Oracle RAC Basic Administration

By Ahmed Baraka

Objectives

In this lecture, you will learn how to do the following:

- Start up and shutdown Oracle RAC databases and instances using srvctl and SQL*Plus utilities
- Switch between Automatic and Manual management policies
- Perform the common connection methods to RAC
- Manage the initialization parameters in RAC
- Manage the Undo in RAC
- Terminate a session in RAC
- Access RAC-wide performance views

Starting up and Stopping Oracle RAC

- RAC database is available when at least one instance is up and running
- Shutting down a RAC database means shutting down all its instances
- RAC instances can be started and stopped by using:
  - Enterprise Manager Cloud Control
  - The Server Control (srvctl) utility
  - SQL*Plus

Starting and Stopping RAC Instances using srvctl syntax

- Start/Stop database/instance syntax:

```
srvctl start|stop instance -db db_unique_name
 {–node node_name | -instance instance_name_list}
[-startoption [open|mount|nomount] |
 -stopoption [normal|transactional|immediate|abort] ]
```

```
srvctl start|stop database -db db_unique_name  [-eval]
[-startoption [open|mount|nomount] |
 -stopoption [normal|transactional|immediate|abort] ]
```

Starting and Stopping RAC Instances using srvctl: Examples

- Start instances `rac1` and `rac1` in the database `rac`:

```
srvctl start instance -db rac -instance rac1,rac2
srvctl start instance -d rac -i rac1,rac2
```

- Stop instances `rac1` and `rac1` in the database `rac`:

```
srvctl stop instance -db rac -instance rac1,rac2
srvctl stop instance -d rac -i rac1,rac2
```

Starting and Stopping RAC Database using srvctl: Examples

- Start the entire database:

```
srvctl start database -db rac -startoption open
srvctl start database -db rac -startoption mount
```

- Stop the entire database:

```
srvctl stop database -db rac -o immediate
srvctl stop database -db rac -o transactional
```

Starting and Stopping RAC Instances using SQL*Plus

- The STARTUP and SHUTDOWN commands take effect on the current instance:

```
# echo $ORACLE_SID
rac

# sqlplus / as sysdba
SQL> startup
SQL> shutdown immediate
```

Switching between Automatic and Manual Management Policies

- Oracle Clusterware controls database restarts in Oracle RAC environments via two management policies:
  - AUTOMATIC (default): the database is automatically restored to its previous running condition
  - MANUAL: the database is never automatically restarted
- To change the current management policy:

```
srvctl modify database -db db_unique_name
        -policy [AUTOMATIC | MANUAL | NORESTART]
```

- To display current policy:

```
srvctl config database -db db_unique_name –all
```

Methods to Connect to Oracle RAC: Examples

- Operating system authenticated connection:

```
sqlplus / as sysdba
```

- Password file authenticated connection:

```
sqlplus sys/oracle as sysdba
sqlplus sys/oracle@rac as sysdba
```

- Oracle database authenticated connection

```
sqlplus system/oracle@rac
```

Methods to Connect to Oracle RAC: Concepts

- Operating system authenticated connection:
  - No password is required
  - It requires the logged on operating system user belongs to the dba group
  - Connects to the local instance defined in ORACLE_SID
  - Supersedes the password file authentication method
- Password file authenticated connection:
  - Uses the credentials saved in the password file
- Oracle database authenticated connection:
  - The provided password must be correct

Managing Initialization Parameters in RAC

- All instances use the same SPFILE at startup
- SPFILE must exist in the shared storage
- A parameter can be set in the database level (apply to all instances) or instance level
- Change a parameter setting in all the instances:

```
ALTER SYSTEM SET param=value SCOPE=[MEMORY|SPFILE|BOTH] SID='*';
```

- Change a parameter setting in a specific instance:

```
ALTER SYSTEM SET param=value SCOPE=[MEMORY|SPFILE|BOTH] SID='sid';
```

Managing Initialization Parameters in RAC (cont)

- To remove a parameter setting from SPFILE:

```
ALTER SYSTEM RESET param SCOPE=SPFILE SID='[*|sid]';
```

- If an instance-level parameter is in place, and you want the instance to use the *.param setting:

```
ALTER SYSTEM RESET param SCOPE=MEMORY SID='sid';
```

RAC Specific Parameters

Daemon/Service	Description
CLUSTER_DATABASE	It is always set to TRUE in a RAC database.
CLUSTER_DATABASE _INSTANCES	The total number of instances in the RAC database
DB_NAME	Specifies a database identifier of up to 8 characters. The setting of this parameter must be identical for all instances
INSTANCE_NAME	The instance's SID. The value for this parameter is automatically set to the database unique name followed by an incrementing number.

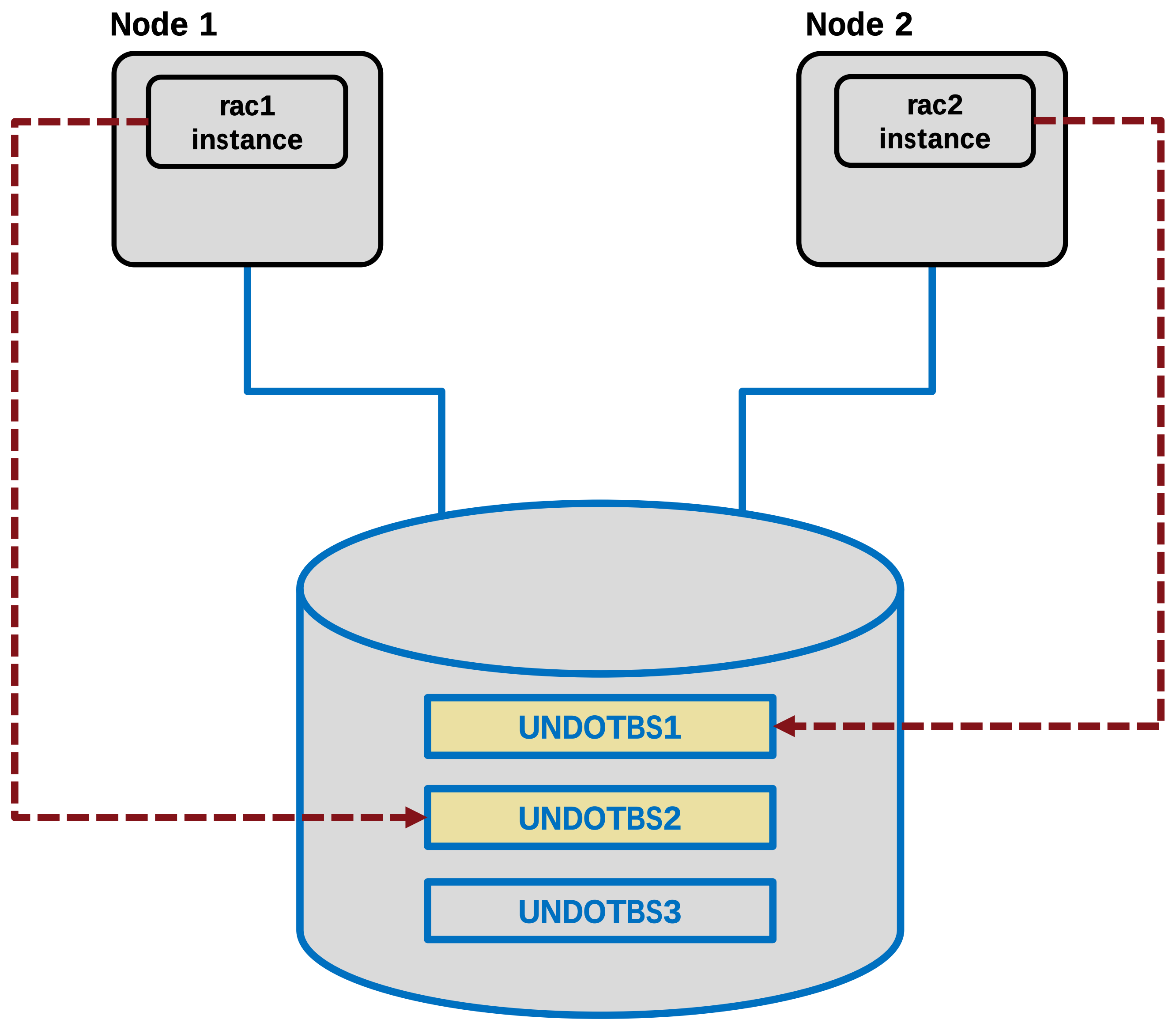
Parameters that Require Identical Settings in All the Instances

- COMPATIBLE
- CONTROL_FILES
- DB_DOMAIN
- DB_NAME
- DB_RECOVERY_FILE_DEST_SIZE
- INSTANCE_TYPE
- REMOTE_LOGIN_PASSWORDFILE
- CLUSTER_DATABASE
- DB_BLOCK_SIZE
- DB_FILES
- DB_RECOVERY_FILE_DEST
- DB_UNIQUE_NAME
- PARALLEL_EXECUTION_MESSAGE_SIZE
- UNDO_MANAGEMENT

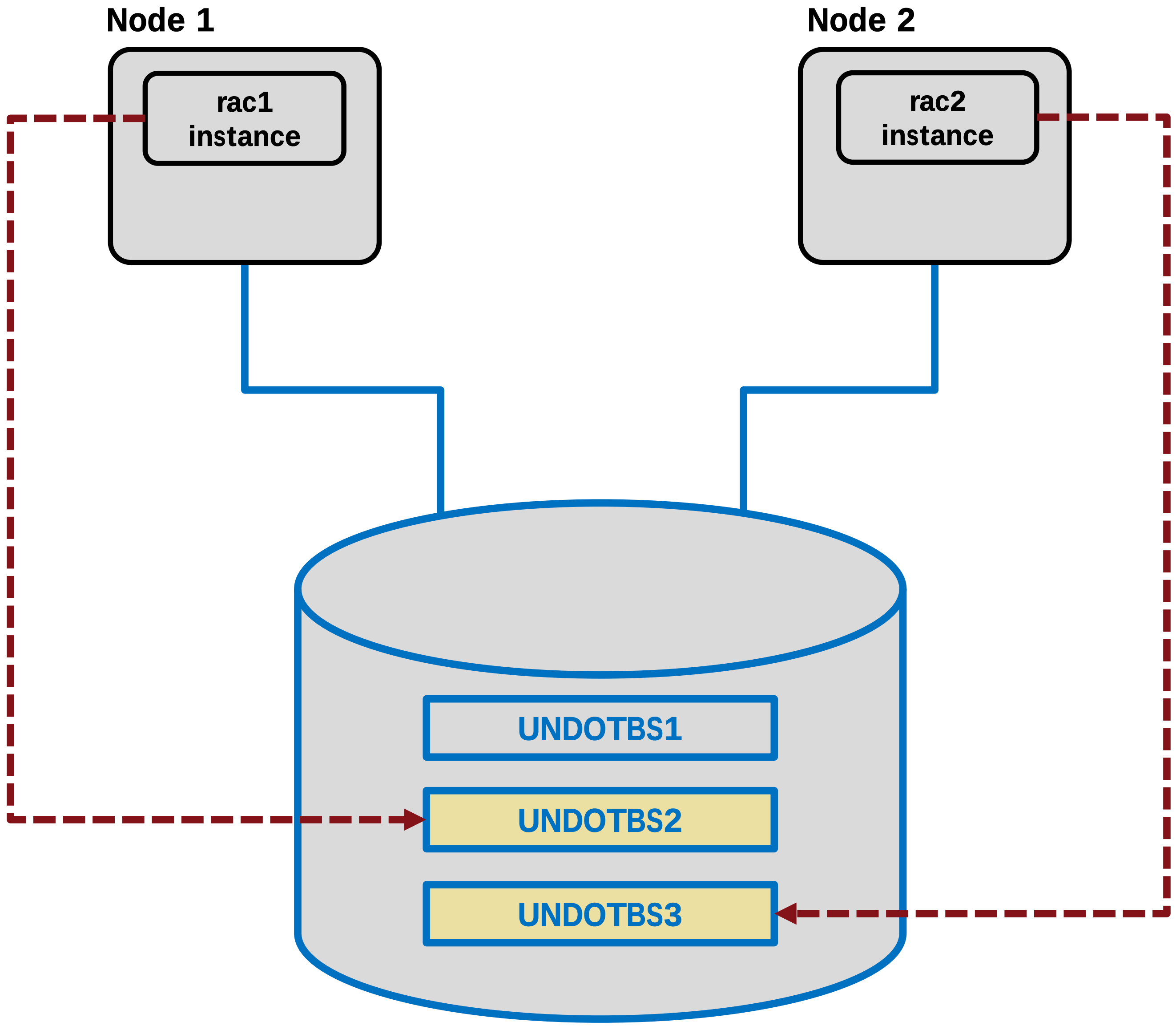
Parameters that Require Unique Settings

```
INSTANCE_NAME
INSTANCE_NUMBER
UNDO_TABLESPACE
CLUSTER_INTERCONNECTS
ROLLBACK_SEGMENTS
```

Automatic Undo Management in RAC



Automatic Undo Management in RAC



Managing UNDO Tablespaces in Oracle RAC

- Each instance is assigned an undo tablespace
- Undo is defined using **UNDO_TABLESPACE**
- In normal operation, only one instance writes to its undo
- All instances can read for all active undo for read consistent-read
- During transaction recovery, an instance can update any undo
- To switch undo assignment:

```
ALTER SYSTEM SET UNDO_TABLESPACE=undotbs1 SID='rac1';
```

```
rac1.UNDO_TABLESPACE=undotbs1
rac2.UNDO_TABLESPACE=undotbs2
```

Terminating Sessions on a Specific Instance

- Retrieve the SID, SERIAL#, and INST_ID:

```
SELECT SID, SERIAL#, INST_ID
FROM GV$SESSION WHERE USERNAME='XYZ';
```

- Kill the session using the following format:

```
ALTER SYSTEM KILL SESSION 'sid,serial#,@inst_id';

ALTER SYSTEM KILL SESSION '125,698,@2';
```

- If the session is marked for termination but not terminated yet:

```
ORA-00031: session marked for kill
```

About GV$ Performance Views

- Performance views are defined as V_$
- Should be accessed by their public synonyms V$
- Global Dymanic Performance views (GV$) retrieve information about all started instances accessing a RAC database
- Usually for every V$ view, there is a corresponding GV$ view
- INST_ID identifies the instance

Summary

In this lecture, you should have learnt how to do the following:

- Start up and shutdown Oracle RAC databases and instances using srvctl and SQL*Plus utilities
- Switch between Automatic and Manual management policies
- Perform the common connection methods to RAC
- Manage the initialization parameters in RAC
- Manage the Undo in RAC
- Terminate a session in RAC
- Access RAC-wide performance views