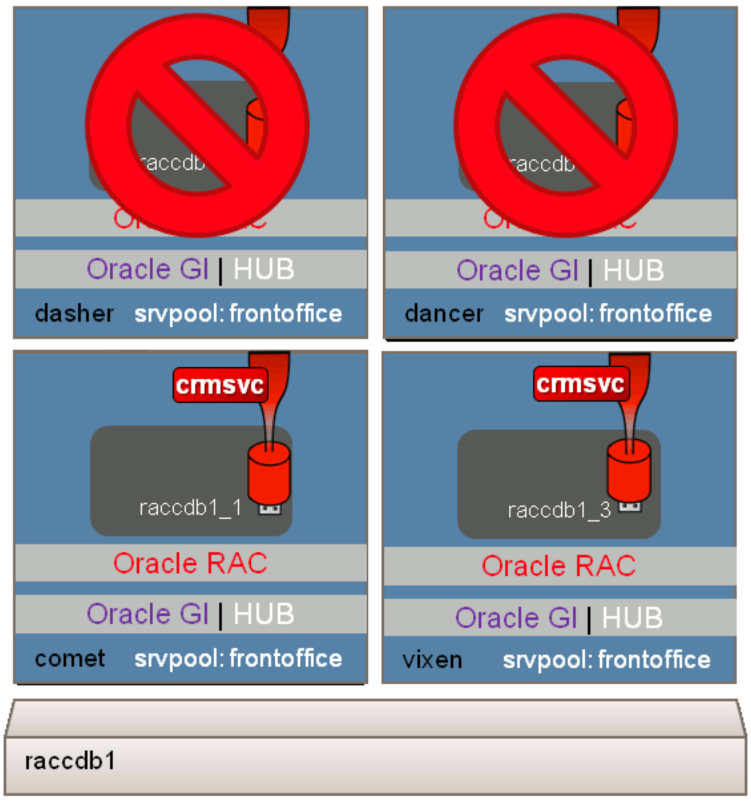


Figure 5: Ensuring the Last Service Standing by Policy

This behavior is solely dependent upon the MIN property being greater than zero for the sever pool with the greatest IMP property as illustrated in the SRVCTL output in *Figure 6.*

```
[RAC]> srvctl config serverpool
Server pool name: frontoffice
Importance: 10, Min: 2, Max 2
Server pool name: backoffice
Importance: 5, Min: 1, Max 1

[RAC]> srvctl status service -db racdb1
Service crmsrv is running on nodes comet,vixen
Service hrsvc is not running
```

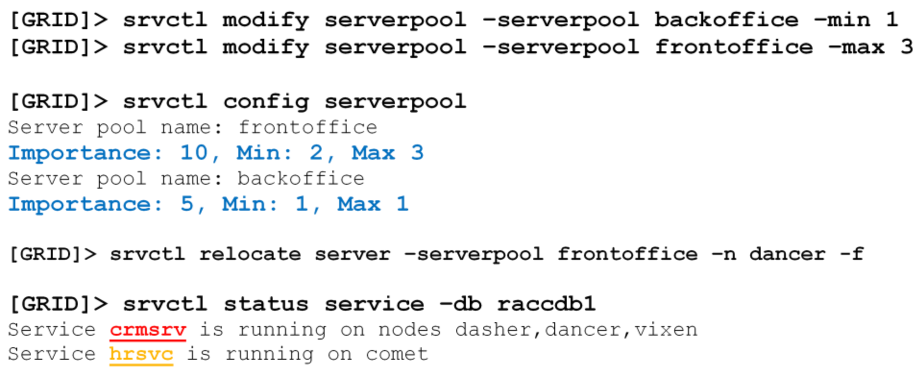
Figure 6: SRVCTL Configuration and Output After a Multi-Node Failure

As in Use Case 2, when dasher or dancer rejoin the cluster, they will be placed in the backoffice server pool preventing disruption to the crmsvc service and starting up the hrsvc service and database instances.

Use Case 4: Dyanmic Resource Provisioning

As illustrated by the previous use cases, servers hosting policy-managed databases and their services are able to move between pools dynamically and online. This can also be useful in non-HA cases in that workloads change based upon period of the day, week, month, quarter, or events which may cause an unexpected increase in transactions. By setting the MIN and MAX properties of a cluster's server pools as a group in a policy as is offered by Oracle Quality of Service Management begining in 11g Release 2 or Oracle Cluaterware in 12c, planned demand changes can be easily anticipated and unplanned ones can be handled through the use of QoS Management, the SRVCTL relocate server command, or manually altering the MIN and MAX properties with SRVCTL.

Returning to the example, for a planned change or an unplanned one that modified the server pool properties, *Figure 7* specifies the SRVCTL commands that allowed for frontoffice to increase to 3 servers and then to actually perform the change.



```
[GRID]> srvctl modify serverpool -serverpool backoffice -min 1
[GRID]> srvctl modify serverpool -serverpool frontoffice -max 3

[GRID]> srvctl config serverpool
Server pool name: frontoffice
Importance: 10, Min: 2, Max 3
Server pool name: backoffice
Importance: 5, Min: 1, Max 1

[GRID]> srvctl relocate server -serverpool frontoffice -n dancer -f

[GRID]> srvctl status service -db raccdb1
Service crmsrv is running on nodes dasher,dancer,vixen
Service hrsvc is running on comet
```

Figure 7: Dynamically Allocating Servers in Response to Demand

Note that the relocate command line utilizes a –f flag which signifies *force* and that if the specfied server is running resources such as database instances, that these should be shut down prior to the move and start of the frontoffice resources. As a result of these commands, whether issued manually as in *Figure 7* or via a QoS Management recommendation or Clusterware policy, the resulting cluster will now have the additional resources allocated to the frontoffice server pool to support the increased crmsvc service demand as shown in *Figure 8*.

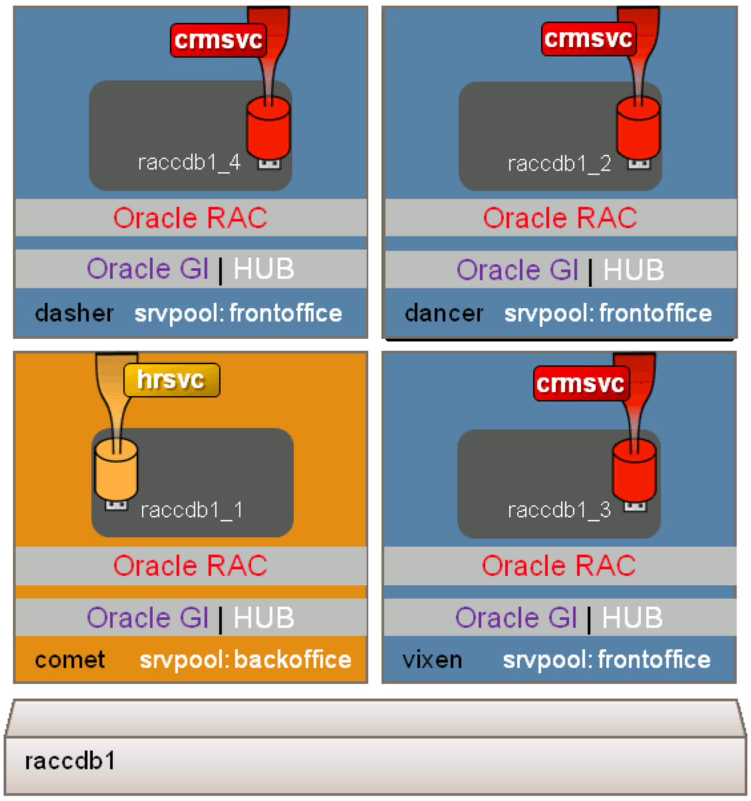


Figure 8: Server Pools Dynamically Sized to Meet Demand

Policy Managed RAC Databases and Policy Sets

In Use Case 4, the concept of policies was introduced. As server pools and policy-managed databases allow the deployments to be more flexible, dynamic and ordered, it makes sense to offer the capability to be able to make changes is a grouped and least disruptive way as a named policy. The collection of these policies would be a policy set. This capability was introduced as a feature of QoS Management in 11g Release 2 and extended into a core Clusterware feature with 12c. The ability to change server pool properties as a group permits Clusterware to optimize the transition to be minimally disruptive which would be very hard to do one command at a time.

To understand how to create and use a policy set and its included policies consider the following use case where frontoffice and backoffice server pools are augmented with a backup server pool. While the number of servers in the cluster has not changed, server pools can be created and services registered to them at any time. When a server is placed in that pool, the services will automatically start. In this case a *backup* service will be created with the intention to run an incremental backup on weeknights and a full backup on weekends.

The command in *Figure 9* creates the new backup server pool and given the high business criticality of its workload sets its importance to be highest, IMP=20, but it minimum servers to initially be zero, MIN=0, to avoid moving servers.

```
[GRID]> srvctl add serverpool -serverpool backup -min 0 -max 2 -importance 20

[GRID] srvctl status serverpool
Server pool name: frontoffice
Active Servers count: 3
Server pool name: backoffice
Active Servers count: 1
Server pool name: backup
Active Servers count: 0
```

Figure 9: Creating the backup Server Pool

Besides creating this pool, the Clusterware 12c new server pool *categories* feature will be used which supports managing clusters with heterogeneous servers very easily. Server categories allow the specification of server attributes to be included as server pool membership requirements. These attributes such as number of CPUs, CPU speed, Memory, etc. as well as user labels are evaluated at the time a server joins the cluster. For this example, comet and dancer have greater IO capacity, so they will be labeled as *IOplus* with the following command run locally on comet and dancer.

```
[GRID]> crsctl set server label IOplus
```

Once this is completed and the Clusterware stack restarted the status of each server can be queried for all of its attributes with the command:

```
[GRID]> crsctl status server comet dancer -f
```

The output would be as follows:

Comet	Dancer
NAME=comet	NAME=dancer
MEMORY_SIZE=3338	MEMORY_SIZE=3338
CPU_COUNT=1	CPU_COUNT=1
CPU_CLOCK_RATE=2132	CPU_CLOCK_RATE=2247
CPU_HYPERTHREADING=0	CPU_HYPERTHREADING=0
CPU_EQUIVALENCY=1000	CPU_EQUIVALENCY=1000
DEPLOYMENT=other	DEPLOYMENT=other
CONFIGURED_CSS_ROLE=hub	CONFIGURED_CSS_ROLE=hub
RESOURCE_USE_ENABLED=1	RESOURCE_USE_ENABLED=1
SERVER_LABEL=IOplus	SERVER_LABEL=IOplus
...	...

All that is left is to create an appropriate server category, for example, moreIO, in order for it to be incorporated into the policy set.

```
[GRID]> crsctl add category moreIO -attr "EXPRESSION='SERVER_LABEL
co IOplus'"

[GRID]> crsctl status category moreIO
     NAME=moreIO
     ACL=owner:grid:rwx,pgrp:oinstall:rwx,other::r--
     ACTIVE_CSS_ROLE=hub
     EXPRESSION=(SERVER_LABEL co IOplus)

[GRID]> crsctl modify serverpool ora.backup -attr
"SERVER_CATEGORY=moreIO" -all_policies
```

Now the policy set can be created as a simple text file in the following format specifying the three policies, DayTime, NightTime, Weekend.

```
SERVER_POOL_NAMES=frontoffice backoffice backup
POLICY
  NAME=DayTime
  SERVERPOOL
    NAME=frontoffice
    IMPORTANCE=10
    MAX_SIZE=2
    MIN_SIZE=2
    SERVER_CATEGORY=
  SERVERPOOL
```

```
    NAME=backoffice
    IMPORTANCE=5
    MAX_SIZE=2
    MIN_SIZE=2
    SERVER_CATEGORY=
  SERVERPOOL
    NAME=backup
    IMPORTANCE=20
    MAX_SIZE=1
    MIN_SIZE=0
    SERVER_CATEGORY=
POLICY
  NAME=NightTime
  SERVERPOOL
    NAME=frontoffice
    IMPORTANCE=10
    MAX_SIZE=1
    MIN_SIZE=1
    SERVER_CATEGORY=
  SERVERPOOL
    NAME=backoffice
    IMPORTANCE=5
    MAX_SIZE=2
    MIN_SIZE=2
    SERVER_CATEGORY=
  SERVERPOOL
    NAME=backup
    IMPORTANCE=20
    MAX_SIZE=1
    MIN_SIZE=1
    SERVER_CATEGORY=moreIO
POLICY
  NAME=Weekend
  SERVERPOOL
    NAME=frontoffice
    IMPORTANCE=5
    MAX_SIZE=1
    MIN_SIZE=1
    SERVER_CATEGORY=
  SERVERPOOL
    NAME=backoffice
    IMPORTANCE=10
    MAX_SIZE=1
    MIN_SIZE=1
    SERVER_CATEGORY=
  SERVERPOOL
    NAME=backup
    IMPORTANCE=20
    MAX_SIZE=2
    MIN_SIZE=2
    SERVER_CATEGORY=moreIO
```

This policy is submitted to Clusterware using the following example CRSCTL command:

```
[GRID]> crsctl modify policyset –file /u01/app/…/policyset.txt
```

Once this is completed, then policies can be activated either manually or via a job scheduler such as Cron using the following command:

```
[GRID]> crsctl modify policyset -attr
"LAST_ACTIVATED_POLICY=Weekend"
```

The results can be verified using the SRVCTL status command as follows with the resulting output based upon the example.

```
[GRID]> srvctl config serverpool
      Server pool name: Free
      Importance: 0, Min: 0, Max: -1
      Category:
      Candidate server names:
      Server pool name: Generic
      Importance: 0, Min: 0, Max: -1
      Category:
      Candidate server names:
      Server pool name: backoffice
      Importance: 10, Min: 1, Max: 1
      Category:
      Candidate server names:
      Server pool name: backup
      Importance: 20, Min: 2, Max: 2
      Category:
      Candidate server names:
      Server pool name: frontoffice
      Importance: 5, Min: 1, Max: 1
      Category:
      Candidate server names:
```

This server pool reallocation for the Weekend policy would result in a new cluster-wide service configuration performed entirely online. *Figure 10* illustrates the resulting service and server pool configuration.

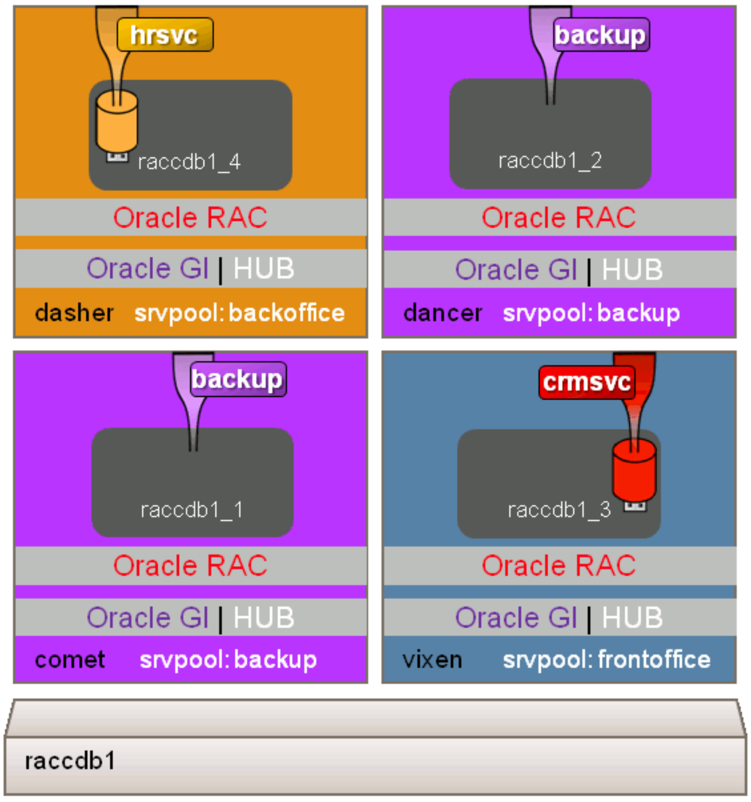


Figure 10: Weekend Policy Cluster Configuration

Note that by using server categories only comet or dancer will ever run the backup service thus assuring the desired resource to task mapping.

Policy-Managed Databases: The Multitenant RAC Database Default Deployment

As evidenced by these examples, policy-managed RAC database deployments afford rich functionality when it comes to the placement and HA management of database services. It turns out these features can be leveraged quite effectively when depolying new Oracle 12c Multitenant RAC databases and multiple PDBs.

Since policy-managed databases can span multiple server pools and as has been demonstrated, these pools can be prioritized in their startup and failover characterisitics. One such use case is illustrated in *Figure 11* where a service provider hosts a suite of applications under different service level and availability terms.

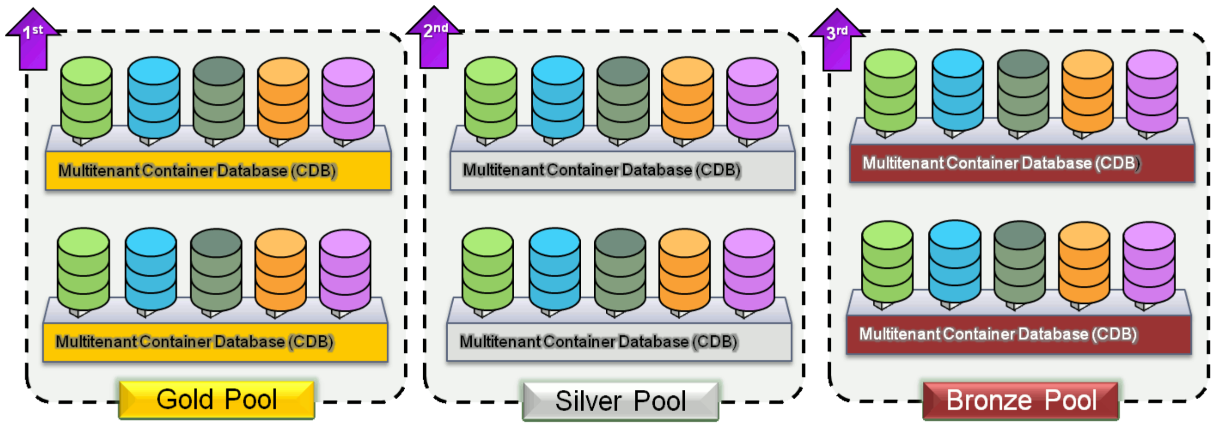

1st
2nd
3rd
Multitenant Container Database (CDB)
Multitenant Container Database (CDB)
Multitenant Container Database (CDB)
Multitenant Container Database (CDB)
Multitenant Container Database (CDB)
Multitenant Container Database (CDB)
Gold Pool
Silver Pool
Bronze Pool

Figure 11: Policy-Managed Multitenant RAC Database Service Level Example

By setting up appropriate MIN, MAX and IMP properties on the Gold, Silver and Bronze server pools both availability and performance levels can be effectively managed even when there are node failures. Not only can the Gold level customers be assured the highest availability, but the dynamic allocation functionality within the server pool and RAC database frameworks permits just-in-time server allocations to meet demand.

An additional desirable characteristic when consolidating multiple databases into a multitenant RAC database as PDBs, is the ability to group those databases that can benefit from sharing the same hardware resources. This type of affinity could be based upon application type, need for DBLinks, tier use, hardware differences, etc, as illustrated in *Figure 12*.

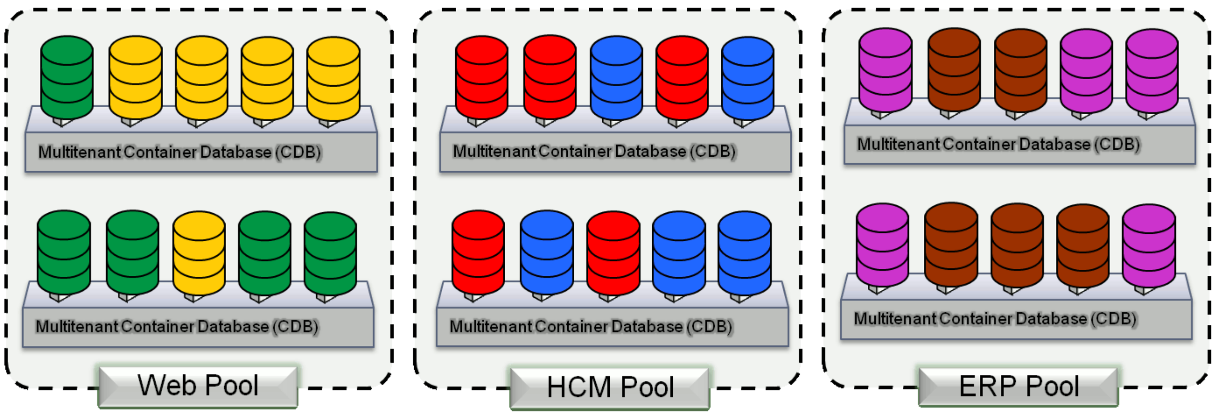

Multitenant Container Database (CDB)
Multitenant Container Database (CDB)
Multitenant Container Database (CDB)
Multitenant Container Database (CDB)
Multitenant Container Database (CDB)
Multitenant Container Database (CDB)
Web Pool
HCM Pool
ERP Pool

Figure 12: Policy-Managed Multitenant RAC Database PDB Affinity Example

In this example PDBs are deployed into server pools to ensure that they share the same hardware resources and remain functionally isolated. Different policies can then be used to adjust their performance and availability priority in order to track business objectives and demand.

Considerations and Tips for Policy Managed RAC Databases

With this new type of database deployment and management paradigm come a number of considerations and tips when planning both a new as well as a converted deployment. Of prime importance is the appreciation that this functionality and flexibility must be planned for – not simply adopted. As previously described, node failures may or may not cause services to failover and in some cases cause instances and their services to be shutdown. While this is entirely deterministic, each scenario should be carefully examined. A new *eval* option on CRSCTL and SRVCTL commands provides details as to what will occur should a server pool property be modified or a policy activated. This is demonstrated below in evaluating activating the DayTime policy from the current Weekend policy.

```
[GRID]>  date;  crsctl modify policyset -attr
"LAST_ACTIVATED_POLICY=DayTime"; date
Mon Sep 16 19:26:42 PDT 2013
CRS-2673: Attempting to stop 'ora.raccdb1.backup.svc' on 'dancer'
CRS-2673: Attempting to stop 'ora.raccdb1.backup.svc' on 'comet'
CRS-2677: Stop of 'ora.raccdb1.backup.svc' on 'dancer' succeeded
CRS-2677: Stop of 'ora.raccdb1.backup.svc' on 'comet' succeeded
CRS-2672: Attempting to start 'ora.raccdb1.crmsvc.svc' on 'dancer'
CRS-2672: Attempting to start 'ora.raccdb1.crmsvc.svc' on 'comet'
CRS-2676: Start of 'ora.raccdb1.crmsvc.svc' on 'dancer' succeeded
CRS-2676: Start of 'ora.raccdb1.crmsvc.svc' on 'comet' succeeded
Mon Sep 16 19:26:43 PDT 2013
```

Since servers no longer host only specific database instances, any log files that are instance-specific may be written on different servers making diagnostics a bit more challenging. Consolidation of these is a priority for a future release.

With regards to database instances, policy-managed RAC or RAC One Node databases have a modified instance naming convention that may have an impact on custom scripts. Instead of identifying an instance with a simple numeric suffix, orcl1, orcl2, etc., an additional underscore is inserted as in orcl_1, orcl_2, etc. The RAC database code uses this indicator to self-provision new UNDO tablespaces when a new instance is created as a server pool is expanded. This can be reverted to the original naming on a per instance basis using SRVCTL if the use of server pools never results in new instances being dynamically created.

Care must be taken when using Data Guard Data Broker, also when creating new instances as a Standby Redo Log thread will not be created with it. These should be pre-created for all instance to node mappings so one is available regardless of where the instance runs.

Existing backup configurations that use RMAN should also be reviewed. While it has been best practice to use services for RMAN channels to load balance for some time, if these are mapped to instance names then they need to be updated to the new naming scheme as in this example:

```
CONFIGURE CHANNEL DEVICE TYPE sbt CONNECT '@raccdb1_1'
CONFIGURE CHANNEL DEVICE TYPE sbt CONNECT '@raccdb1_2'
…
```

GoldenGate usage is also impacted by policy-managed RAC databases. Upon conversion, after database instance additions, the Extract Process requires manual intervention. The additional instance will not be added to the Extract Group until it is manually dropped and updated. As servers do move in the case of HA events the GoldenGate Extract and Replicate processes must do likewise. This can be accomplished by deploying the Clusterware Bundled Agents available from OTN.

Conclusion and Further Information

The desire for consolidation and a provision-on-demand DBaaS to meet the growing demand without the costs of growing datacenters brings with it new infrastructure functionality and a management paradigm that is both flexible in its resource allocation and deterministic in its operational and failure behavior. There is a learning curve with this type of change, but the ability to implement these types of deployments on bare metal resources using enterprise-leading Oracle RAC database technology without converting to new virtual architectures will pay performance and manageability dividends in the future while leveraging in-house Oracle expertise.

Further information on Policy-Managed RAC databases, Clusterware policies and server pools as well as QoS Management is available from the following links:

http://www.oracle.com/technology/documentation/database.html
To convert to a policy-managed database, see:
http://docs.oracle.com/cd/E11882_01/rac.112/e16795/admin.htm#RACAD803
For using EMCA after conversions, see:
http://docs.oracle.com/cd/E11882_01/install.112/e24660/srvpool.htm#BHBJIIDC
http://www.oracle.com/goto/rac
http://www.oracle.com/goto/clusterware

Contact address:

Mark V. Scardina
Oracle Corporation
500 Oracle Parkway
94065, Redwood Shores

Phone:	+1 650-506-7000
Fax:	+1 650-506-7203
Email	mark.scardina@oracle.com
Internet:	www.oracle.com