Oracle Database 10*g*: Real Application Clusters

Volume 2 • Student Guide

D17276GC10

Edition 1.0

January 2005

D40140

Authors

James Womack
Jean-Francois Verrier

Technical Contributors and Reviewers

Troy Anthony
Harald van Breederode
Bill Bridge
Michael Cebulla
Carol Colrain
Jonathan Creighton
Joel Goodman
Yunrui Li
Yi Lu
Vijay Lunawat
Paul Manning
John McHugh
Erik Peterson
Javier Seen
Nitin Vengurlekar

Publisher

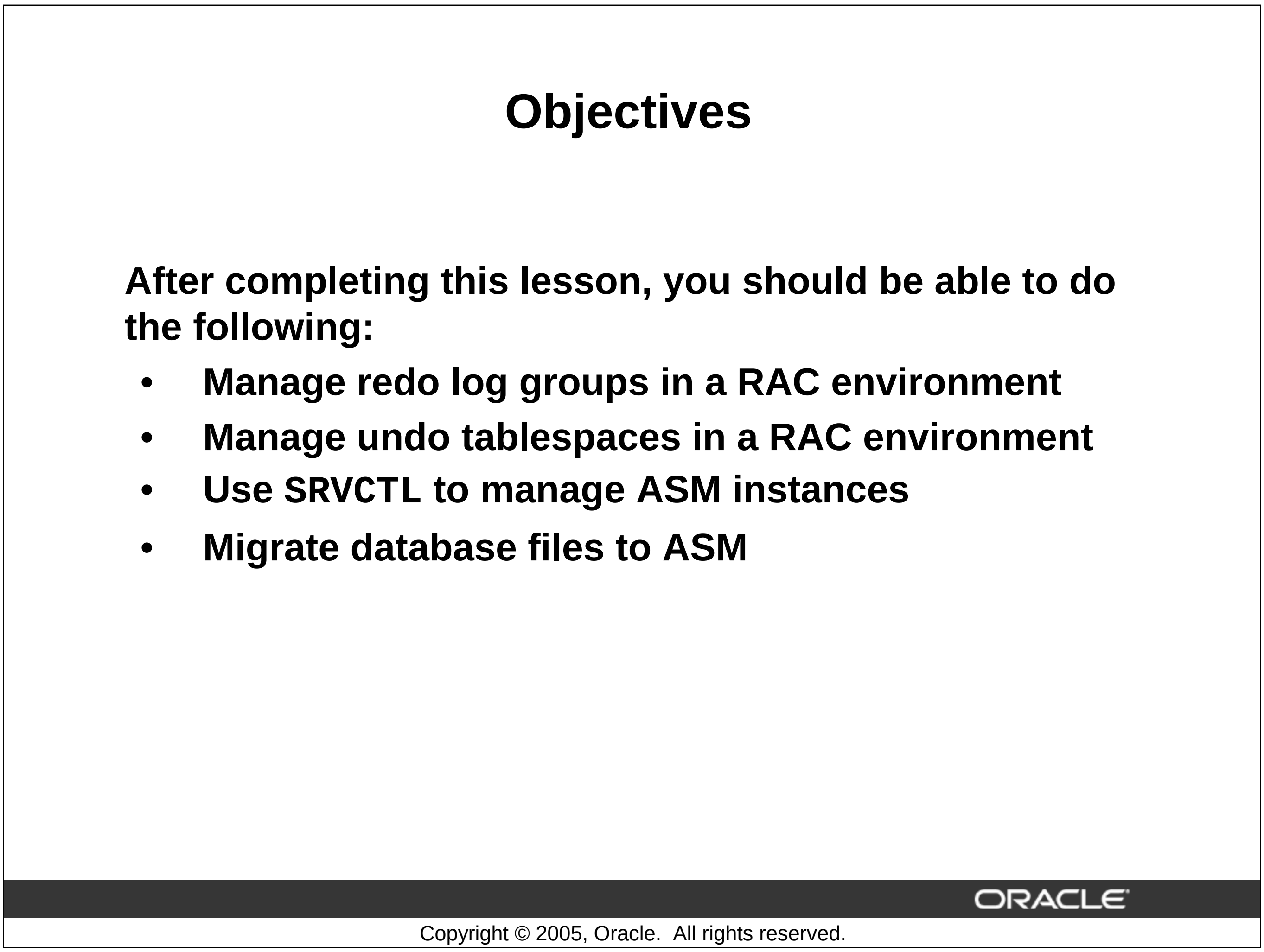
Joseph Fernandez

Contents

3 RAC Installation and Configuration (Part II)

6

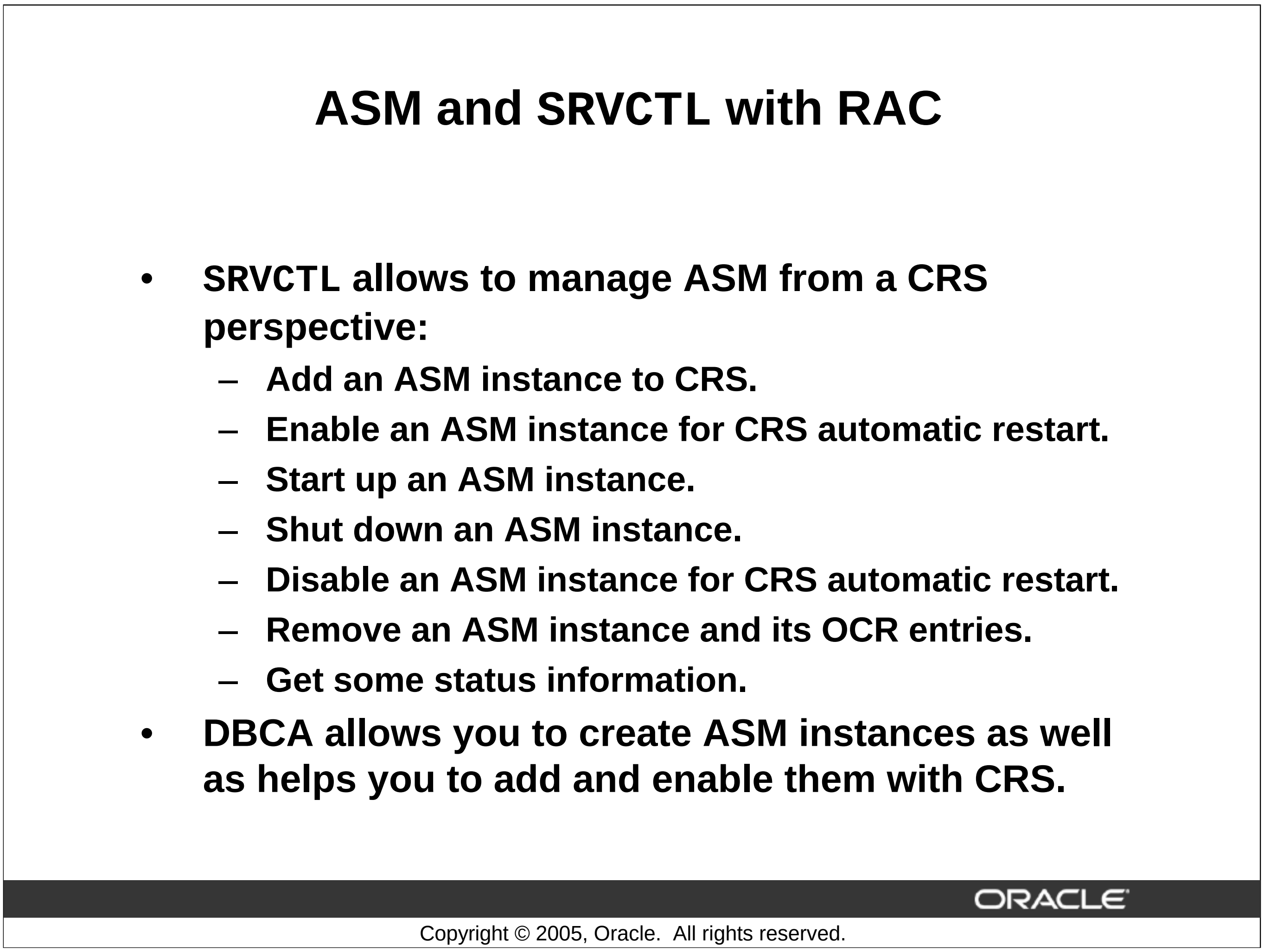
Administering Storage in RAC (Part II)



Objectives

After completing this lesson, you should be able to do the following:

- Manage redo log groups in a RAC environment
- Manage undo tablespaces in a RAC environment
- Use `SRVCTL` to manage ASM instances
- Migrate database files to ASM



ASM and SRVCTL with RAC

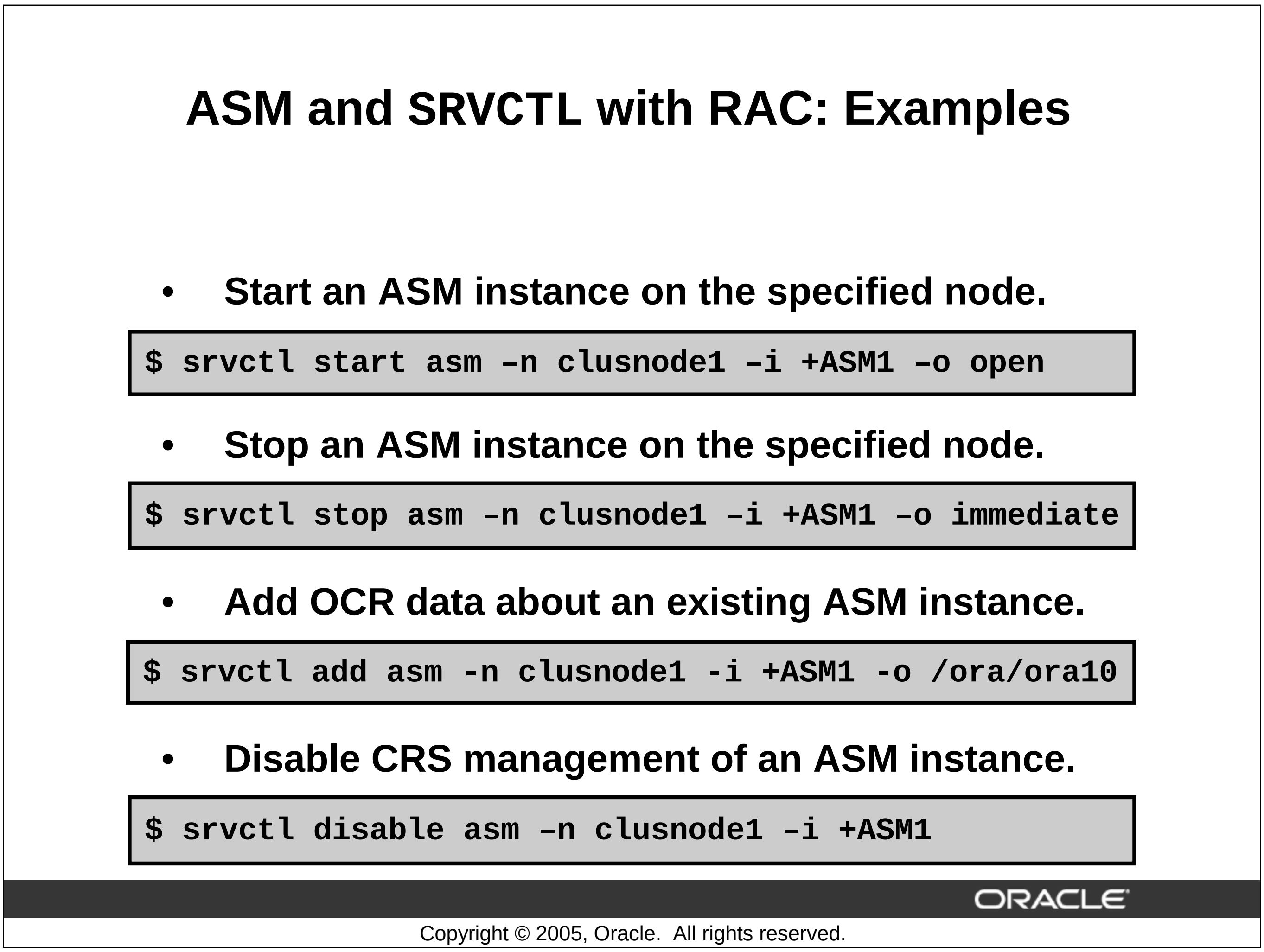
- SRVCTL allows to manage ASM from a CRS perspective:
  - Add an ASM instance to CRS.
  - Enable an ASM instance for CRS automatic restart.
  - Start up an ASM instance.
  - Shut down an ASM instance.
  - Disable an ASM instance for CRS automatic restart.
  - Remove an ASM instance and its OCR entries.
  - Get some status information.
- DBCA allows you to create ASM instances as well as helps you to add and enable them with CRS.

ASM and SRVCTL with RAC

You can use SRVCTL to perform the following ASM administration tasks:

- The ADD option adds Oracle Cluster Registry (OCR) information about an ASM instance to run under CRS. This option also enables the recourse.
- The ENABLE option enables an ASM instance to run under CRS for automatic startup, or restart.
- The DISABLE option disables an ASM instance to prevent CRS inappropriate automatic restarts.
- The START option starts a CRS-enabled ASM instance. SRVCTL uses the SYSDBA connection to perform the operation.
- The STOP option stops an ASM instance by using either shutdown normal, transactional, immediate, or abort option.
- The CONFIG option displays the configuration information stored in the OCR for a particular ASM instance.
- The STATUS option obtains the current status of an ASM instance.
- The REMOVE option removes the configuration of an ASM instance, as well as its corresponding instance from a node. Before you can remove an ASM instance, you must first stop and disable it.

Note: Adding and enabling an ASM instance is automatically performed by the DBCA when creating the ASM instance.



ASM and SRVCTL with RAC: Examples

- Start an ASM instance on the specified node.

```
$ srvctl start asm –n clusnode1 –i +ASM1 –o open
```

- Stop an ASM instance on the specified node.

```
$ srvctl stop asm –n clusnode1 –i +ASM1 –o immediate
```

- Add OCR data about an existing ASM instance.

```
$ srvctl add asm -n clusnode1 -i +ASM1 -o /ora/ora10
```

- Disable CRS management of an ASM instance.

```
$ srvctl disable asm –n clusnode1 –i +ASM1
```

ASM and SRVCTL with RAC (continued)

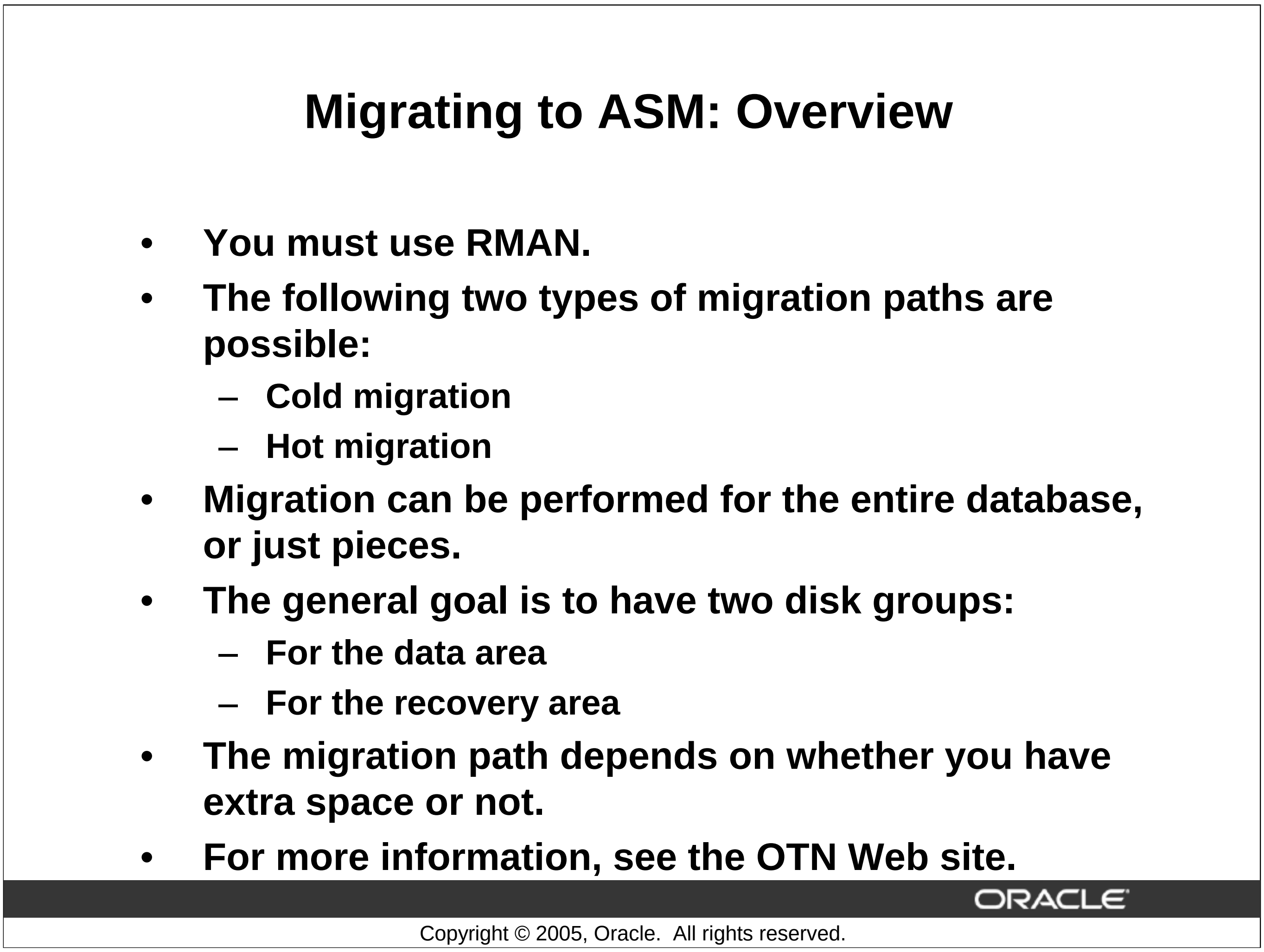
The first example starts up the +ASM1 ASM instance on the CLUSNODE1 node.

The second example shuts down immediate +ASM1 on CLUSNODE1.

The third example adds to the OCR the CRS information for +ASM1 on CLUSNODE1. You need to specify the ORACLE_HOME of the instance.

The fourth example prevents CRS to automatically restart +ASM1.

Note: For more information, refer to the *Oracle Real Application Clusters Administrator's Guide*.



Migrating to ASM: Overview

- You must use RMAN.
- The following two types of migration paths are possible:
  - Cold migration
  - Hot migration
- Migration can be performed for the entire database, or just pieces.
- The general goal is to have two disk groups:
  - For the data area
  - For the recovery area
- The migration path depends on whether you have extra space or not.
- For more information, see the OTN Web site.

Migrating to ASM: Overview

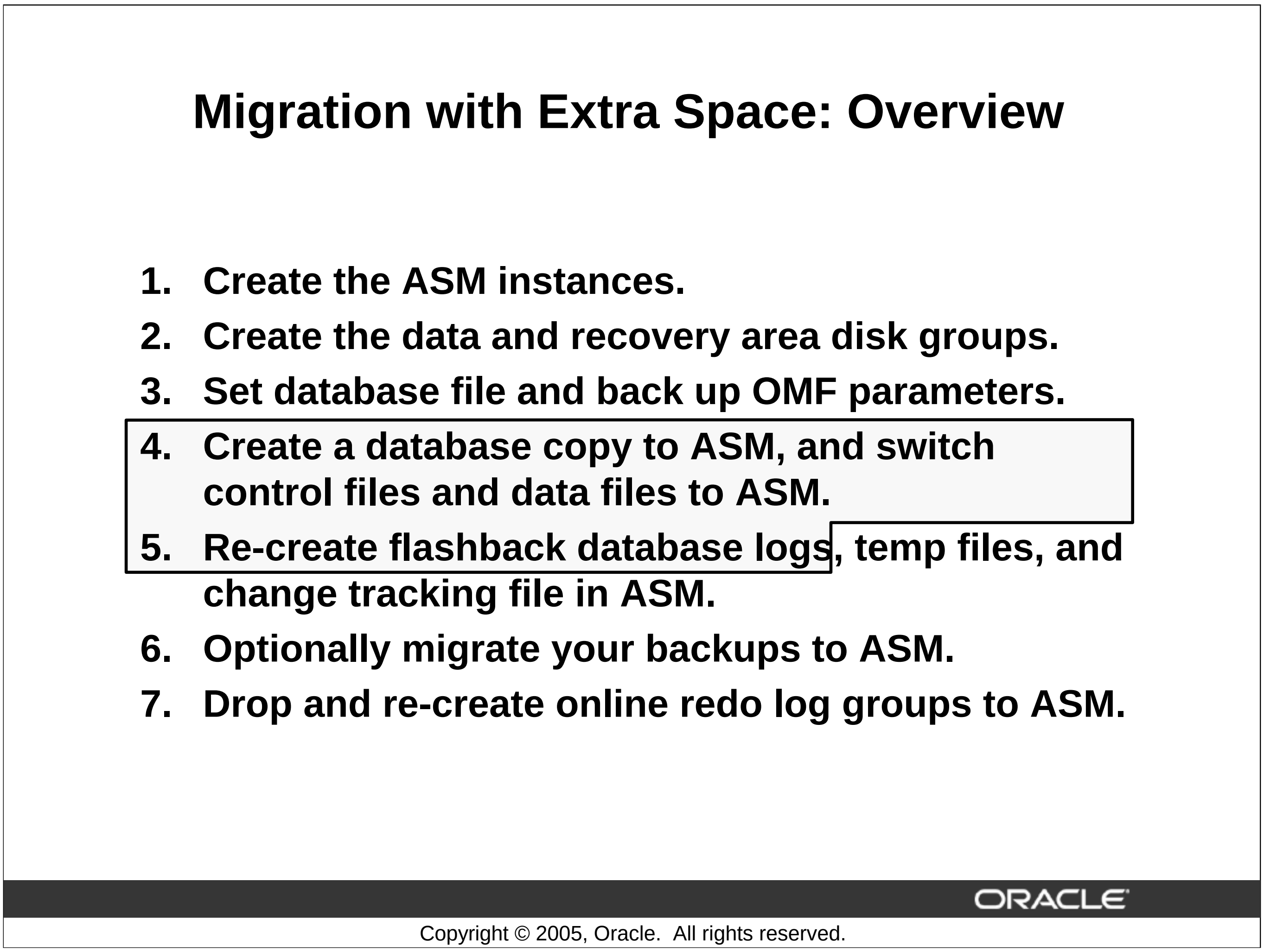
Whenever you want to migrate your database to ASM, the only possibility is to use Recovery Manager (RMAN). This is because each file stored in a disk group is physically spread across all disks in the disk group, and RMAN commands enable non-ASM files to be relocated to an ASM disk group.

Although it is not a requirement, most of the time, the goal of a migration to ASM is to distribute your database on two disk groups: one that contains all the database files and the other that contains the flash recovery area files.

The possible migration paths are organized around the concepts of hot and cold migration paths. With a cold migration path, you do not care to shut down your database for a long period of time. Whereas, with a hot migration path, you want to minimize the down time. However, it is not possible to do a complete online migration to ASM.

Also, the procedures you need to follow depend on your disk space capacity. The easiest paths are when you have enough disk space capacity to store the database both on the file system and ASM. Nevertheless, it is possible to migrate your database to ASM even in the case where you cannot add disks to your system.

In this lesson, you explore one possible migration path. However, for more information about other alternatives, refer to the OTN Web site at otn.oracle.com, and the *Backup and Recovery Advanced User's Guide*.



Migration with Extra Space: Overview

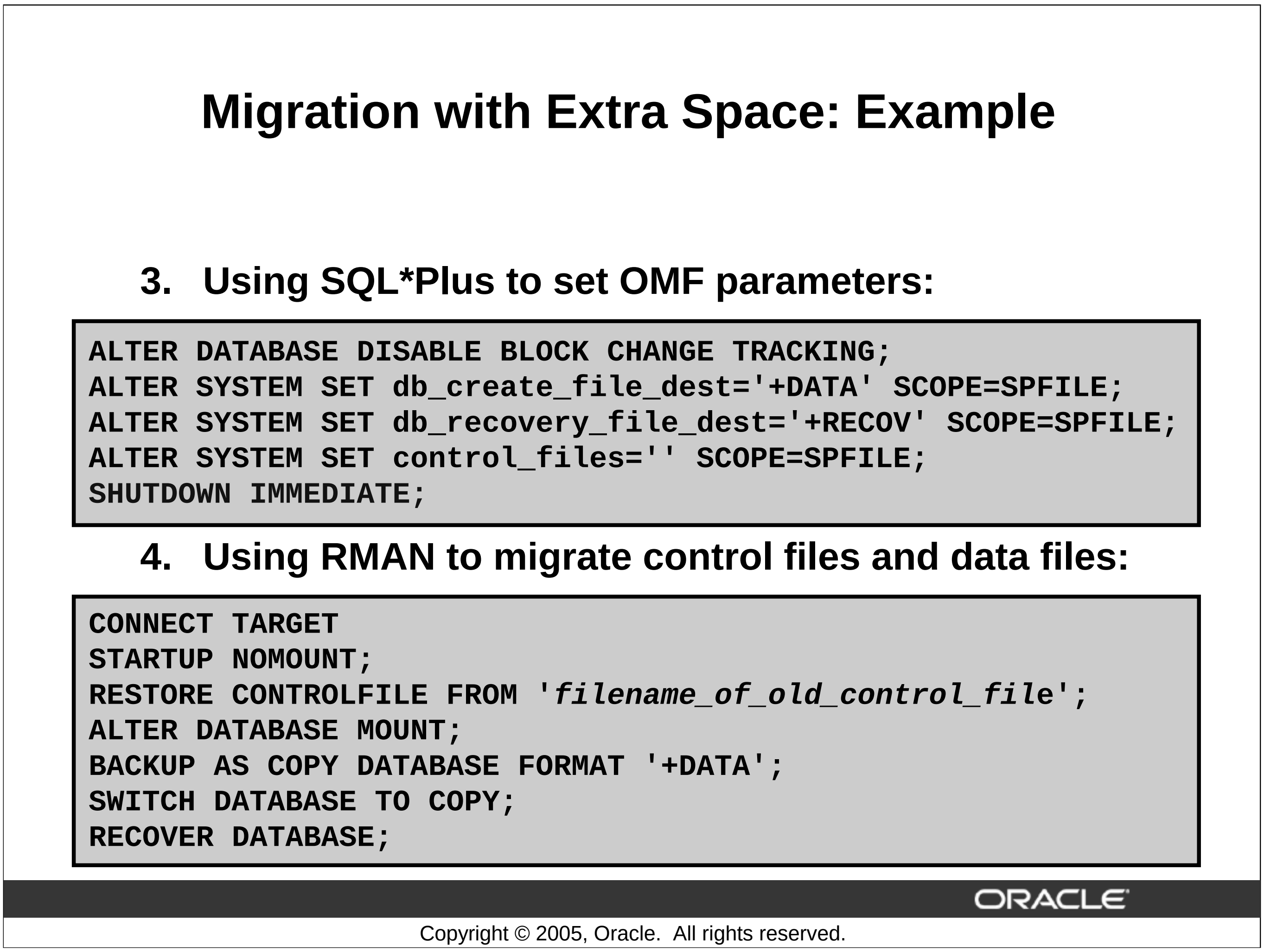
1. Create the ASM instances.
2. Create the data and recovery area disk groups.
3. Set database file and back up OMF parameters.
4. Create a database copy to ASM, and switch control files and data files to ASM.
5. Re-create flashback database logs, temp files, and change tracking file in ASM.
6. Optionally migrate your backups to ASM.
7. Drop and re-create online redo log groups to ASM.

Migration with Extra Space: Overview

The migration path that you are going to study with this case produces a minimum down time if your database is relatively small. As indicated in the slide, the down time is between steps four and six. However, only the flashback logs reconstitution in step five needs to be done offline.

In this example, it is assumed that your database is currently using a flash recovery area, and that you have enough disk space capacity to simultaneously store your database both on the file system and ASM.

Note: If your database is too big to afford a down time corresponding to the whole database backup, then you can create the database image copies to ASM while the database is online. Then, you can use an incrementally updated backup strategy to reduce the recovery time to its minimum before initiating the switch.



Migration with Extra Space: Example

3. Using SQL*Plus to set OMF parameters:

```
ALTER DATABASE DISABLE BLOCK CHANGE TRACKING;
ALTER SYSTEM SET db_create_file_dest='+DATA' SCOPE=SPFILE;
ALTER SYSTEM SET db_recovery_file_dest='+RECOV' SCOPE=SPFILE;
ALTER SYSTEM SET control_files='' SCOPE=SPFILE;
SHUTDOWN IMMEDIATE;
```

4. Using RMAN to migrate control files and data files:

```
CONNECT TARGET
STARTUP NOMOUNT;
RESTORE CONTROLFILE FROM 'filename_of_old_control_file';
ALTER DATABASE MOUNT;
BACKUP AS COPY DATABASE FORMAT '+DATA';
SWITCH DATABASE TO COPY;
RECOVER DATABASE;
```

Migration with Extra Space: Example

It is assumed that you already created ASM instances, as well as the DATA disk group for the database area and the RECOV disk group for the recovery area. It is also assumed that you are using an SPFILE.

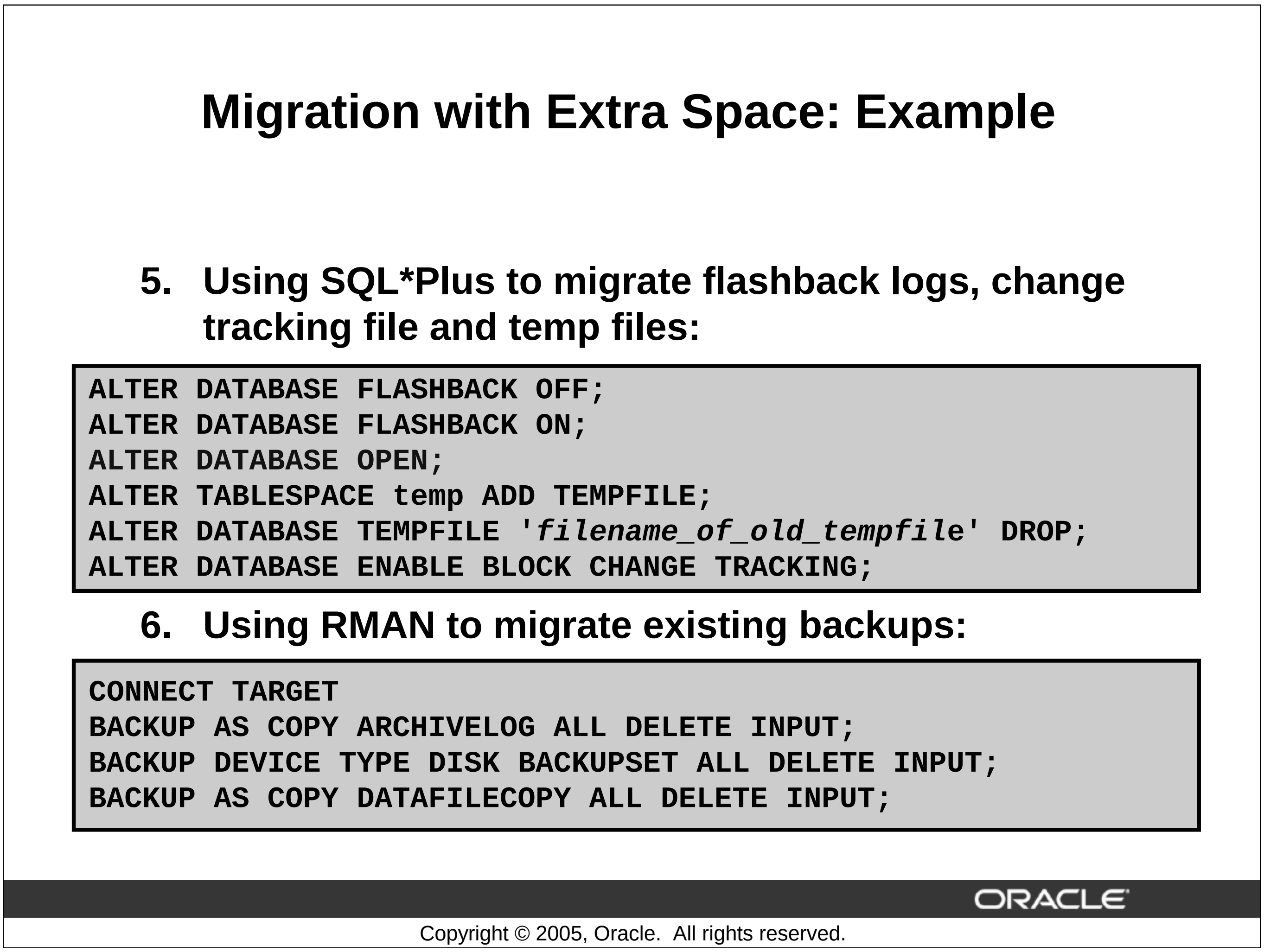
You can execute the first script using SQL*Plus from any instance that has the database open. However, you need to shut down any remaining instance immediately.

The goal of this script is to modify the OMF parameters of each instance to point to the new disk groups. By also resetting the value of CONTROL_FILES, you make sure that the OMF control files will be created in ASM. The script also turns off the block change tracking mechanism.

After the database is shut down, you can run the second script by using RMAN. This script create two multiplexed OMF control files stored in DATA and RECOV. Before executing this script, you need to specify the full name of one of the control files that has been used so far. Then, the script mounts the database by using the newly created control files, and backs up the existing file-system database to DATA using image copies. After completing, the control file pointers are switched to the ASM database image copies.

Then, if needed, the database is recovered. This might not be needed if the database was shut down correctly by the previous script.

Note: In a RAC/SPFILE environment, SID='*' is assumed for ALTER SYSTEM statements.



Migration with Extra Space: Example

5. Using SQL*Plus to migrate flashback logs, change tracking file and temp files:

```
ALTER DATABASE FLASHBACK OFF;
ALTER DATABASE FLASHBACK ON;
ALTER DATABASE OPEN;
ALTER TABLESPACE temp ADD TEMPFILE;
ALTER DATABASE TEMPFILE 'filename_of_old_tempfile' DROP;
ALTER DATABASE ENABLE BLOCK CHANGE TRACKING;
```

6. Using RMAN to migrate existing backups:

```
CONNECT TARGET
BACKUP AS COPY ARCHIVELOG ALL DELETE INPUT;
BACKUP DEVICE TYPE DISK BACKUPSET ALL DELETE INPUT;
BACKUP AS COPY DATAFILECOPY ALL DELETE INPUT;
```

Migration with Extra Space: Example (continued)

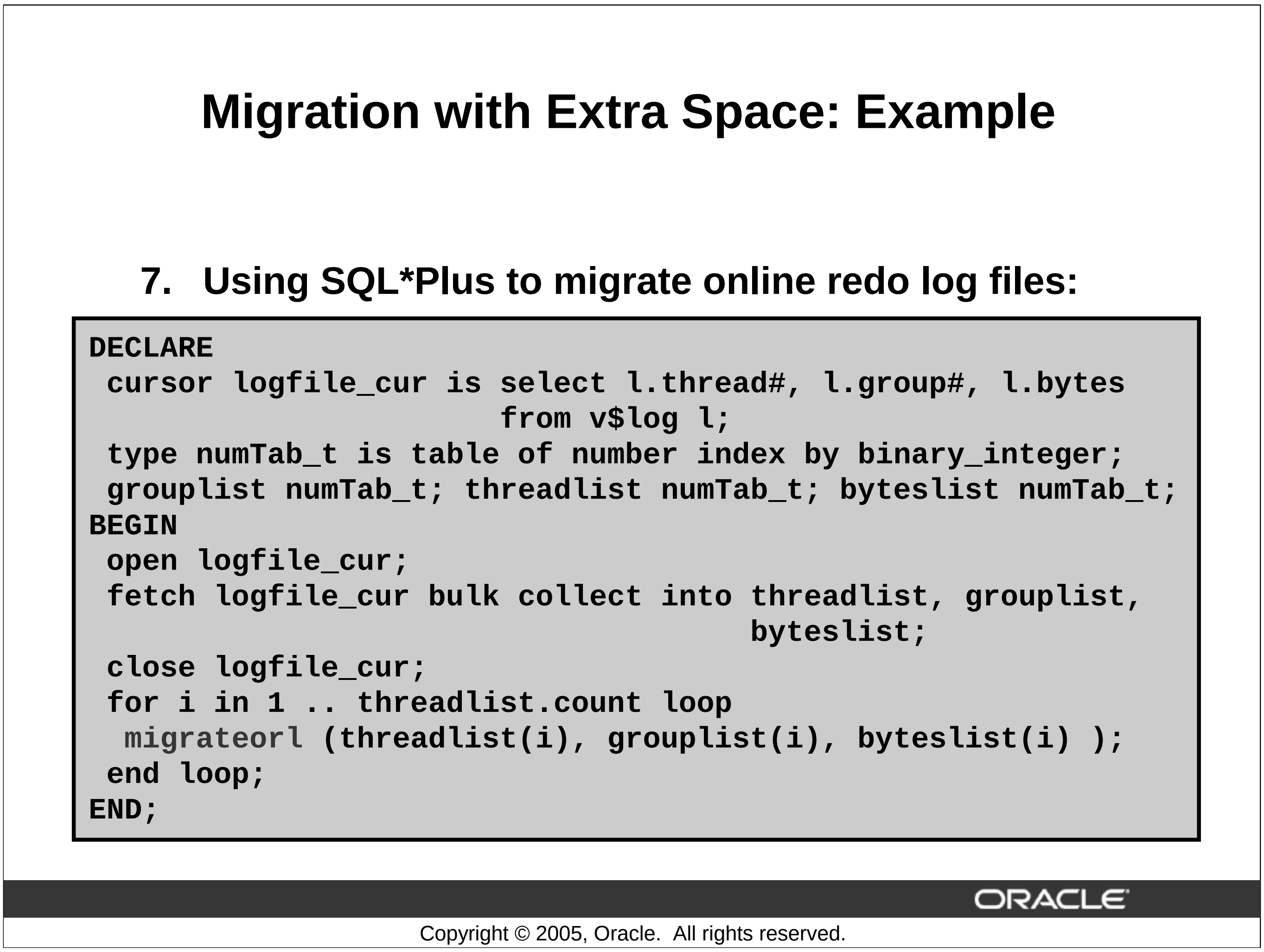
The third script disables and enables again flashback logging. By doing this, you re-create the flashback logs into RECOV. At that point, you can open the database again.

Because RMAN does not take into account tempfiles, you need to re-create them for each existing TEMPORARY tablespace. All you need for that is to add at least a new tempfile, and drop the ones that are still in the file system. You need to do that for each TEMPORARY tablespace. Therefore, before executing the third script, make sure that you specify the right tempfile names.

The third script also re-creates the change-tracking file directly inside DATA.

The fourth script is optional but might be recommended if you want to get rid of the old flash recovery area files. The goal of this script is to transfer the existing backups to RECOV.

The first BACKUP command is used to move all the current archived log files that have not yet been backed up. The second BACKUP command is used to move all the current backup sets. The last BACKUP command is used to move all the current data file copies, including the ones corresponding to the old file system database.



Migration with Extra Space: Example

7. Using SQL*Plus to migrate online redo log files:

```
DECLARE
 cursor logfile_cur is select l.thread#, l.group#, l.bytes
                       from v$log l;
 type numTab_t is table of number index by binary_integer;
 grouplist numTab_t; threadlist numTab_t; byteslist numTab_t;
BEGIN
 open logfile_cur;
 fetch logfile_cur bulk collect into threadlist, grouplist,
                                     byteslist;
 close logfile_cur;
 for i in 1 .. threadlist.count loop
  migrateorl (threadlist(i), grouplist(i), byteslist(i) );
 end loop;
END;
```

Migration with Extra Space: Example (continued)

The last step of this procedure is to create new online redo log files in ASM, and drop the existing ones from the file system.

The above script can be used to automate this process. It is assumed that you have already created the `MIGRATEORL` procedure discussed later in this lesson.

The basic idea of the above script is to add a new redo log group inside ASM for each existing redo log group on the file system, and then drop the corresponding existing group from the file system.

Therefore, the script retrieves all the groups of each thread, and for each of them, it invokes the `MIGRATEORL` procedure.

Migration with Extra Space: Example

```
CREATE PROCEDURE migrateorl(thread# number, group# number,
                            bytes number) is
 stmt  varchar2(1024):='alter database add logfile thread'||
                        thread#||' size '||bytes;
 asalc varchar2(1024):='alter system archive log current';
BEGIN
 execute immediate stmt;
 stmt := 'alter database drop logfile group '||group#;
 for i in 1 .. 5 loop
  begin
   execute immediate stmt;
   exit;
  exception
   when others then execute immediate asalc;
  end;
 end loop;
END;
```

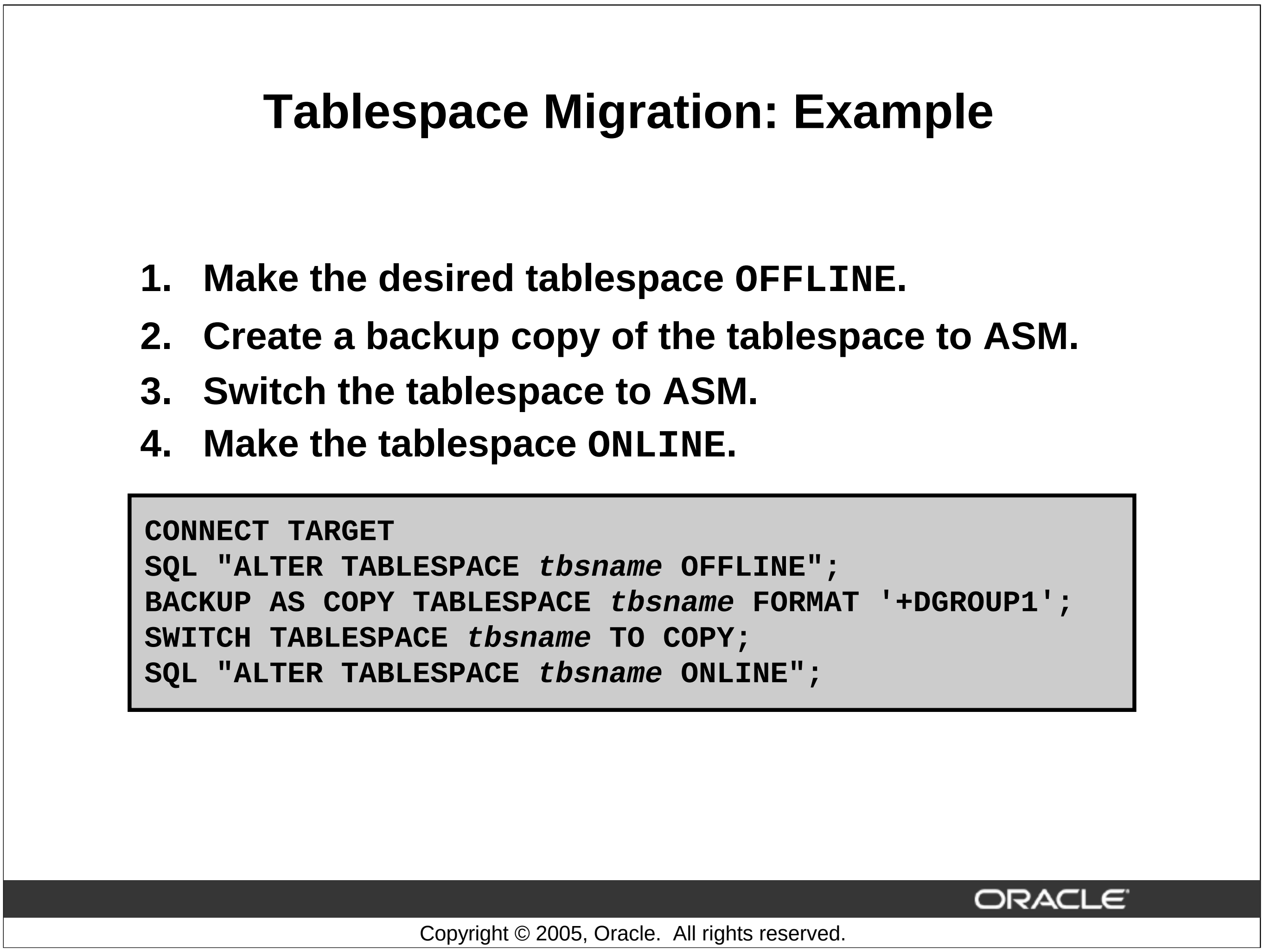
Migration with Extra Space: Example (continued)

The goal of the MIGRATEORL procedure is to drop a particular online redo log group from one thread, and to create a new one inside ASM for the same thread.

The only issue is with the CURRENT log of each thread. Because it is not possible to drop a CURRENT group, you need to generate an artificial log switch before you can drop a CURRENT group. In a RAC environment, when the database is open, the global switch is achieved by using the ALTER SYSTEM ARCHIVELOG CURRENT command. This command archives all redo log file groups from all enabled threads, which forces a switch to occur for each enabled thread.

Therefore, at the beginning, the procedure adds a new group for the specified thread. Then it tries to drop the given group. If it succeeds, the procedure has successfully migrated one group. If it fails to drop the group, this is because the group is a CURRENT group. At that point, the procedure tries to switch and then tries to drop the group again. The procedure retries this five times before it stops. This ensures that there is never a problem.

Note: This procedure is not part of the set standard set of procedure. You need to manually create this procedure in your database.



Tablespace Migration: Example

1. Make the desired tablespace `OFFLINE`.
2. Create a backup copy of the tablespace to ASM.
3. Switch the tablespace to ASM.
4. Make the tablespace `ONLINE`.

```
CONNECT TARGET
SQL "ALTER TABLESPACE tbsname OFFLINE";
BACKUP AS COPY TABLESPACE tbsname FORMAT '+DGROUP1';
SWITCH TABLESPACE tbsname TO COPY;
SQL "ALTER TABLESPACE tbsname ONLINE";
```

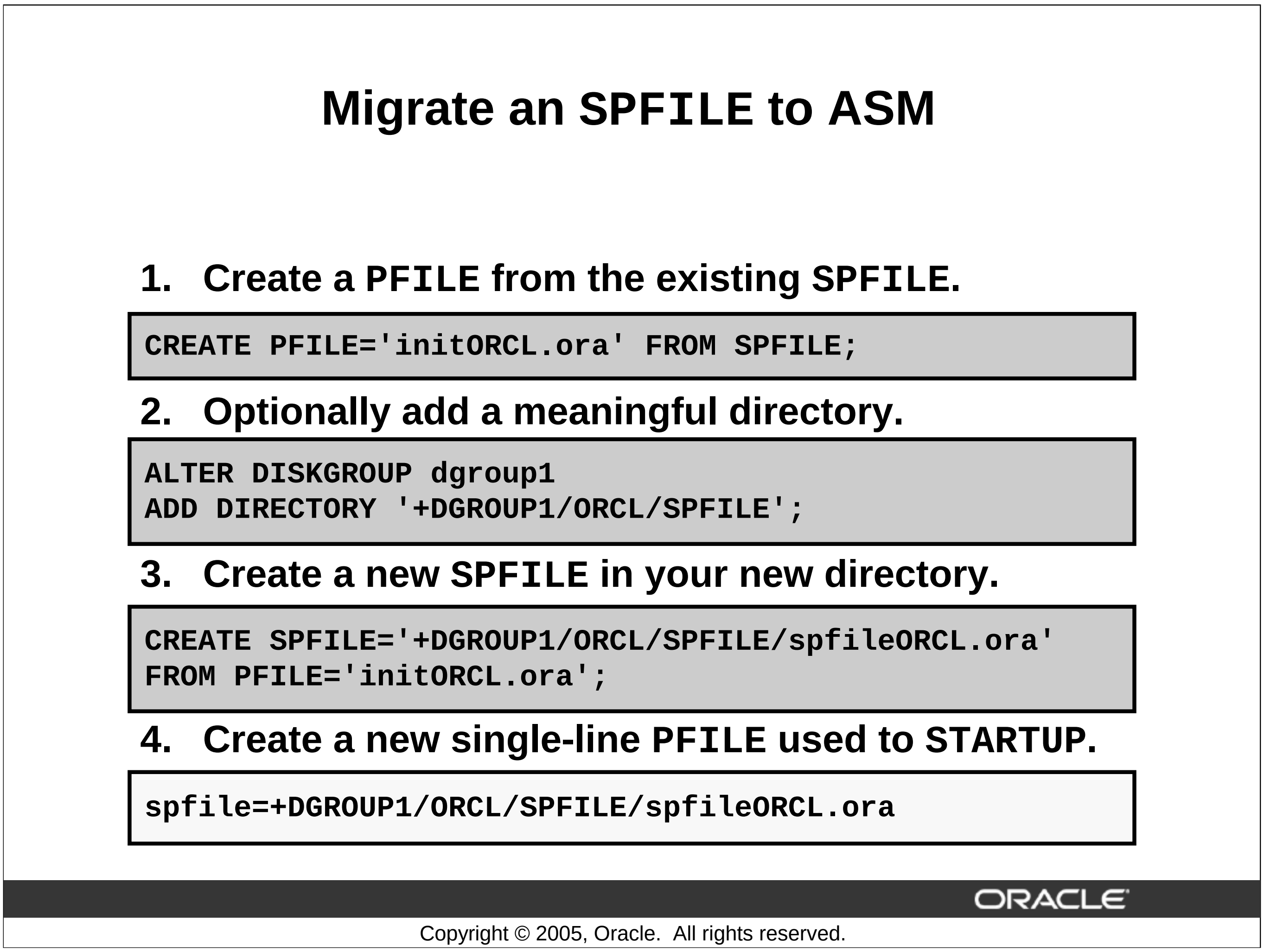
Tablespace Migration: Example

This procedure describes one possible way of migrating an individual tablespace to ASM while the database is online.

It is assumed that ASM instances are already created, and that the `DATA` disk group is currently mounted by all ASM instances.

1. You need to use RMAN to connect to the target database.
2. Make the target tablespace `OFFLINE` or `READ ONLY`.
3. Copy the tablespace to the ASM disk group.
4. Switch the control file pointer to the ASM copy.
5. Make the target tablespace `ONLINE` or `READ WRITE` again.

Note: Before you execute the above RMAN script, replace *`tbsname`* occurrences with the corresponding name of the tablespace that you want to migrate.



Migrate an SPFILE to ASM

1. Create a PFILE from the existing SPFILE.

```
CREATE PFILE='initORCL.ora' FROM SPFILE;
```

2. Optionally add a meaningful directory.

```
ALTER DISKGROUP dgroup1
ADD DIRECTORY '+DGROUP1/ORCL/SPFILE';
```

3. Create a new SPFILE in your new directory.

```
CREATE SPFILE='+DGROUP1/ORCL/SPFILE/spfileORCL.ora'
FROM PFILE='initORCL.ora';
```

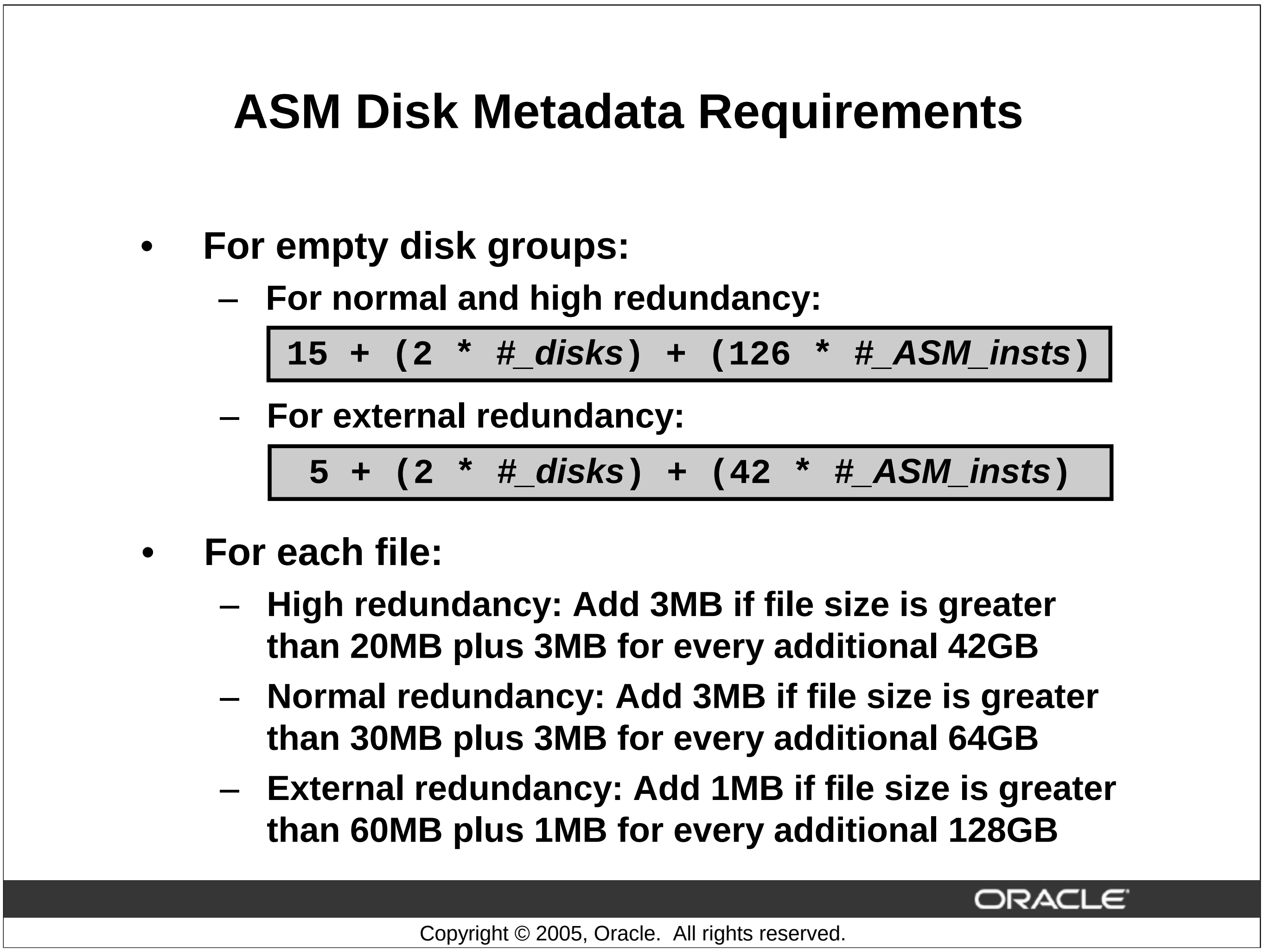
4. Create a new single-line PFILE used to STARTUP.

```
spfile=+DGROUP1/ORCL/SPFILE/spfileORCL.ora
```

Migrate an SPFILE to ASM

It is possible to store SPFILEs inside an ASM disk group. From a database instance, you can use the CREATE SPFILE statement to do this. The slide shows you the procedure to follow in order to migrate an existing file system SPFILE to an ASM disk group:

1. First of all, you need to create a PFILE from the existing SPFILE. This is because the CREATE SPFILE command needs a PFILE as parameter. You create the PFILE in your file system.
2. Although you can create an SPFILE by just specifying a disk group name, it might be important to create a meaningful directory alias in your disk group to precisely locate your SPFILE. This is especially relevant if you want to create backups, or have multiple databases stored in the same disk group. In the example, it is assumed that you have already created a database file for the ORCL database inside the DGROUP1 ASM disk group. By default, ASM adds a directory corresponding to the name of your database. The example just adds the SPFILE directory under the existing ORCL directory.
3. You can then create the new SPFILE directly in the new directory with a specified alias. The alias is automatically created by ASM.
4. The last step is to then create a new PFILE with only the SPFILE parameter pointing to the new SPFILE. This PFILE should then be used to start up your instances.



ASM Disk Metadata Requirements

- For empty disk groups:
  - For normal and high redundancy:

```
15 + (2 * #_disks) + (126 * #_ASM_insts)
```

  - For external redundancy:

```
5 + (2 * #_disks) + (42 * #_ASM_insts)
```

- For each file:
  - High redundancy: Add 3MB if file size is greater than 20MB plus 3MB for every additional 42GB
  - Normal redundancy: Add 3MB if file size is greater than 30MB plus 3MB for every additional 64GB
  - External redundancy: Add 1MB if file size is greater than 60MB plus 1MB for every additional 128GB

ASM Disk Metadata Requirements

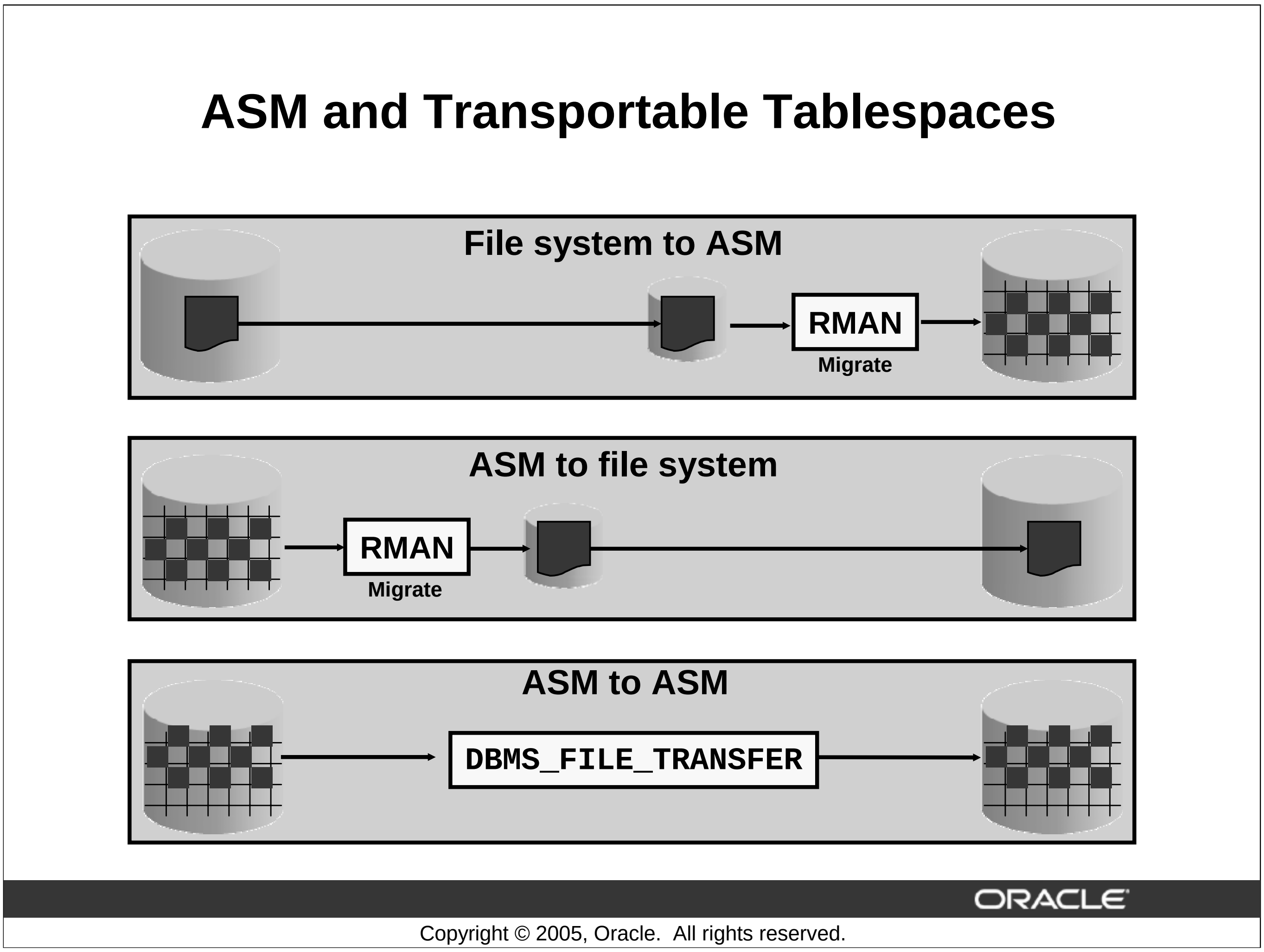
You must also add additional disk space for the ASM metadata. You can use the above formulas to calculate the additional disk space requirements (in MB) for an empty disk group.

For example, for a four-node RAC installation, using three disks in a high-redundancy disk group, you require an additional 525 MB of disk space: (15 + (2 * 3) +(126 * 4)) = 525.

As files are created, there is additional metadata overhead:

- With high redundancy, every file grater than 20MB adds 3MB of metadata, and another 3MB for every additional 42GB in that file.
- With normal redundancy, every file grater than 30MB adds 3MB of metadata, and another 3MB for every additional 64GB in that file.
- With external redundancy, every file greater than 60MB adds 1MB of metadata, and another 1MB for every additional 128GB in that file.

Note: Compared to the space used for storing user data, this should all be noise.



ASM and Transportable Tablespaces

File system to ASM

RMAN

Migrate

ASM to file system

RMAN

Migrate

ASM to ASM

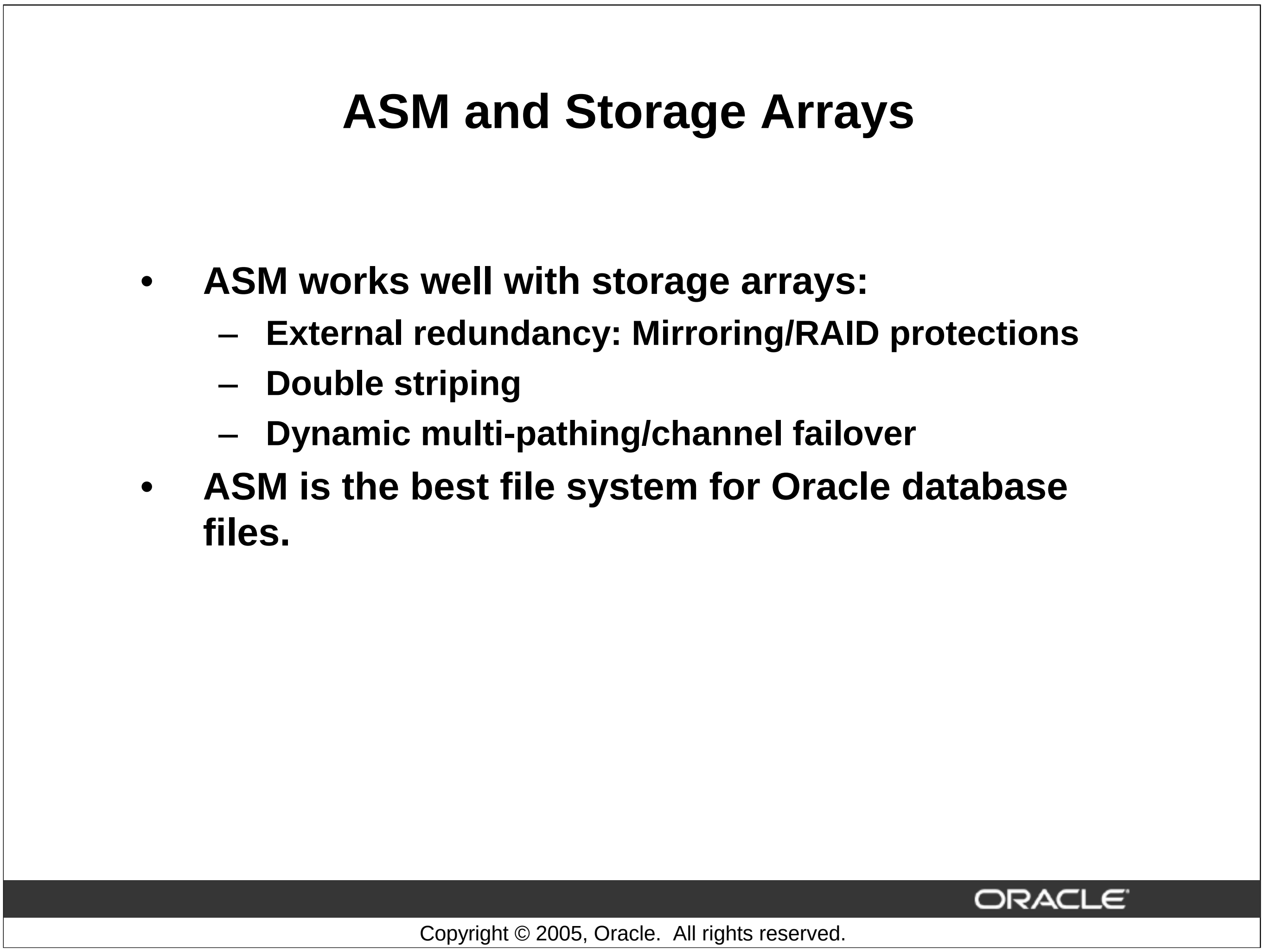
DBMS_FILE_TRANSFER

ASM and Transportable Tablespaces

You can copy a data file stored inside an ASM disk group on one machine to an ASM disk group on another machine via the `DBMS_FILE_TRANSFER` package running in one of the database instances. This operation can be performed directly without having to covert the data file.

However, if you want to transport a data file stored in a traditional file system to an ASM disk group on another database, then you need to plug the data file in the target database by using the classic transportable tablespace procedure, and then use `RMAN` to migrate the plugged tablespace to ASM.

Note: For more information about the `DBMS_FILE_TRANSFER` package, refer to the *PL/SQL Packages and Types Reference* guide.



ASM and Storage Arrays

- ASM works well with storage arrays:
  - External redundancy: Mirroring/RAID protections
  - Double striping
  - Dynamic multi-pathing/channel failover
- ASM is the best file system for Oracle database files.

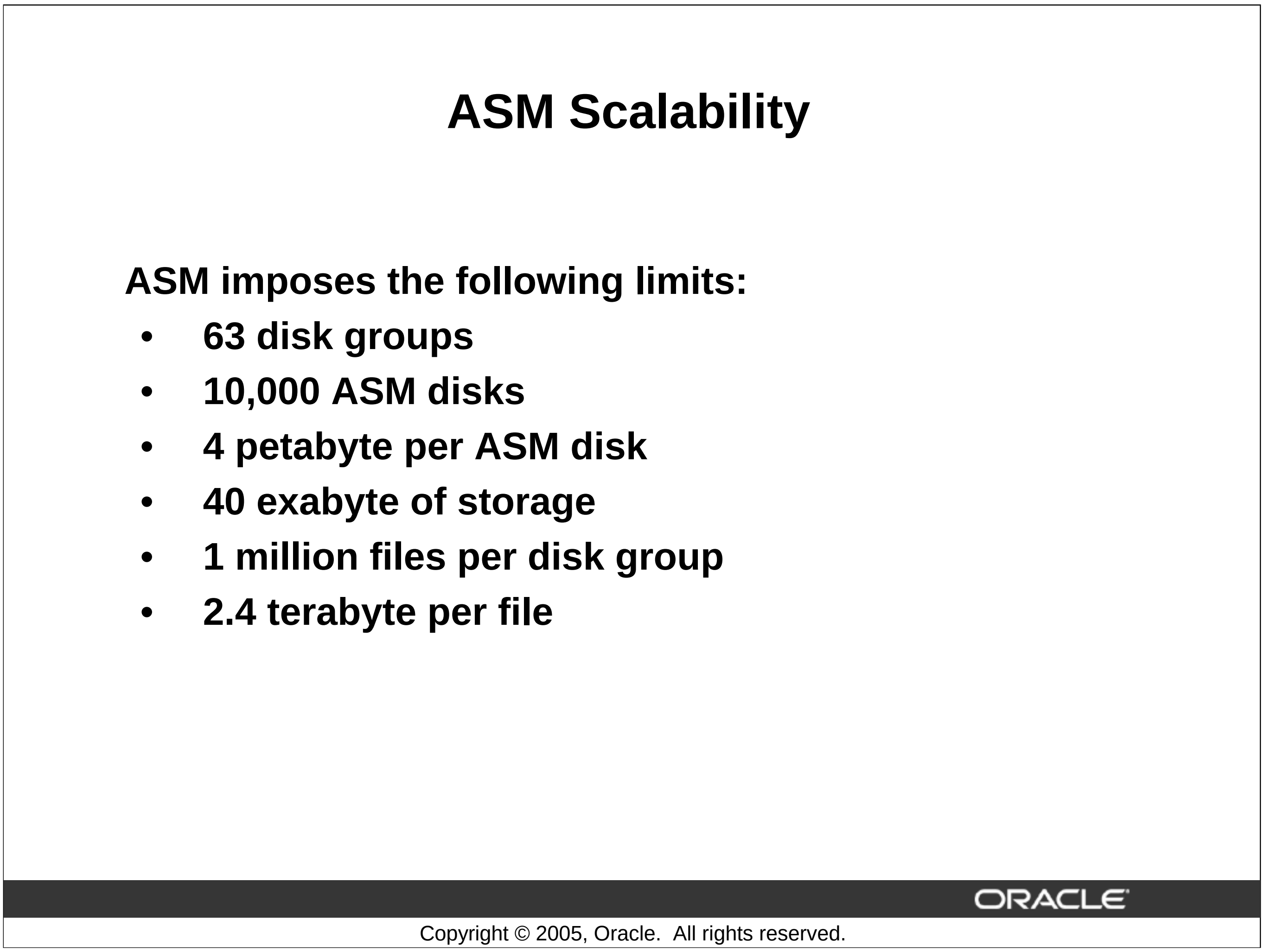
ASM and Storage Arrays

Using ASM does not imply that you have to discard your storage and replace it with something new.

Although ASM offers mirroring functionality, it is considered best practice to offload these tasks to an external storage array that supports the mirroring or RAID 5 technology, if available. ASM can be configured to use these external redundancy mechanisms to protect data.

Server-level striping can be used to complement the performance benefit of storage-level striping. This technique is known as double striping. Because server-level striping provides an even distribution of I/O across the back end of the storage layer, and dynamic multi-pathing provides a dynamic distribution across the available channels, use of ASM to evenly distribute the database I/O across striped metavolumes provides an efficient double striping strategy.

Use of ASM with the recovery area provides the purpose-built file system of choice for Oracle Database 10*g* environments. It provides automation of file naming best practices, placement, and data file automatic expansion. Combined with the recovery area, there is no better purpose-built clustered file system for Oracle database files than ASM.



ASM Scalability

ASM imposes the following limits:

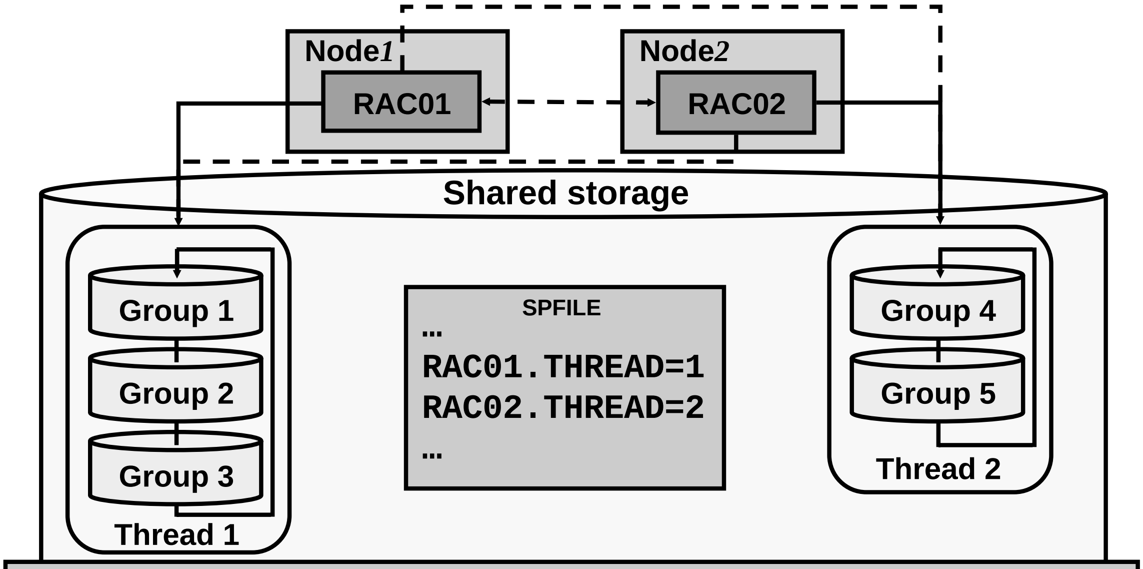
- 63 disk groups
- 10,000 ASM disks
- 4 petabyte per ASM disk
- 40 exabyte of storage
- 1 million files per disk group
- 2.4 terabyte per file

ASM Scalability

ASM imposes the following limits:

- 63 disk groups in a storage system
- 10,000 ASM disks in a storage system
- 4 petabyte maximum storage for each ASM disk
- 40 exabyte maximum storage for each storage system
- 1 million files for each disk group
- 2.4 terabyte maximum storage for each file

Redo Log Files and RAC



```
ALTER DATABASE ADD LOGFILE THREAD 2 GROUP 4;
ALTER DATABASE ADD LOGFILE THREAD 2 GROUP 5;
```

```
ALTER DATABASE ENABLE THREAD 2;
```

Redo Log Files and RAC

With Real Application Clusters (RAC), each instance writes to its own set of online redo log files, and the redo written by an instance is called a thread of redo, or thread. Thus, each redo log file group used by an instance is associated with the same thread number determined by the value of the THREAD initialization parameter. If you set the THREAD parameter to a nonzero value for a particular instance, the next time the instance is started, it will try to use that thread. Because an instance can use a thread as long as that thread is enabled, and not in use by another instance, it is recommended to set the THREAD parameter to a nonzero value with each instance having different values.

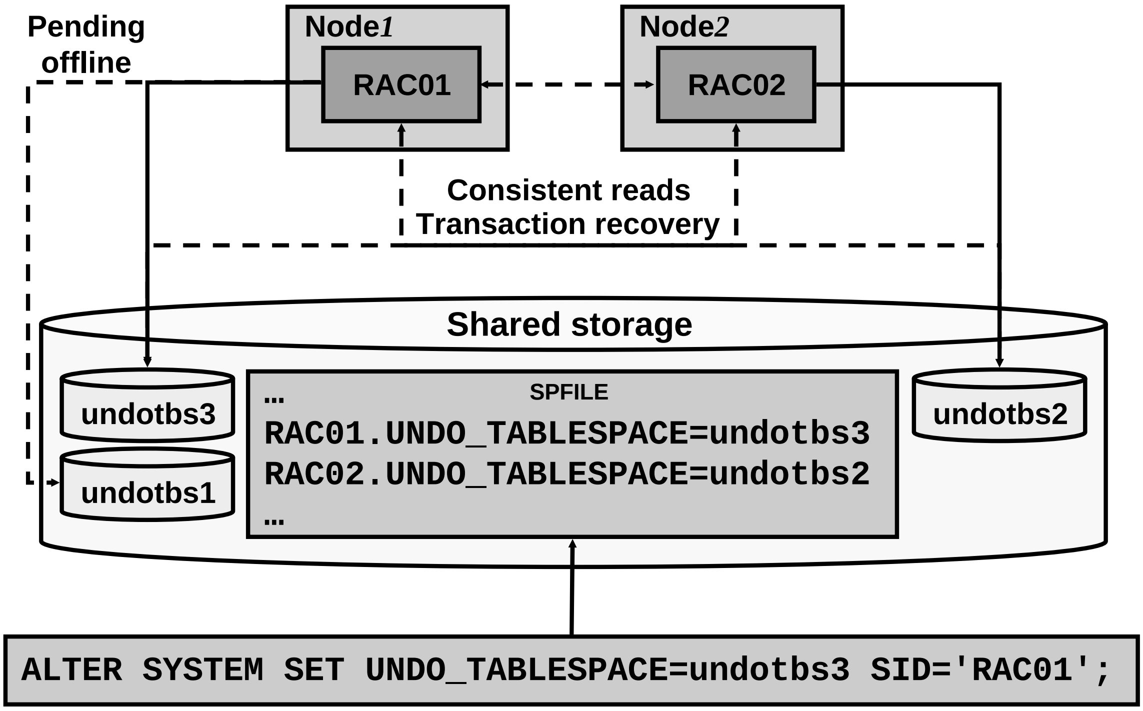
You associate a thread number with a redo log file group by using the ALTER DATABASE ADD LOGFILE THREAD statement. You enable a thread number by using the ALTER DATABASE ENABLE THREAD statement. Before you can enable a thread, it must have at least two redo log file groups.

By default a database is created with one enabled public thread. An enabled public thread is a thread that has been enabled by using the ALTER DATABASE ENABLE PUBLIC THREAD statement. Such a thread can be acquired by an instance with its THREAD parameter set to zero.

Therefore, you need to create and enable additional threads when you add instances to your database.

Note: The maximum possible value for the THREAD parameter is the value assigned to the MAXINSTANCES parameter specified in the CREATE DATABASE statement.

Automatic Undo Management and RAC



```
…
RAC01.UNDO_TABLESPACE=undotbs3
RAC02.UNDO_TABLESPACE=undotbs2
…
```

```
ALTER SYSTEM SET UNDO_TABLESPACE=undotbs3 SID='RAC01';
```

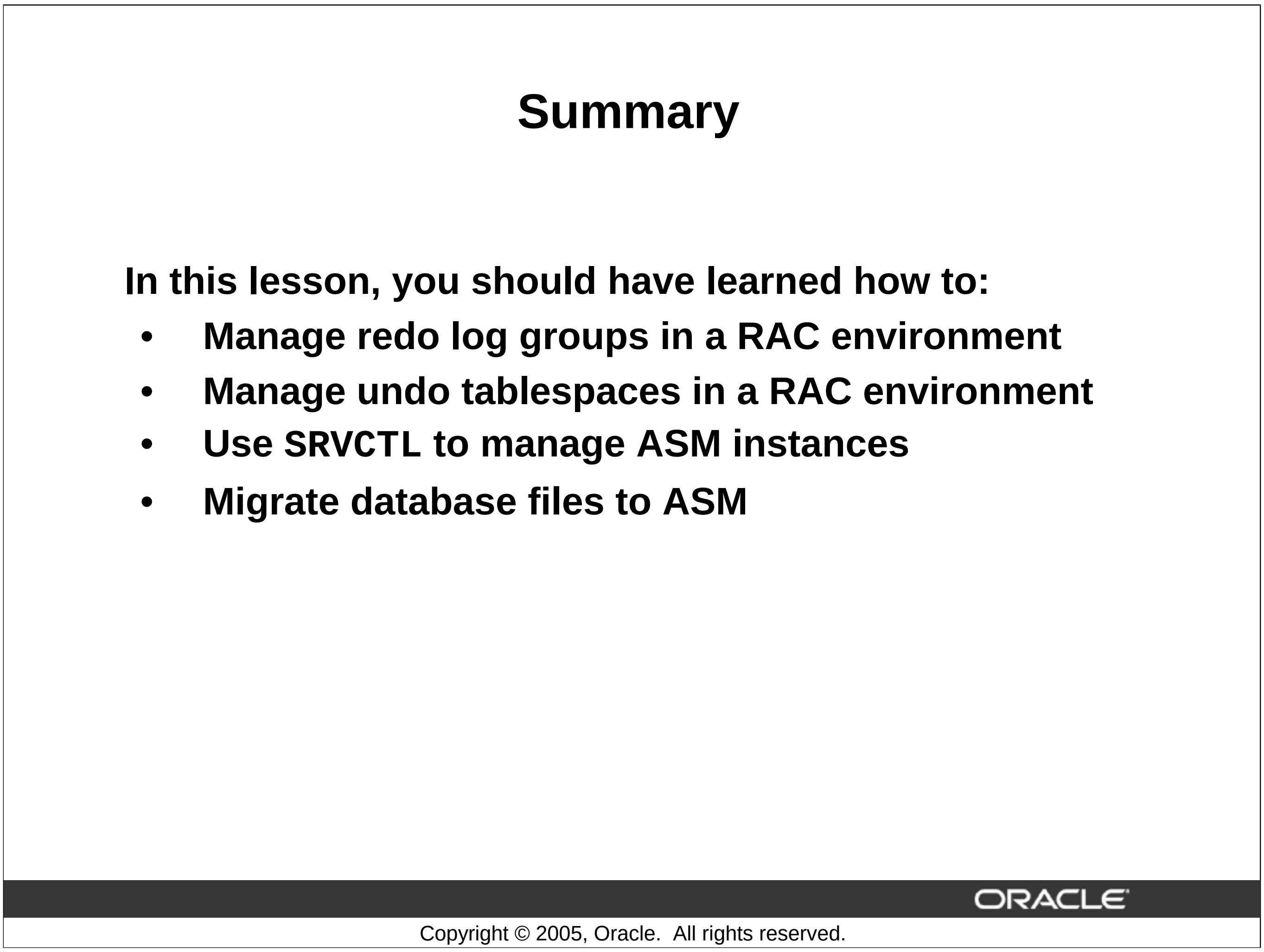
Automatic Undo Management in RAC

The Oracle database automatically manages undo segments within a specific undo tablespace that is assigned to an instance. Under normal circumstances, only the instance assigned to the undo tablespace can modify the contents of that tablespace. However, all instances can always read all undo blocks for consistent read purposes. Also, any instance can update any undo tablespace during transaction recovery, as long as that undo tablespace is not currently used by another instance for undo generation or transaction recovery.

You assign undo tablespaces in your RAC database by specifying a different value for the `UNDO_TABLESPACE` parameter for each instance in your `SPFILE` or individual `PFILE`s. If you do not set the `UNDO_TABLESPACE` parameter, then each instance uses the first available undo tablespace. If undo tablespaces are not available, the `SYSTEM` rollback segment is used.

You can dynamically switch undo tablespace assignments by executing the `ALTER SYSTEM SET UNDO_TABLESPACE` statement with the `SID` clause. You can run this command from any instance. In this example, the previously used undo tablespace assigned to instance `RAC01` remains assigned to it until the `RAC01` instance’s last active transaction commits. The pending offline tablespace may be unavailable for other instances until all transactions against that tablespace are committed.

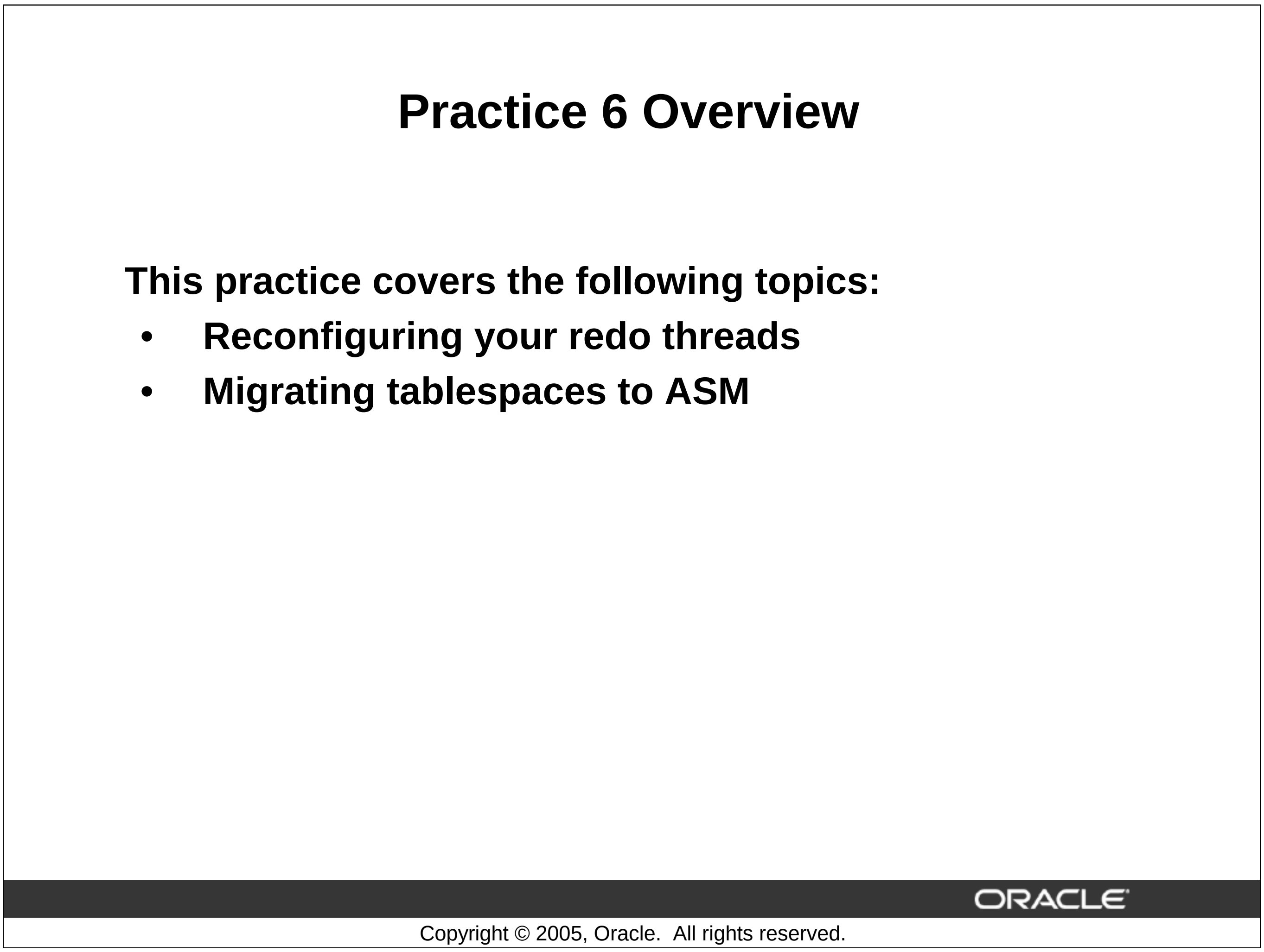
Note: You cannot simultaneously use automatic undo management (AUM) and manual undo management in a RAC database. It is highly recommended that you use the AUM mode.



Summary

In this lesson, you should have learned how to:

- Manage redo log groups in a RAC environment
- Manage undo tablespaces in a RAC environment
- Use `SRVCTL` to manage ASM instances
- Migrate database files to ASM



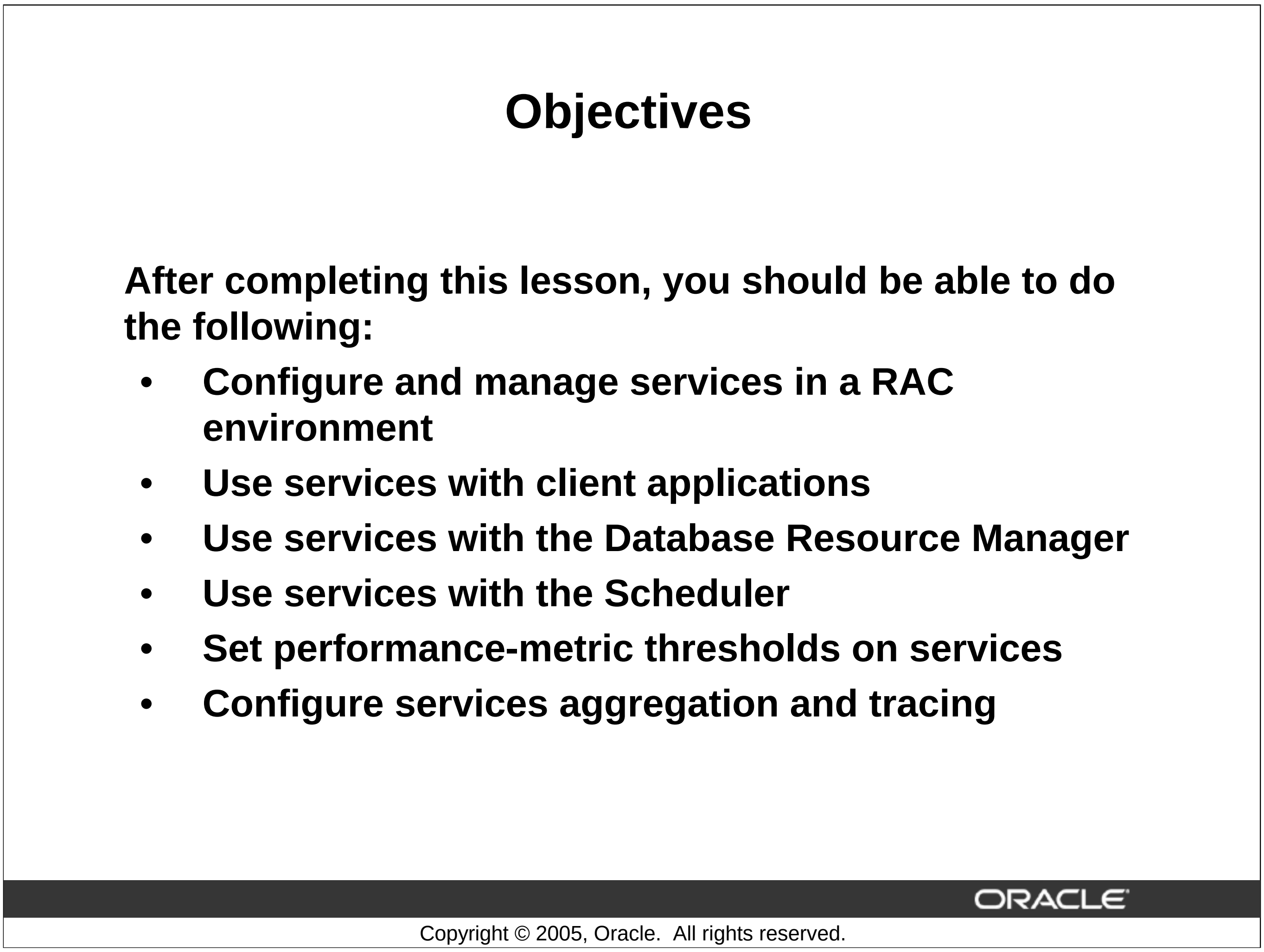
Practice 6 Overview

This practice covers the following topics:

- Reconfiguring your redo threads
- Migrating tablespaces to ASM

7

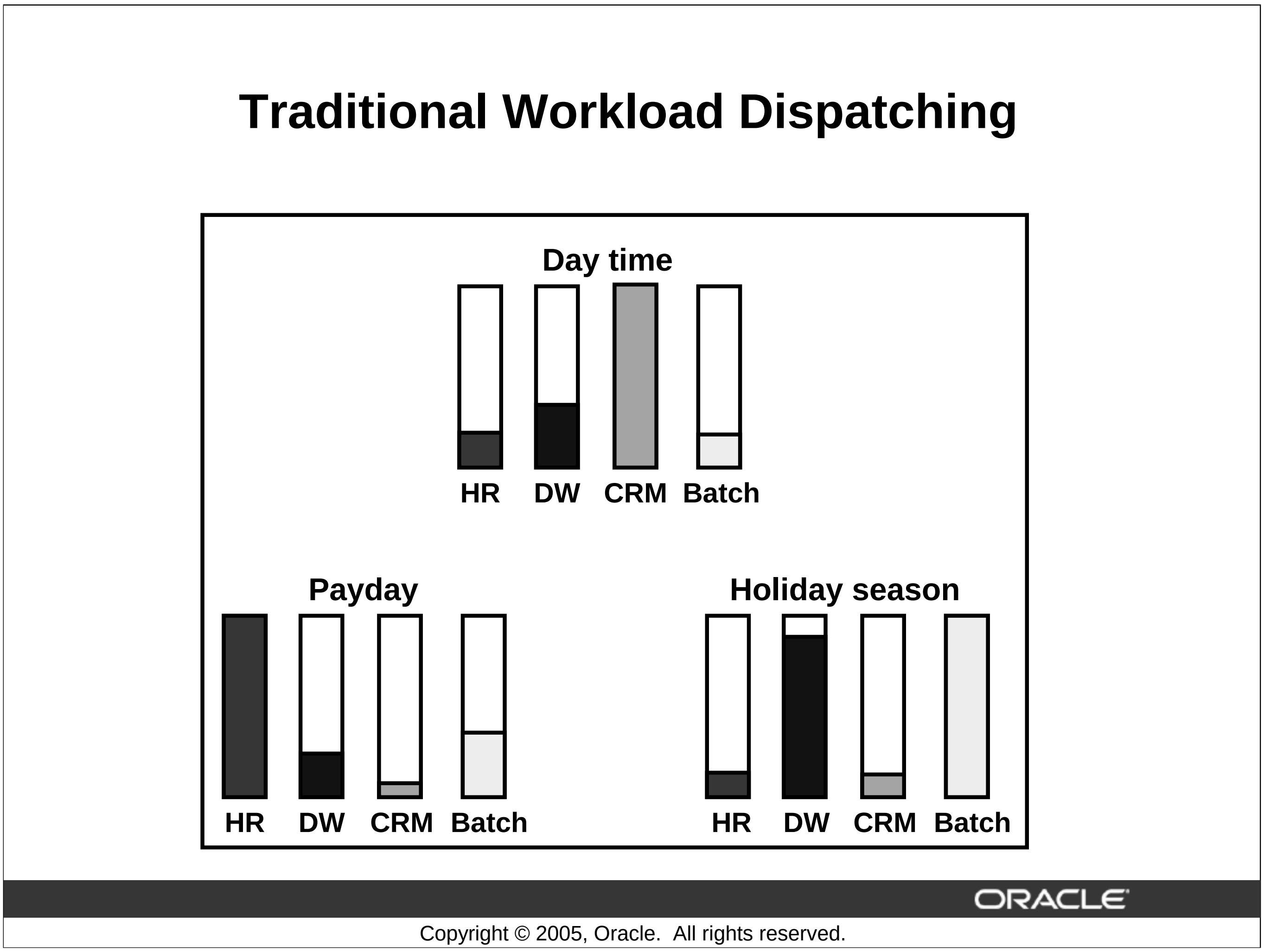
Services



Objectives

After completing this lesson, you should be able to do the following:

- Configure and manage services in a RAC environment
- Use services with client applications
- Use services with the Database Resource Manager
- Use services with the Scheduler
- Set performance-metric thresholds on services
- Configure services aggregation and tracing

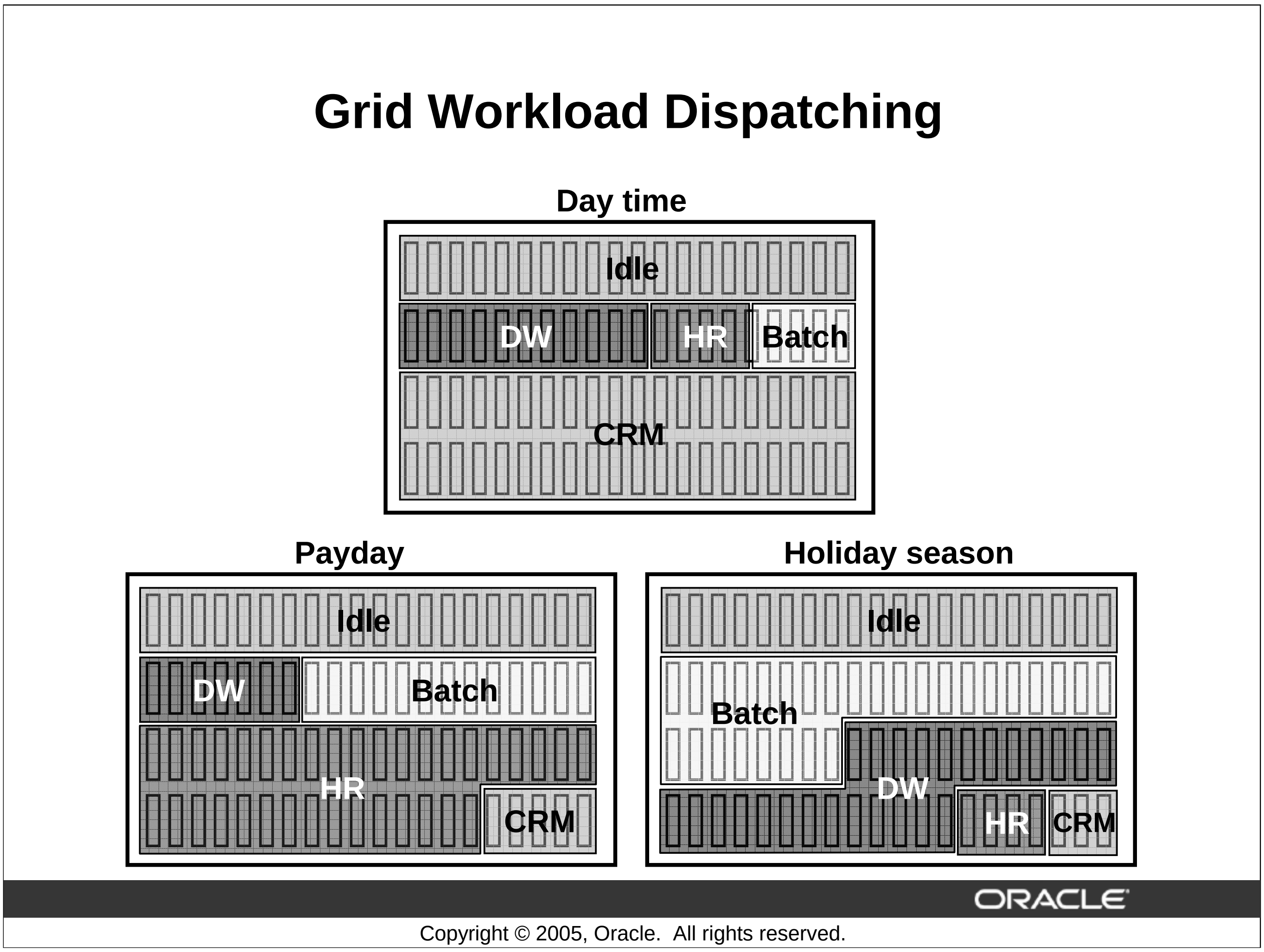


Traditional Workload Dispatching

Traditional Workload Dispatching

In a standard environment, isolated computing units of different sizes are permanently dedicated to specific applications such as Human Resources, Data Warehouses, Customer Relationship Management, and Retail Batches.

These computing units need to be sized for their peak workload. As the peak workload occurs for some hours only, a considerable amount of resources is idle for a long time.



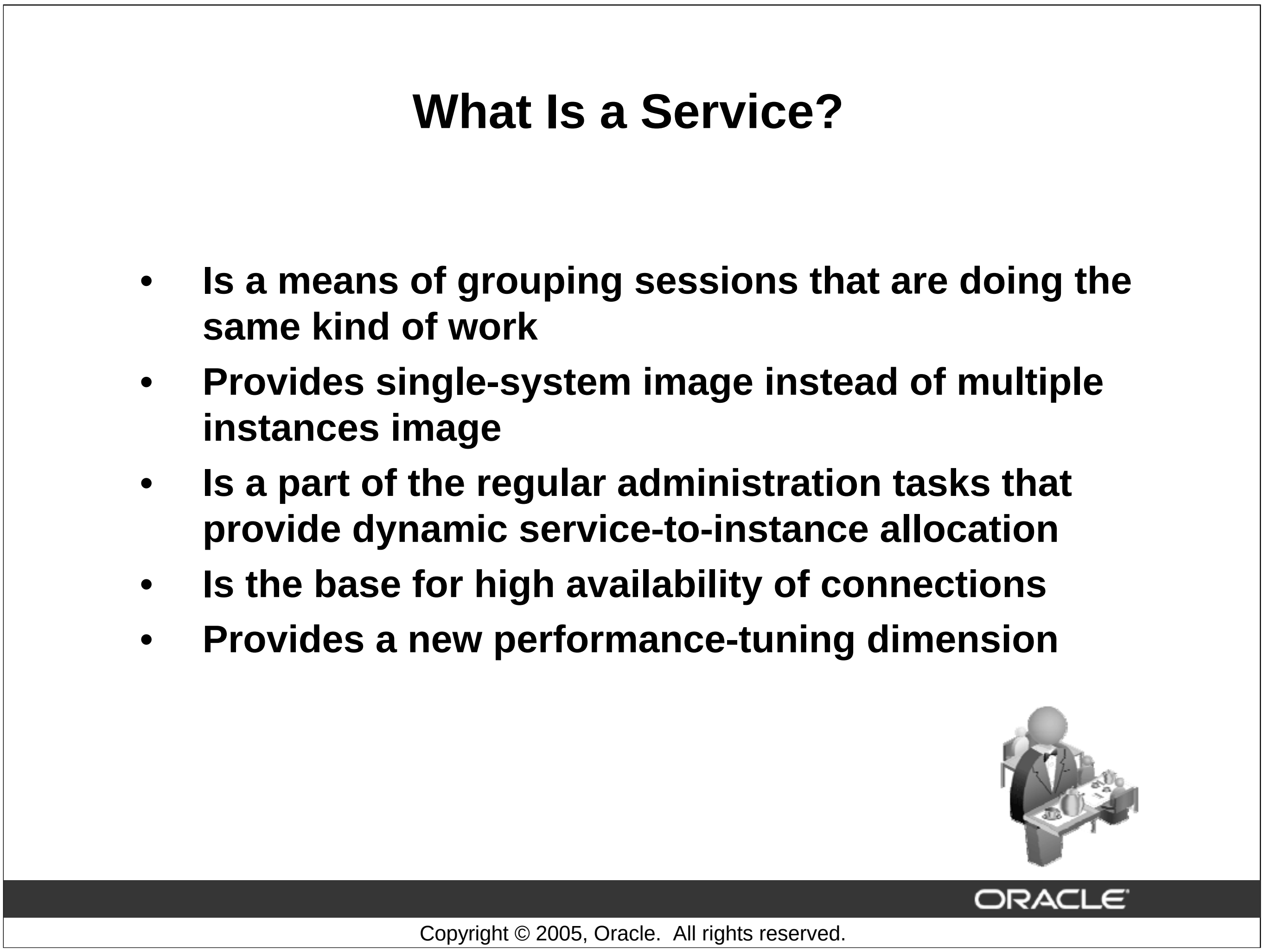
Grid Workload Dispatching

Grid Workload Dispatching

With Grid Computing, a global pool of computing units can be provided, and the computing units can be temporarily assigned to specific applications. Computing units can then be dynamically exchanged between applications. During business hours, more units can be used for CRM applications, and after business hours, some of them can be transferred to Retail Batches.

Grid Computing minimizes unused resources. This means that overall a grid-enabled environment needs less computing power than an environment that is not grid enabled.

In the example, 25 percent of the computing resource units are idle. This unused extra capacity is there so that service levels can still be met when there are components failures, such as nodes or instances, and also to deal with unexpected workloads. This is much better than the industry average of 70 to 90 percent idle rates when each machine is sized for its individual maximum.



What Is a Service?

- Is a means of grouping sessions that are doing the same kind of work
- Provides single-system image instead of multiple instances image
- Is a part of the regular administration tasks that provide dynamic service-to-instance allocation
- Is the base for high availability of connections
- Provides a new performance-tuning dimension

What Is a Service?

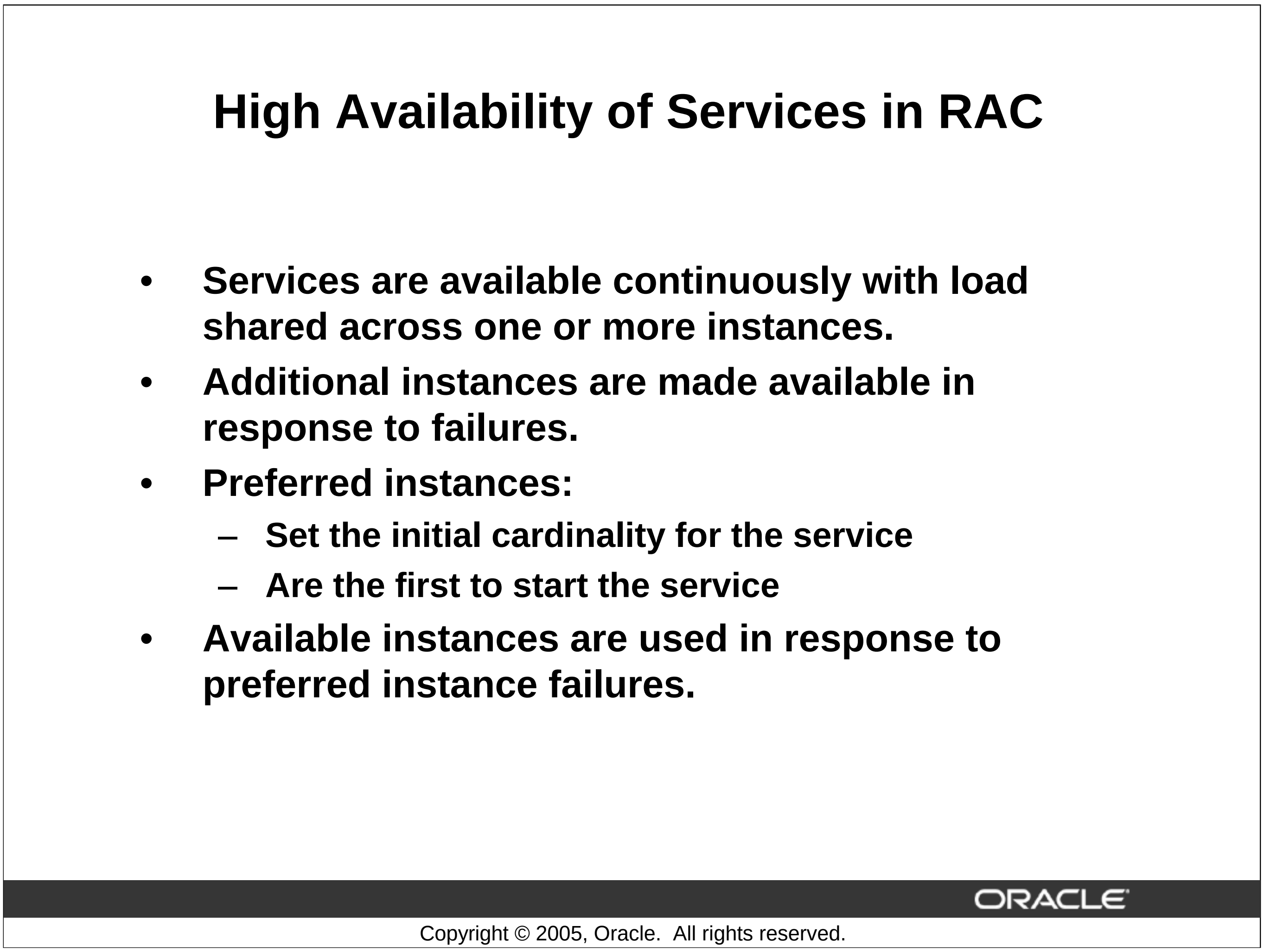
The concept of a service was first introduced in Oracle8*i* as a means for the listener to do connection load balancing between nodes and instances of a cluster. However, the concept, definition, and implementation of services have been dramatically expanded. Services are a feature for workload management that organizes the universe of work execution within the database to make that work more manageable, measurable, tunable, and recoverable. A service is a grouping of related tasks within the database with common functionality, quality expectations, and priority relative to other services. The notion of service provides a single-system image for managing competing applications running within a single instance and across multiple instances and databases.

Using standard interfaces, such as DBCA, Enterprise Manager, and SRVCTL, services can be configured, administered, enabled, disabled, and measured as a single entity.

Services provide availability. Following outages, a service is recovered fast and automatically at surviving instances.

Services provide a new dimension to performance tuning. With services, workloads are visible and measurable. Tuning by “service and SQL” replaces tuning by “session and SQL” in the majority of systems where sessions are anonymous and shared.

Services are dynamic in that the number of instances a service runs on can be augmented when load increases, and reduced when load declines. This dynamic resource allocation enables a cost-effective solution for meeting demands as they occur.



High Availability of Services in RAC

- Services are available continuously with load shared across one or more instances.
- Additional instances are made available in response to failures.
- Preferred instances:
  - Set the initial cardinality for the service
  - Are the first to start the service
- Available instances are used in response to preferred instance failures.

High Availability of Services in RAC

With RAC, the focus of high availability (HA) is on protecting the logically defined application services. This focus is more flexible than focusing on high availability of instances.

Services must be location-independent and the RAC HA framework is used to implement this. Services are made available continuously with load shared across one or more instances in the cluster. Any instance can offer services in response to run-time demands, failures, and planned maintenance. Services are always available somewhere in the cluster.

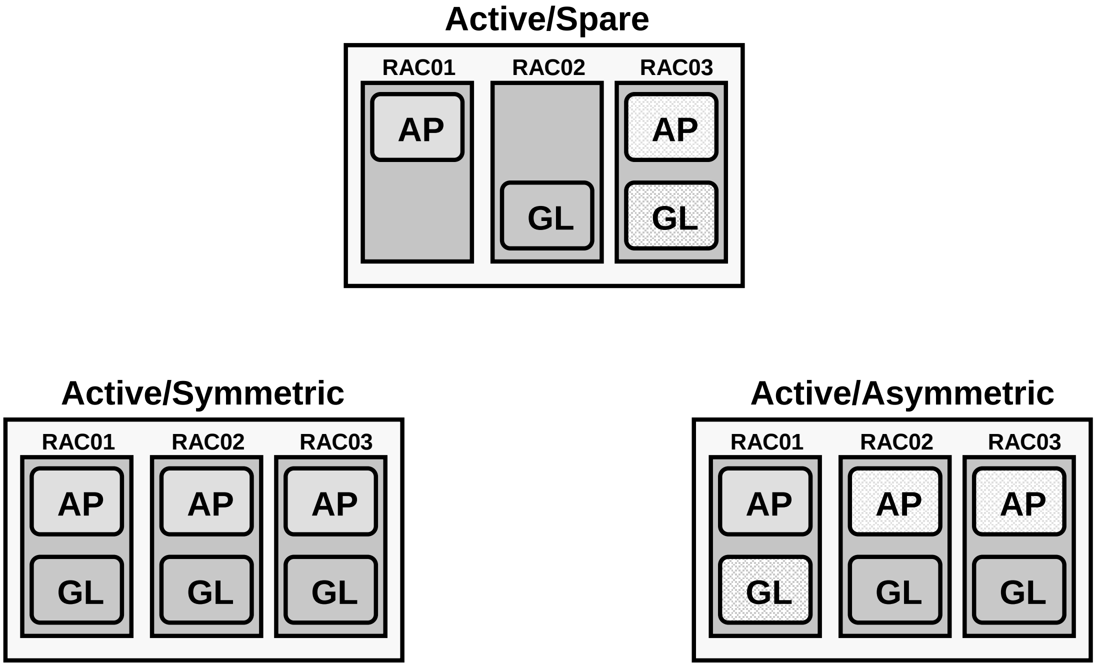
To implement the workload balancing and continuous availability features of services, CRS stores the HA configuration for each service in the Oracle Cluster Registry (OCR). The HA configuration defines a set of preferred and available instances that support the service.

A preferred instance set defines the number of instances (cardinality) that support the corresponding service. It also identifies every instance in the cluster that the service will run on when the system first starts up.

An available instance does not initially support a service. However, it begins accepting connections for the service when a preferred instance cannot support the service. If a preferred instance fails, then the service is transparently restored to an available instance defined for the service.

Note: An available instance can become a preferred instance and vice versa.

Possible Service Configuration with RAC



Active/Spare

RAC01 RAC02 RAC03

AP AP

GL GL

Active/Symmetric

RAC01 RAC02 RAC03

AP AP AP

GL GL GL

Active/Asymmetric

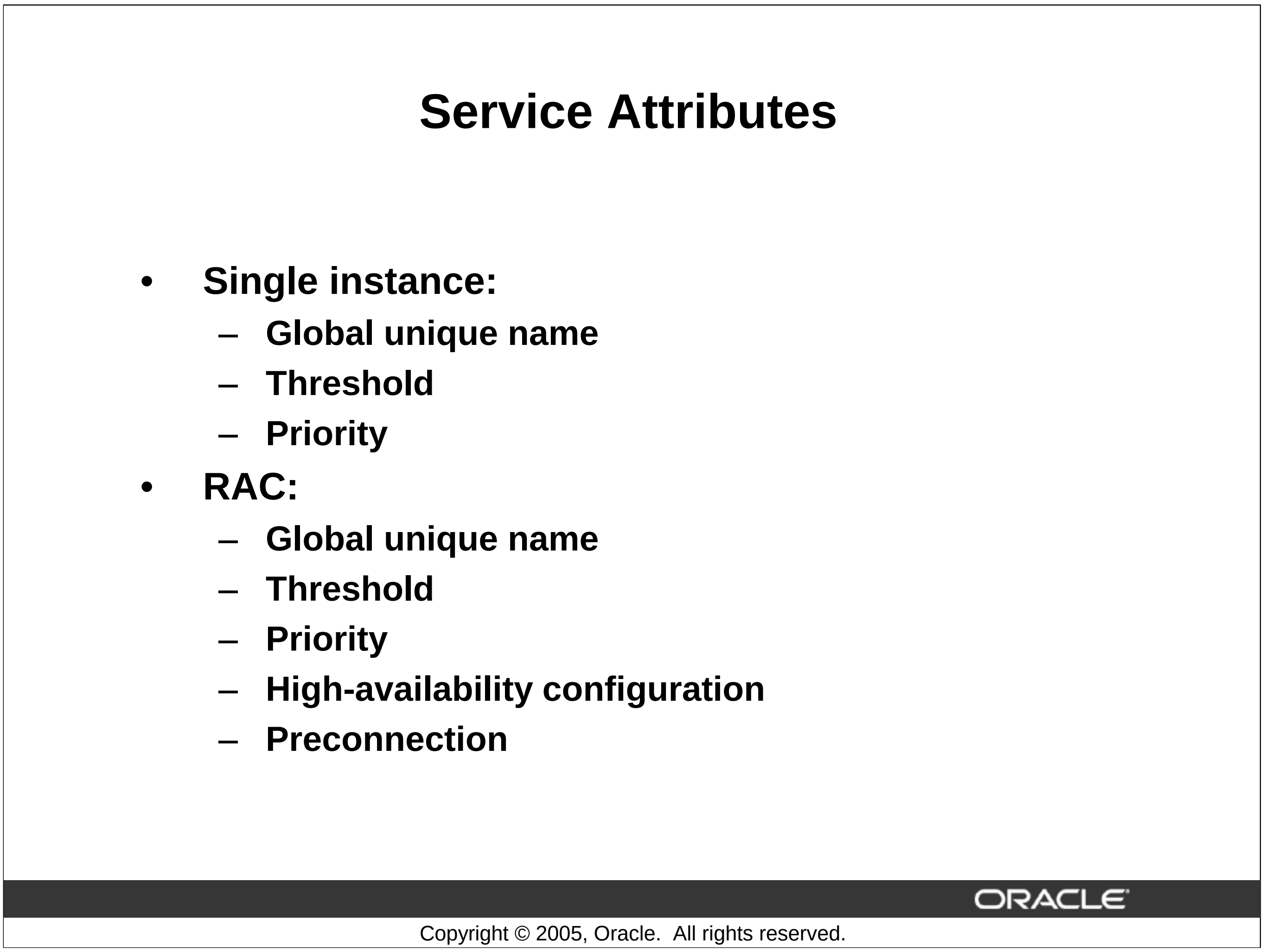
RAC01 RAC02 RAC03

AP AP AP

GL GL GL

Possible Service Configuration with RAC

- **Active/Spare:** With this service configuration, the simplest redundancy known as primary/secondary, or 1+1 redundancy is extended to the general case of N+M redundancy, where N is the number of primary RAC instances providing service, and M is the number of spare RAC instances available to provide the service. An example of this solution is a three-node configuration in which one instance provides the AP service; the second instance provides the GL service; the third instance provides service failover capability for both services. The spare node can still be available for other applications during normal operation.
- **Active/Symmetric:** With this service configuration, the same set of services is active on every instance. An example of this is illustrated in the slide, with both AP and GL services being offered on all three instances. Each instance provides service load-sharing and service failover capabilities for the other.
- **Active/Asymmetric:** With this service configuration, services with lower capacity needs can be defined with single cardinality and configured as having all other instances capable of providing the service in the event of failure. The slide shows the AP service running on only one instance, and the GL service running on two instances. The first instance supports the AP services and offers failover for the GL service. Likewise, the second and third instances support the GL service and offer failover for AP. If either the first and third instance dies, then GL and AP are still offered through the second instance.



Service Attributes

- Single instance:
  - Global unique name
  - Threshold
  - Priority
- RAC:
  - Global unique name
  - Threshold
  - Priority
  - High-availability configuration
  - Preconnection

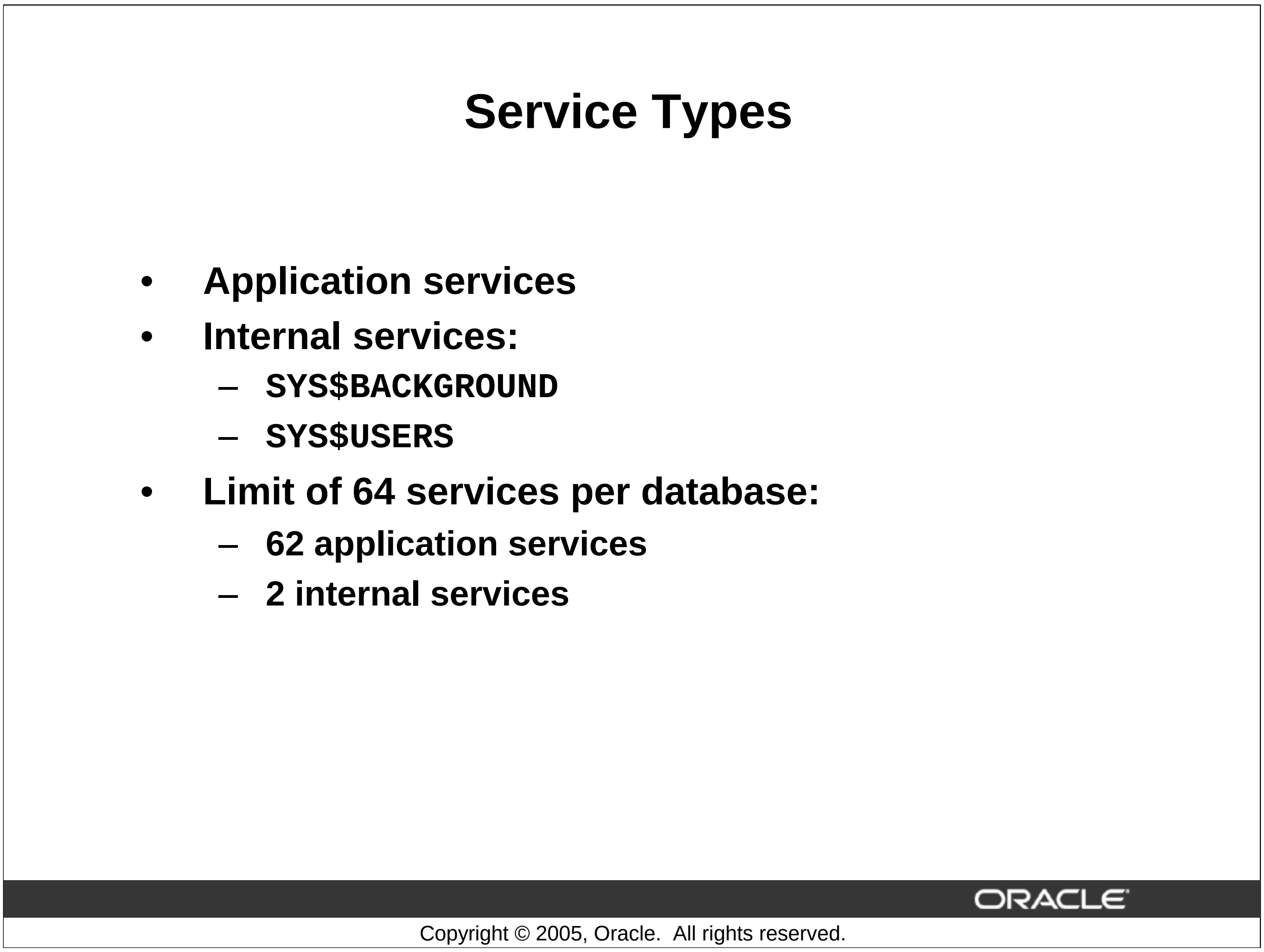
Service Attributes

Each service has the following attributes:

- Globally unique name that identifies the service in the local cluster and globally for data guard
- Quality of service thresholds for response time and CPU consumption
- Priority relative to other services, defined in terms of either ratio of resource consumption or priority

In a RAC environment, services have two additional attributes:

- High-availability configuration: A description of how to distribute the service across instances when the system first starts.
- Preconnection: Definition of a corresponding preconnected service, also called a shadow service. The preconnect service spans the set of instances that are *available* to support a service in the event of a failure. When a service is added by using the Database Configuration Assitant (DBCA) or Server Control (`SRVCTL`), the preconnect service is created automatically and is then managed by CRS. This service name `<SERVICE>_PRECONNECT` is used in the backup clause for transparent application failover (TAF) connect descriptors for directly connected applications that are using the preconnected TAF feature. Preconnect services are non-overlapping with their matching active services. This feature eliminates sessions looping back to the same instance as the original, and enables load balancing for both active and preconnected sessions with TAF.



Service Types

- Application services
- Internal services:
  - `SYS$BACKGROUND`
  - `SYS$USERS`
- Limit of 64 services per database:
  - 62 application services
  - 2 internal services

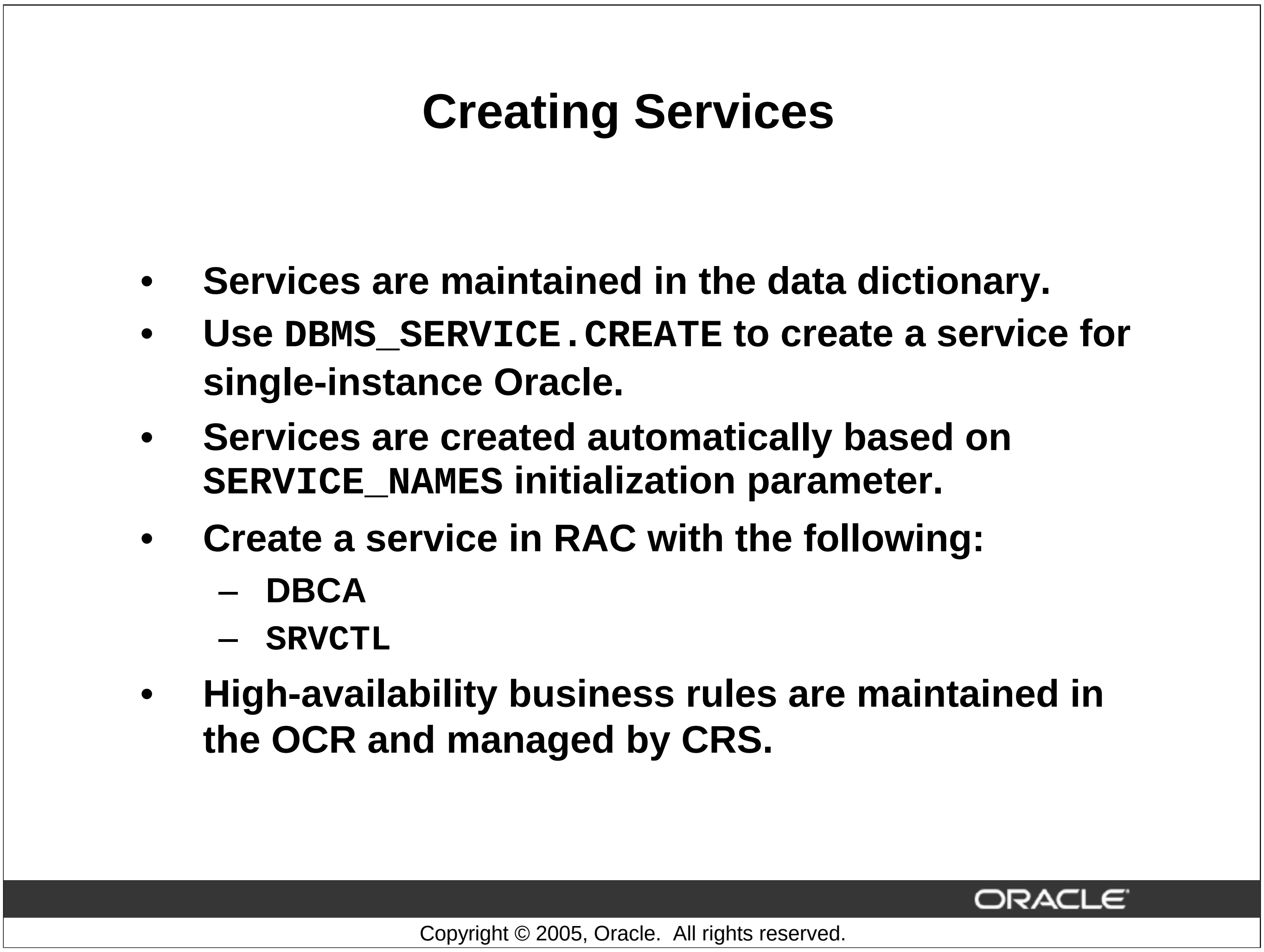
Service Types

Oracle Database 10*g* supports two broad types of services: application services and internal services. Application services are mainly functional maps to workloads. Sessions doing work for a common business function are grouped together. For Oracle Applications, AP, AR, GL, MFG, WIP, BOM, and so on create a functional division of work within the database and can thus be categorized as services.

In addition to application services, the RDBMS also supports two internal services. `SYS$BACKGROUND` is used by the background processes only. `SYS$USERS` is the default service for user sessions that are not associated with any application service. Both internal services support all the workload management features and neither one can be stopped or disabled.

There is a limitation of 64 services per database, 62 application services, and 2 internal services. Also, a service name is restricted to 64 characters.

Note: Shadow services are also included in the application service category. In addition, a service is also created for each Advanced Queue created.



Creating Services

- **Services are maintained in the data dictionary.**
- **Use `DBMS_SERVICE.CREATE` to create a service for single-instance Oracle.**
- **Services are created automatically based on `SERVICE_NAMES` initialization parameter.**
- **Create a service in RAC with the following:**
  - **DBCA**
  - **`SRVCTL`**
- **High-availability business rules are maintained in the OCR and managed by CRS.**

Creating Services

Like other database objects, services are maintained and tracked through the data dictionary and dynamic performance views.

Each service has a unique name that identifies it locally in the cluster and globally for Data Guard.

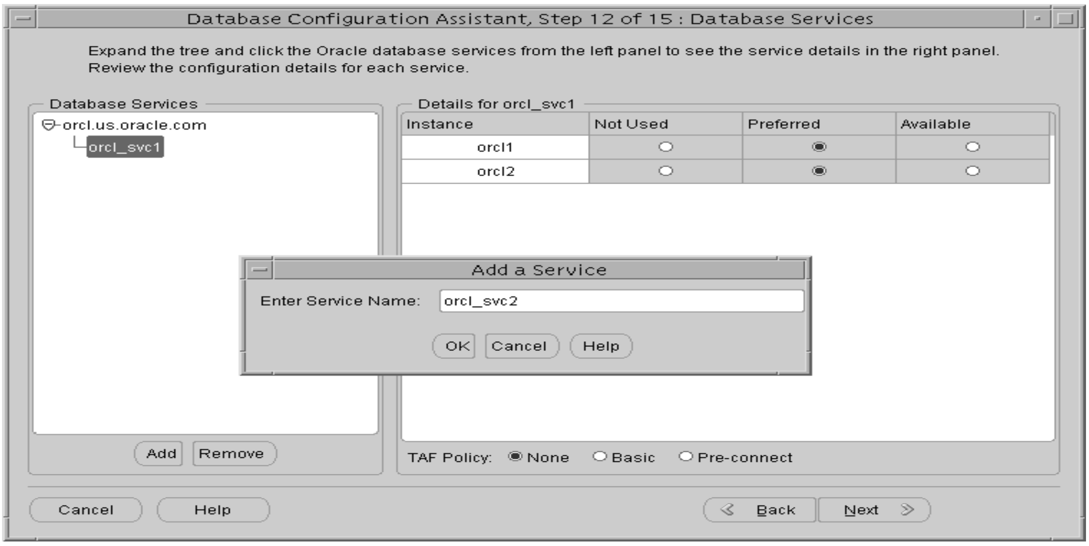
For single-instance Oracle, services can be created with the `DBMS_SERVICE` package.

Services are also created implicitly at startup of the instance according to the values set for the `SERVICE_NAMES` initialization parameter.

For high-availability features in RAC environments, services should be defined either by the DBCA, or by the command-line tool `SRVCTL`. This definition process implicitly creates high-availability business rules that are managed automatically by CRS to keep the services available. The high-availability business rules are kept in the OCR.

Creating Services with DBCA

DBCA configures both the CRS resources and the Net Service entries for each service.



Database Configuration Assistant, Step 12 of 15 : Database Services

Expand the tree and click the Oracle database services from the left panel to see the service details in the right panel. Review the configuration details for each service.

Database Services

- orcl.us.oracle.com
  - orcl_svc1

Details for orcl_svc1

Instance	Not Used	Preferred	Available
orcl1	○	●	○
orcl2	○	●	○

Add a Service

Enter Service Name: orcl_svc2

TAF Policy: ● None ○ Basic ○ Pre-connect

Creating Services with DBCA

The Database Configuration Assistant (DBCA) enables you to perform simple management operations on database services. When creating a RAC database by using DBCA, you can add and remove services, establish a preferred configuration, and set up a Transparent Application Failover (TAF) policy for your services.

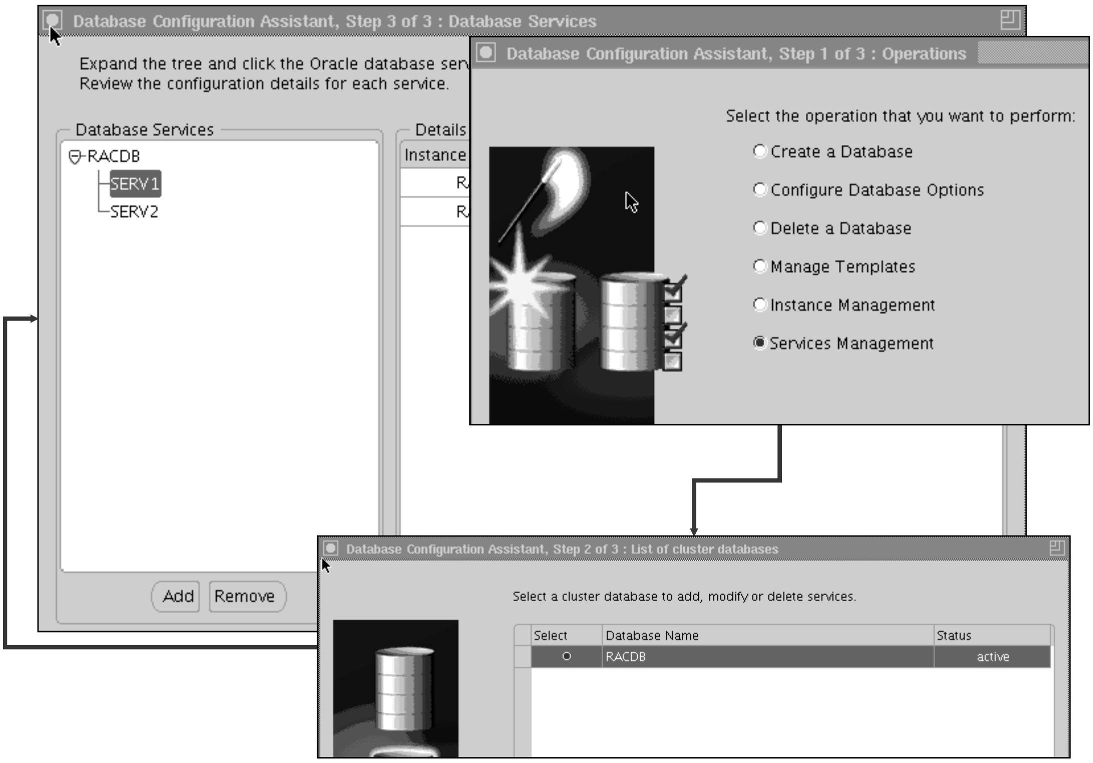
The DBCA lists the available instances for your RAC database. By clicking the appropriate option button, you can configure an instance as being preferred or available for a service. If you want to prevent a service from running on a specific instance, then click the option button in the Not Used column for the prohibited instance.

The entries you make in the Add a Service dialog box are appended to the `SERVICE_NAMES` parameter entry, which has a 4 KB limit. Therefore, the total length of the names of all services assigned to an instance cannot exceed 4 KB.

When you click Finish, the DBCA configures the CRS resources for the services that you added, modified, or removed. The DBCA also configures the net service entries for these services and starts them. When you use the DBCA to remove services, the DBCA stops the service, removes the CRS resource for the service, and removes the net service entries.

Note: You can also set up a service for transparent application failover using the TAF Policy section as shown in the slide above.

Creating Services with DBCA



Database Configuration Assistant, Step 3 of 3 : Database Services

Expand the tree and click the Oracle database ser
Review the configuration details for each service.

Database Services
⊖-RACDB
SERV1
SERV2
Details
Instance
Add
Remove

Database Configuration Assistant, Step 1 of 3 : Operations

Select the operation that you want to perform:

○ Create a Database
○ Configure Database Options
○ Delete a Database
○ Manage Templates
○ Instance Management
● Services Management

Database Configuration Assistant, Step 2 of 3 : List of cluster databases

Select a cluster database to add, modify or delete services.

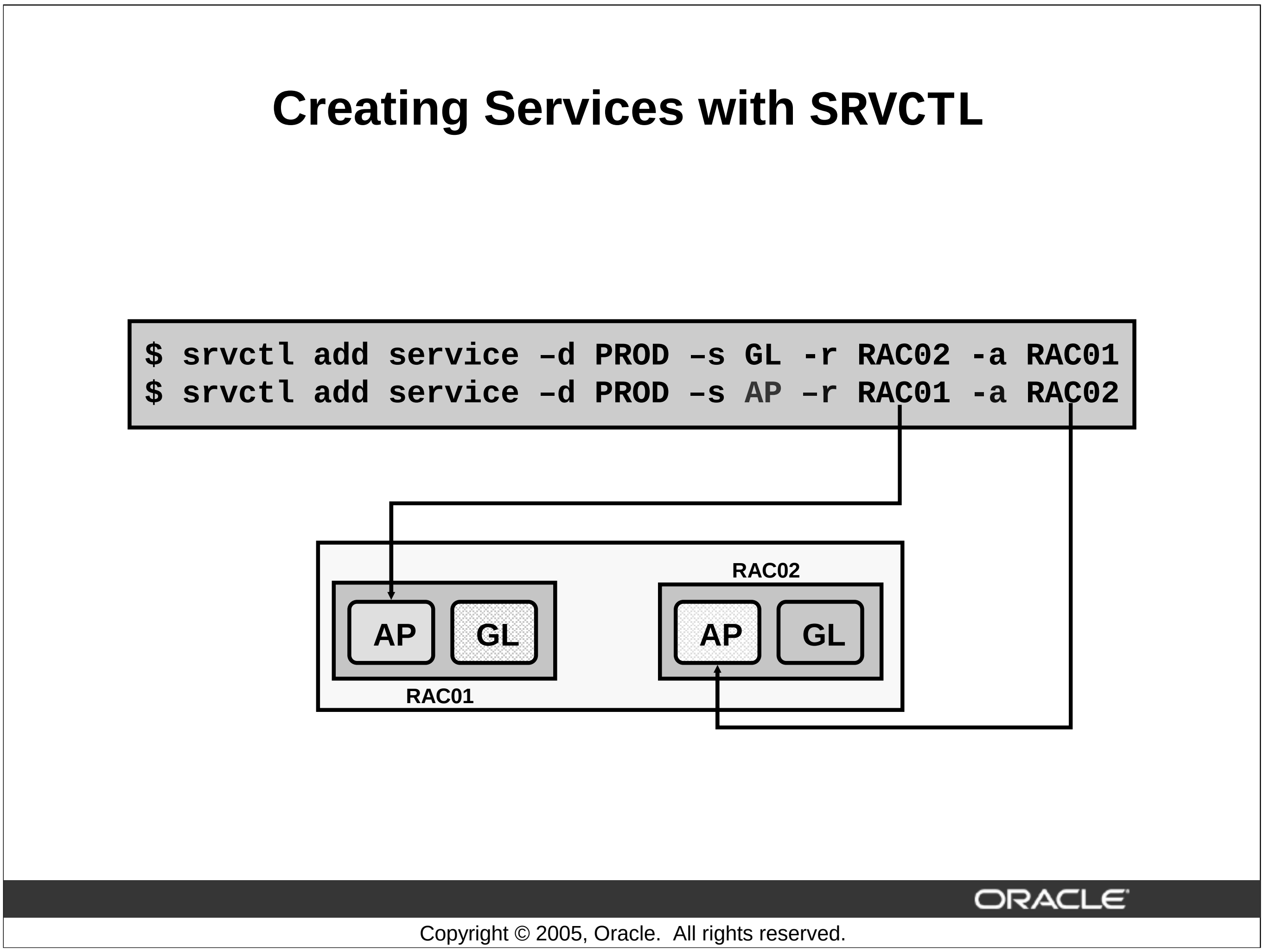
Select	Database Name	Status
○	RACDB	active

Creating Services with DBCA (continued)

The Database Configuration Assistant (DBCA) also enables you to perform simple management operations on database services after the database has been created.

You can do so by selecting the **Services Management** option in the first step. Then select the corresponding database in step two. In step three, you can add and remove services, establish a preferred configuration, and set up a transparent application failover (TAF) policy for your services.

Note: This page is identical to the **Database Services** page you see when creating a database.



Creating Services with SRVCTL

```
$ srvctl add service –d PROD –s GL -r RAC02 -a RAC01
$ srvctl add service –d PROD –s AP –r RAC01 -a RAC02
```

Creating Services with SRVCTL

The example in the slide shows a two-node cluster with an instance named RAC01 on one node and an instance called RAC02 on the other. The cluster database name is PROD.

Two services are created, AP and GL, and stored in the cluster repository to be managed by CRS. The AP service is defined with a preferred instance of RAC01 and an available instance of RAC02.

If RAC01 dies, the AP service member on RAC01 is restored automatically on RAC02. The same scenarios hold true for the GL service.

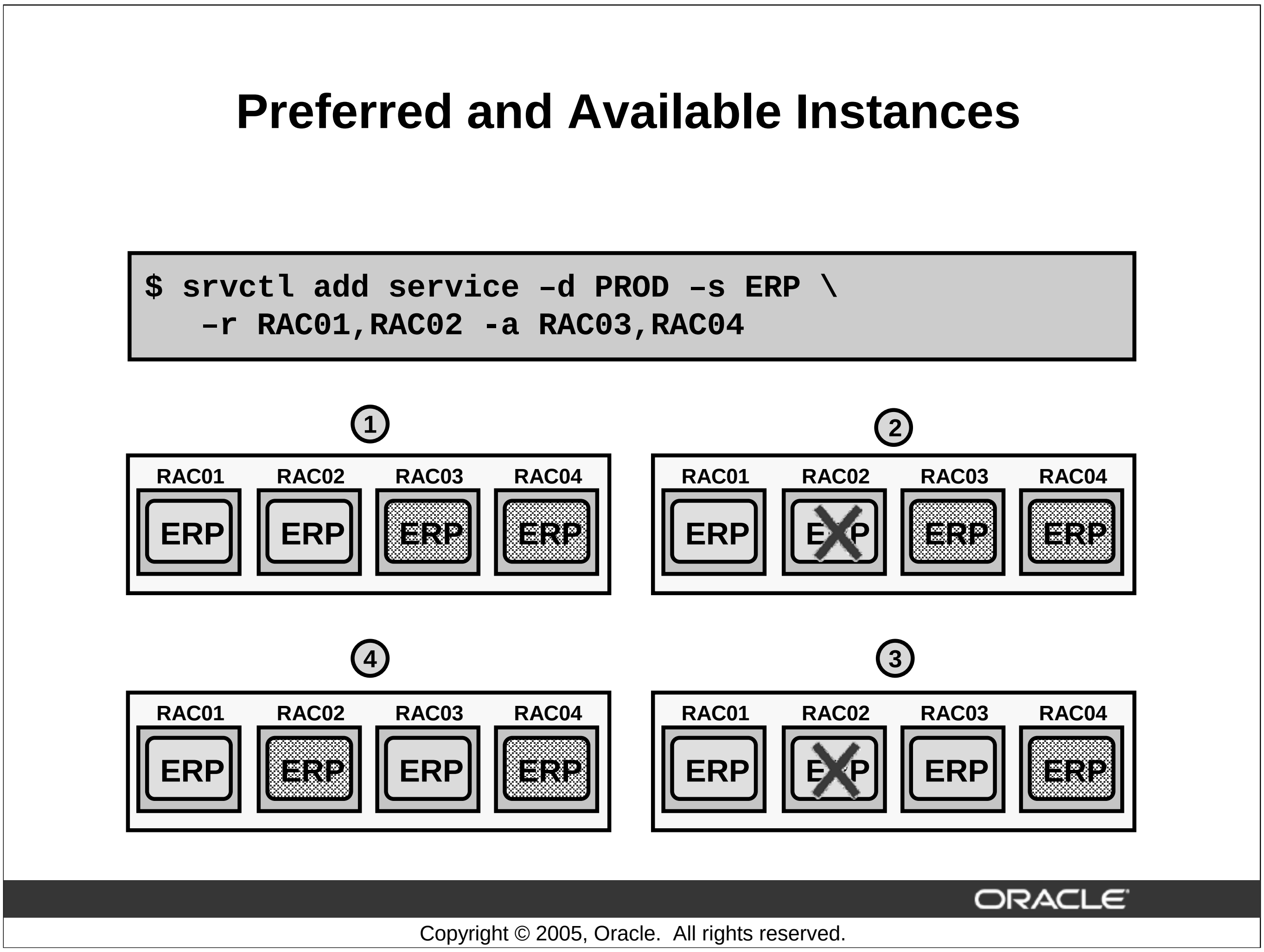
Note that it is possible to assign more than one instance with both the `-r` and `-a` options. However, `-r` is mandatory but `-a` is optional.

Services enable you to move beyond the simple two-node primary/secondary configuration of RAC Guard in Oracle9*i*.

With Oracle Database 10*g*, multiple primary nodes can support a service with RAC.

Possible configurations for service placement are active/spare, active/symmetric, and active/asymmetric.

Note: You can also set up a service for transparent application failover by using the `–P` option of `SRVCTL`. Possible values are `NONE`, `BASIC`, and `PRECONNECT`.



Preferred and Available Instances

```
$ srvctl add service –d PROD –s ERP \
   –r RAC01,RAC02 -a RAC03,RAC04
```

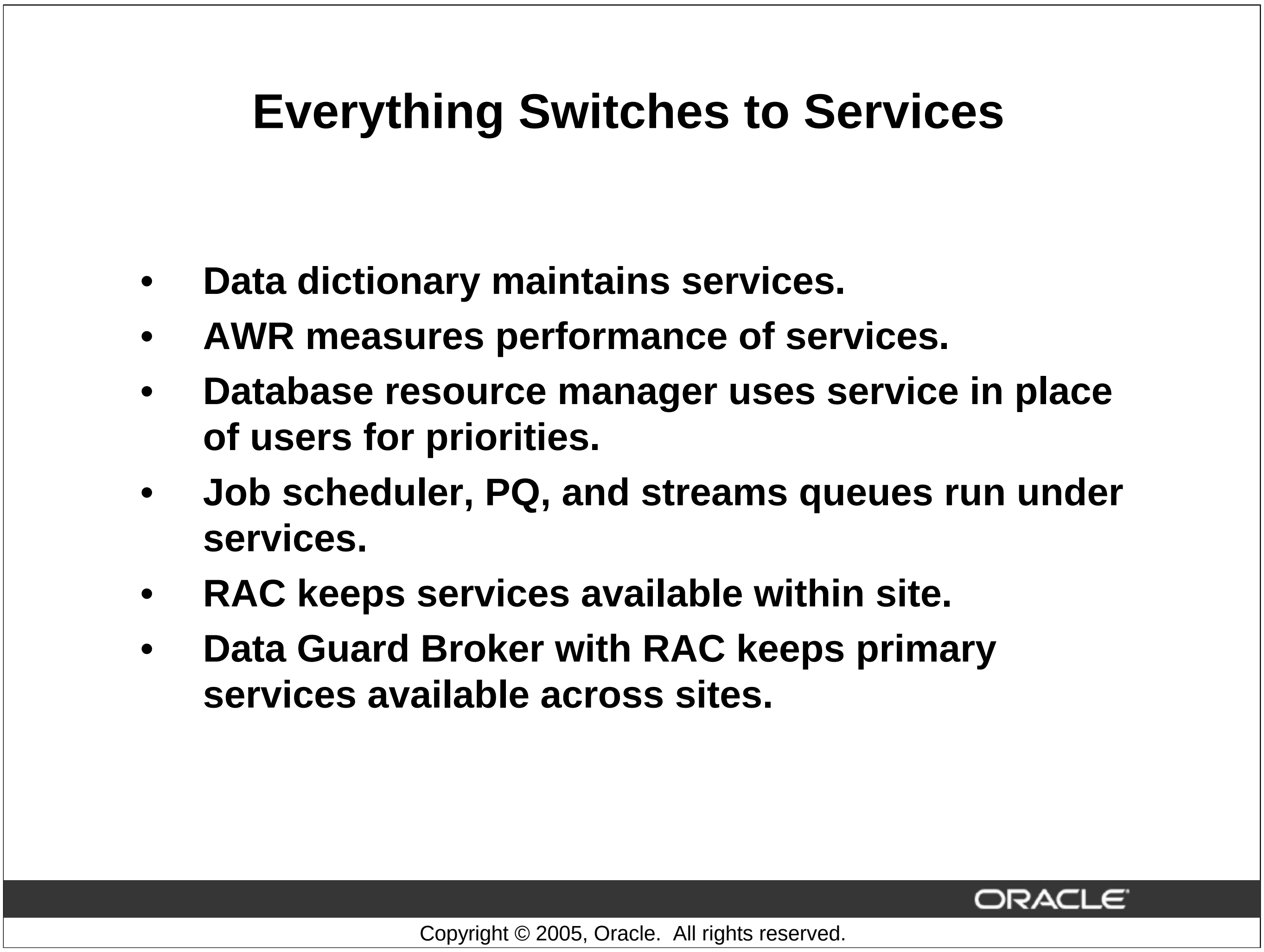
Preferred and Available Instances

In this example, it is assumed that you have a four-node cluster.

You define a service called ERP. The preferred instances for ERP are RAC01 and RAC02. The available instances for ERP are RAC03 and RAC04.

1. Initially, ERP connections are only directed to RAC01 and RAC02.
2. RAC02 is failing and goes down.
3. CRS detects the failure of RAC02, and because the cardinality of ERP is 2, CRS restores the service on one of the available instances, in this case RAC03.
4. ERP connection requests are now directed to RAC01 and RAC03, which are the instances that currently offer the service. Although CRS is able to restart RAC02, the ERP service does not fall back to RAC02. RAC02 and RAC04 are now the instances that are accessible if subsequent failures occur.

Note: If you want to fall back to RAC02 you can use `SRVCTL` to relocate the service. This operation can be done manually by the DBA, or by coding the SRVCTL relocation command using a call back mechanism to automate the fallback. However, relocating a service is a disruptive operation.



Everything Switches to Services

- Data dictionary maintains services.
- AWR measures performance of services.
- Database resource manager uses service in place of users for priorities.
- Job scheduler, PQ, and streams queues run under services.
- RAC keeps services available within site.
- Data Guard Broker with RAC keeps primary services available across sites.

Everything Switches to Services

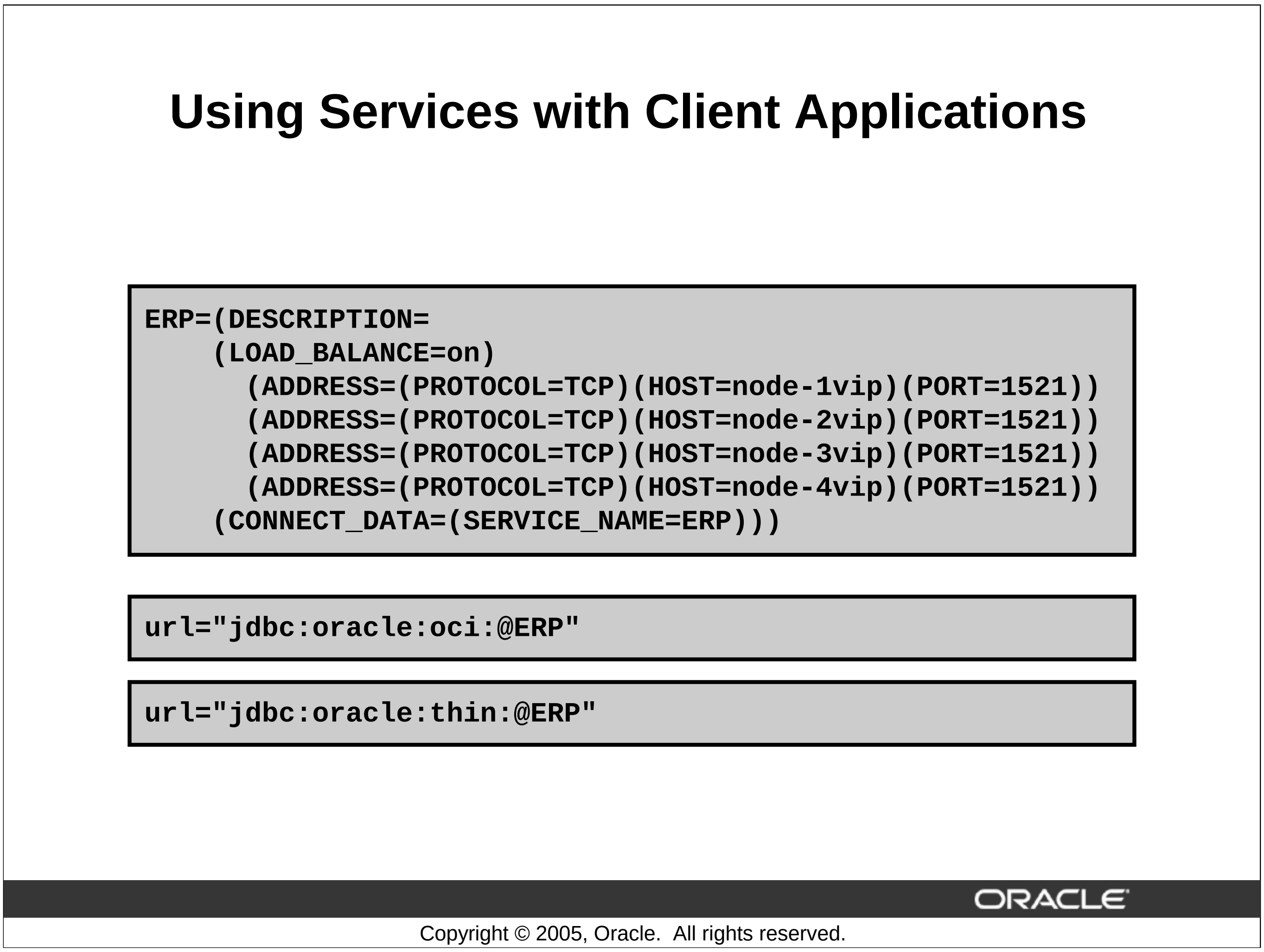
Several database features support services. Sessions are tracked by the service with which they connect. In addition, performance-related statistics and wait events are also tracked by service.

The Automatic Workload Repository (AWR) manages the performance of services. The AWR records the service performance, including SQL execution times, wait classes, and resources consumed by service. The AWR alerts when service response time thresholds are exceeded. Specific dynamic performance views report current service status with one hour of history.

The Database Resource Manager is now capable of managing services for prioritizing application workloads within an instance. In addition, jobs can now run under a service, as opposed to a specific instance. Parallel slave processes inherit the service of their coordinator.

The RAC High Availability framework keeps services available within a site. Data Guard Broker, in conjunction with RAC, migrates the primary service across Data Guard sites for disaster tolerance.

Note: For more information about RAC and Data Guard Broker integration, refer to the lesson titled “Design for High Availability” in this course.



Using Services with Client Applications

```
ERP=(DESCRIPTION=
    (LOAD_BALANCE=on)
      (ADDRESS=(PROTOCOL=TCP)(HOST=node-1vip)(PORT=1521))
      (ADDRESS=(PROTOCOL=TCP)(HOST=node-2vip)(PORT=1521))
      (ADDRESS=(PROTOCOL=TCP)(HOST=node-3vip)(PORT=1521))
      (ADDRESS=(PROTOCOL=TCP)(HOST=node-4vip)(PORT=1521))
    (CONNECT_DATA=(SERVICE_NAME=ERP)))
```

```
url="jdbc:oracle:oci:@ERP"
```

```
url="jdbc:oracle:thin:@ERP"
```

Using Services with Client Applications

Applications and mid-tier connection pools select a service by using the TNS connection descriptor.

The service must match the service that has been created using `SRVCTL` or DBCA.

The address lists in each example use virtual IP addresses. The address lists must not use hostnames. Using the virtual addresses for client communication ensures that connections and SQL statements issued against a node that is down do not result in a TCP/IP timeout.

The first example on the slide above shows the TNS connect descriptor that could be used to access the ERP service.

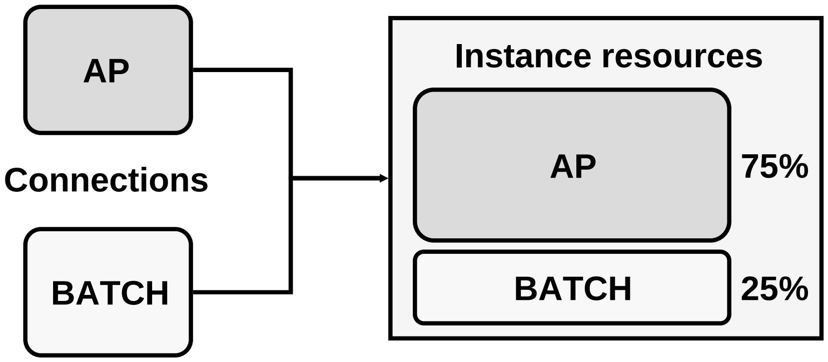
The second example shows the thick JDBC connection description using the previously defined TNS connect descriptor.

The third example shows the thin JDBC connection description using the previously defined TNS connect descriptor.

Note: The `LOAD_BALANCE=ON` clause is used by Oracle Net to randomize its progress through the protocol addresses of the connect descriptor. This feature is called client load balancing.

Using Services with Resource Manager

- Consumer groups are automatically assigned to sessions based on session services.
- Work is prioritized by service inside one instance.



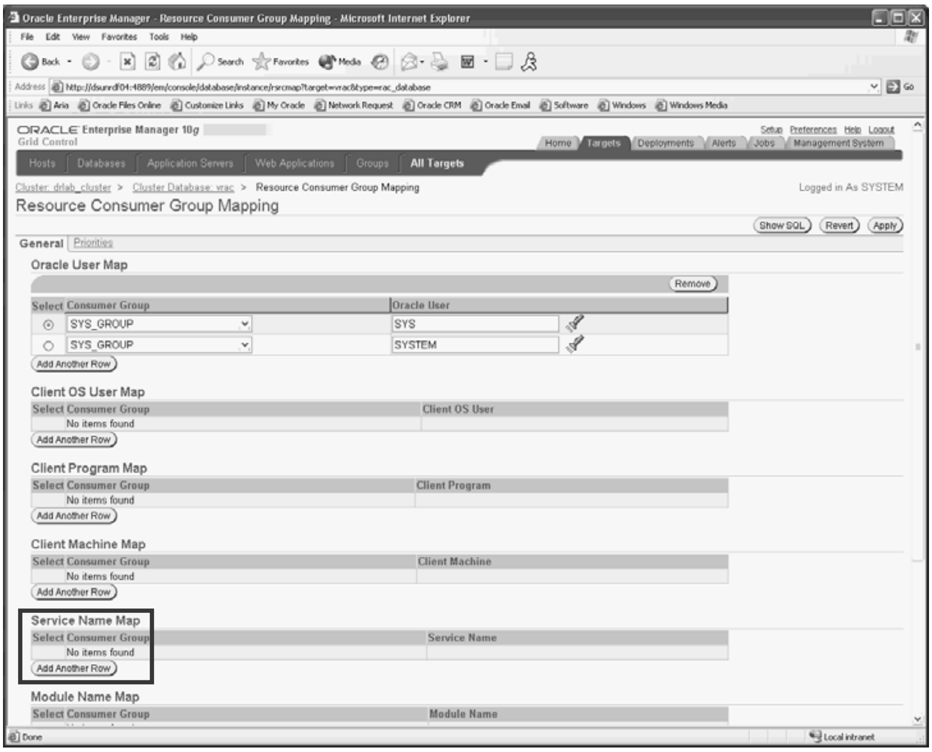
Using Services with Resource Manager

The Database Resource Manager enables you to identify work by using services. It manages the relative priority of services within an instance by binding services directly to consumer groups. When a client connects by using a service, the consumer group is assigned transparently at connect time. This enables Resource Manager to manage the work requests by service in the order of their importance.

For example, you define the AP and BATCH services to run on the same instance, and assign AP to a high-priority consumer group and BATCH to a low-priority consumer group. Sessions that connect to the database with the AP service specified in their TNS connect descriptor get priority over those that connect to the BATCH service.

This offers benefits in managing workloads because priority is given to business functions rather than the sessions that support those business functions.

Services and Resource Manager with EM


Oracle Enterprise Manager - Resource Consumer Group Mapping - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://dsunrdf04:4889/em/console/database/instance/rsrcmap?target=vrac&type=rac_database
Links Aria Oracle Files Online Customize Links My Oracle Network Request Oracle CRM Oracle Email Software Windows Windows Media
ORACLE Enterprise Manager 10g
Grid Control
Setup Preferences Help Logout
Home Targets Deployments Alerts Jobs Management System
Hosts Databases Application Servers Web Applications Groups All Targets
Cluster: drlab_cluster > Cluster Database: vrac > Resource Consumer Group Mapping
Logged in As SYSTEM
Show SQL Revert Apply
General Priorities
Remove
Add Another Row
Done
Local intranet

Resource Consumer Group Mapping

Oracle User Map

Select	Consumer Group	Oracle User
●	SYS_GROUP	SYS
○	SYS_GROUP	SYSTEM

Client OS User Map

Select	Consumer Group	Client OS User
	No items found	

Client Program Map

Select	Consumer Group	Client Program
	No items found	

Client Machine Map

Select	Consumer Group	Client Machine
	No items found	

Service Name Map

Select	Consumer Group	Service Name
	No items found	

Module Name Map

Select	Consumer Group	Module Name

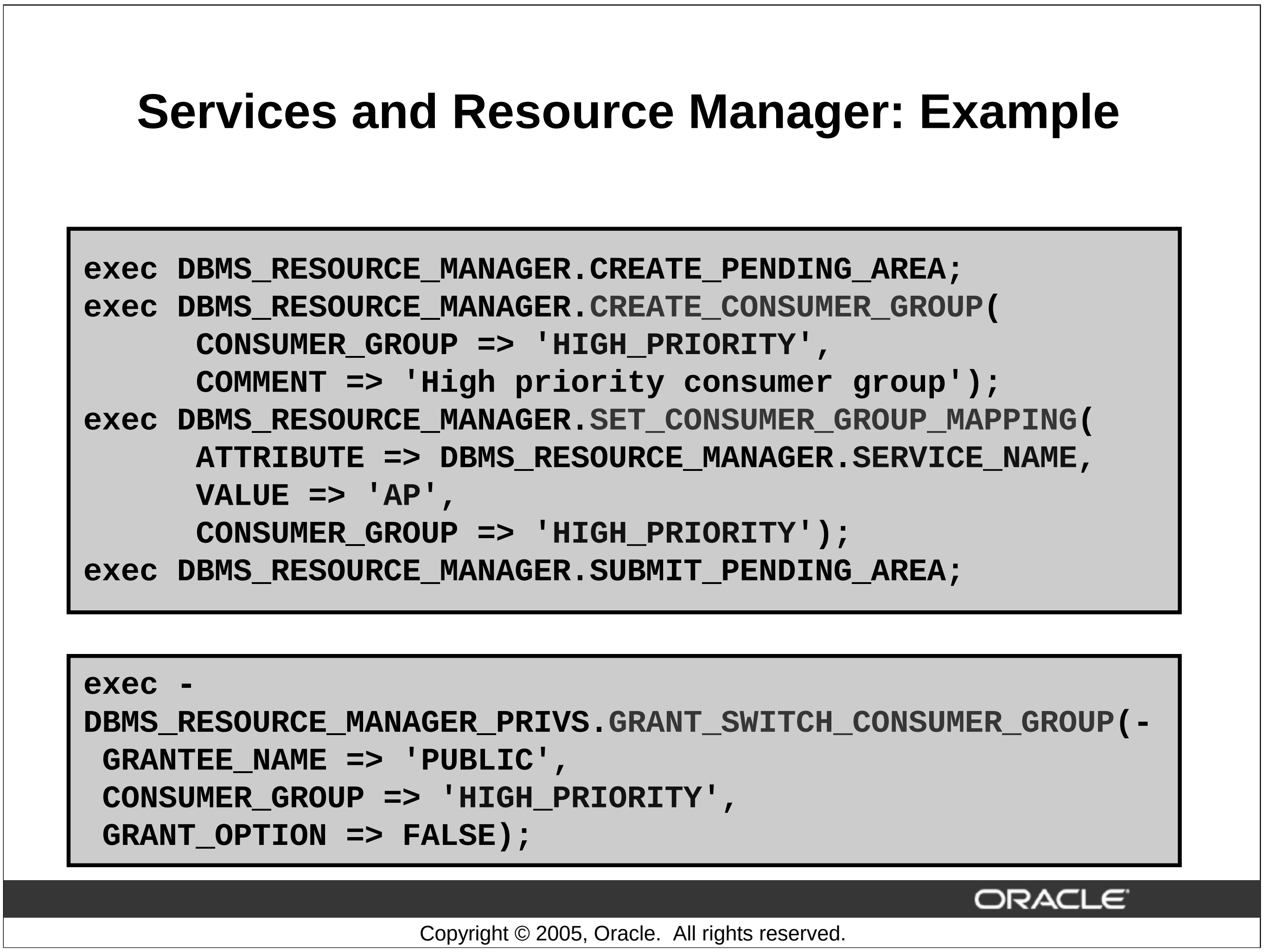
Services and Resource Manager with EM

Enterprise Manager (EM) gives you a GUI interface through the Resource Consumer Group Mapping page to automatically map sessions to consumer groups. This page can be accessed by clicking the Resource Consumer Group Mappings link on the Cluster Database Administration page.

Using the General page of this page, you can set up a mapping of sessions connecting with a service name to consumer groups. At the bottom of the page (not visible in this screenshot), there is an option for a module name and action mapping.

With the ability to map sessions to consumer groups by service, module, and action, you have greater flexibility when it comes to managing the performance of different application workloads.

Note: The Priorities page of the Resource Consumer Group Mapping page allows you to set priorities for the mappings that you set up on the General page. The mapping options correspond to columns in `V$SESSION`. When multiple mapping columns have values, the priorities you set determine the precedence for assigning sessions to consumer groups.



Services and Resource Manager: Example

```
exec DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA;
exec DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP(
      CONSUMER_GROUP => 'HIGH_PRIORITY',
      COMMENT => 'High priority consumer group');
exec DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING(
      ATTRIBUTE => DBMS_RESOURCE_MANAGER.SERVICE_NAME,
      VALUE => 'AP',
      CONSUMER_GROUP => 'HIGH_PRIORITY');
exec DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA;
```

```
exec -
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER_GROUP(-
 GRANTEE_NAME => 'PUBLIC',
 CONSUMER_GROUP => 'HIGH_PRIORITY',
 GRANT_OPTION => FALSE);
```

Services and Resource Manager: Example

Assume that your site has two consumer groups called `HIGH_PRIORITY` and `LOW_PRIORITY`. These consumer groups map to a resource plan for the database that reflects either the intended ratios or the intended resource consumption.

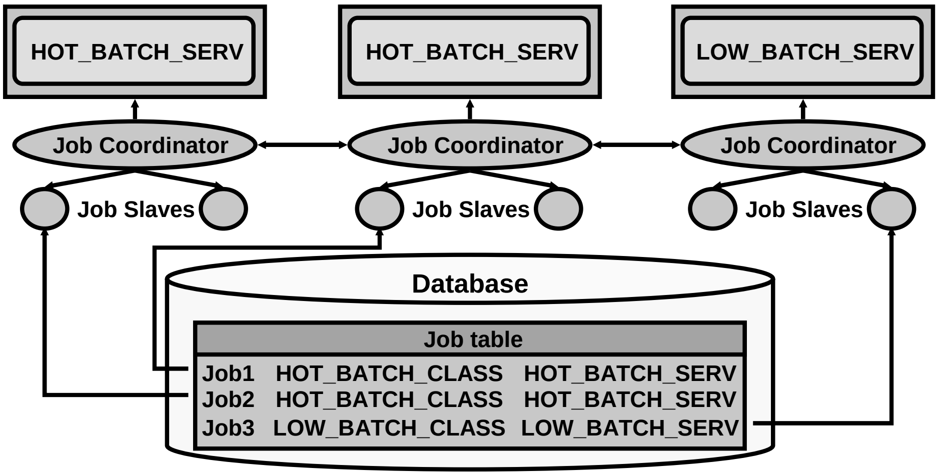
Before mapping services to consumer groups, you must first create the consumer groups and the resource plan for these consumer groups. The resource plan can be priority based or ratio based. The above PL/SQL calls are used to create the `HIGH_PRIORITY` consumer group, and map the AP service to the `HIGH_PRIORITY` consumer group. You can use similar calls to create the `LOW_PRIORITY` consumer groups and map the BATCH service to the `LOW_PRIORITY` consumer group.

The last PL/SQL call in the example in the slide above is executed because sessions are automatically assigned only to consumer groups for which they have been granted switch privileges. A similar call should be executed for the `LOW_PRIORITY` consumer group.

Note: For more information about Database Resource Manager, refer to the *Oracle Database Administrator's Guide and PL/SQL Packages and Types Reference*.

Using Services with Scheduler

- Services are associated with Scheduler classes.
- Scheduler jobs have service affinity:
  - High availability
  - Load balancing



Job table		
Job1	HOT_BATCH_CLASS	HOT_BATCH_SERV
Job2	HOT_BATCH_CLASS	HOT_BATCH_SERV
Job3	LOW_BATCH_CLASS	LOW_BATCH_SERV

Using Services with Scheduler

Just as in other environments, the Scheduler in a RAC environment uses one job table for each database and one job coordinator for each instance. The job coordinators communicate with each other to keep information current. Each instance's job coordinator exchanges information with the others.

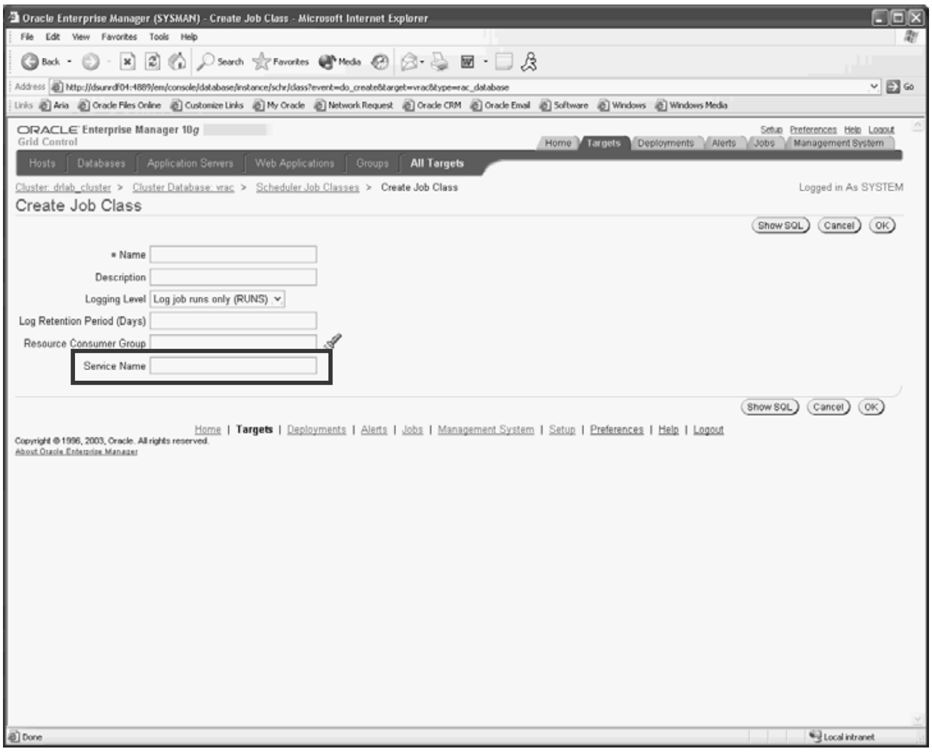
The Scheduler can use the services and the benefits they offer in a RAC environment. The service that a specific job class uses is defined when the job class is created. During execution, jobs are assigned to job classes and job classes run within services. Using services with job classes ensures that the work of the Scheduler is identified for workload management and performance tuning.

For example, jobs inherit server-generated alerts and performance thresholds for the service they run under.

For high availability, the Scheduler offers service affinity instead of instance affinity. Jobs are not scheduled to run on any specific instance. They are scheduled to run under a service. So, if an instance dies, the job can still run on any other instance in the cluster that offers the service.

Note: By specifying the service where you want the jobs to run, the job coordinators balance the load on your system for better performance.

Services and Scheduler with EM



Oracle Enterprise Manager (SYSMAN) - Create Job Class - Microsoft Internet Explorer

Address http://dsunrdf04:4889/em/console/database/instance/schr/class?event=do_create&target=vrac&type=rac_database

Cluster: drlab_cluster > Cluster Database: vrac > Scheduler Job Classes > Create Job Class

Logged in As SYSTEM

Create Job Class

* Name	
Description	
Logging Level	Log job runs only (RUNS)
Log Retention Period (Days)	
Resource Consumer Group	
Service Name	

Copyright © 1996, 2003, Oracle. All rights reserved.

About Oracle Enterprise Manager

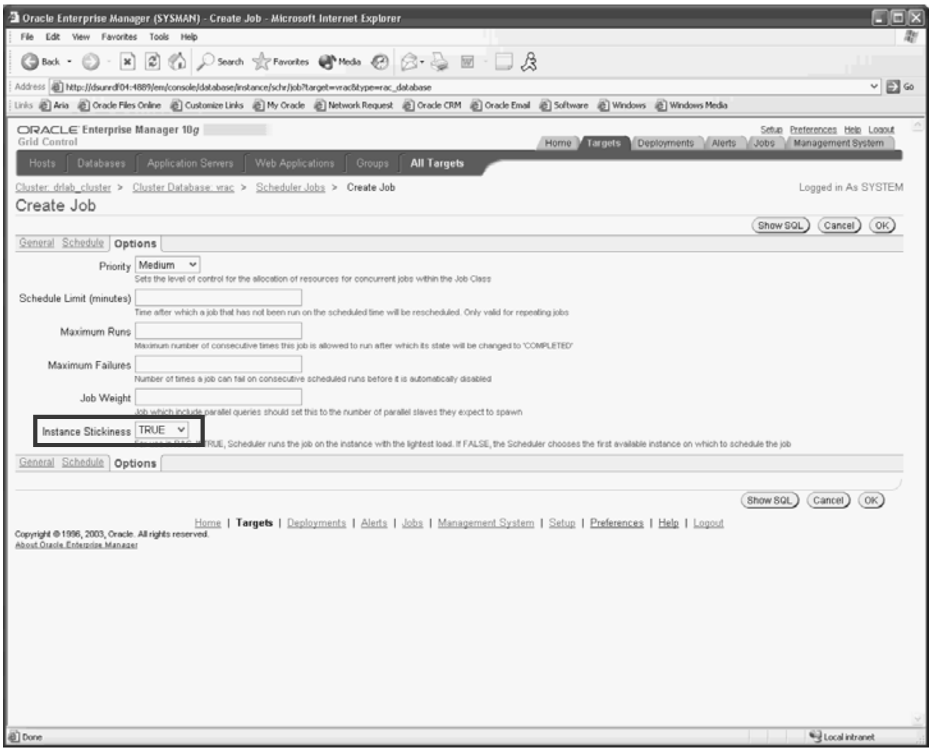
File Edit View Favorites Tools Help
Back Search Favorites Media
Links Aria Oracle Files Online Customize Links My Oracle Network Request Oracle CRM Oracle Email Software Windows Windows Media
ORACLE Enterprise Manager 10g
Grid Control
Setup Preferences Help Logout
Home Targets Deployments Alerts Jobs Management System
Hosts Databases Application Servers Web Applications Groups All Targets
Show SQL Cancel OK
Home | Targets | Deployments | Alerts | Jobs | Management System | Setup | Preferences | Help | Logout
Done
Local intranet

Services and Scheduler with EM

To configure a job to run under a specific service, click the Job Classes link under the Scheduler section of the Cluster Database Administration page. That takes you to the Scheduler Job Class page. On the Scheduler Job Class page, you can see services assigned to job classes.

When you click the Create button on the Scheduler Job Classes page, you access the Create Job Class page. On this page, you can enter details of a new job class, including what service it must run under.

Services and Scheduler with EM



Oracle Enterprise Manager (SYSMAN) - Create Job - Microsoft Internet Explorer

Address: http://dsunrdf04:4889/em/console/database/instance/schr/job?target=vrac&type=rac_database

Cluster: drlab_cluster > Cluster Database: vrac > Scheduler Jobs > Create Job

Logged in As SYSTEM

Create Job

Priority	Medium Sets the level of control for the allocation of resources for concurrent jobs within the Job Class
Schedule Limit (minutes)	 Time after which a job that has not been run on the scheduled time will be rescheduled. Only valid for repeating jobs
Maximum Runs	 Maximum number of consecutive times this job is allowed to run after which its state will be changed to 'COMPLETED'
Maximum Failures	 Number of times a job can fail on consecutive scheduled runs before it is automatically disabled
Job Weight	 Job which include parallel queries should set this to the number of parallel slaves they expect to spawn
Instance Stickiness	TRUE For use in RAC. If TRUE, Scheduler runs the job on the instance with the lightest load. If FALSE, the Scheduler chooses the first available instance on which to schedule the job

Copyright © 1996, 2003, Oracle. All rights reserved.

File Edit View Favorites Tools Help
Back Search Favorites Media
Links Aria Oracle Files Online Customize Links My Oracle Network Request Oracle CRM Oracle Email Software Windows Windows Media
ORACLE Enterprise Manager 10g
Grid Control
Setup Preferences Help Logout
Home Targets Deployments Alerts Jobs Management System
Hosts Databases Application Servers Web Applications Groups All Targets
Show SQL Cancel OK
General Schedule Options
Home | Targets | Deployments | Alerts | Jobs | Management System | Setup | Preferences | Help | Logout
About Oracle Enterprise Manager
Done
Local intranet

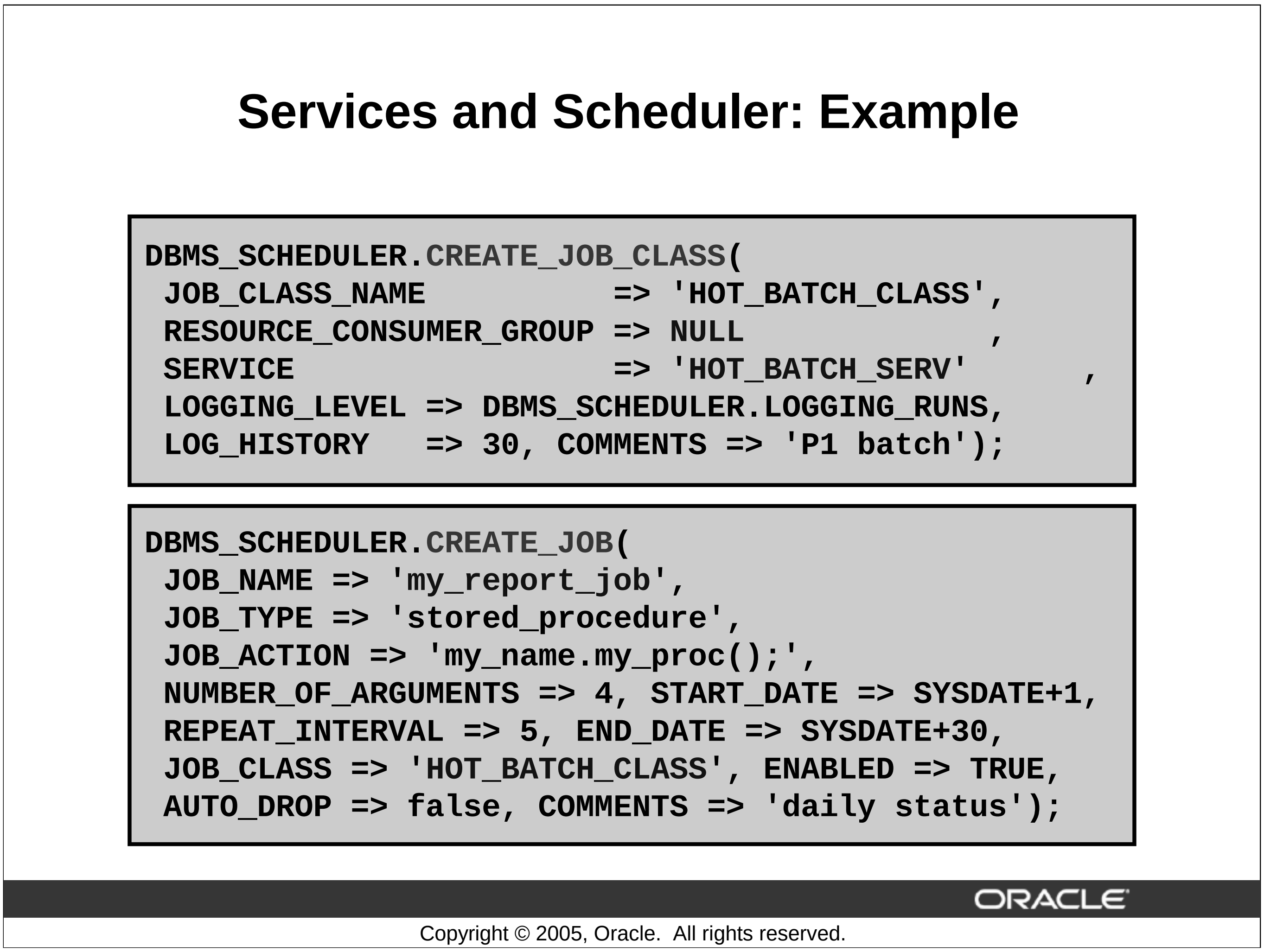
Services and Scheduler with EM (continued)

After your job class is set up with the service that you want it to run under, you can create the job.

To create the job, click the Jobs link just above the Job Classes link on the Cluster Database Administration page. The Scheduler Jobs page appears where you can click the Create button to create a new job. When you click the Create button, the Create Job page is displayed. This page has different pages, represented by the General, Schedule, and Options tabs. Use the General page to assign your job to a job class.

Use the Options page displayed in the slide above to set the Instance Stickiness attribute for your job. Basically, this attribute causes the job to be load balanced across the instances for which the service of the job is running. The job can run only on one instance. If the Instance Stickiness value is set to `TRUE`, which is the default value, the Scheduler runs the job on the instance where the service is offered with the lightest load. If Instance Stickiness is set to `FALSE`, then the job is run on the first available instance where the service is offered.

Note: It is possible to set jobs attributes, such as `INSTANCE_STICKINESS`, by using the `SET_ATTRIBUTE` procedure of the `DBMS_SCHEDULER` PL/SQL package.



Services and Scheduler: Example

```
DBMS_SCHEDULER.CREATE_JOB_CLASS(
 JOB_CLASS_NAME          => 'HOT_BATCH_CLASS',
 RESOURCE_CONSUMER_GROUP => NULL            ,
 SERVICE                 => 'HOT_BATCH_SERV'      ,
 LOGGING_LEVEL => DBMS_SCHEDULER.LOGGING_RUNS,
 LOG_HISTORY   => 30, COMMENTS => 'P1 batch');
```

```
DBMS_SCHEDULER.CREATE_JOB(
 JOB_NAME => 'my_report_job',
 JOB_TYPE => 'stored_procedure',
 JOB_ACTION => 'my_name.my_proc();',
 NUMBER_OF_ARGUMENTS => 4, START_DATE => SYSDATE+1,
 REPEAT_INTERVAL => 5, END_DATE => SYSDATE+30,
 JOB_CLASS => 'HOT_BATCH_CLASS', ENABLED => TRUE,
 AUTO_DROP => false, COMMENTS => 'daily status');
```

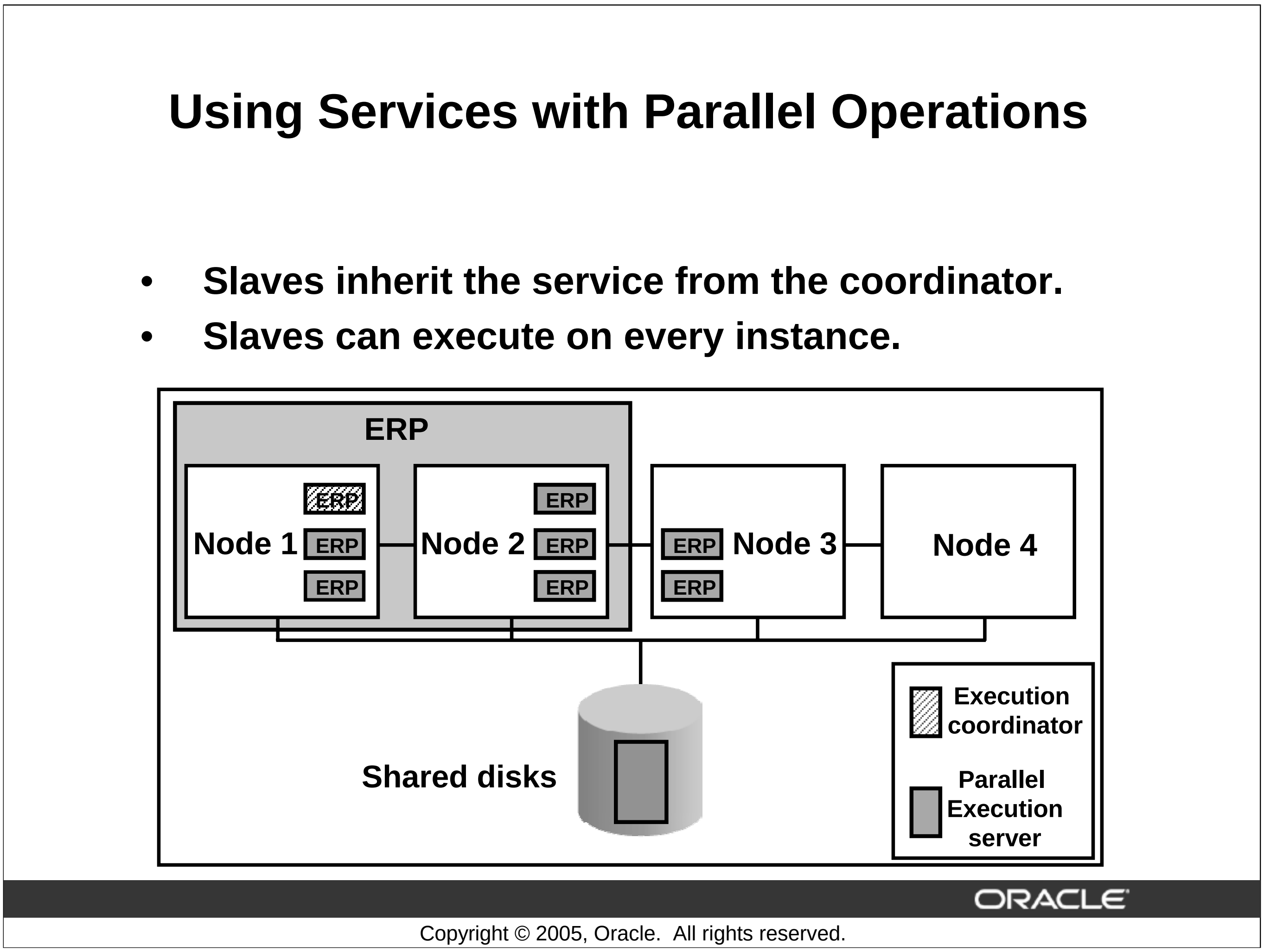
Services and Scheduler: Example

In this PL/SQL example, you define a batch queue managed by the Scheduler called HOT_BATCH_CLASS. You associate the HOT_BATCH_SERV service to the HOT_BATCH_CLASS queue. It is assumed that you had already defined the HOT_BATCH_SERV service.

After the class is defined, you can define your job. In this example, the MY_REPORT_JOB job executes in the HOT_BATCH_CLASS job class at instances offering the HOT_BATCH_SERV service.

In this example, you do not assign a resource consumer group to the HOT_BATCH_CLASS job class. However, it is possible to assign a consumer group to a class. Regarding services, this allows you to combine Scheduler jobs and service prioritization using Database Resource Manager.

Note: For more information about Scheduler, refer to the *Oracle Database Administrator's Guide* and *PL/SQL Packages and Types Reference*.



Using Services with Parallel Operations

- Slaves inherit the service from the coordinator.
- Slaves can execute on every instance.

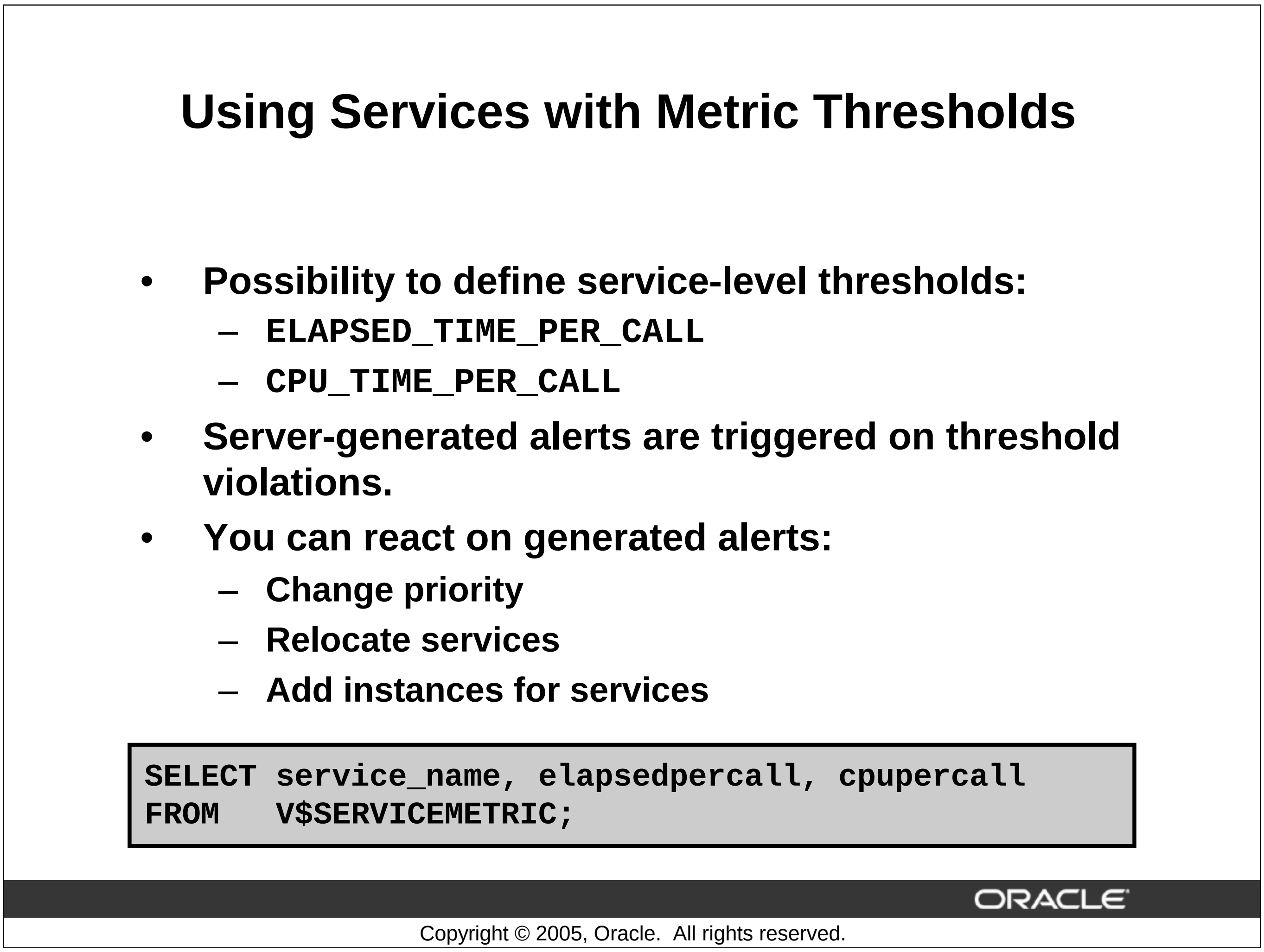
Using Services with Parallel Operations

For parallel query and parallel DML operations, the parallel query slaves inherit the service from the query coordinator for the duration of the operation. ERP is the name of the service used by the example shown on the slide.

However, services currently do not restrict the set of instances that are used by a parallel query. Connecting via a service and then issuing a parallel query may use instances that are not part of the service that was specified during the connection.

A slave appears to belong under the service even on an instance that does not support the service, if that slave is being used by a query coordinator that was started on an instance that does support that service.

Note: At the end of the execution, the slaves revert to the default database service.



Using Services with Metric Thresholds

- Possibility to define service-level thresholds:
  - `ELAPSED_TIME_PER_CALL`
  - `CPU_TIME_PER_CALL`
- Server-generated alerts are triggered on threshold violations.
- You can react on generated alerts:
  - Change priority
  - Relocate services
  - Add instances for services

```
SELECT service_name, elapsedpercall, cpupercall
FROM   V$SERVICEMETRIC;
```

Using Services with Metric Thresholds

Service-level thresholds permit the comparison of achieved service levels against accepted minimum required level. This provides accountability with respect to delivery or failure to deliver an agreed service level.

You can specify explicitly two metric thresholds for each service on a particular instance:

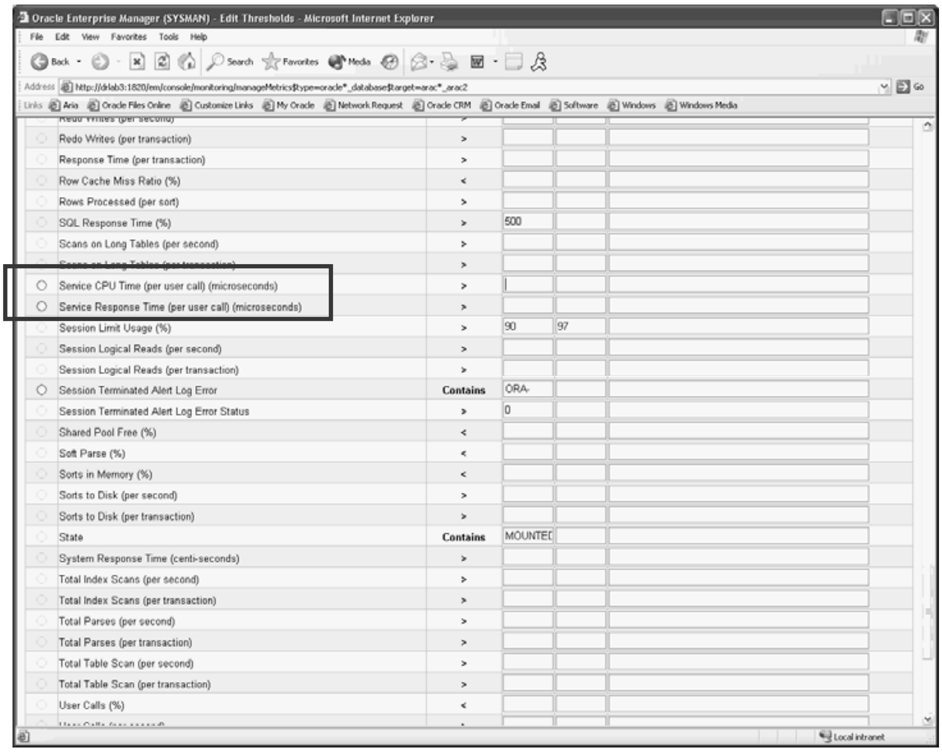
- The response time for calls: `ELAPSED_TIME_PER_CALL`. The response time goal indicates a desire for the elapsed time to be, at most, a certain value. The response time represents the wall clock time. It is a fundamental measure that reflects all delays and faults blocking the call from running on behalf of the user.
- CPU time for calls: `CPU_TIME_PER_CALL`

The AWR monitors the service time and publishes AWR alerts when the performance exceeds the thresholds. You can then respond to these alerts by changing the priority of a job, stopping overloaded processes, or relocating, expanding, shrinking, starting or stopping a service. Using automated tasks, you can automate the reaction.

This allows you to maintain service quality despite changes in demand.

Note: The `SELECT` statement shown in the slide above gives you the accumulated instance statistics for elapsed time and for CPU used metrics for each service for the most recent 60-second interval. For the last hour history, look at `V$SERVICEMETRIC_HISTORY`.

Changing Service Thresholds Using EM



Oracle Enterprise Manager (SYSMAN) - Edit Thresholds - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://drlab3:1820/em/console/monitoring/manageMetrics$type=oracle*_database$target=arac*_arac2
Go
Links Aria Oracle Files Online Customize Links My Oracle Network Request Oracle CRM Oracle Email Software Windows Windows Media
Local intranet

○	Redo Writes (per second)	>			
○	Redo Writes (per transaction)	>			
○	Response Time (per transaction)	>			
○	Row Cache Miss Ratio (%)	<			
○	Rows Processed (per sort)	>			
○	SQL Response Time (%)	>	500		
○	Scans on Long Tables (per second)	>			
○	Scans on Long Tables (per transaction)	>			
○	Service CPU Time (per user call) (microseconds)	>			
○	Service Response Time (per user call) (microseconds)	>			
○	Session Limit Usage (%)	>	90	97	
○	Session Logical Reads (per second)	>			
○	Session Logical Reads (per transaction)	>			
○	Session Terminated Alert Log Error	**Contains**	ORA-		
○	Session Terminated Alert Log Error Status	>	0		
○	Shared Pool Free (%)	<			
○	Soft Parse (%)	<			
○	Sorts in Memory (%)	<			
○	Sorts to Disk (per second)	>			
○	Sorts to Disk (per transaction)	>			
○	State	**Contains**	MOUNTED		
○	System Response Time (centi-seconds)	>			
○	Total Index Scans (per second)	>			
○	Total Index Scans (per transaction)	>			
○	Total Parses (per second)	>			
○	Total Parses (per transaction)	>			
○	Total Table Scan (per second)	>			
○	Total Table Scan (per transaction)	>			
○	User Calls (%)	<			
○	User Calls (per second)	>			

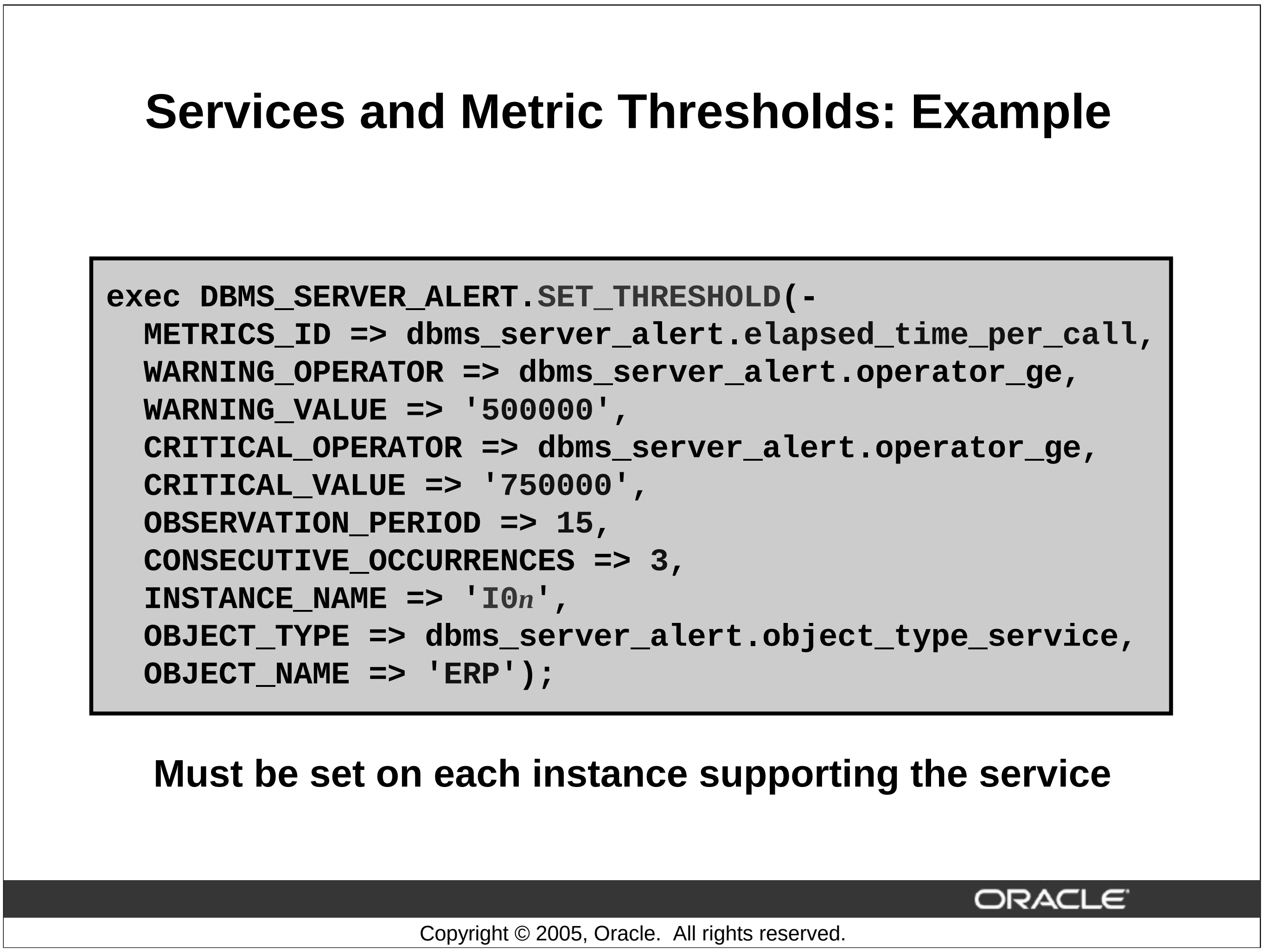
Changing Service Thresholds Using EM

The Edit Thresholds page is displayed in the slide. The screenshot shows a portion of the page where you can see the Service CPU Time (per user call) and Service Response Time (per user call) metrics.

To access the Edit Thresholds page, click the Manage Metrics link on the All Metrics page. After the Manage Metrics page is displayed, click the Edit Thresholds button.

Using the Edit Thresholds page, you can change the critical and warning values for the service metrics. If you modify the critical and warning values on this page, the thresholds apply to all services of the instance.

If you want different thresholds for different services, click the Specify Multiple Thresholds button at the top of the page. Another page appears where you can set critical and warning thresholds for individual services.



Services and Metric Thresholds: Example

```
exec DBMS_SERVER_ALERT.SET_THRESHOLD(-
  METRICS_ID => dbms_server_alert.elapsed_time_per_call,
  WARNING_OPERATOR => dbms_server_alert.operator_ge,
  WARNING_VALUE => '500000',
  CRITICAL_OPERATOR => dbms_server_alert.operator_ge,
  CRITICAL_VALUE => '750000',
  OBSERVATION_PERIOD => 15,
  CONSECUTIVE_OCCURRENCES => 3,
  INSTANCE_NAME => 'I0n',
  OBJECT_TYPE => dbms_server_alert.object_type_service,
  OBJECT_NAME => 'ERP');
```

Must be set on each instance supporting the service

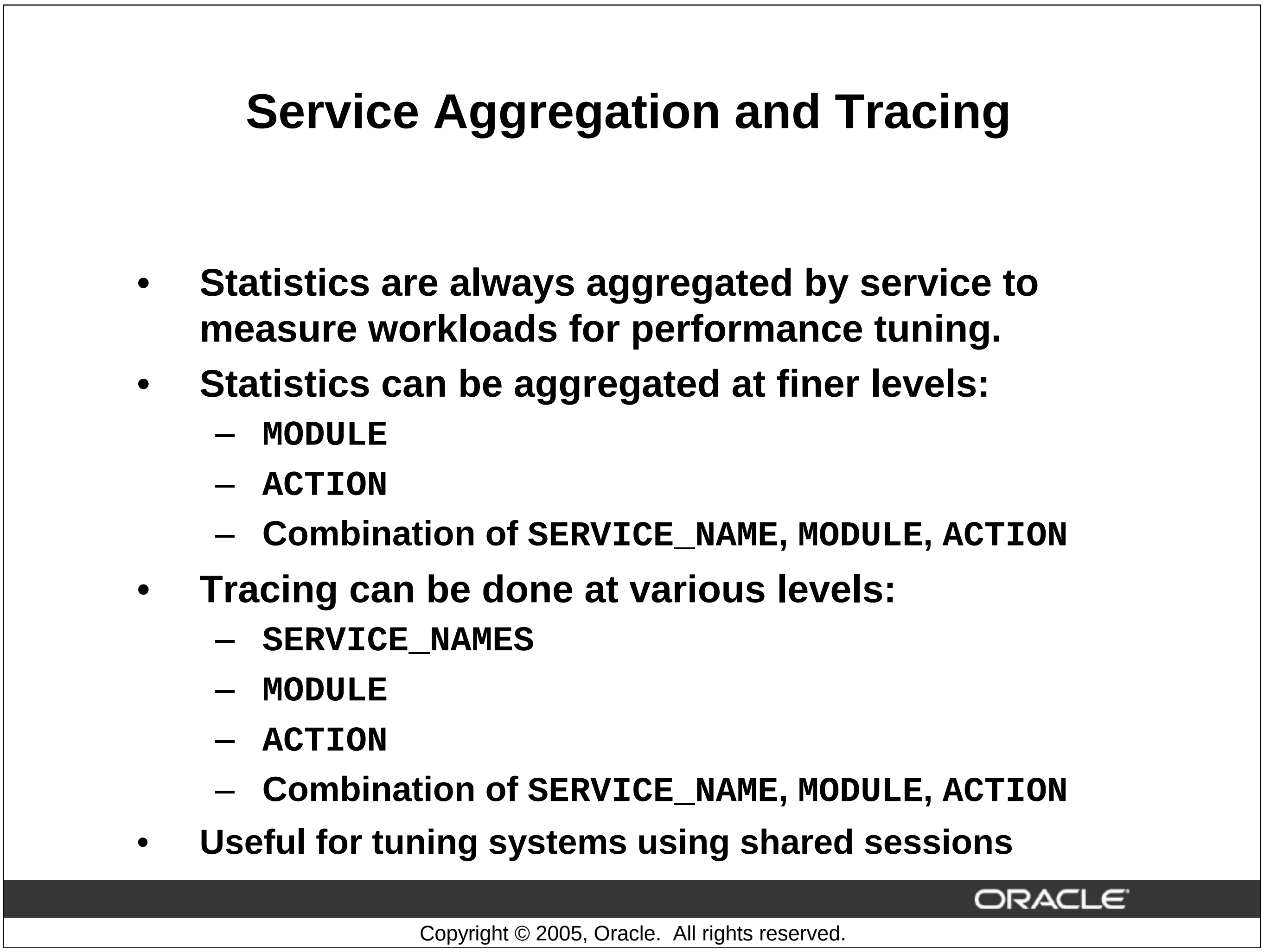
Services and Metric Thresholds: Example

In this example, thresholds are added for the ERP service for the `ELAPSED_TIME_PER_CALL` metric. This metric measures the elapsed time for each user call for the corresponding service. The time must be expressed in microseconds.

A warning alert is raised by the server whenever the average elapsed time per call for the ERP service over a 15-minutes period exceeds 0.5 seconds three consecutive times.

A critical alert is raised by the server whenever the average elapsed time per call for the ERP service over a 15-minutes period exceeds 0.75 seconds three consecutive times.

Note: The thresholds must be created for each RAC instance that potentially supports the service.



Service Aggregation and Tracing

- Statistics are always aggregated by service to measure workloads for performance tuning.
- Statistics can be aggregated at finer levels:
  - `MODULE`
  - `ACTION`
  - Combination of `SERVICE_NAME`, `MODULE`, `ACTION`
- Tracing can be done at various levels:
  - `SERVICE_NAMES`
  - `MODULE`
  - `ACTION`
  - Combination of `SERVICE_NAME`, `MODULE`, `ACTION`
- Useful for tuning systems using shared sessions

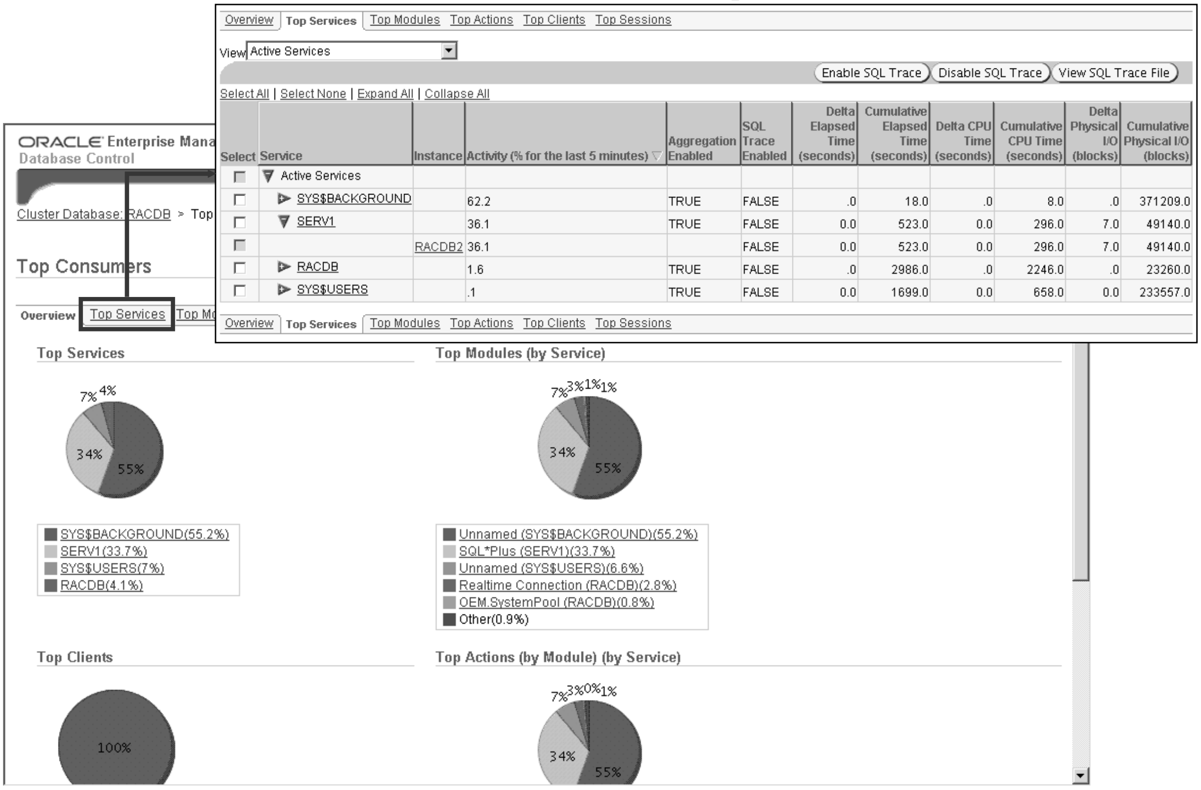
Service Aggregation and Tracing

By default, important statistics and wait events are collected for the work attributed to every service. An application can further qualify a service by `MODULE` and `ACTION` names to identify the important transactions within the service. This enables you to locate exactly the poorly performing transactions for categorized workloads. This is especially important when monitoring performance in systems by using connection pools or transaction processing monitors. For these systems, the sessions are shared which makes accountability difficult.

`SERVICE_NAME`, `MODULE`, and `ACTION` are actual columns in `V$SESSION`. `SERVICE_NAME` is set automatically at login time for the user. `MODULE` and `ACTION` names are set by the application by using the `DBMS_APPLICATION_INFO` PL/SQL package or special OCI calls. `MODULE` should be set to a user recognizable name for the program that is currently executing. Likewise, `ACTION` should be set to a specific action or task that a user is performing within a module (for example, entering a new customer).

Another aspect of this workload aggregation is tracing by service. The traditional method of tracing each session produces trace files with SQL commands that can span workloads. This results in a hit-or-miss approach to diagnose problematic SQL. With the criteria that you provide, `SERVICE_NAME`, `MODULE`, or `ACTION`, specific trace information is captured in a set of trace files and combined into a single output trace file. This enables you to produce trace files that contain SQL that is relevant to a specific workload being done.

Cluster Database: Top Services



Overview | **Top Services** | Top Modules | Top Actions | Top Clients | Top Sessions

View: Active Services

Enable SQL Trace | Disable SQL Trace | View SQL Trace File

Select All | Select None | Expand All | Collapse All

Select	Service	Instance	Activity (% for the last 5 minutes)	Aggregation Enabled	SQL Trace Enabled	Delta Elapsed Time (seconds)	Cumulative Elapsed Time (seconds)	Delta CPU Time (seconds)	Cumulative CPU Time (seconds)	Delta Physical I/O (blocks)	Cumulative Physical I/O (blocks)
□	▼ Active Services										
□	▶ SYS$BACKGROUND		62.2	TRUE	FALSE	.0	18.0	.0	8.0	.0	371209.0
□	▼ SERV1		36.1	TRUE	FALSE	0.0	523.0	0.0	296.0	7.0	49140.0
□		RACDB2	36.1		FALSE	0.0	523.0	0.0	296.0	7.0	49140.0
□	▶ RACDB		1.6	TRUE	FALSE	.0	2986.0	.0	2246.0	.0	23260.0
□	▶ SYS$USERS		.1	TRUE	FALSE	0.0	1699.0	0.0	658.0	0.0	233557.0

Overview | **Top Services** | Top Modules | Top Actions | Top Clients | Top Sessions

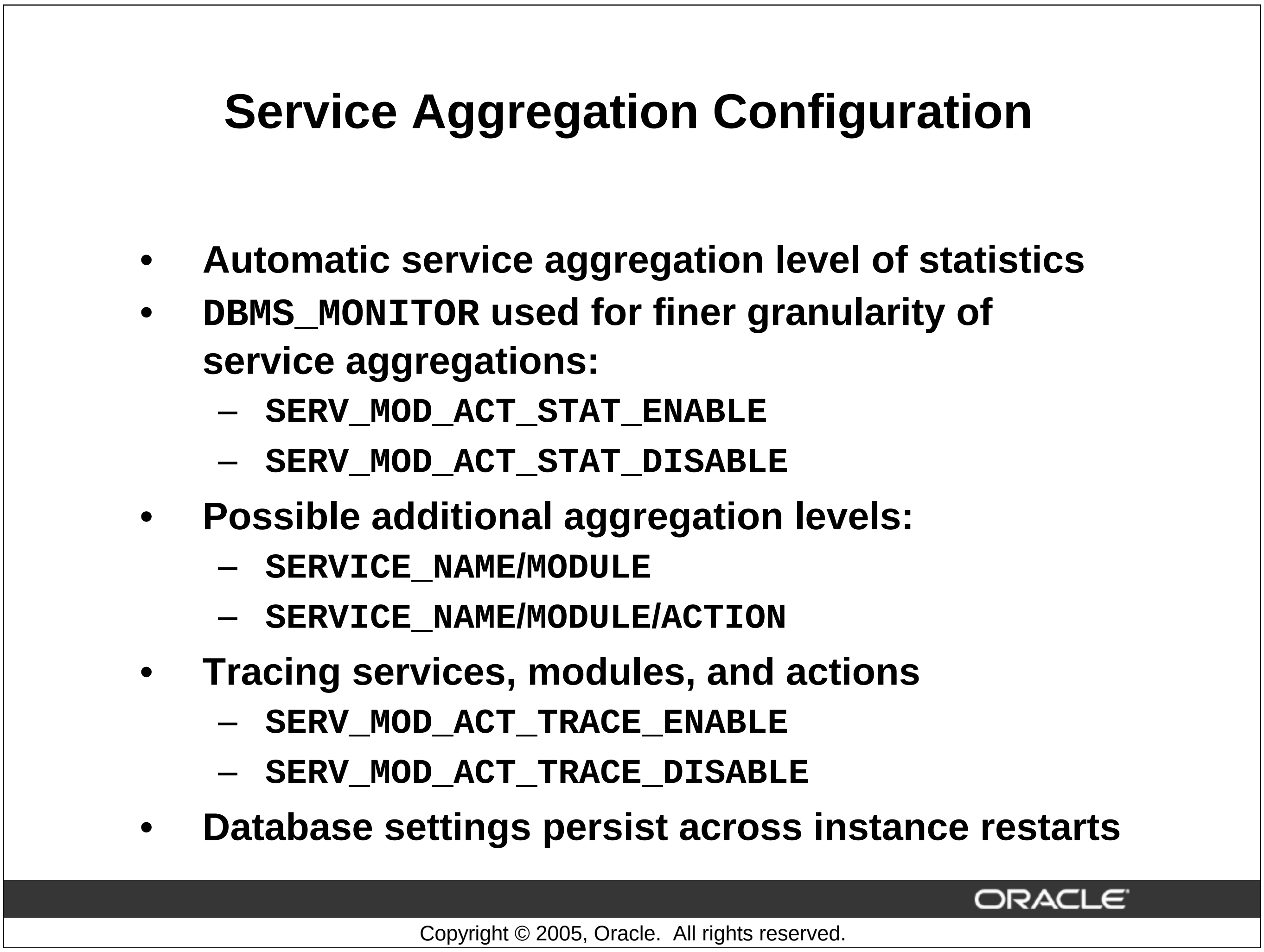
Cluster Database: Top Services

From the **Cluster Database Performance** page, you can access **Top Consumers** page by clicking the **Top Consumers** link.

The **Top Consumers** page has several tabs for displaying your RAC database as a single-system image. The **Overview** tab contains four pie charts: **Top Clients**, **Top Services**, **Top Modules**, and **Top Actions**. Each chart provides a different perspective regarding the top resource consumers across all instances of a particular RAC database.

The **Top Services** tab displays performance-related information for the services that are defined in your cluster. Performance data is broken down by each instance that the service has run on since startup and is also summarized across all instances.

Through this page, you can enable or disable tracing at the service level, as well as view the resulting SQL trace file.



Service Aggregation Configuration

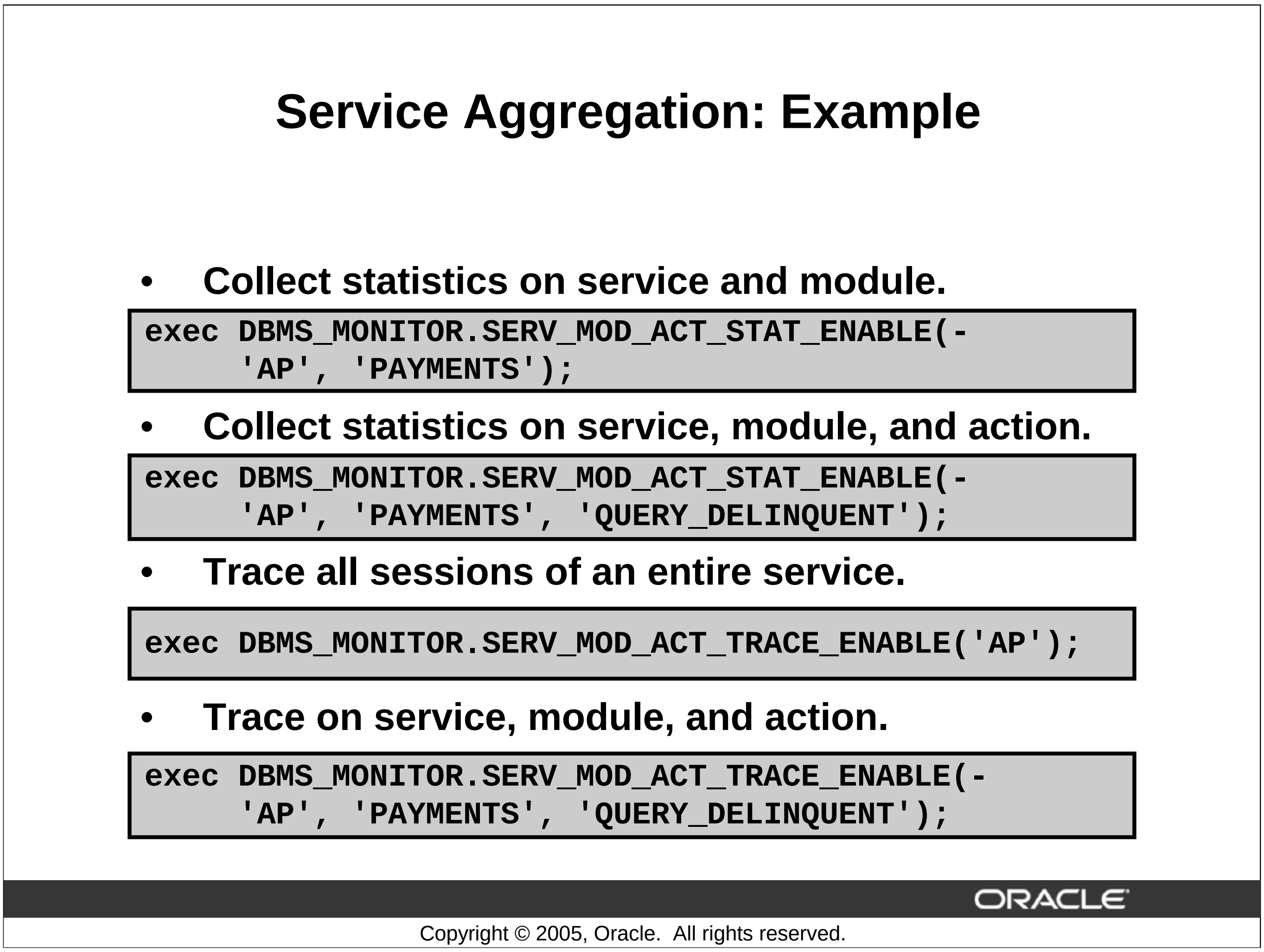
- **Automatic service aggregation level of statistics**
- **`DBMS_MONITOR` used for finer granularity of service aggregations:**
  - **`SERV_MOD_ACT_STAT_ENABLE`**
  - **`SERV_MOD_ACT_STAT_DISABLE`**
- **Possible additional aggregation levels:**
  - **`SERVICE_NAME/MODULE`**
  - **`SERVICE_NAME/MODULE/ACTION`**
- **Tracing services, modules, and actions**
  - **`SERV_MOD_ACT_TRACE_ENABLE`**
  - **`SERV_MOD_ACT_TRACE_DISABLE`**
- **Database settings persist across instance restarts**

Service Aggregation Configuration

On each instance, important statistics and wait events are automatically aggregated and collected by service. You do not have to do anything to set this up, except connect with different connect strings using the services you want to connect to. However, to achieve a finer level of granularity of statistics collection for services, you must make use of the `SERV_MOD_ACT_STAT_ENABLE` procedure in the `DBMS_MONITOR` package. This procedure enables statistics gathering for additional hierarchical combinations of `SERVICE_NAME/MODULE` and `SERVICE_NAME/MODULE/ACTION`. The `SERV_MOD_ACT_STAT_DISABLE` procedure stops the statistics gathering that was turned on.

The enabling and disabling of statistics aggregation within the service applies to every instance accessing the database. Furthermore, these settings are persistent across instance restarts.

The `SERV_MOD_ACT_TRACE_ENABLE` procedure enables tracing for services with three hierarchical possibilities: `SERVICE_NAME`, `SERVICE_NAME/MODULE`, and `SERVICE_NAME/MODULE/ACTION`. The default is to trace for all instances that access the database. A parameter is provided that restricts tracing to specified instances where poor performance is known to exist. This procedure also gives you the option of capturing relevant waits and bind variable values in the generated trace files. `SERV_MOD_ACT_TRACE_DISABLE` disables the trace at all enabled instances for a given combination of service, module, and action. Like the statistics gathering mentioned previously, service tracing persists across instance restarts.



Service Aggregation: Example

- Collect statistics on service and module.

```
exec DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(-
     'AP', 'PAYMENTS');
```

- Collect statistics on service, module, and action.

```
exec DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(-
     'AP', 'PAYMENTS', 'QUERY_DELINQUENT');
```

- Trace all sessions of an entire service.

```
exec DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE('AP');
```

- Trace on service, module, and action.

```
exec DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE(-
     'AP', 'PAYMENTS', 'QUERY_DELINQUENT');
```

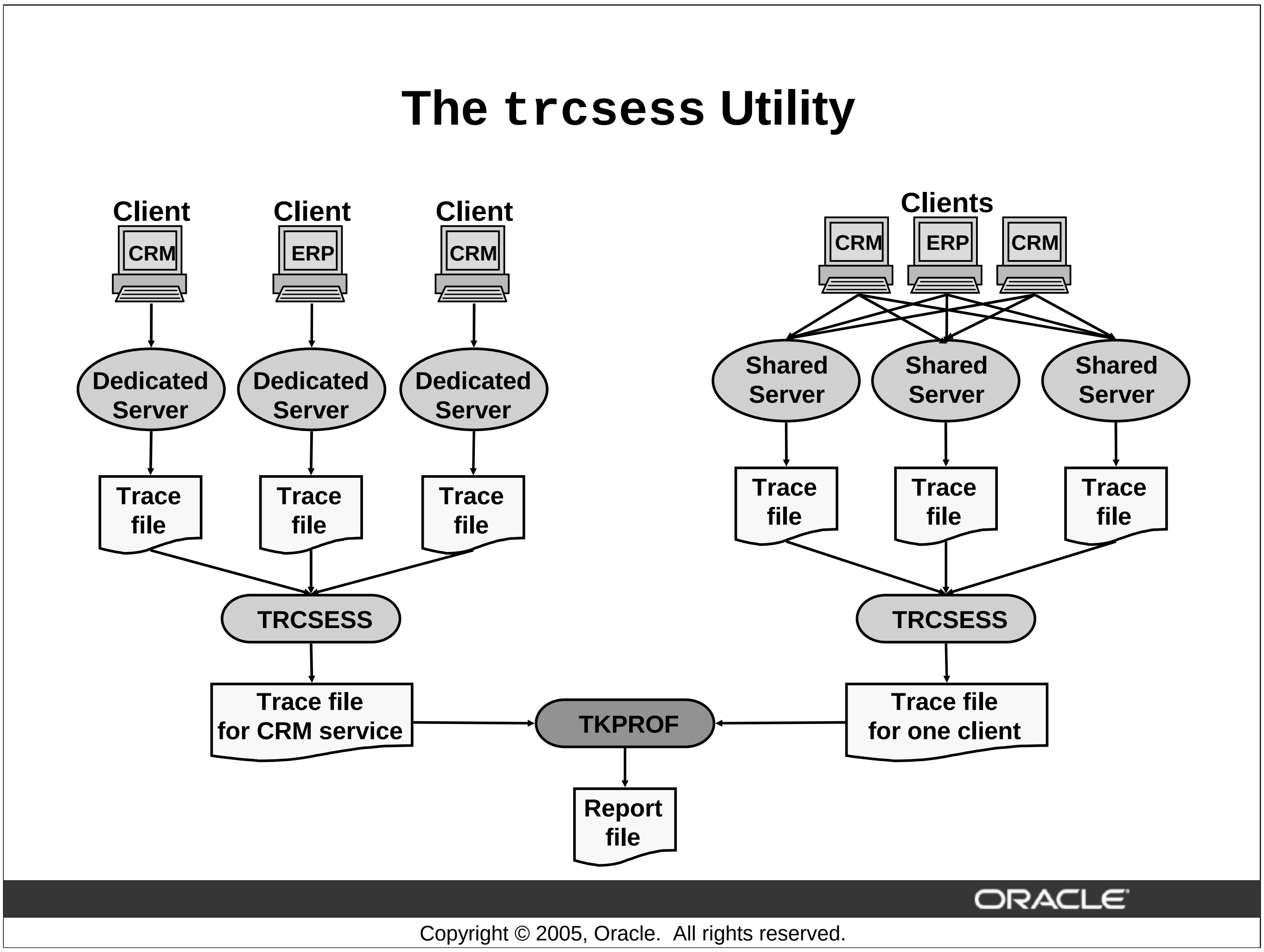
Service Aggregation: Example

The first piece of sample code begins collecting statistics for the PAYMENTS module within the AP service. The second example collects statistics only for the QUERY_DELINQUENT program that runs in the PAYMENTS module under the AP service. This enables statistics collection on specific tasks that run in the database.

In the third code box, all sessions that log in under the AP service are traced. A trace file is created for each session that uses the service, regardless of the module and action. To be precise, you can trace only specific tasks within a service. This is illustrated in the last example where all sessions of the AP service that execute the QUERY_DELINQUENT action within the PAYMENTS module are traced.

Tracing by service, module, and action enables you to focus your tuning efforts on specific SQL, rather than sifting through trace files with SQL from different programs. Only the SQL statements that define this one task are recorded in the trace file. This complements collecting statistics by service, module, and action because relevant wait events for an action can be identified.

Note: For more information about the DBMS_MONITOR package, refer to *PL/SQL Packages and Types Reference*.



The trcsess Utility

The trcsess Utility

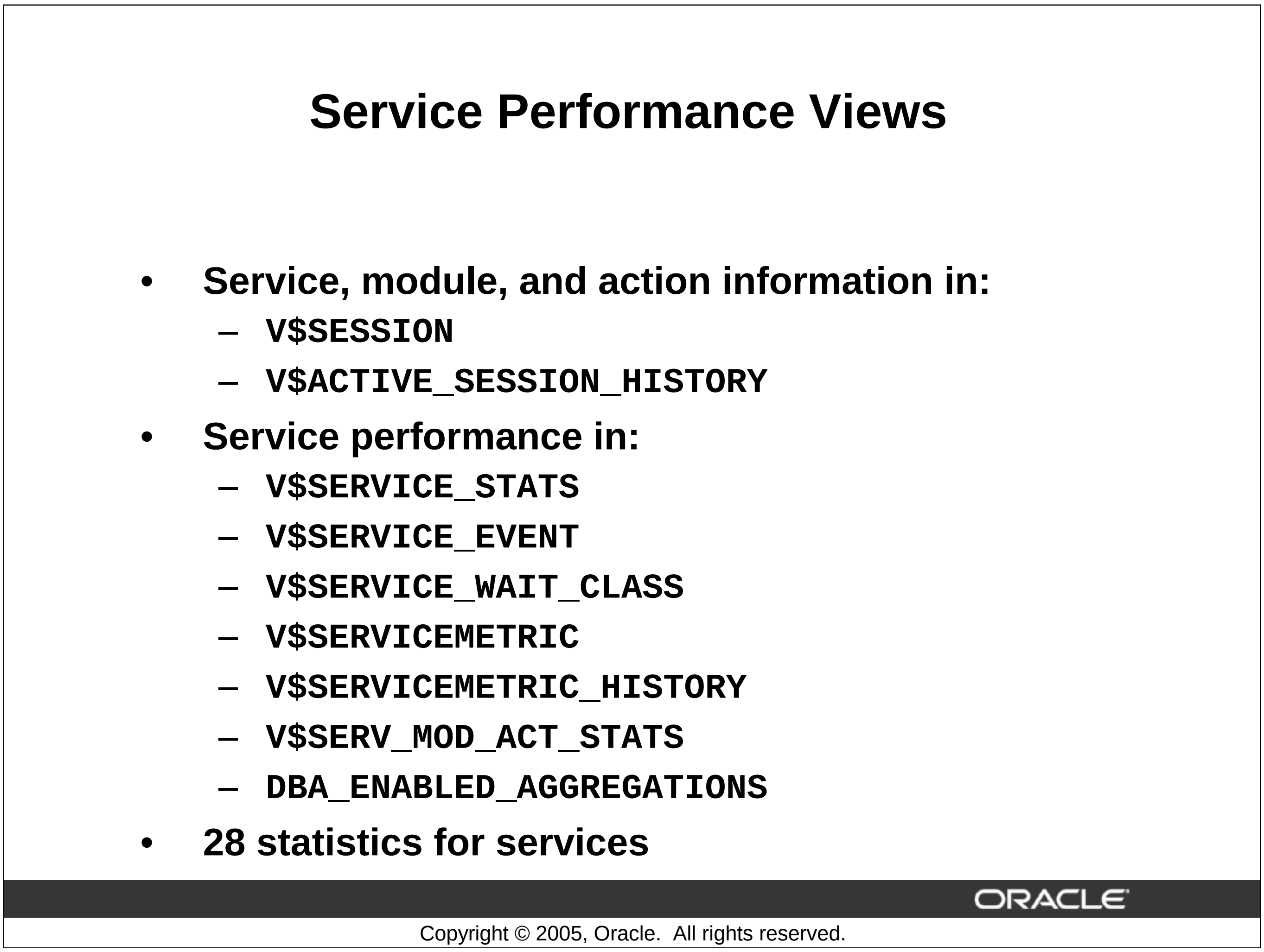
The trcsess utility consolidates trace output from selected trace files based on several criteria: Session Id, client Id, service name, action name, and module name. After trcsess merges the trace information into a single output file, the output file can be processed by tkprof.

When using the DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE procedure, tracing information is present in multiple trace files and you must use the trcsess tool to collect it into a single file.

The trcsess utility is useful for consolidating the tracing of a particular session or service for performance or debugging purposes.

Tracing a specific session is usually not a problem in the dedicated server model as a single dedicated process serves a session during its lifetime. All the trace information for the session can be seen from the trace file belonging to the dedicated server serving it. However, tracing a service might become a complex task even in dedicated server model.

Moreover, in a shared server configuration a user session is serviced by different processes from time to time. The trace pertaining to the user session is scattered across different trace files belonging to different processes. This makes it difficult to get a complete picture of the life cycle of a session.



Service Performance Views

- Service, module, and action information in:
  - `V$SESSION`
  - `V$ACTIVE_SESSION_HISTORY`
- Service performance in:
  - `V$SERVICE_STATS`
  - `V$SERVICE_EVENT`
  - `V$SERVICE_WAIT_CLASS`
  - `V$SERVICEMETRIC`
  - `V$SERVICEMETRIC_HISTORY`
  - `V$SERV_MOD_ACT_STATS`
  - `DBA_ENABLED_AGGREGATIONS`
- 28 statistics for services

Service Performance Views

The service, module, and action information are visible in `V$SESSION` and `V$ACTIVE_SESSION_HISTORY`.

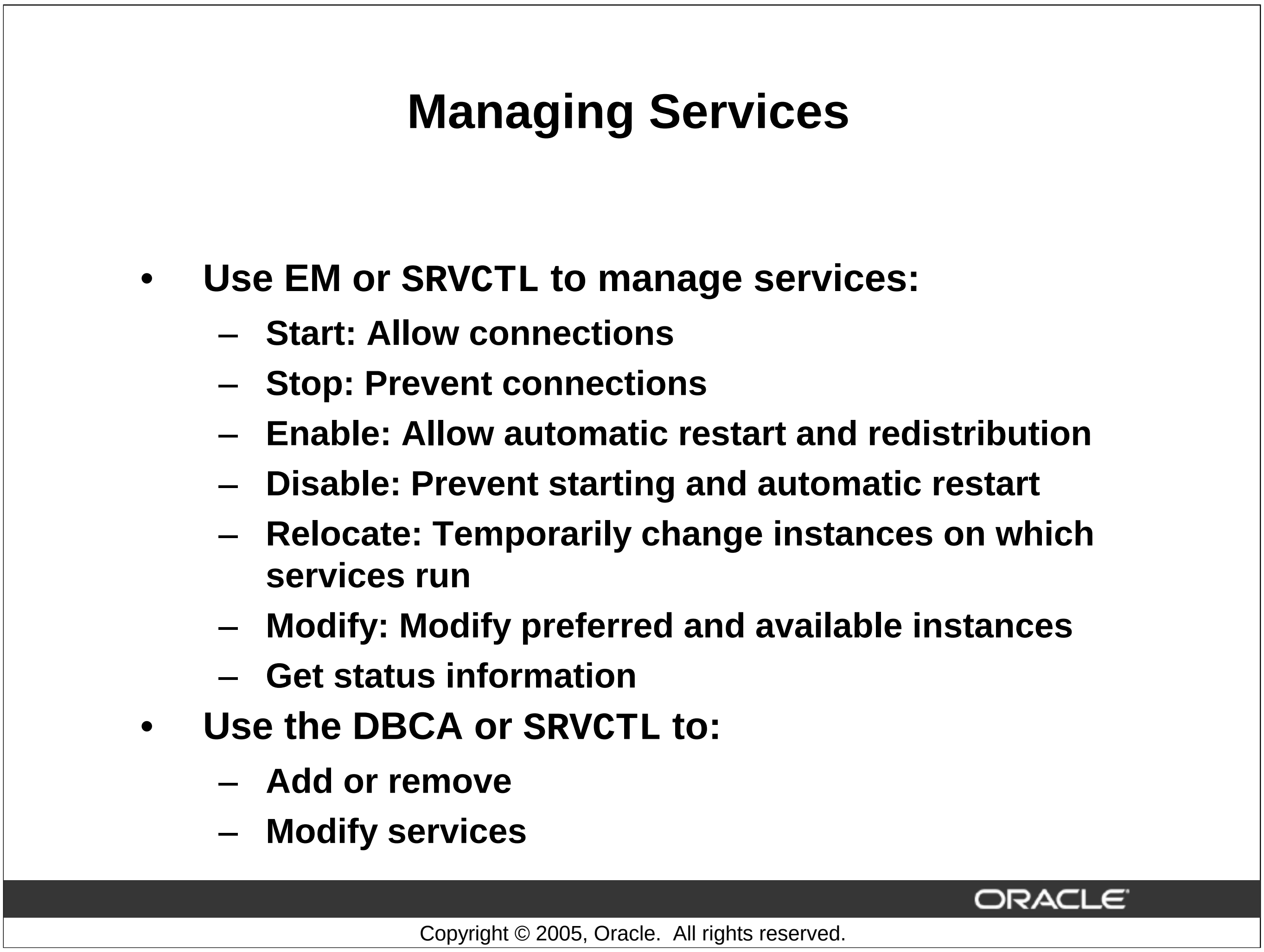
The call times and performance statistics are visible in `V$SERVICE_STATS`, `V$SERVICE_EVENT`, `V$SERVICE_WAIT_CLASS`, `V$SERVICEMETRIC`, and `V$SERVICEMETRIC_HISTORY`.

When statistics collection for specific modules and actions is enabled, performance measures are visible at each instance in `V$SERV_MOD_ACT_STATS`.

There are over 300 performance-related statistics that are tracked and visible in `V$SYSSTAT`. Of these, 28 statistics are tracked for services. To see the statistics measured for services, run the following query: `SELECT DISTINCT stat_name FROM v$service_stats`

Of the 28 statistics, `DB time` and `DB CPU` are worth mentioning. `DB time` is a statistic that measures the average response time per call. It represents the actual wall clock time for a call to complete. `DB CPU` is an average of the actual CPU time spent per call. The difference between response time and CPU time is the wait time for the service. After the wait time is known and if it consumes a large percentage of response time, then you can trace at the action level to identify the waits.

Note: `DBA_ENABLED_AGGREGATIONS` displays information about enabled on-demand statistic aggregation.



Managing Services

- **Use EM or `SRVCTL` to manage services:**
  - **Start: Allow connections**
  - **Stop: Prevent connections**
  - **Enable: Allow automatic restart and redistribution**
  - **Disable: Prevent starting and automatic restart**
  - **Relocate: Temporarily change instances on which services run**
  - **Modify: Modify preferred and available instances**
  - **Get status information**
- **Use the DBCA or `SRVCTL` to:**
  - **Add or remove**
  - **Modify services**

Managing Services

Depending on the type of management tasks that you want to perform, you can use Enterprise Manager, DBCA, or `SRVCTL`.

The following is the description of the management tasks related to services in a RAC environment:

- Disabling a service is used to disable a specified service on all or specified instances. The disable state is used when a service is down for maintenance to prevent inappropriate automatic CRS restarts. Disabling an entire service affects all the instances by disabling each one.
- Enabling a service is used to enable a service to run under CRS for automatic restart and redistribution. You can enable a service even if that service is stopped. Enable is the default value when a service is created. If the service is already enabled, then the command is ignored. Enabled services can be started, and disabled services cannot be started. Enabling an entire service affects the enabling of the service over all the instances by enabling the service at each one.
- Starting a service is used to start a service or multiple services on the specified instance. Only enabled services can be started. The command fails if you attempt to start a service on an instance and if the number of instances that are currently running the service already reaches its cardinality.

Managing Services (continued)

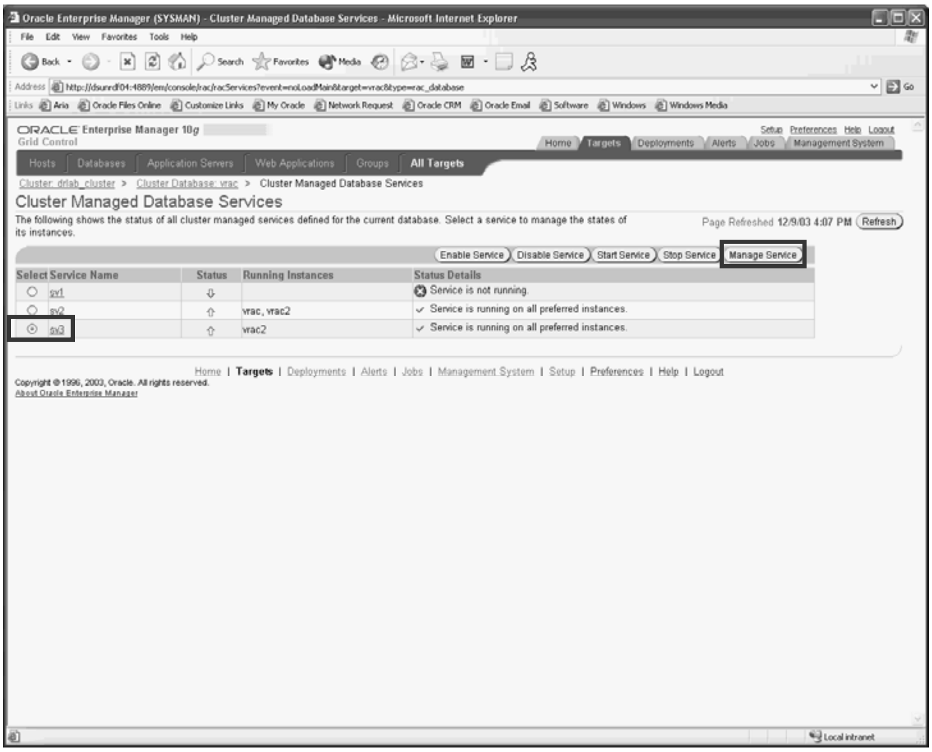
- Stopping is used to stop one or more services globally across the cluster database, or on the specified instance. Only CRS services that are starting or started are stopped. You should disable a service that you intend to remain stopped after you stopped that service because if the service is stopped and is not disabled, then it can be restarted automatically as a result of another planned operation. This operation can force sessions to be disconnected transactionally.
- Removing a service is used to remove its configuration from the cluster database on all or specified instances. You must first stop the corresponding service before you can remove it. You can remove a service from specific instances only.
- Relocating a service is used to relocate a service from a source instance to a target instance. The target instance must be on the preferred or available list for the service. This operation can force sessions to be disconnected transactionally. The relocated service is temporary until you permanently modify the configuration.
- Modifying a service configuration is used to permanently modify a service configuration. The change takes effect when the service is restarted later. This allows you to move a service from one instance to another. Additionally, this command changes the instances that are to be the preferred and available instances for a service.
- Displaying the current state of a named service.

You can only administer a service with Enterprise Manager after creating the service with DBCA or `SRVCTL`. When using the DBCA to add services, the DBCA also configures the net service entries for these services and starts them. When you use DBCA to remove services, DBCA stops the service, removes the CRS resource for the service, and removes the net service entries.

When you create a service with `SRVCTL`, you must start it with a separate `SRVCTL` command.

Note: When you create a service it is automatically enabled.

Managing Services with EM



Oracle Enterprise Manager (SYSMAN) - Cluster Managed Database Services - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://dsunrdf04:4889/em/console/rac/racServices?event=noLoadMain&target=vrac&type=rac_database
Links Aria Oracle Files Online Customize Links My Oracle Network Request Oracle CRM Oracle Email Software Windows Windows Media
ORACLE Enterprise Manager 10g
Grid Control
Setup Preferences Help Logout
Home Targets Deployments Alerts Jobs Management System
Hosts Databases Application Servers Web Applications Groups All Targets
Cluster: drlab_cluster > Cluster Database: vrac > Cluster Managed Database Services
Page Refreshed 12/9/03 4:07 PM Refresh
Enable Service Disable Service Start Service Stop Service Manage Service
Home | Targets | Deployments | Alerts | Jobs | Management System | Setup | Preferences | Help | Logout
Copyright © 1996, 2003, Oracle. All rights reserved.
About Oracle Enterprise Manager
Local intranet

Cluster Managed Database Services

The following shows the status of all cluster managed services defined for the current database. Select a service to manage the states of its instances.

Select	Service Name	Status	Running Instances	Status Details
○	sv1	⇩		Service is not running.
○	sv2	⇧	vrac, vrac2	✓ Service is running on all preferred instances.
●	sv3	⇧	vrac2	✓ Service is running on all preferred instances.

Managing Services with EM

EM provides you with some ability to manage services within a GUI framework. The screenshot shown in the slide above is the main page for administering services within RAC. It shows you some basic status information about defined service.

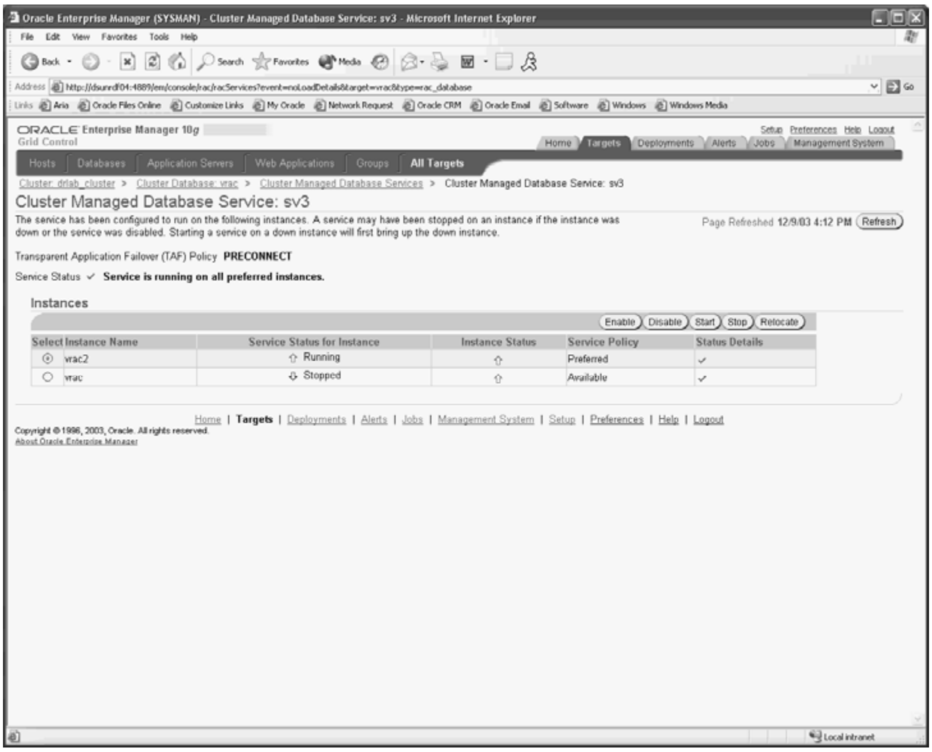
To access this page, click the Cluster Managed Database Services link on the Cluster Database Administration page.

With the initial release of EM, you can perform simple service management such as enabling, disabling, starting, stopping, and relocating services.

If you choose to start a service on the Cluster Managed Database Services page, then EM attempts to start the service on every preferred instance. Stopping the service stops it on all instances that it is currently running.

To start or stop a service on individual instances or to relocate a service, choose the service that you want to administer and then click the Manage Service button.

Managing Services with EM



Oracle Enterprise Manager (SYSMAN) - Cluster Managed Database Service: sv3 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://dsunrdf04:4889/em/console/rac/racServices?event=noLoadDetails&target=vrac&type=rac_database
Go
Links Aria Oracle Files Online Customize Links My Oracle Network Request Oracle CRM Oracle Email Software Windows Windows Media
ORACLE Enterprise Manager 10g
Grid Control
Setup Preferences Help Logout
Home Targets Deployments Alerts Jobs Management System
Hosts Databases Application Servers Web Applications Groups All Targets
Local intranet

Cluster: drlab_cluster > Cluster Database: vrac > Cluster Managed Database Services > Cluster Managed Database Service: sv3

Cluster Managed Database Service: sv3

The service has been configured to run on the following instances. A service may have been stopped on an instance if the instance was down or the service was disabled. Starting a service on a down instance will first bring up the down instance.

Page Refreshed 12/9/03 4:12 PM (Refresh)

Transparent Application Failover (TAF) Policy **PRECONNECT**

Service Status ✓ **Service is running on all preferred instances.**

Instances

(Enable) (Disable) (Start) (Stop) (Relocate)

Select	Instance Name	Service Status for Instance	Instance Status	Service Policy	Status Details
●	vrac2	⇧ Running	⇧	Preferred	✓
○	vrac	⇩ Stopped	⇧	Available	✓

Home | **Targets** | Deployments | Alerts | Jobs | Management System | Setup | Preferences | Help | Logout

Copyright © 1996, 2003, Oracle. All rights reserved.
About Oracle Enterprise Manager

ORACLE

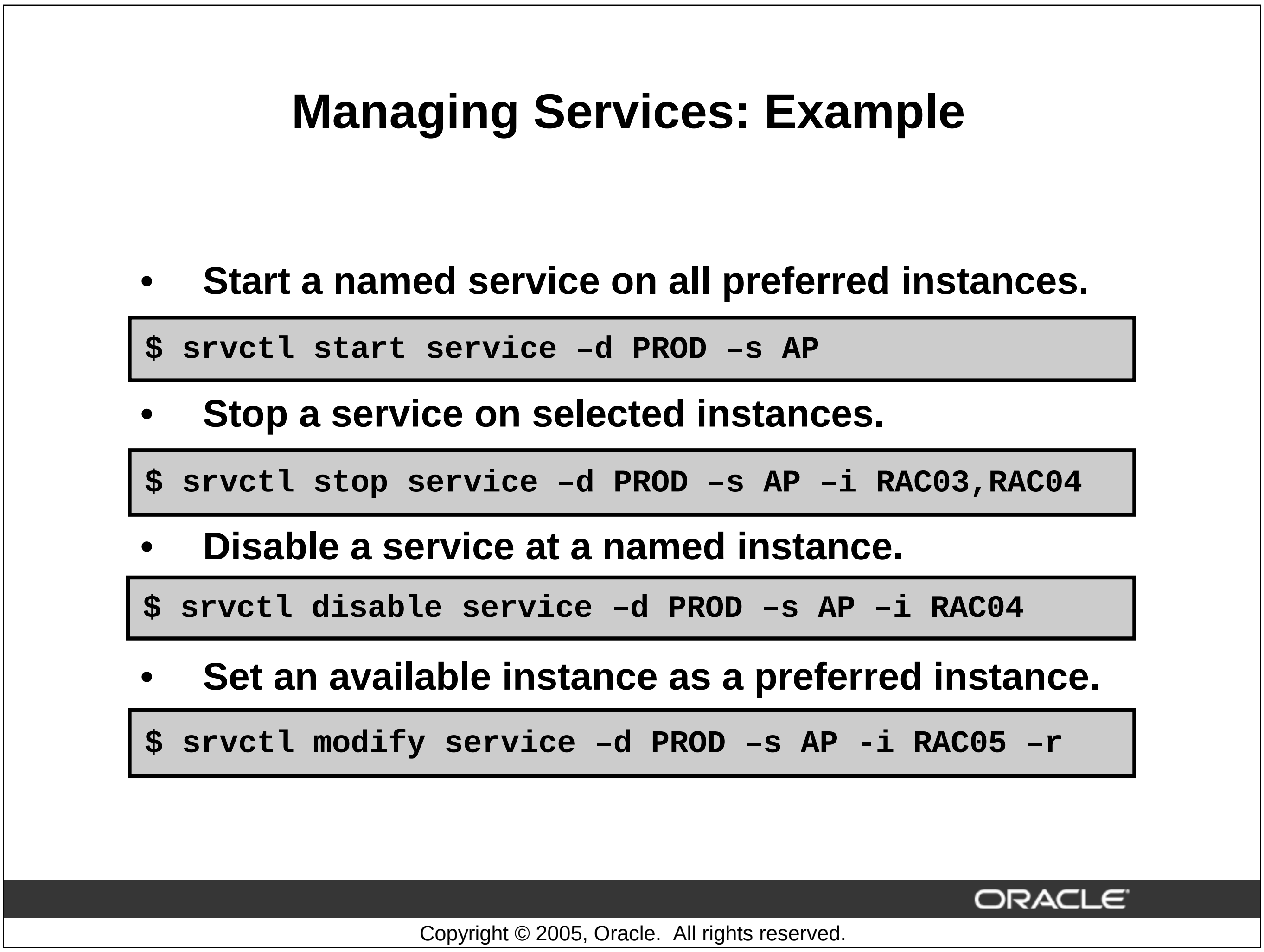
Managing Services with EM (continued)

To access this page, you must choose a service from the Cluster Managed Database Services page and then click the Manage Service button.

This is the Cluster Managed Database Service page for an individual service. It offers you the same functionality as the previous page, except that actions performed here apply to specific instances of a service.

This page also offers you the added functionality of relocating a service to an available instance. Relocating a service from one instance to another stops the service on the first instance and then starts it on the second.

Note: This page also shows you the TAF policy set for this particular service.



Managing Services: Example

- Start a named service on all preferred instances.

```
$ srvctl start service –d PROD –s AP
```

- Stop a service on selected instances.

```
$ srvctl stop service –d PROD –s AP –i RAC03,RAC04
```

- Disable a service at a named instance.

```
$ srvctl disable service –d PROD –s AP –i RAC04
```

- Set an available instance as a preferred instance.

```
$ srvctl modify service –d PROD –s AP -i RAC05 –r
```

Managing Services: Example

The slide demonstrates some management tasks with services by using `SRVCTL`.

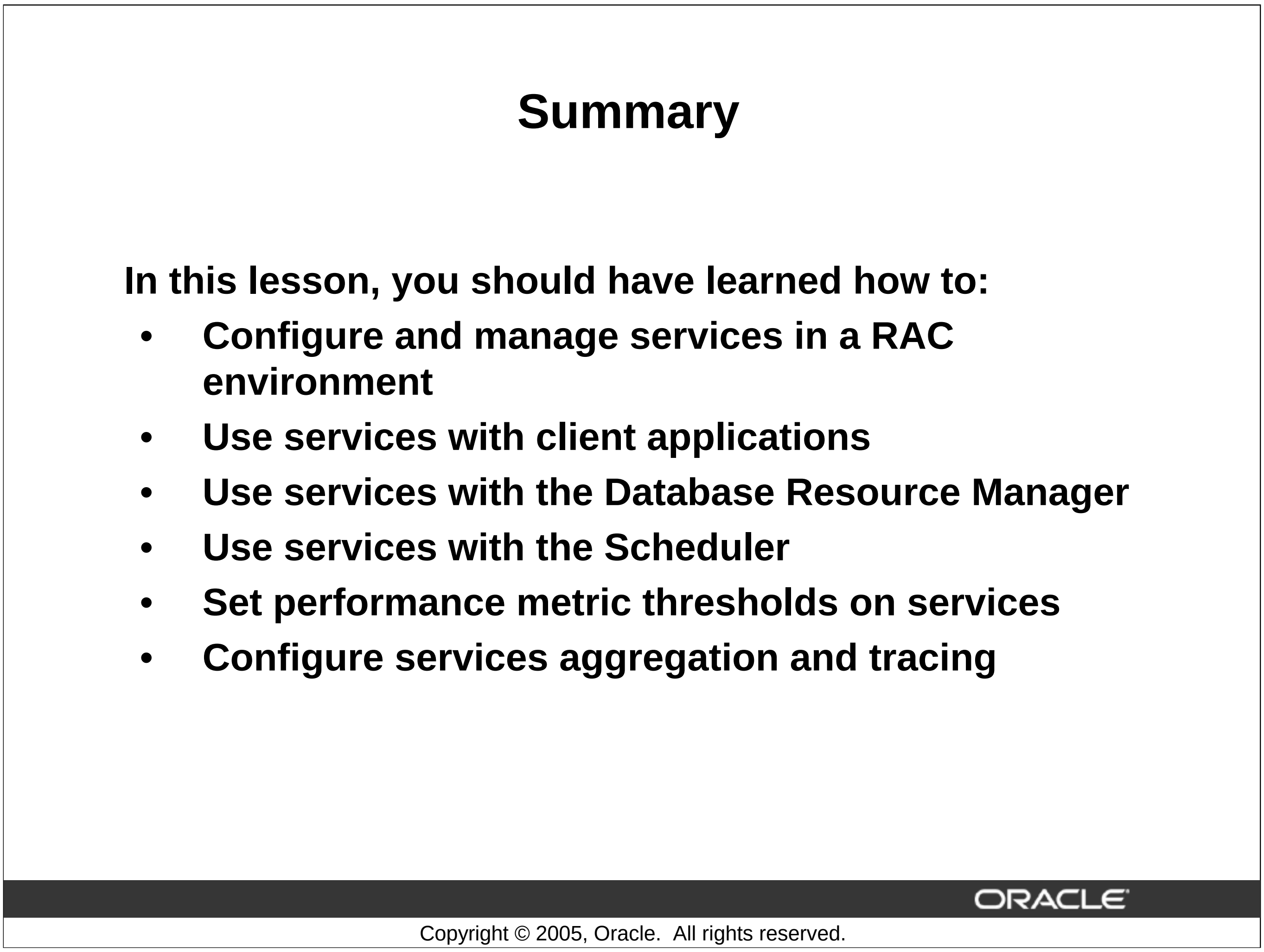
Assume that an `AP` service has been created with four preferred instances: `RAC01`, `RAC02`, `RAC03`, and `RAC04`. An available instance, `RAC05`, has also been defined for `AP`.

In the first example, the `AP` service is started on all preferred instances. If any of the preferred or available instances that support `AP`, are not running but are enabled, then they are started.

The stop command stops the `AP` service on instances `RAC03` and `RAC04`. The instances themselves are not shut down, but remain running possibly supporting other services. The `AP` service continues to run on `RAC01` and `RAC02`. The intention might have been to do maintenance on `RAC04`, and so the `AP` service was disabled on that instance to prevent automatic restart of the service on that instance. The OCR records the fact that `AP` is disabled for `RAC04`. Thus, CRS will not run `AP` on `RAC04` until the service is enabled later.

The last command in the slide changes `RAC05` from being an available instance to a preferred one. This is beneficial if the intent is to always have four instances run the service because `RAC04` was previously disabled.

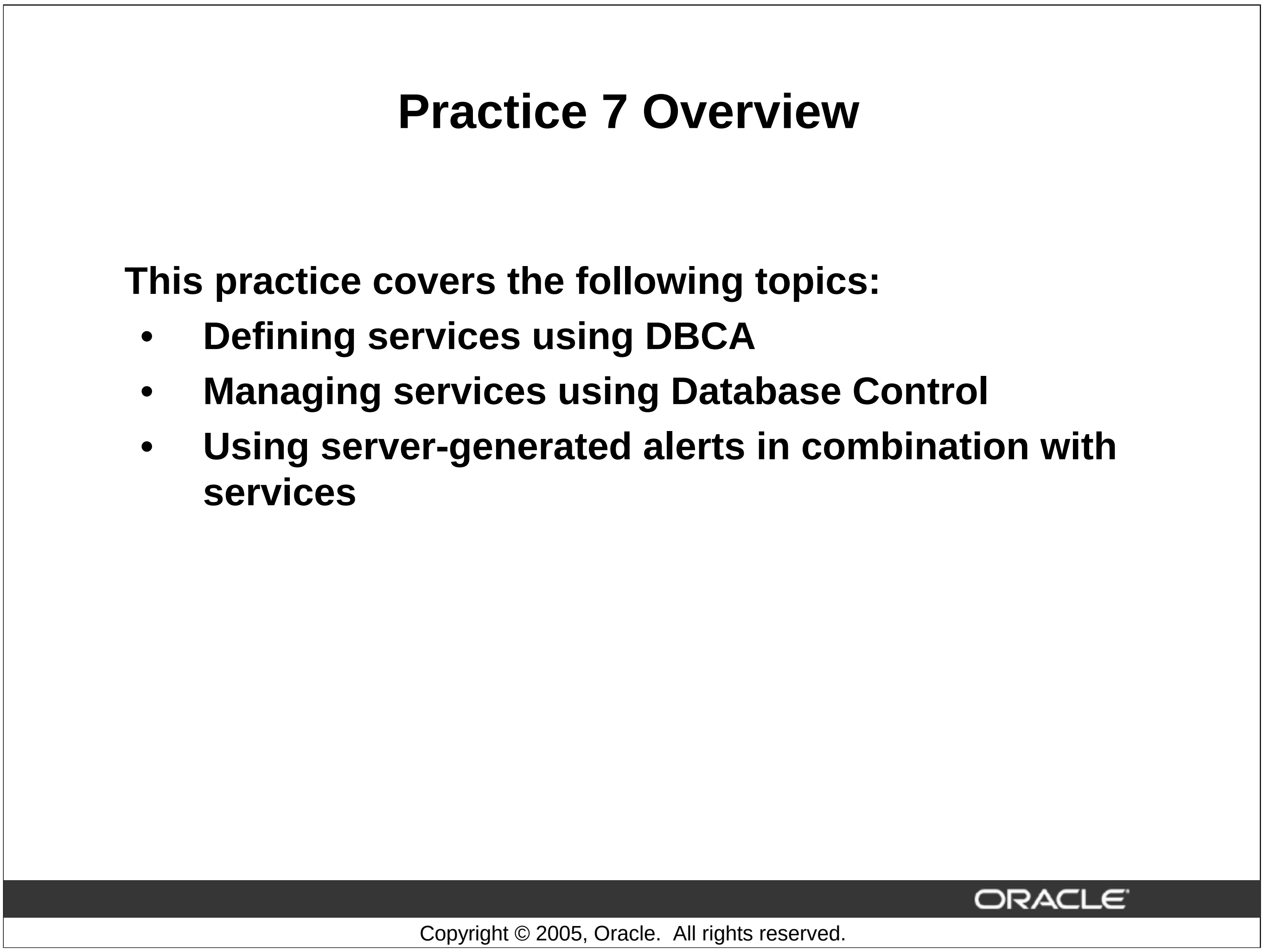
Note: For more information, refer to the *Oracle Real Application Clusters Administrator's Guide*.



Summary

In this lesson, you should have learned how to:

- Configure and manage services in a RAC environment
- Use services with client applications
- Use services with the Database Resource Manager
- Use services with the Scheduler
- Set performance metric thresholds on services
- Configure services aggregation and tracing



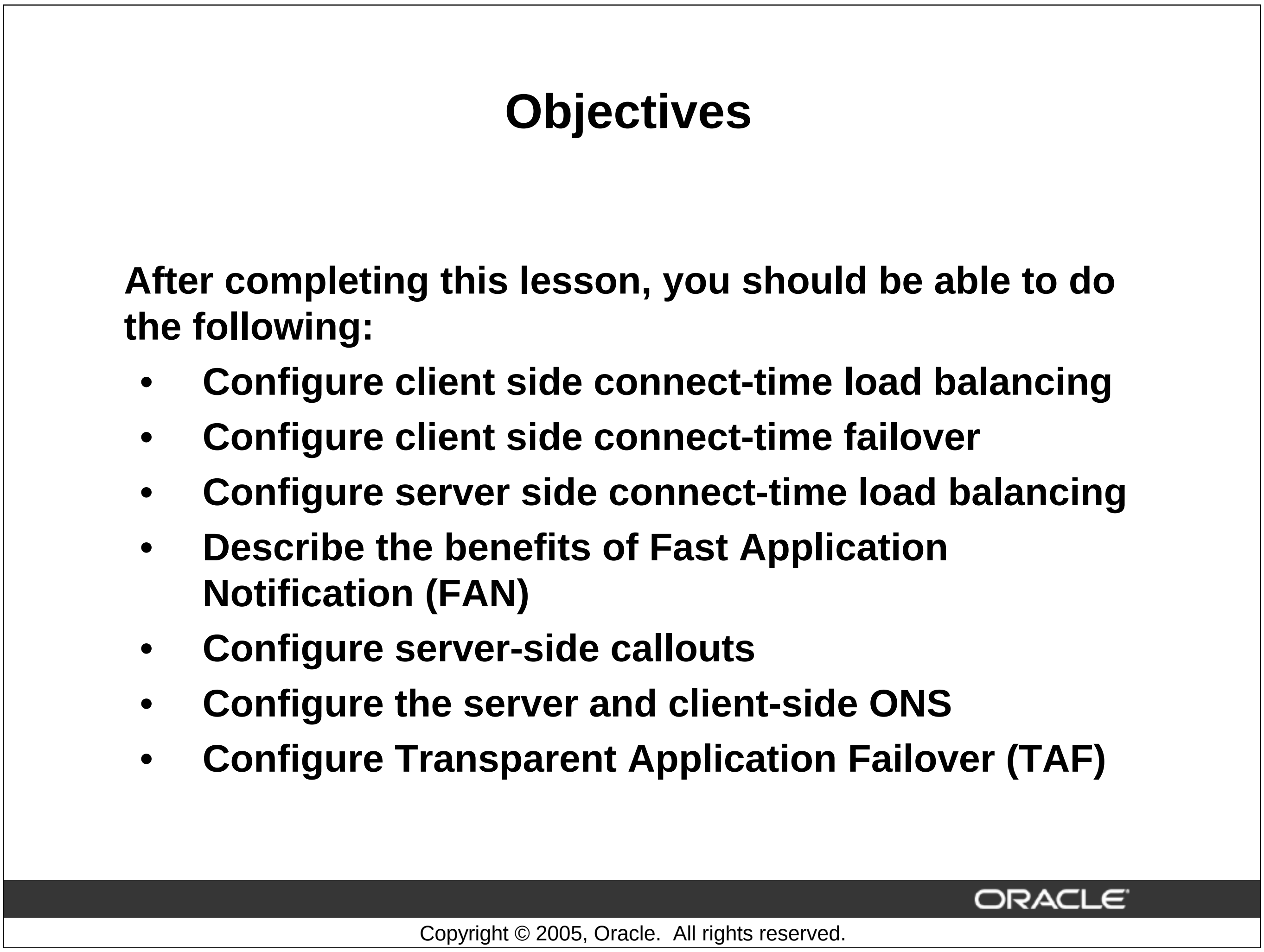
Practice 7 Overview

This practice covers the following topics:

- Defining services using DBCA
- Managing services using Database Control
- Using server-generated alerts in combination with services

8

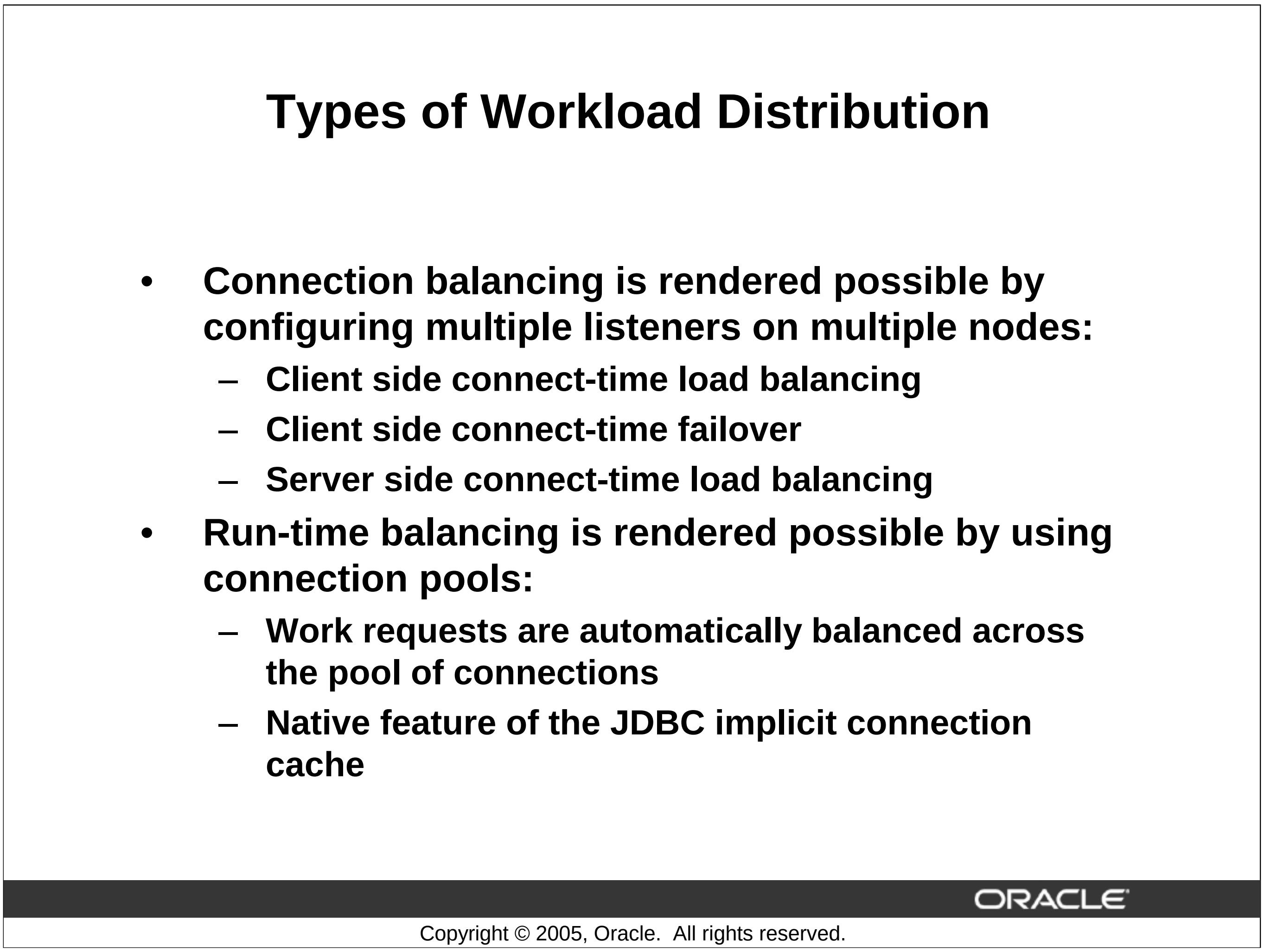
High Availability of Connections



Objectives

After completing this lesson, you should be able to do the following:

- Configure client side connect-time load balancing
- Configure client side connect-time failover
- Configure server side connect-time load balancing
- Describe the benefits of Fast Application Notification (FAN)
- Configure server-side callouts
- Configure the server and client-side ONS
- Configure Transparent Application Failover (TAF)



Types of Workload Distribution

- **Connection balancing is rendered possible by configuring multiple listeners on multiple nodes:**
  - **Client side connect-time load balancing**
  - **Client side connect-time failover**
  - **Server side connect-time load balancing**
- **Run-time balancing is rendered possible by using connection pools:**
  - **Work requests are automatically balanced across the pool of connections**
  - **Native feature of the JDBC implicit connection cache**

Types of Workload Distribution

With RAC, multiple listeners on multiple nodes can be configured to handle client connection requests for the same database service.

A multiple-listener configuration enables you to leverage the following failover and load-balancing features:

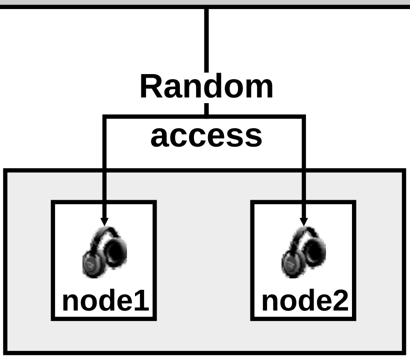
- Client side connect-time load balancing
- Client side connect-time failover
- Server side connect-time load balancing

These features can be implemented either one by one, or in combination with each other.

Moreover, if you are using connection pools, you can benefit from run-time balancing to distribute the client work requests across the pool of connections established by the middle tier. This possibility is offered by the Oracle JDBC implicit connection cache feature.

Client Side Connect-Time Load Balancing

```
ERP =
  (DESCRIPTION =
   (LOAD_BALANCE=ON)
    (ADDRESS_LIST =
      (ADDRESS=(PROTOCOL=TCP)(HOST=node1vip)(PORT=1521))
      (ADDRESS=(PROTOCOL=TCP)(HOST=node2vip)(PORT=1521))
    )
    (CONNECT_DATA=(SERVICE_NAME=ERP)))
```

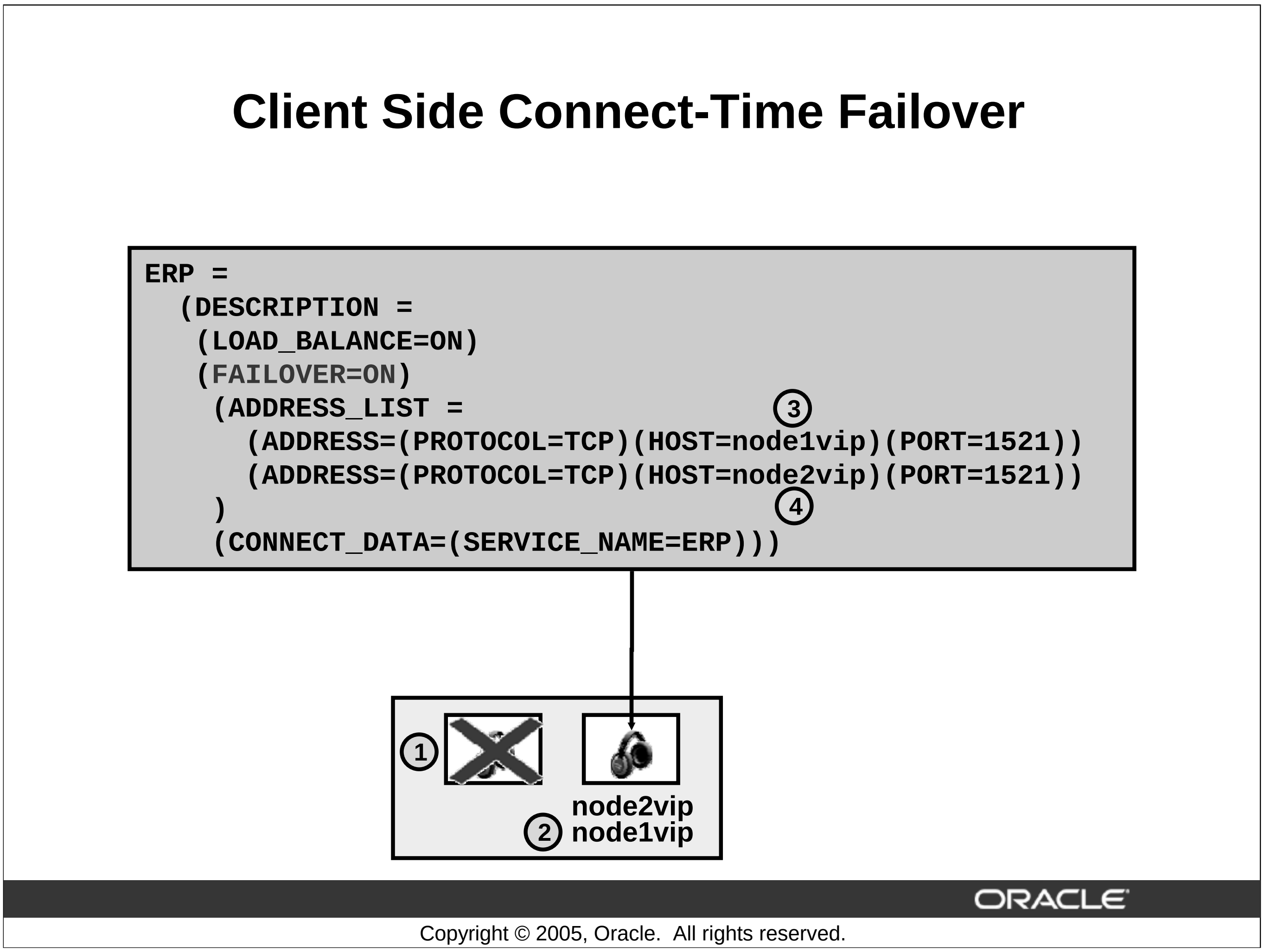


Client Side Connect-Time Load Balancing

The client side connect-time load balancing feature enables clients to randomize connection requests among a list of available listeners. Oracle Net progresses through the list of protocol addresses in a random sequence, balancing the load on the various listeners. Without this feature, Oracle Net always takes the first protocol address to attempt a connection.

You enable this feature by setting the `LOAD_BALANCE=ON` clause in the corresponding client side TNS entry.

Note: For a small number of connections, random sequence is not always even.



Client Side Connect-Time Failover

```
ERP =
  (DESCRIPTION =
   (LOAD_BALANCE=ON)
   (FAILOVER=ON)
    (ADDRESS_LIST =
      (ADDRESS=(PROTOCOL=TCP)(HOST=node1vip)(PORT=1521))
      (ADDRESS=(PROTOCOL=TCP)(HOST=node2vip)(PORT=1521))
    )
    (CONNECT_DATA=(SERVICE_NAME=ERP)))
```

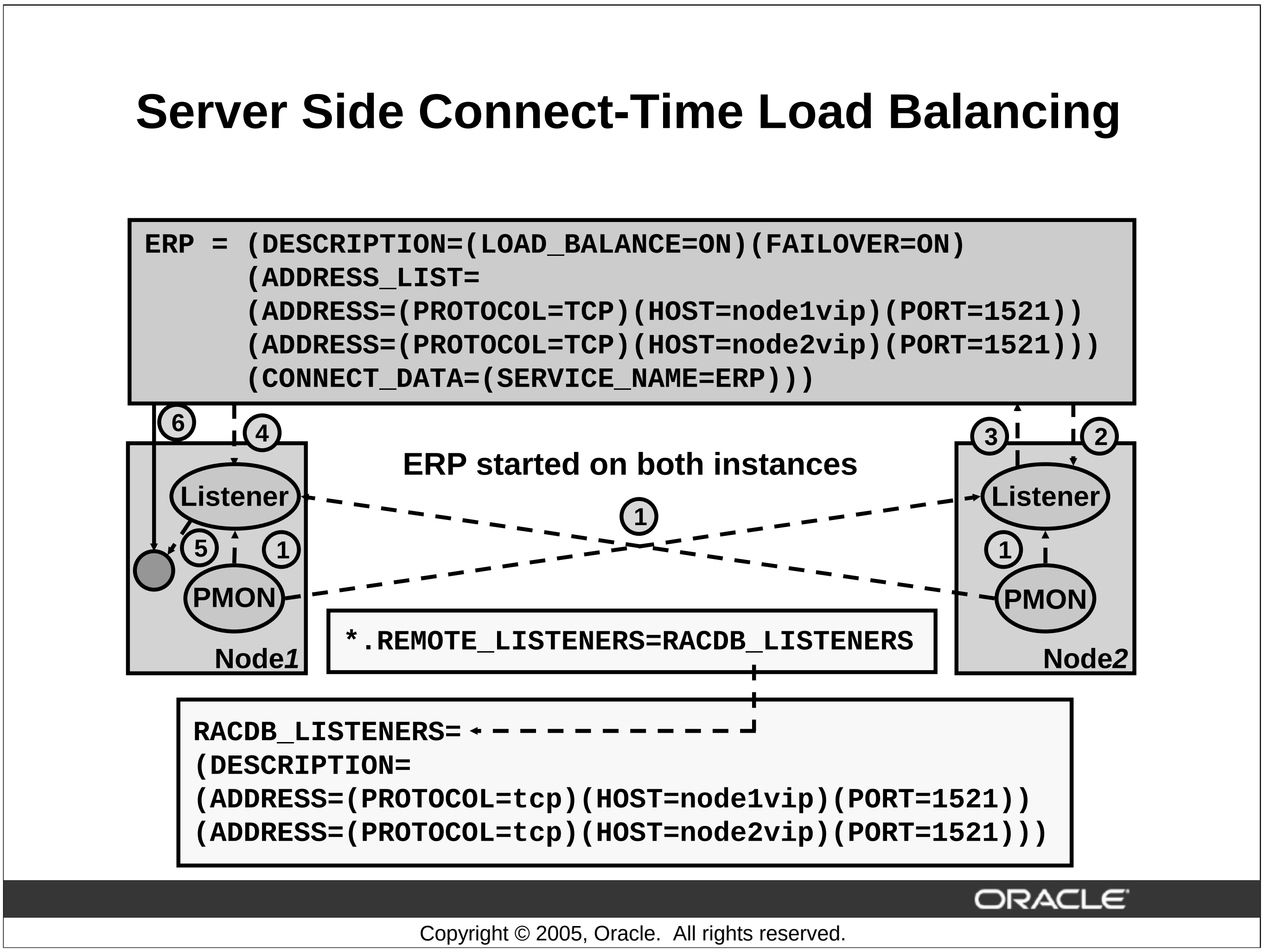
Client Side Connect-Time Failover

This feature enables clients to connect to another listener if the initial connection to the first listener fails. The number of listener protocol addresses in the connect descriptor determines how many listeners are tried. Without client side connect-time failover, Oracle Net attempts a connection with only one listener. As shown by the example, client side connect-time failover is enabled by setting the `FAILOVER=ON` clause in the corresponding client side TNS entry.

In the example, you expect the client to randomly attempt connections to either `NODE1VIP` or `NODE2VIP`, because `LOAD_BALANCE` is set to `ON`. In a case where one of the nodes is down, the client cannot know this. If a connection attempt is made to a down node, the client needs to wait until it receives the notification that the node is not accessible, before an alternate address in the `ADDRESS_LIST` is tried.

So, it is highly recommended to use virtual host names in the `ADDRESS_LIST` of your connect descriptors. Should a failure of a node occur (1), the Virtual IP address assigned to that node is failed over and brought online on another node in the cluster (2). Thus, all client connection attempts are still able to get a response from the IP address, without the need to wait for the operating system TCP/IP timeout (3). Therefore, clients get an immediate acknowledgement from the IP address, and are notified that the service on that node is not available. The next address in the `ADDRESS_LIST` can then be tried immediately with no delay (4).

Note: If using connect-time failover, do no set `GLOBAL_DBNAME` in your `listener.ora` file.



Server Side Connect-Time Load Balancing

```
ERP = (DESCRIPTION=(LOAD_BALANCE=ON)(FAILOVER=ON)
      (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=TCP)(HOST=node1vip)(PORT=1521))
      (ADDRESS=(PROTOCOL=TCP)(HOST=node2vip)(PORT=1521)))
      (CONNECT_DATA=(SERVICE_NAME=ERP)))
```

ERP started on both instances

```
*.REMOTE_LISTENERS=RACDB_LISTENERS
```

```
RACDB_LISTENERS=
(DESCRIPTION=
(ADDRESS=(PROTOCOL=tcp)(HOST=node1vip)(PORT=1521))
(ADDRESS=(PROTOCOL=tcp)(HOST=node2vip)(PORT=1521)))
```

Server Side Connect-Time Load Balancing

The slide shows you how listeners evenly distribute service connection requests across a RAC cluster. Here, the client application connects to the ERP service. On the server side, the database is using the dynamic service registration feature. This allows the PMON process of each instance in the cluster to register workload information with each listener in the cluster (1). Each listener is then aware of which instance has a particular service started, as well as the absolute session count per instance, and the run queue length of each node.

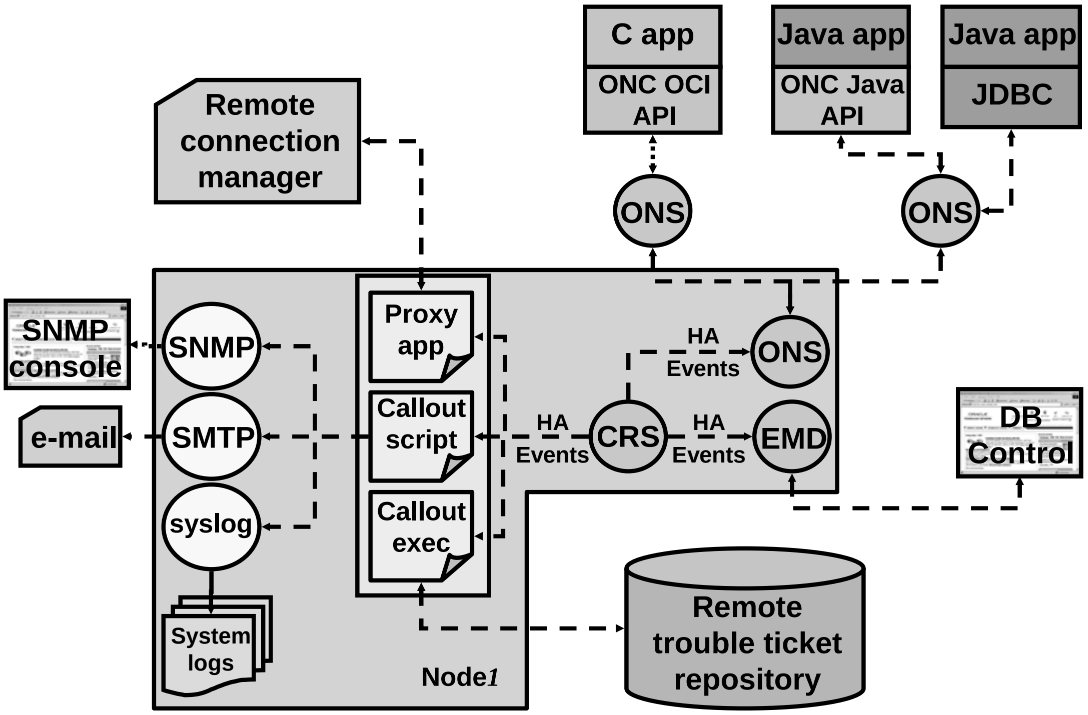
You configure this feature by setting the REMOTE_LISTENER initialization parameter of each instance to a TNS name that describes the list of all available listeners. The slide shows the shared entry in the SPFILE as well as its corresponding server-side TNS entry.

Depending on the load information sent by each PMON process, a listener, by default, redirects the incoming connection request (2) to the listener of the least-loaded instance on the least-loaded node (3). If you want to guarantee that the listener redirects the connection request to the least loaded instance, then set the PREFER_LEAST_LOADED_NODE_[listener_name] to OFF in the listener.ora files. This is better for applications that use connection pools.

In the example, the listener on NODE2 is tried first. Based on workload information dynamically updated by PMON processes, the listener determines that the best instance is the one residing on NODE1. The listener redirects the connection request to the listener on NODE1 (4). That listener then starts a dedicated server process (5), and the connection is made to that process (6).

Note: For more information, refer to the *Net Services Administrator’s Guide.*

Fast Application Notification: Overview



Fast Application Notification: Overview

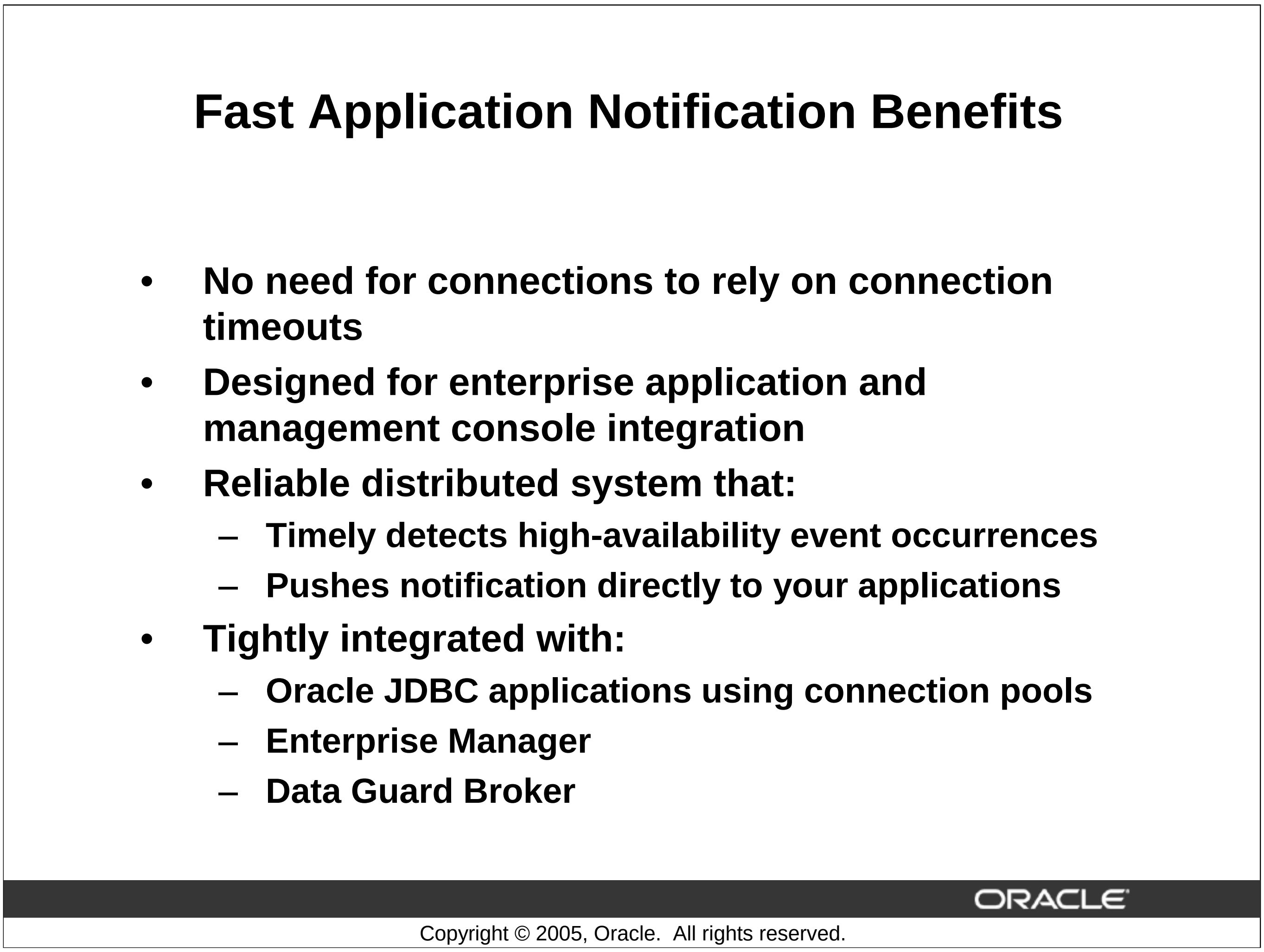
A typical RAC installation may generate a myriad of cluster events, which are used by the internal components of a cluster to manage its high availability. Fast Application Notification (FAN) is a feature that filters and publishes only those high-availability events that are considered as meaningful to specific targets. There are currently three main methods to integrate FAN with your applications:

- Server-side callouts that are using wrapper scripts or executables installed in a particular directory on the database server. Callouts should be coded to affect local components. However, callouts can also affect remote components if proxy applications are deployed. Proxy applications may include existing server components that expose command-line interfaces, such as SMTP mail servers and SNMP agents, or custom programs that connect exclusively to a specific set of remote applications.
- The Oracle Notification Client (ONC) API, which provides a distributed, publish-and-subscribe protocol over TCP/IP. This API is available for C and Java applications. With this method, RAC is the default event publisher, and any custom C or Java application reachable from the database server, simply links the ONC API library and subscribes to FAN events. Upon reception, the application can execute event-handling actions. An Oracle Notification Services (ONS) listening daemon needs to be started on each host where one or more ONC subscribers are present. This includes each node of your RAC environment.

Fast Application Notification: Overview (continued)

- Applications can also enable the Oracle JDBC implicit connection cache and let FAN events be handled by the Oracle JDBC libraries directly. Using this method, you no longer need to write any custom Java or C code to handle FAN events, nor invoke the ONC API directly. Think of this as an out-of-the-box integration with the ONS.

Note: Enterprise manager is tightly integrated with FAN, and no configuration is required.



Fast Application Notification Benefits

- No need for connections to rely on connection timeouts
- Designed for enterprise application and management console integration
- Reliable distributed system that:
  - Timely detects high-availability event occurrences
  - Pushes notification directly to your applications
- Tightly integrated with:
  - Oracle JDBC applications using connection pools
  - Enterprise Manager
  - Data Guard Broker

Fast Application Notification Benefits

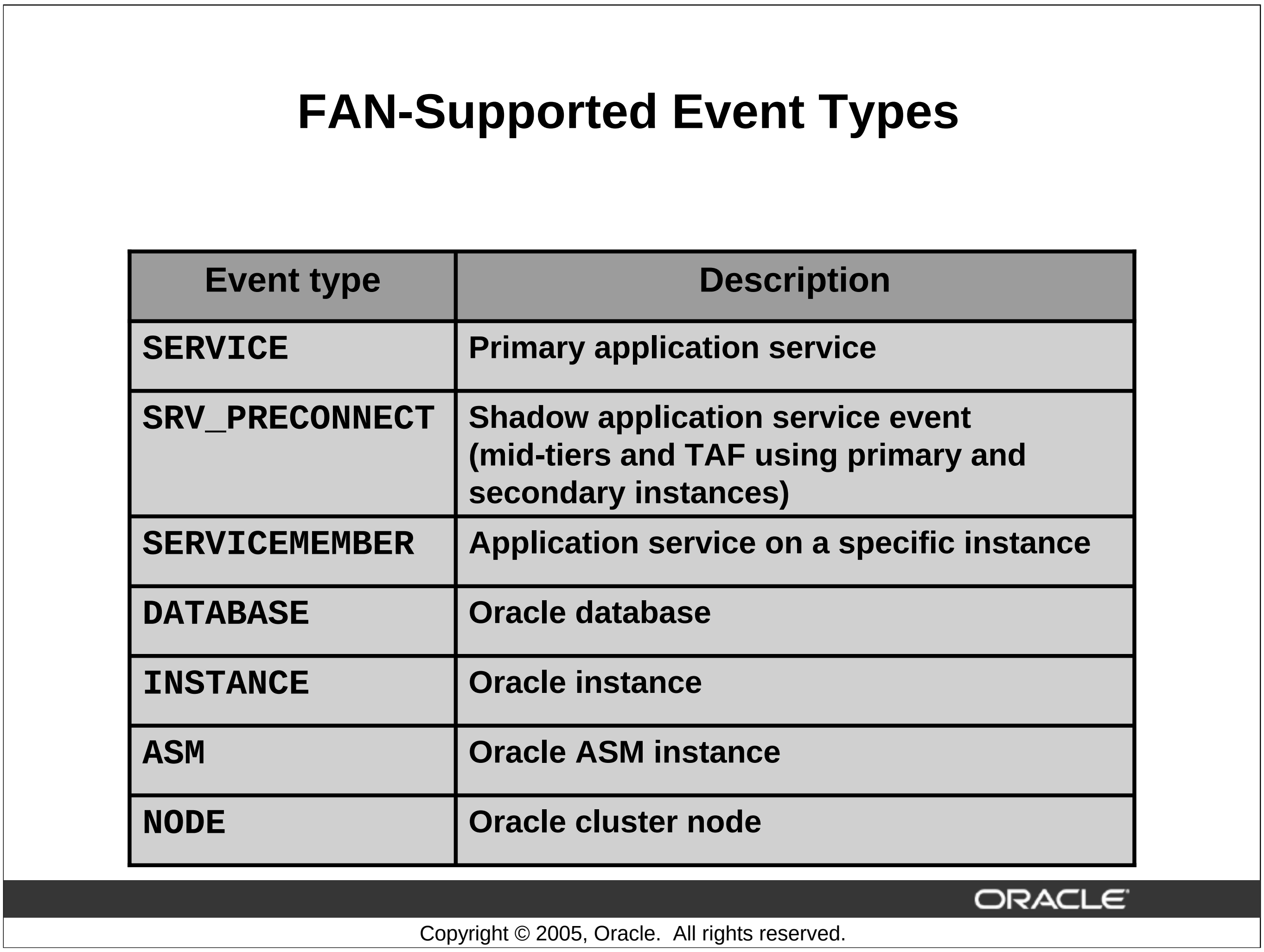
Traditionally, client or mid-tier applications connected to the database have relied on connection timeouts, out-of-band polling mechanisms, or other custom solutions to realize that a system component has failed. This approach has huge implications in application availability, because down times are extended and more noticeable.

With FAN, important high-availability events are pushed as soon as they are detected, which results in a more efficient use of existing computing resources, and a better integration with your enterprise applications, including mid-tier connection managers, or IT management consoles, including trouble ticket loggers and e-mail/paging servers.

FAN is in fact a distributed system that is enabled on each participating node. This makes it very reliable and fault-tolerant because the failure of one component is detected by another one. Therefore, event notification can be detected and pushed by any of the participating nodes.

FAN events are tightly integrated with Oracle Data Guard Broker, Oracle JDBC implicit connection cache, and Enterprise Manager. Oracle Database 10*g* JDBC applications managing connection pools do not need custom code development. They are automatically integrated with the ONS if implicit connection cache and fast connection failover are enabled.

Note: For more information about FAN and Data Guard integration, refer to the lesson titled *Design for High Availability* in this course.



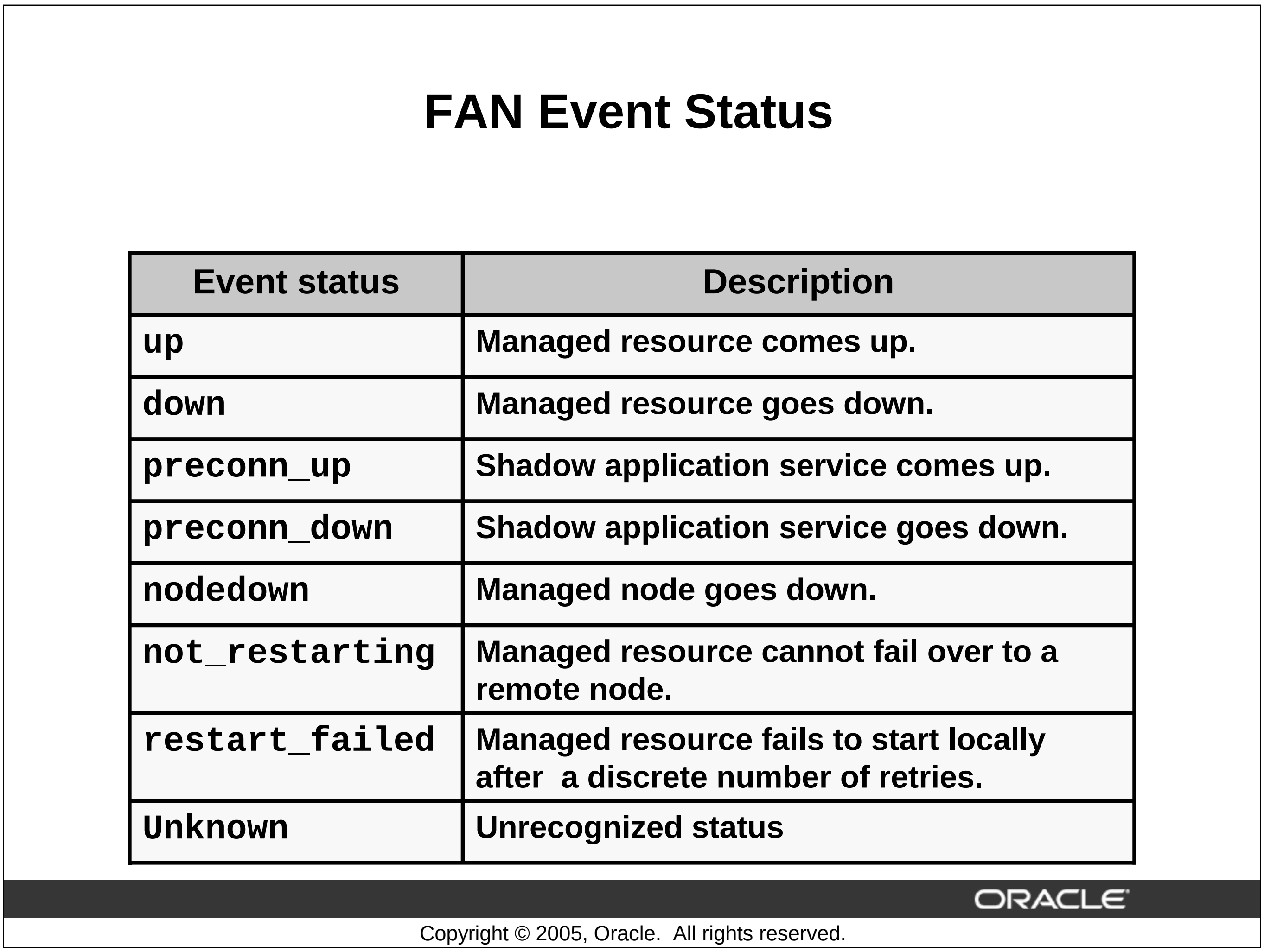
FAN-Supported Event Types

Event type	Description
`SERVICE`	Primary application service
`SRV_PRECONNECT`	Shadow application service event (mid-tiers and TAF using primary and secondary instances)
`SERVICEMEMBER`	Application service on a specific instance
`DATABASE`	Oracle database
`INSTANCE`	Oracle instance
`ASM`	Oracle ASM instance
`NODE`	Oracle cluster node

FAN-Supported Event Types

FAN delivers events pertaining to the above list of managed cluster resources. The table describes each of the resources.

Note: `SRV_PRECONNECT` is discussed later in this lesson.

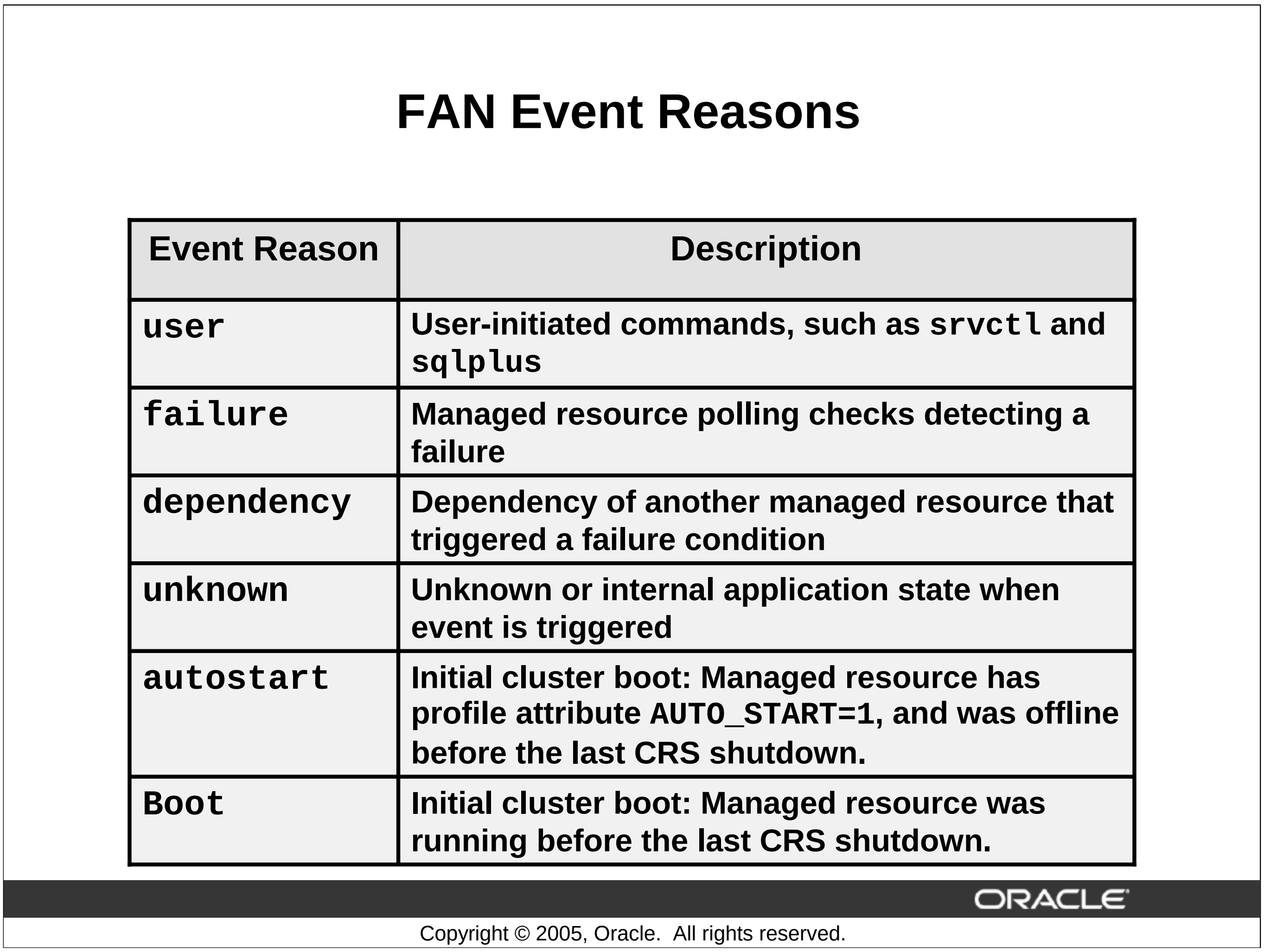


FAN Event Status

Event status	Description
`up`	Managed resource comes up.
`down`	Managed resource goes down.
`preconn_up`	Shadow application service comes up.
`preconn_down`	Shadow application service goes down.
`nodedown`	Managed node goes down.
`not_restarting`	Managed resource cannot fail over to a remote node.
`restart_failed`	Managed resource fails to start locally after a discrete number of retries.
`Unknown`	Unrecognized status

FAN Event Status

This table describes the event status for each of the managed cluster resources seen previously.

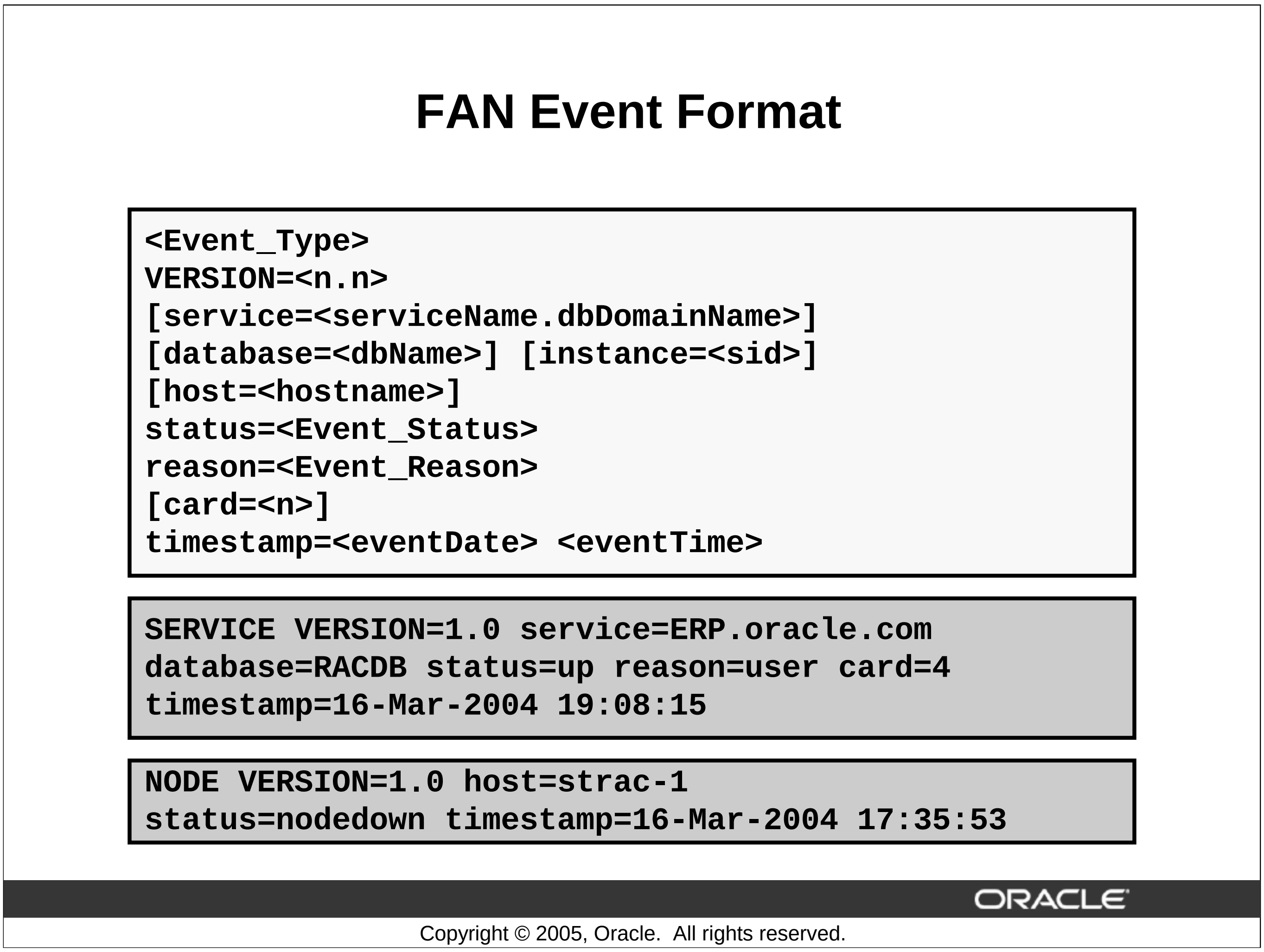


FAN Event Reasons

Event Reason	Description
`user`	User-initiated commands, such as `srvctl` and `sqlplus`
`failure`	Managed resource polling checks detecting a failure
`dependency`	Dependency of another managed resource that triggered a failure condition
`unknown`	Unknown or internal application state when event is triggered
`autostart`	Initial cluster boot: Managed resource has profile attribute `AUTO_START=1`, and was offline before the last CRS shutdown.
`Boot`	Initial cluster boot: Managed resource was running before the last CRS shutdown.

FAN Event Reasons

The event status for each managed resource is associated with an event reason. The reason further describes what triggered the event. The above table gives you the list of possible reasons with a corresponding description.



FAN Event Format

```
<Event_Type>
VERSION=<n.n>
[service=<serviceName.dbDomainName>]
[database=<dbName>] [instance=<sid>]
[host=<hostname>]
status=<Event_Status>
reason=<Event_Reason>
[card=<n>]
timestamp=<eventDate> <eventTime>
```

```
SERVICE VERSION=1.0 service=ERP.oracle.com
database=RACDB status=up reason=user card=4
timestamp=16-Mar-2004 19:08:15
```

```
NODE VERSION=1.0 host=strac-1
status=nodedown timestamp=16-Mar-2004 17:35:53
```

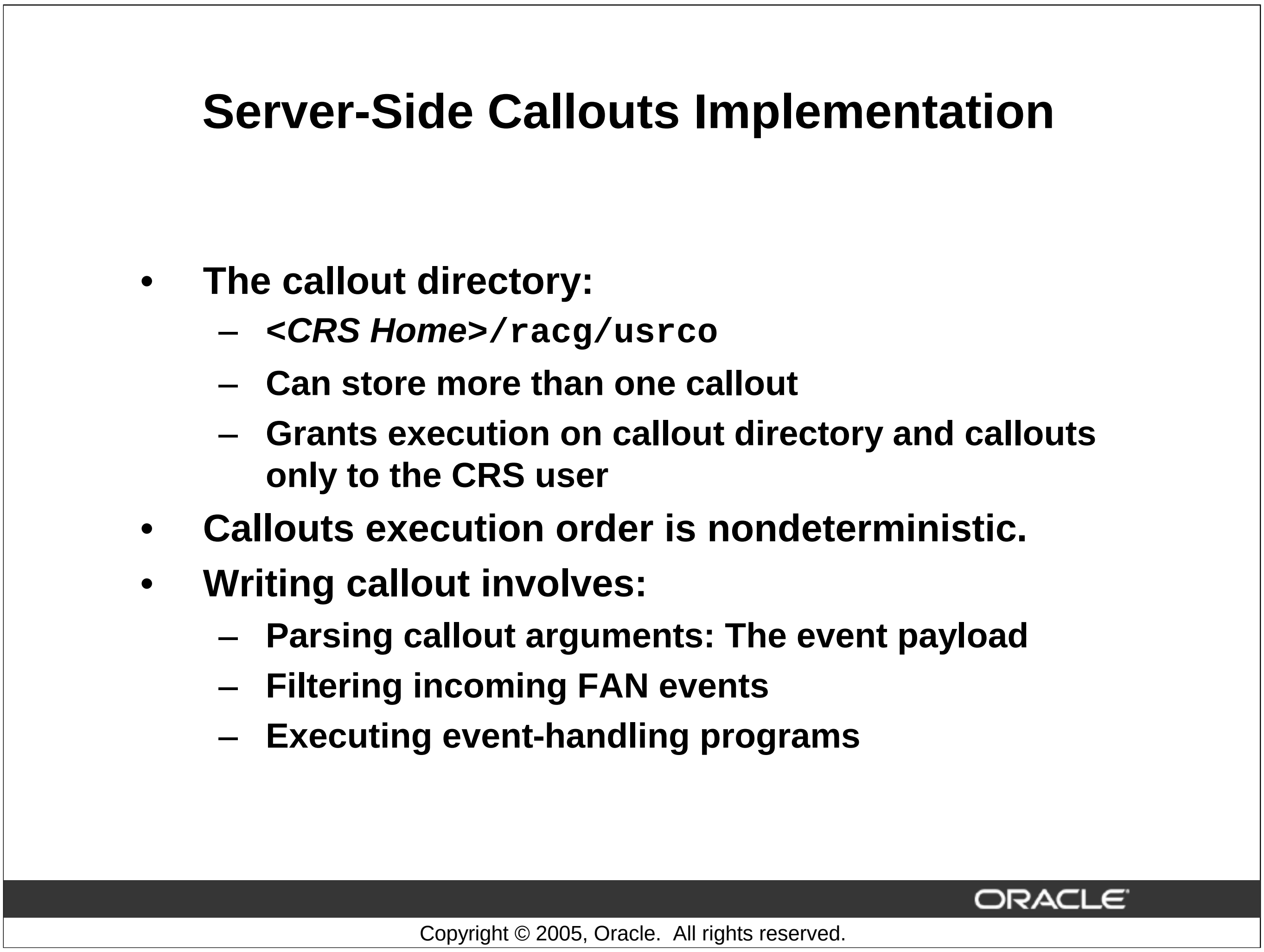
ORACLE

FAN Event Format

In addition to its type, status, and reason, a FAN event has other payload fields to further describe the unique cluster resource whose status is being monitored and published:

- The event payload version which is currently 1.0.
- The name of the primary or shadow application service. This name is excluded from NODE events.
- The name of the RAC database which is also excluded from NODE events.
- The name of the RAC instance which is excluded from SERVICE, DATABASE, and NODE events.
- The name of the cluster host machine which is excluded from SERVICE and DATABASE events.
- The service cardinality which is excluded from all events except for SERVICE status=up events.
- The server-side date and time when the event is detected.

The general FAN event format is described in the slide along with possible FAN event examples. Note the differences in event payload for each FAN event type.



Server-Side Callouts Implementation

- The callout directory:
  - *<CRS Home>*`/racg/usrco`
  - Can store more than one callout
  - Grants execution on callout directory and callouts only to the CRS user
- Callouts execution order is nondeterministic.
- Writing callout involves:
  - Parsing callout arguments: The event payload
  - Filtering incoming FAN events
  - Executing event-handling programs

Server-Side Callouts Implementation

Each database event detected by the RAC High Availability (HA) framework results in the execution of each callout deployed in the standard CRS callout directory. On UNIX, it is `$ORACLE_BASE/product/10.1.0/crs/racg/usrco`. Unless your CRS home directory is shared across the network, you must deploy each new callout on each RAC node.

The order in which these callouts are executed is nondeterministic. However, RAC guarantees that all callouts are invoked once for each recognized event in an asynchronous fashion. Thus, it is recommended to merge callouts whose executions need to be in a particular order.

You can install as many callout scripts or programs as your business requires, provided that each callout does not incur expensive operations that delay the propagation of HA events. If many callouts are going to be written to perform different operations based on the event received, it might be more efficient to write a single callout program that merges each single callout.

Writing server-side callouts involve the above shown steps. In order for your callout to identify an event, it must parse the event payload sent by the RAC HA framework to your callout. After the sent event is identified, your callout can filter it to avoid execution on each event notification. Then, your callout needs to implement a corresponding event-handler that depends on the event itself and the recovery process required by your business.

Note: As a security measure, make sure that the callout directory and its contained callouts have write permissions only to the system user who installed CRS.

Server-Side Callout Parse: Example

```
#!/bin/sh
NOTIFY_EVENTTYPE=$1
for ARGS in $*; do
   PROPERTY=`echo $ARGS | $AWK -F"=" '{print $1}'`
   VALUE=`echo $ARGS | $AWK -F"=" '{print $2}'`
   case $PROPERTY in
     VERSION|version)     NOTIFY_VERSION=$VALUE ;;
     SERVICE|service)     NOTIFY_SERVICE=$VALUE ;;
     DATABASE|database)   NOTIFY_DATABASE=$VALUE ;;
     INSTANCE|instance)   NOTIFY_INSTANCE=$VALUE ;;
     HOST|host)           NOTIFY_HOST=$VALUE ;;
     STATUS|status)       NOTIFY_STATUS=$VALUE ;;
     REASON|reason)       NOTIFY_REASON=$VALUE ;;
     CARD|card)           NOTIFY_CARDINALITY=$VALUE ;;
     TIMESTAMP|timestamp) NOTIFY_LOGDATE=$VALUE ;;
     ??:??:??)            NOTIFY_LOGTIME=$PROPERTY ;;
   esac
done
```

Server-Side Callout Parse: Example

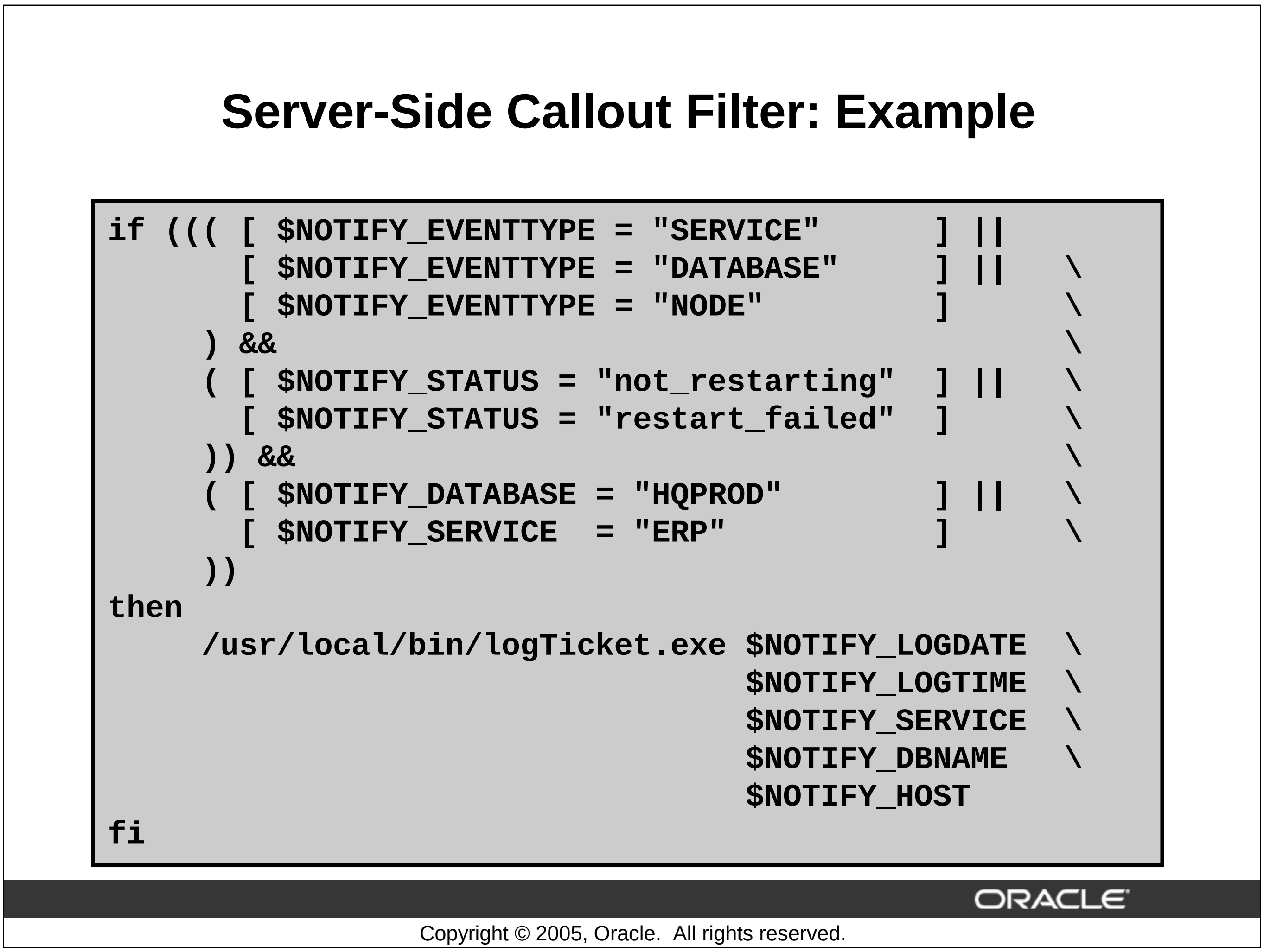
Unless you want your callouts to be executed on each event notification, you must first identify the event parameters that are passed automatically to your callout during its execution. The example in the slide above shows you how to parse these arguments by using a sample Bourne shell script.

The first argument that is passed to your callout is the type of event that is detected. Then, depending on the event type, a set of `PROPERTY=VALUE` strings are passed to identify exactly the event itself.

The above script identifies the event type and each pair of `PROPERTY=VALUE` string. The data is then dispatched into a set of variables that can be used later in the callout for filtering purposes.

As mentioned in the previous slide, it might be better to have a single callout that parses the event payload, and then executes a function or another program based on the information in the event, as opposed to having to filter information in each callout. This becomes necessary only if many callouts are required.

Note: In the above example, `#!/bin/sh` should be the very first line of the script to be correctly interpreted by the shell script interpreter.



Server-Side Callout Filter: Example

```
if ((( [ $NOTIFY_EVENTTYPE = "SERVICE"      ] ||
       [ $NOTIFY_EVENTTYPE = "DATABASE"     ] ||   \
       [ $NOTIFY_EVENTTYPE = "NODE"         ]      \
     ) &&                                          \
     ( [ $NOTIFY_STATUS = "not_restarting"  ] ||   \
       [ $NOTIFY_STATUS = "restart_failed"  ]      \
     )) &&                                         \
     ( [ $NOTIFY_DATABASE = "HQPROD"        ] ||   \
       [ $NOTIFY_SERVICE  = "ERP"           ]      \
     ))
then
     /usr/local/bin/logTicket.exe $NOTIFY_LOGDATE  \
                                  $NOTIFY_LOGTIME  \
                                  $NOTIFY_SERVICE  \
                                  $NOTIFY_DBNAME   \
                                  $NOTIFY_HOST
fi
```

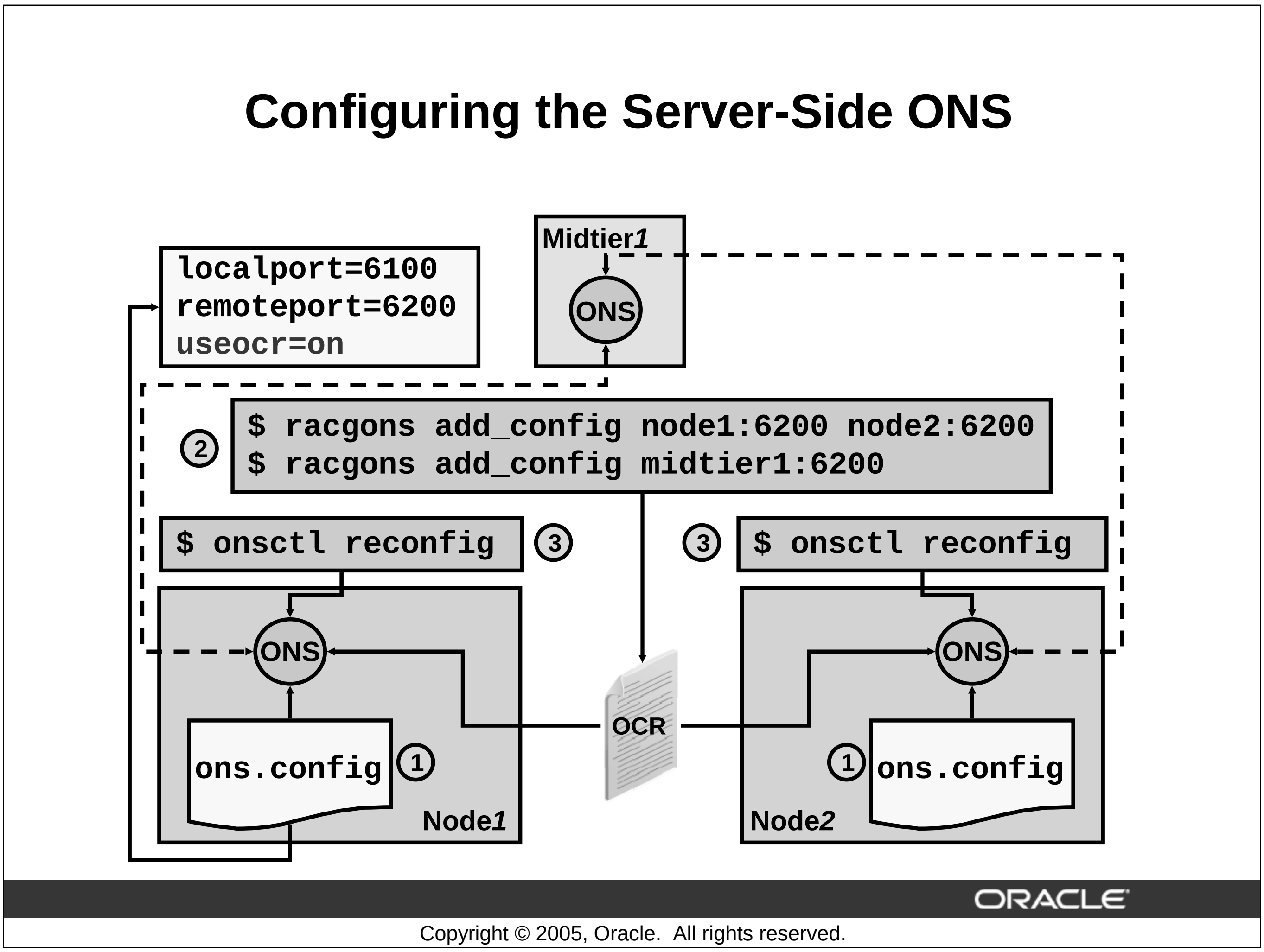
Server-Side Callout Filter: Example

The example in the slide above shows you a way to filter FAN events from a callout script. This example is based on the example in the previous slide.

Now that the event characteristics are identified, this script triggers the execution of the trouble logging program `/usr/local/bin/logTicket.exe` only when the RAC HA framework posts a `SERVICE`, `DATABASE` or `NODE` event type, with a status sets to either `not_restarting`, or `restart_failed`, and only for the production `HQPROD` RAC database or the `ERP` service.

It is assumed that the `logTicket.exe` program is already created and that it takes the above shown arguments.

It is also assumed that a ticket is logged only for `not_restarting` or `restart_failed` events, because they are the ones that exceeded internally monitored timeouts and seriously need human intervention for full resolution.



Configuring the Server-Side ONS

```
localport=6100
remoteport=6200
useocr=on
```

```
$ racgons add_config node1:6200 node2:6200
$ racgons add_config midtier1:6200
```

```
$ onsctl reconfig
```

Configuring the Server-Side ONS

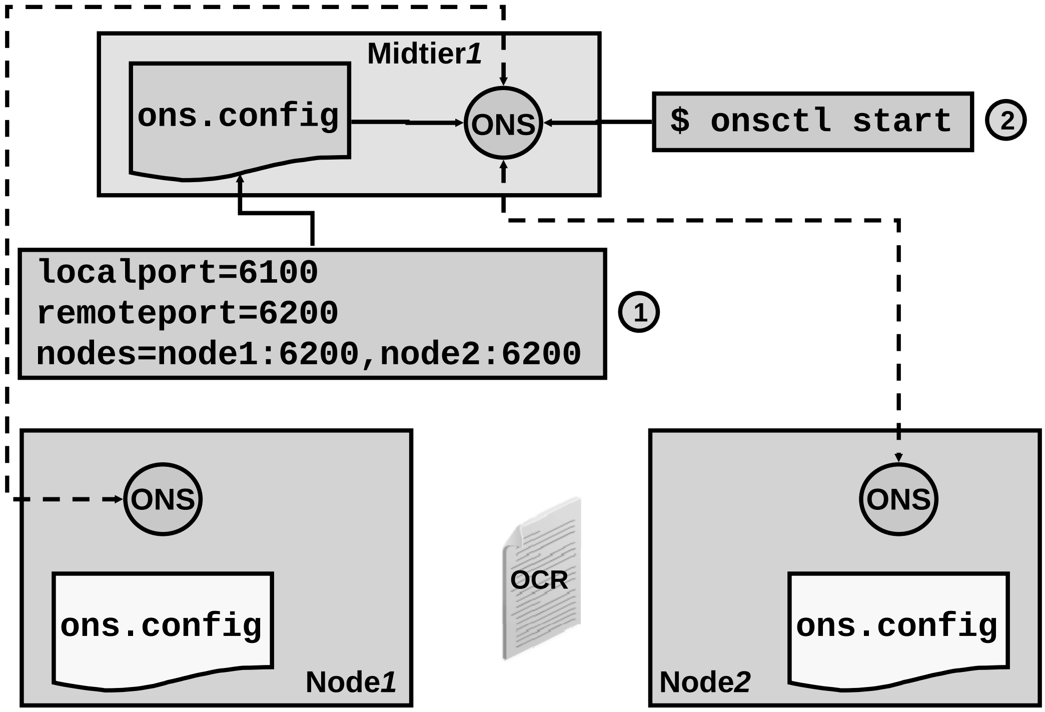
The ONS configuration is controlled by the `$ORACLE_HOME/opmn/conf/ons.config` configuration file. This file is automatically created during installation. There are three important parameters that should always be configured for each ONS:

- The first is `localport`, the port that the ONS uses to talk to local clients.
- The second is `remoteport`, the port that the ONS uses to talk to other ONS daemons.
- The third parameter is called `nodes`. It specifies the list of other ONS daemons to talk to. This list should include all RAC ONS daemons, and all mid-tier ONS daemons. Node values are given as either host names or IP addresses followed by its `remoteport`. Instead, you can store this data in the Oracle Cluster Registry (OCR) using the `racgons add_config` command and having the `useocr` parameter set to `on` in the `ons.config` file. By storing nodes information in the OCR, you do not need to edit a file on every node to change the configuration. Instead, you only need to run a single command on one of the cluster nodes.

In the slide above, it is assumed that ONS daemons are already started on each cluster node. This should be the default situation after a correct RAC installation. However, if you want to use the OCR, you should edit the `ons.config` file on each node, and then add the configuration to the OCR before reloading it on each cluster node. This is illustrated in the slide above.

Note: You should run `racgons` whenever you add or remove a node that runs an ONS daemon. To remove a node from the OCR, you can use the `racgons remove_config` command.

Configuring the Client-Side ONS



```
$ onsctl start
```

```
localport=6100
remoteport=6200
nodes=node1:6200,node2:6200
```

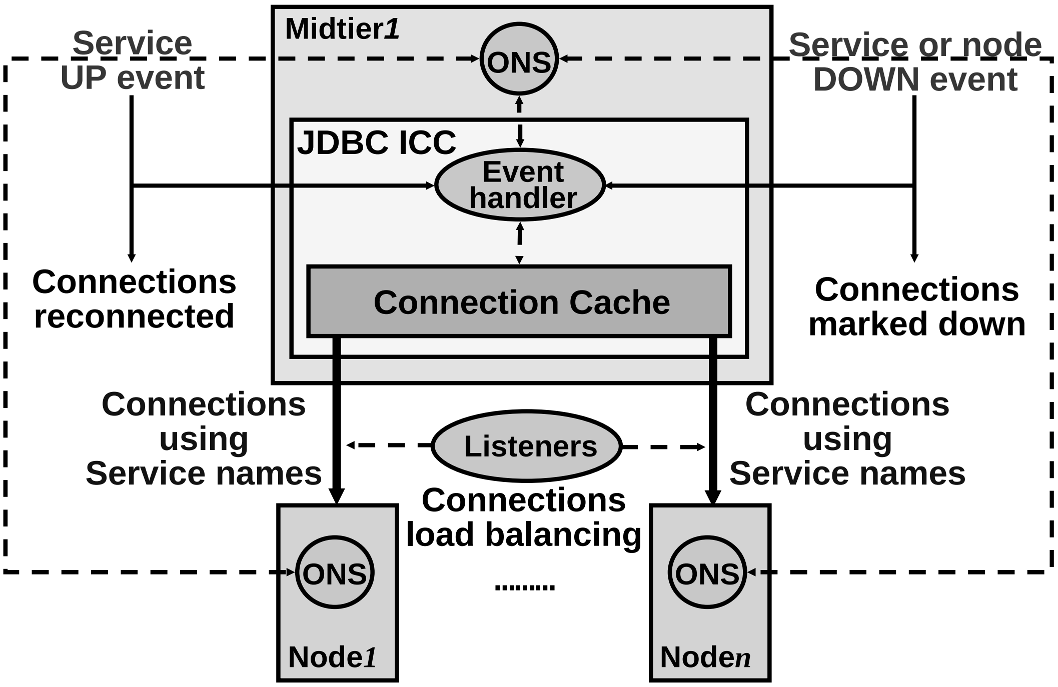
Configuring the Client-Side ONS

You must install the ONS on each host where you have client applications that need to be integrated with FAN. Most of the time, these hosts play the role of a mid-tier application server. The ONS is shipped as part of the 10*g* Release 1 (10.1).

Therefore, on the client side, you must configure all the RAC nodes in the ONS configuration file. A sample configuration file might look like the one shown in the slide.

After configuring the ONS, you start the ONS daemon with the `onsctl start` command. It is your responsibility to make sure that an ONS daemon is running at all times. You can check that the ONS daemon is active by executing the `onsctl ping` command.

JDBC Fast Connection Failover: Overview



JDBC Fast Connection Failover: Overview

Oracle Application Server 10*g* integrates JDBC Implicit Connection Cache (ICC) with the ONC by having application developers enable Fast Connection Failover (FCF). FCF works in conjunction with the JDBC ICC to quickly and automatically recover lost or damaged connections. This automatic connection management results from FAN events received by the local ONS daemon, and handled by a special event handler thread. Both JDBC thin and JDBC OCI drivers are supported.

Therefore, if JDBC ICC and FCF are enabled, your Java program automatically becomes an ONC subscriber without having to manage FAN events directly.

Whenever a service or node down event is received by the mid-tier ONS, the event handler automatically mark the corresponding connections as down. This prevents applications that request connections from the cache, from receiving invalid or bad connections.

Whenever a service up event is received by the mid-tier ONS, the event handler recycles some unused connections, and reconnect them using the event service name. The number of recycled connections is automatically determined by the connection cache. Because the listeners perform connect-time load balancing, this will automatically rebalance the connections across the preferred instances of the service without waiting for application connection requests or retries.

Note: For more information, refer to the *Oracle Database JDBC Developer’s Guide and Reference*.

JDBC Fast Connection Failover Benefits

- **Database connections are balanced across all preferred RAC instances.**
- **Database connections are anticipated.**
- **Database connection failures are immediately detected and stopped.**
- **No need to add extra coding to your existing Java applications except to set the `setFastConnectionFailoverEnabled` flag.**

ORACLE

JDBC Fast Connection Failover Benefits

By enabling the JDBC implicit connection cache and fast connection failover features, your existing Java applications connecting through Oracle JDBC and application services benefit from the following:

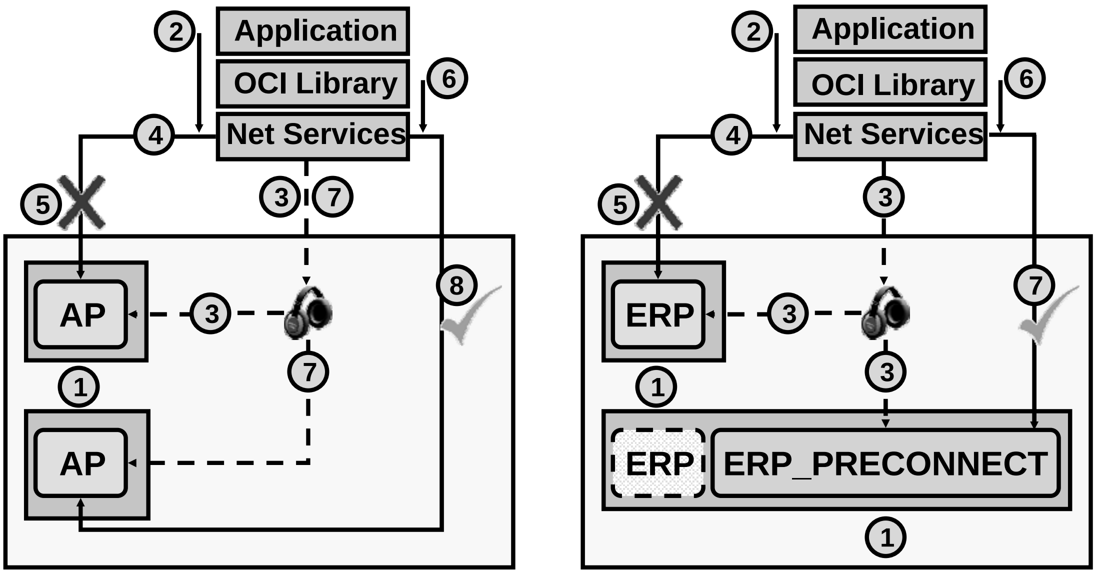
- All database connections are balanced across all RAC instances that support the new service name, instead of having the first batch of sessions routed to the first RAC instance. Connection pools are rebalanced upon service, instance, or node up events.
- The connection cache immediately starts placing connections to a particular RAC instance when a new service is started on that instance.
- The connection cache immediately shuts down stale connections to RAC instances where the service is stopped on that instance, or whose node goes down.
- Your Java program automatically becomes an ONS subscriber without having to manage FAN events directly. The data source needs to have the attribute `setFastConnectionFailoverEnabled` set to `true` to enable fast connection failover in the implicit connection cache.

Note: An exception is immediately thrown as soon as the service status becomes `not_restarting`, which avoids wasteful service connection retries.

Transparent Application Failover: Overview

TAF Basic

TAF Preconnect



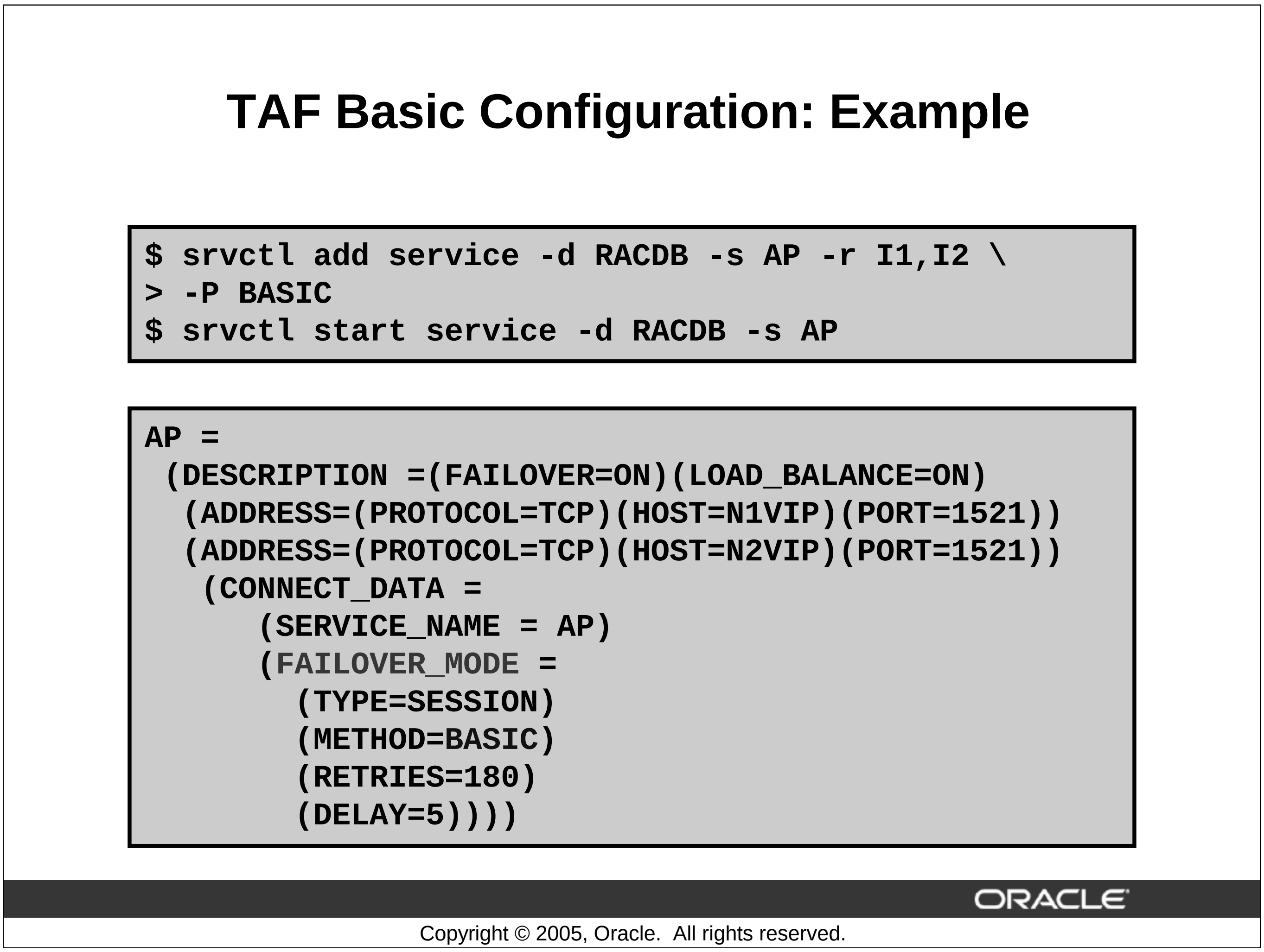
ORACLE

Transparent Application Failover (TAF): Overview

TAF is a run-time feature of the OCI driver. It enables your application to automatically reconnect to the service should the initial connection fails. During the reconnection, although your active transactions are rolled back, TAF can optionally resume the execution of a `SELECT` statement that was in progress. TAF supports two failover methods:

- With the `BASIC` method, the reconnection is established at failover time. The initial connection (2) is made after the corresponding service has been started on the nodes (1). The listener establishes the connection (3), and your application accesses the database (4) until the connection fails (5). Your application then receives an error the next time it tries to access the database (6). Then, the OCI driver reconnects to the same service (7), and the next time your application tries to access the database, it transparently uses the newly created connection (8).
- The `PRECONNECT` method, is similar to the `BASIC` method except that it is during the initial connection that a shadow connection is also created to anticipate the failover. TAF guarantees that the shadow connection is always created on the available instances of your service. This is achieved by using an automatically created and maintained shadow service on available instances only.

Note: Optionally you can register TAF callbacks with the OCI layer. These callback functions are automatically invoked at failover detection and allow you to have some control of the failover process. For more information, refer to the *Oracle Call Interface Programmer’s Guide*.



TAF Basic Configuration: Example

```
$ srvctl add service -d RACDB -s AP -r I1,I2 \
> -P BASIC
$ srvctl start service -d RACDB -s AP
```

```
AP =
 (DESCRIPTION =(FAILOVER=ON)(LOAD_BALANCE=ON)
  (ADDRESS=(PROTOCOL=TCP)(HOST=N1VIP)(PORT=1521))
  (ADDRESS=(PROTOCOL=TCP)(HOST=N2VIP)(PORT=1521))
   (CONNECT_DATA =
      (SERVICE_NAME = AP)
      (FAILOVER_MODE =
        (TYPE=SESSION)
        (METHOD=BASIC)
        (RETRIES=180)
        (DELAY=5))))
```

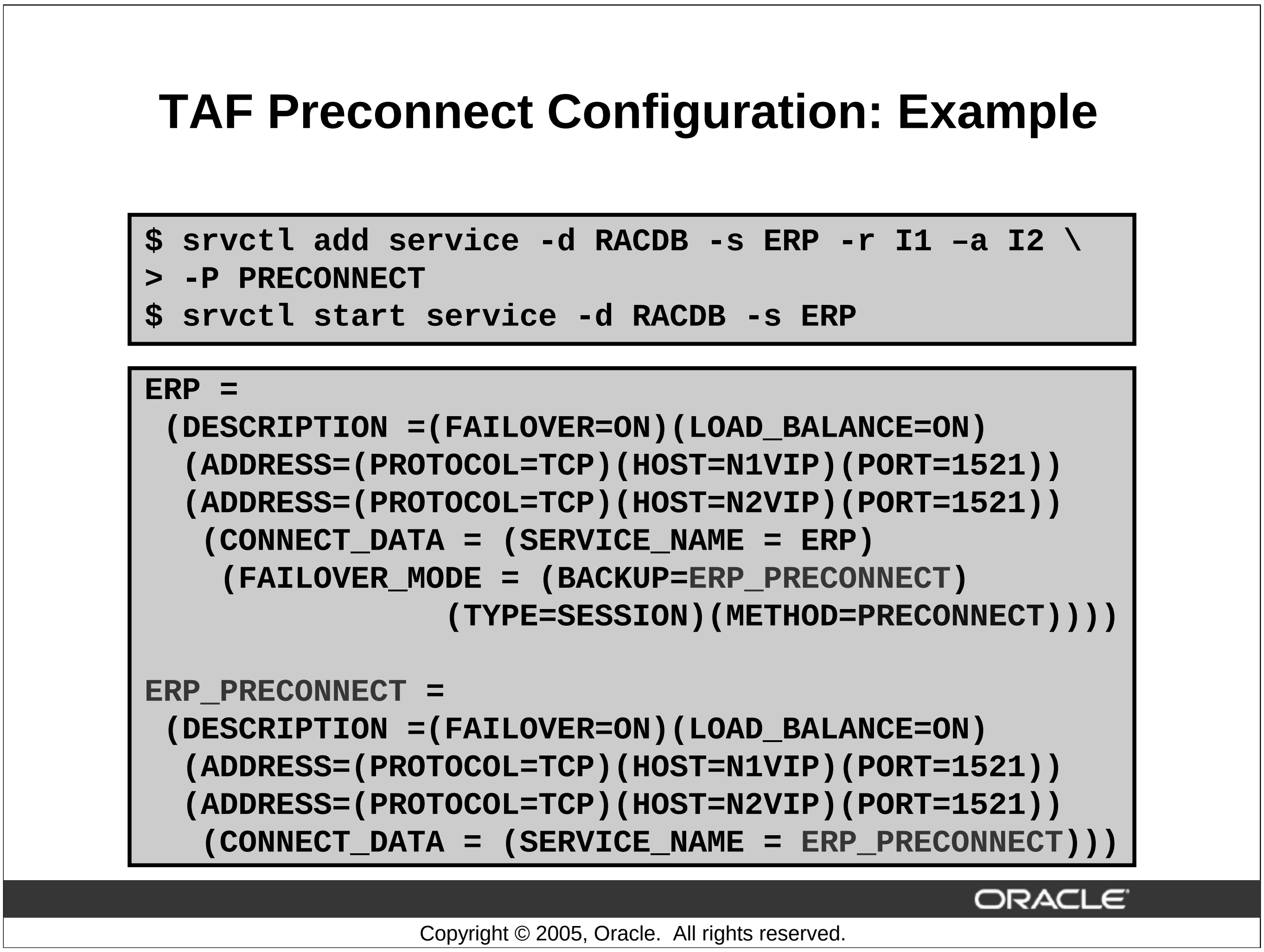
TAF Basic Configuration: Example

Before using TAF, it is recommended that you create and start a service that is used during connections. By doing so, you benefit from TAF and services integration. When you want to use `BASIC` TAF with a service, you should have the `-P BASIC` option when creating the service. After the service is created, you simply start it on your database.

Then, your application needs to connect to the service using a connection descriptor similar to the one shown in the slide above. The `FAILOVER_MODE` parameter must be included in the `CONNECT_DATA` section of your connection descriptor:

- `TYPE` specifies the type of failover. The `SESSION` value means that only the user session is reauthenticated on the server-side, whereas open cursors in the OCI application need to be reexecuted. The `SELECT` value means that not only the user session is reauthenticated on the server side, but the open cursors in the OCI can continue fetching. This implies that the client-side logic maintains fetch-state of each open cursor. A `SELECT` statement is reexecuted by using the same snapshot, discarding those rows already fetched, and retrieving those rows that were not fetched initially. TAF verifies that the discarded rows are those that were returned initially, or it returns an error message.
- `METHOD=BASIC` is used to reconnect at failover time.
- `RETRIES` specifies the number of times to attempt to connect after a failover.
- `DELAY` specifies the amount of time in seconds to wait between connect attempts.

Note: If using TAF, do no set the `GLOBAL_DBNAME` parameter in your `listener.ora` file.



TAF Preconnect Configuration: Example

```
$ srvctl add service -d RACDB -s ERP -r I1 –a I2 \
> -P PRECONNECT
$ srvctl start service -d RACDB -s ERP
```

```
ERP =
 (DESCRIPTION =(FAILOVER=ON)(LOAD_BALANCE=ON)
  (ADDRESS=(PROTOCOL=TCP)(HOST=N1VIP)(PORT=1521))
  (ADDRESS=(PROTOCOL=TCP)(HOST=N2VIP)(PORT=1521))
   (CONNECT_DATA = (SERVICE_NAME = ERP)
    (FAILOVER_MODE = (BACKUP=ERP_PRECONNECT)
                (TYPE=SESSION)(METHOD=PRECONNECT))))

ERP_PRECONNECT =
 (DESCRIPTION =(FAILOVER=ON)(LOAD_BALANCE=ON)
  (ADDRESS=(PROTOCOL=TCP)(HOST=N1VIP)(PORT=1521))
  (ADDRESS=(PROTOCOL=TCP)(HOST=N2VIP)(PORT=1521))
   (CONNECT_DATA = (SERVICE_NAME = ERP_PRECONNECT)))
```

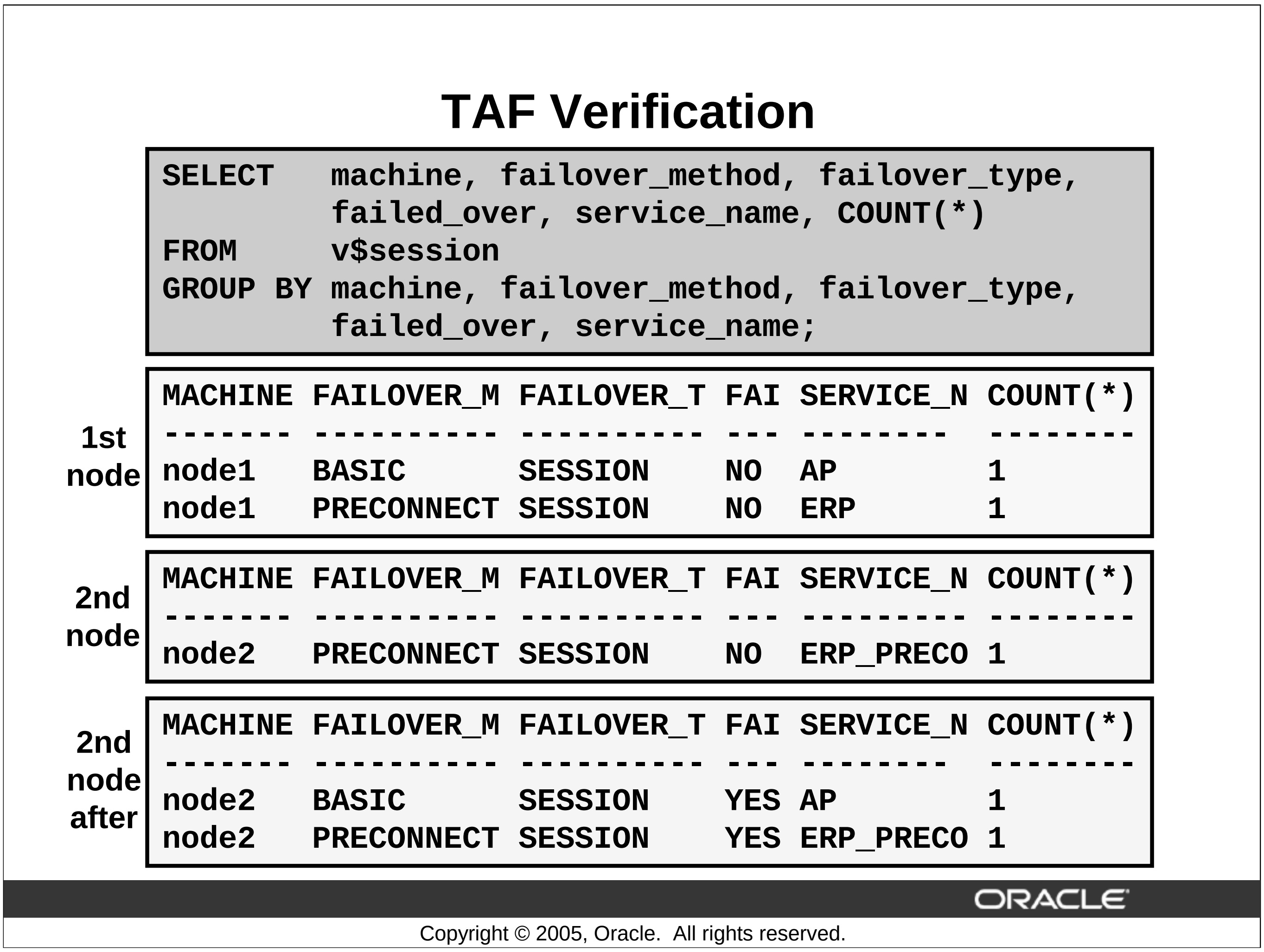
TAF Preconnect Configuration: Example

In order to use PRECONNECT TAF, it is recommended that you create a service with preferred and available instances. Also, in order for the shadow service to be created and managed automatically by the CRS, you must define the service with the –P PRECONNECT option. The shadow service is always named <service_name>_PRECONNECT.

Like with the BASIC method, you need to use a special connection descriptor to use the PRECONNECT method while connecting to the service. One such connection descriptor is shown in the slide above.

The main differences with the previous example is that METHOD is set to PRECONNECT, and an addition parameter is added. This parameter is called BACKUP and must be set to another entry in your tnsnames.ora that points to the shadow service.

Note: In all cases where TAF cannot use the PRECONNECT method, TAF falls back to the BASIC method automatically.



TAF Verification

```
SELECT   machine, failover_method, failover_type,
         failed_over, service_name, COUNT(*)
FROM     v$session
GROUP BY machine, failover_method, failover_type,
         failed_over, service_name;
```

1st node

```
MACHINE FAILOVER_M FAILOVER_T FAI SERVICE_N COUNT(*)
------- ---------- ---------- --- --------  --------
node1   BASIC      SESSION    NO  AP        1
node1   PRECONNECT SESSION    NO  ERP       1
```

2nd node

```
MACHINE FAILOVER_M FAILOVER_T FAI SERVICE_N COUNT(*)
------- ---------- ---------- --- --------- --------
node2   PRECONNECT SESSION    NO  ERP_PRECO 1
```

2nd node after

```
MACHINE FAILOVER_M FAILOVER_T FAI SERVICE_N COUNT(*)
------- ---------- ---------- --- --------  --------
node2   BASIC      SESSION    YES AP        1
node2   PRECONNECT SESSION    YES ERP_PRECO 1
```

TAF Verification

To determine whether TAF is correctly configured and that connections are associated with a failover option, you can examine the V$SESSION view. To obtain information about the connected clients and their TAF status, examine the FAILOVER_TYPE, FAILOVER_METHOD, FAILED_OVER, and SERVICE_NAME columns. The example includes one query that you could execute to verify that you have correctly configured TAF.

The above example is based on the previously configured AP and ERP services, and their corresponding connection descriptors.

The first output in the slide is the result of the execution of the query on the first node after two SQL*Plus sessions from the first node have connected to the AP and ERP services respectively. The output shows that the AP connection ended up on the first instance. Because of the load-balancing algorithm, it can end up on the second instance. Alternatively, the ERP connection must end up on the first instance because it is the only preferred one.

The second output is the result of the execution of the query on the second node before any connection failure. Note that there is currently one unused connection established under the ERP_PROCONNECT service which is automatically started on the ERP available instance.

The third output is the one corresponding to the execution of the query on the second node after the first instance failure. A second connection has been created automatically for the AP service connection, and the original ERP connection is now using the preconnected connection.

FAN Connection Pools and TAF Considerations

- Both techniques are integrated with services, and provide service connection load balancing.
- Do not mix FAN and TAF.
- Connection pools using FAN are always preconnected.
- TAF may rely on OS timeouts to detect failures.
- FAN never relies on OS timeouts to detect failures.

FAN Connection Pools and TAF Considerations

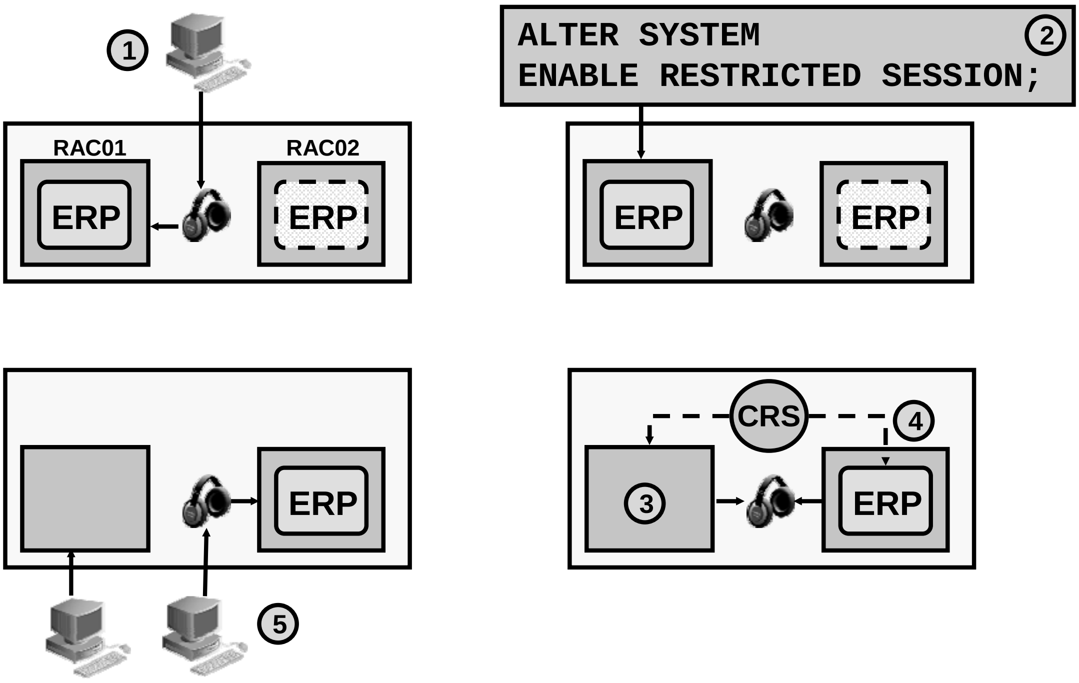
Because the connection load balancing is a listener functionality, both FAN connection pools and TAF automatically benefit from connection load balancing for services.

When you use FAN connection pools, there is no need to use TAF. Moreover, FAN connection pools and TAF cannot work together.

For example, you do not need to preconnect if you use FAN connection pools. The connection pool is always preconnected.

With both techniques, you automatically benefit from VIPs at connection time. This means that your application does not rely on lengthy operating system connection timeouts at connect time, or when issuing a SQL statement. However, when in the SQL stack, and the application is blocked on a read/write call, the application needs to be integrated with FAN in order to receive an interrupt if a node goes down. In a similar case, TAF will rely on OS timeouts to detect the failure. This takes much more time to fail over the connection than when using FAN.

Restricted Session and Services



```
ALTER SYSTEM
ENABLE RESTRICTED SESSION;
```

ORACLE

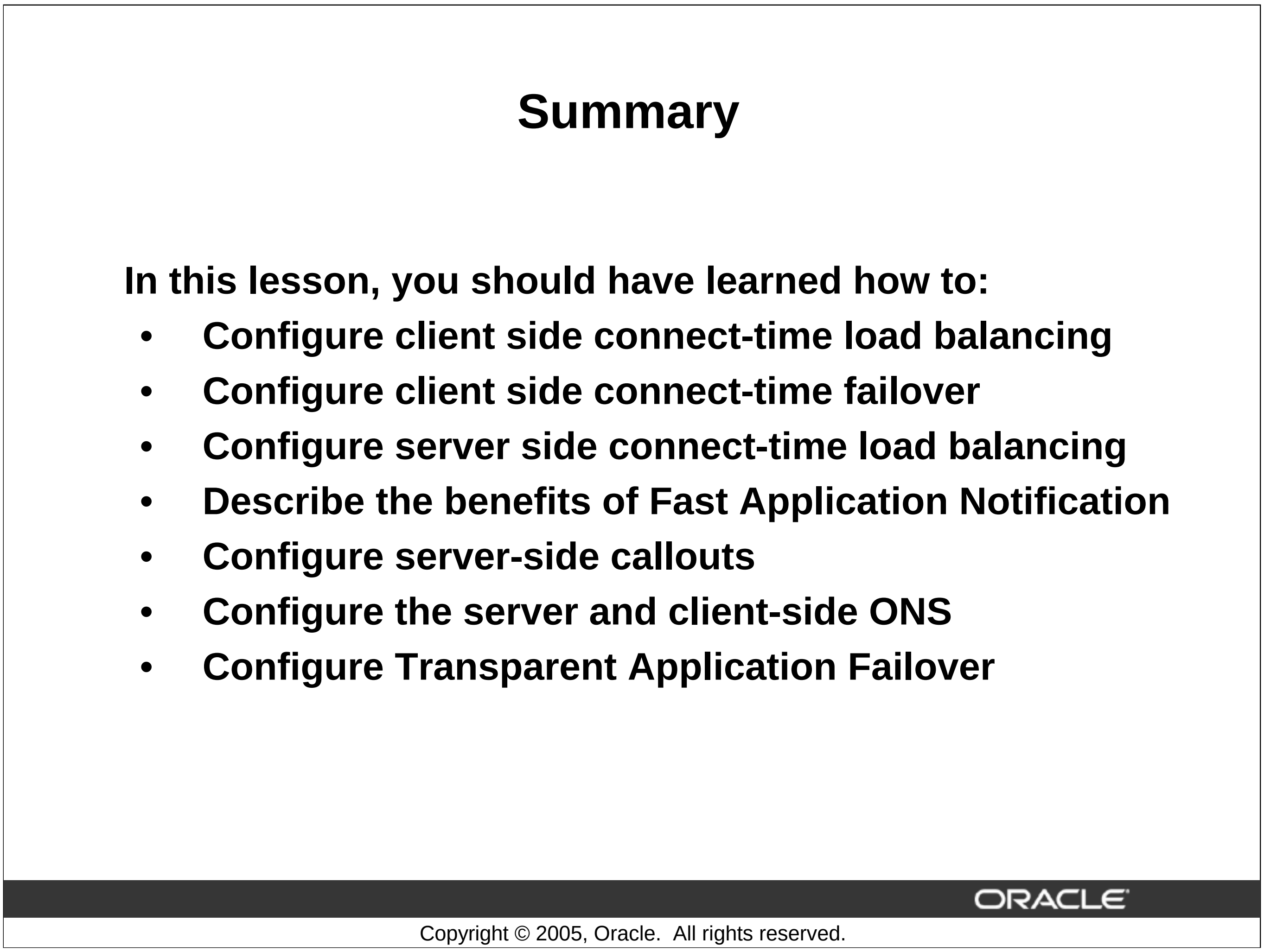
Restricted Session and Services

Whenever you put one instance of the cluster in restricted mode, the corresponding CRS stops the services running on the restricted instance, and it starts them on available instances if they exist. That way, the listeners are dynamically informed of the changes, and they no longer attempt to route requests to the restricted instance, regardless of its current load. In effect, the listeners exempt the restricted instance from their connection load balancing algorithm.

This feature comes with two important considerations:

- First, even users with `RESTRICTED SESSION` privilege are not able to connect remotely through the listeners to an instance that is in the restricted mode. They need to connect locally to the node supporting the instance and use the bequeath protocol.
- Second, this new feature works only when the restricted instance dynamically registers with the listeners. In other words, if you configure the `listener.ora` file with `SID_LIST` entries, and you do not use dynamic registration, the listener cannot block connection attempts to a restricted instance. In this case, and because the unrestricted instances of the cluster are still accessible, the restricted instance will eventually become least loaded, and the listener will start routing connection requests to that instance. Unable to accept the connection request because of its restricted status, the instance will deny the connection and return an error. This situation has the potential for blocking access to an entire service.

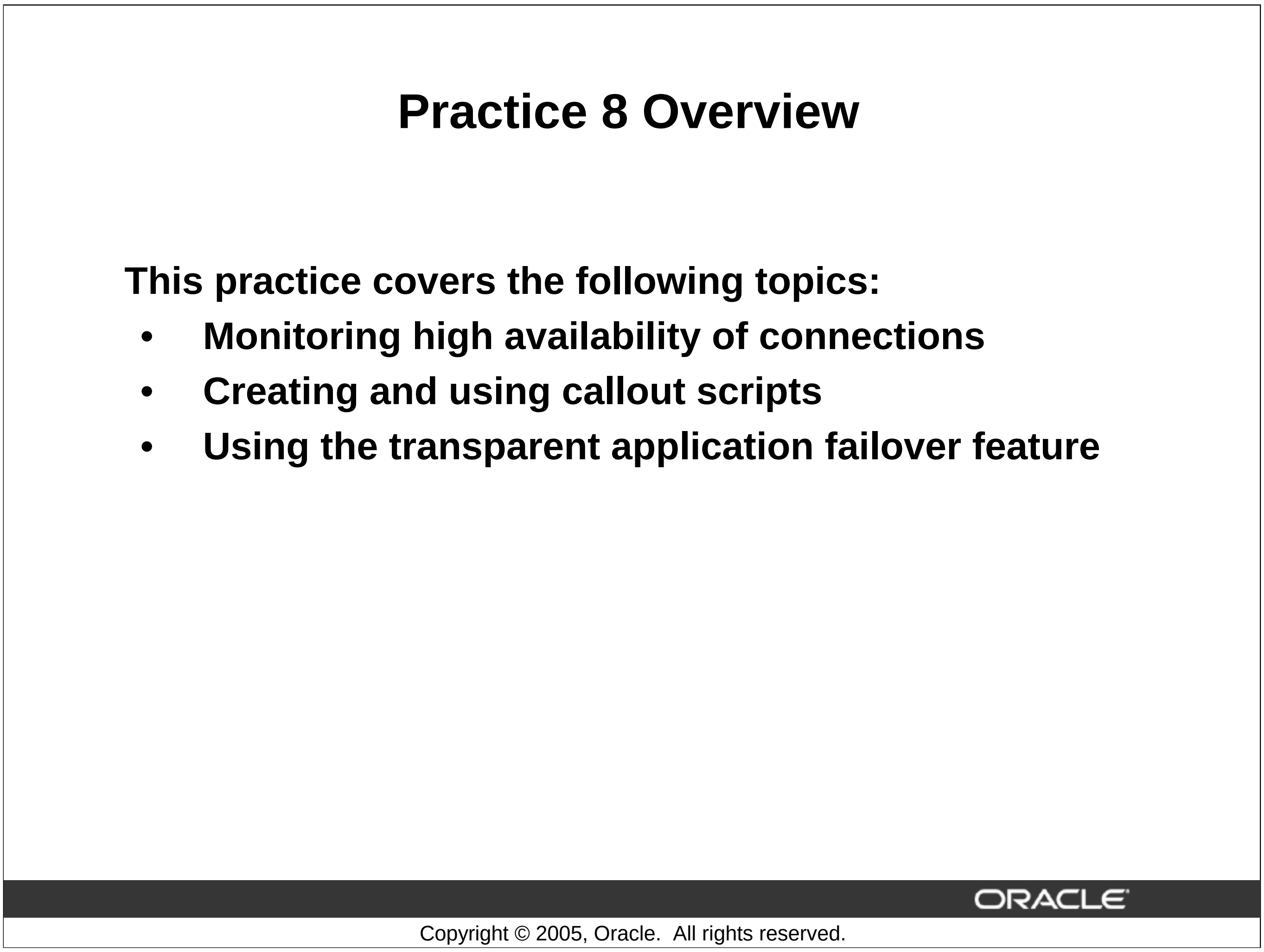
Note: The listener uses dynamic service registration information before static configurations.



Summary

In this lesson, you should have learned how to:

- Configure client side connect-time load balancing
- Configure client side connect-time failover
- Configure server side connect-time load balancing
- Describe the benefits of Fast Application Notification
- Configure server-side callouts
- Configure the server and client-side ONS
- Configure Transparent Application Failover



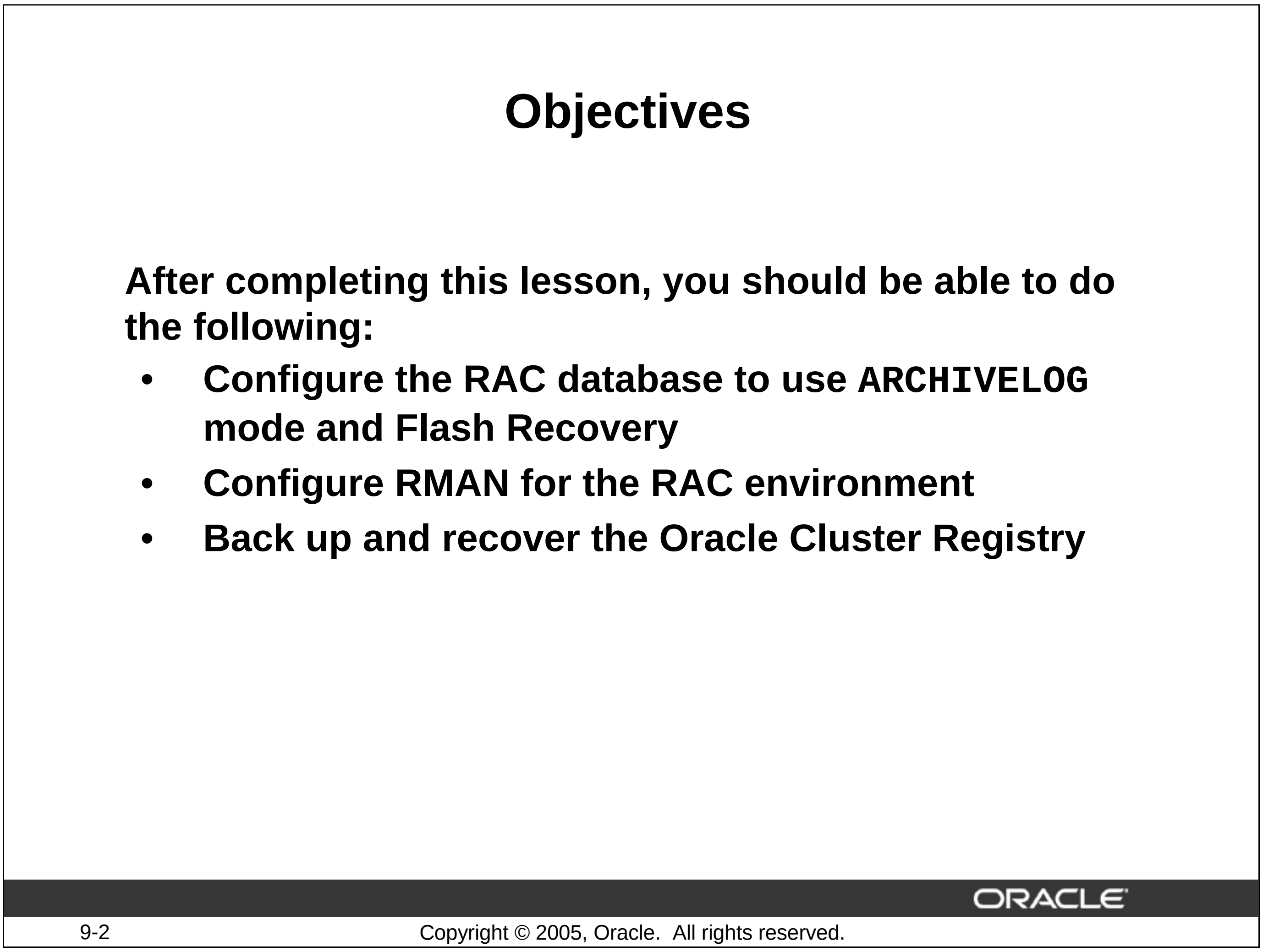
Practice 8 Overview

This practice covers the following topics:

- Monitoring high availability of connections
- Creating and using callout scripts
- Using the transparent application failover feature

9

Managing Backup and Recovery in RAC

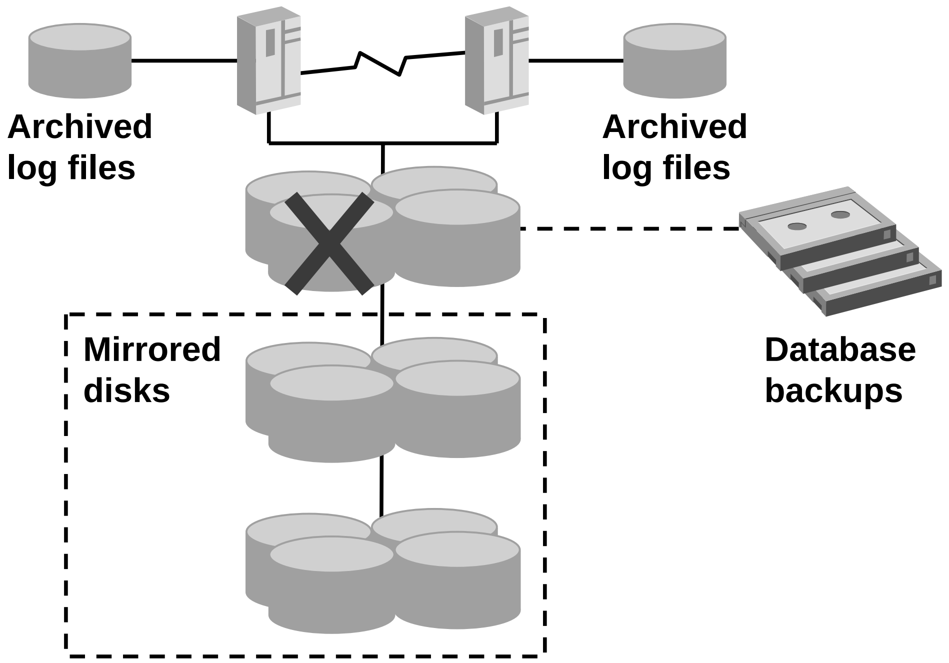


Objectives

After completing this lesson, you should be able to do the following:

- Configure the RAC database to use `ARCHIVELOG` mode and Flash Recovery
- Configure RMAN for the RAC environment
- Back up and recover the Oracle Cluster Registry

Protecting Against Media Failure

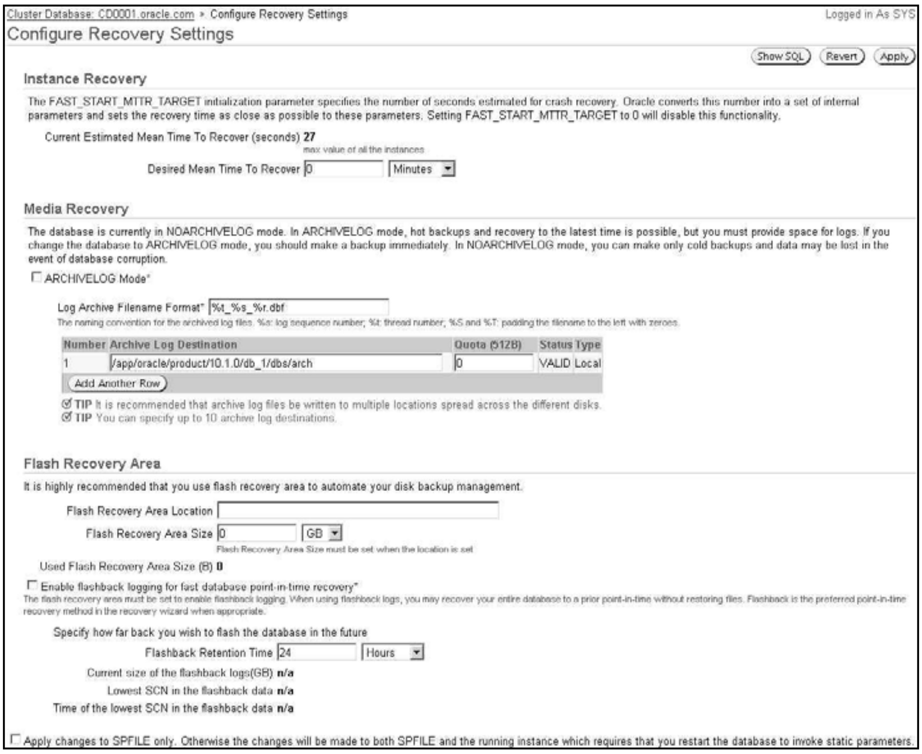


Protecting Against Media Failure

Although RAC provides you with methods to avoid or to reduce down time due to a failure of one or more (but not all) of your instances, you must still protect the database itself, which must be shared by all the instances. This means that you need to consider disk backup and recovery strategies for you cluster database just as you would for a nonclustered database.

To minimize the potential loss of data due to disk failures, you may want to use disk mirroring technology (available from your server or disk vendor). As in nonclustered databases, you can have more than one mirror if your vendor allows it, to help reduce the potential for data loss and to provide you with alternate backup strategies. For example, with your database in `ARCHIVELOG` mode and with three copies of your disks, you can remove one mirror copy and perform your backup from it while the two remaining mirror copies continue to protect ongoing disk activity. To do this correctly, you must first put the tablespaces into backup mode and then, if required by your cluster or disk vendor, temporarily halt disk operations by issuing the `ALTER SYSTEM SUSPEND` command. After the statement completes, you can break the mirror and then resume normal operations by executing the `ALTER SYSTEM RESUME` command and taking the tablespaces out of backup mode.

Configure RAC Recovery Settings with EM



Cluster Database: CD0001.oracle.com > Configure Recovery Settings

Logged in As SYS

Configure Recovery Settings

Instance Recovery

The FAST_START_MTTR_TARGET initialization parameter specifies the number of seconds estimated for crash recovery. Oracle converts this number into a set of internal parameters and sets the recovery time as close as possible to these parameters. Setting FAST_START_MTTR_TARGET to 0 will disable this functionality.

Current Estimated Mean Time To Recover (seconds) **27**
max value of all the instances

Desired Mean Time To Recover 0 Minutes

Media Recovery

The database is currently in NOARCHIVELOG mode. In ARCHIVELOG mode, hot backups and recovery to the latest time is possible, but you must provide space for logs. If you change the database to ARCHIVELOG mode, you should make a backup immediately. In NOARCHIVELOG mode, you can make only cold backups and data may be lost in the event of database corruption.

□ ARCHIVELOG Mode*

Log Archive Filename Format* %t_%s_%r.dbf
The naming convention for the archived log files. %s: log sequence number; %t: thread number; %S and %T: padding the filename to the left with zeroes.

Number	Archive Log Destination	Quota (512B)	Status	Type
1	/app/oracle/product/10.1.0/db_1/dbs/arch	0	VALID	Local

☑ **TIP** It is recommended that archive log files be written to multiple locations spread across the different disks.
☑ **TIP** You can specify up to 10 archive log destinations.

Flash Recovery Area

It is highly recommended that you use flash recovery area to automate your disk backup management.

Flash Recovery Area Location

Flash Recovery Area Size 0 GB
Flash Recovery Area Size must be set when the location is set

Used Flash Recovery Area Size (B) **0**

□ Enable flashback logging for fast database point-in-time recovery*
The flash recovery area must be set to enable flashback logging. When using flashback logs, you may recover your entire database to a prior point-in-time without restoring files. Flashback is the preferred point-in-time recovery method in the recovery wizard when appropriate.

Specify how far back you wish to flash the database in the future

Flashback Retention Time 24 Hours

Current size of the flashback logs(GB) **n/a**

Lowest SCN in the flashback data **n/a**

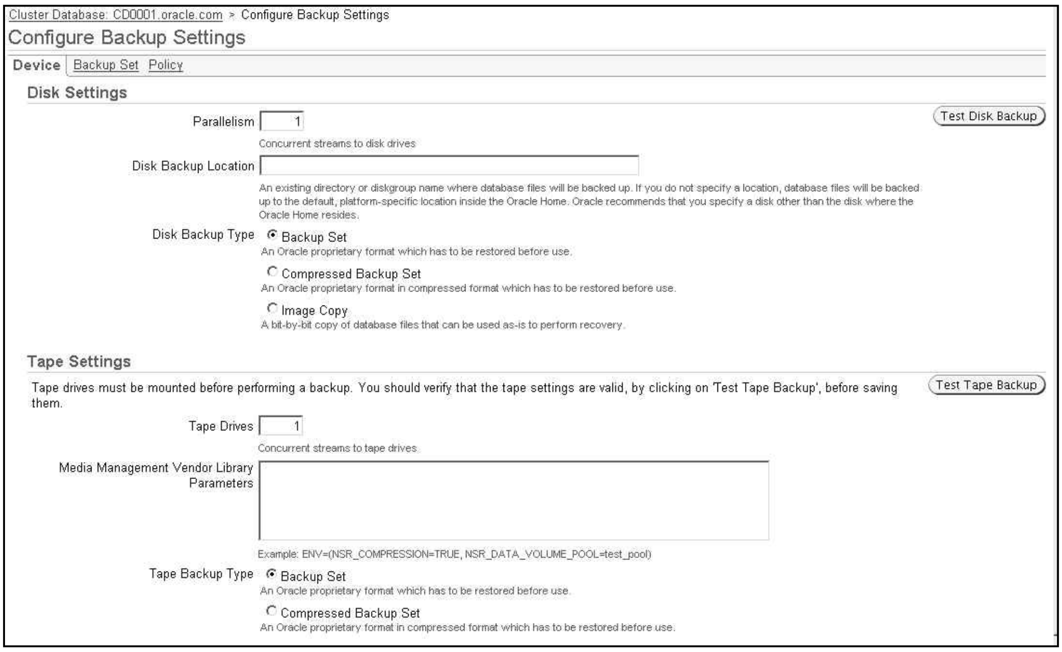
Time of the lowest SCN in the flashback data **n/a**

□ Apply changes to SPFILE only. Otherwise the changes will be made to both SPFILE and the running instance which requires that you restart the database to invoke static parameters.

Configure Recovery Settings with EM

Enterprise Manager can be used to configure important recovery settings for your cluster database. From the Database Control home page, click on the Maintenance folder tab, then click on Configure Recovery Settings link. From here you can ensure that your database is in archivelog mode and configure flash recovery settings.

Configure RAC Backup Settings with EM



Cluster Database: CD0001.oracle.com > Configure Backup Settings

Configure Backup Settings

Disk Settings

Parallelism 1

Concurrent streams to disk drives

Disk Backup Location

An existing directory or diskgroup name where database files will be backed up. If you do not specify a location, database files will be backed up to the default, platform-specific location inside the Oracle Home. Oracle recommends that you specify a disk other than the disk where the Oracle Home resides.

Disk Backup Type

● Backup Set
An Oracle proprietary format which has to be restored before use.

○ Compressed Backup Set
An Oracle proprietary format in compressed format which has to be restored before use.

○ Image Copy
A bit-by-bit copy of database files that can be used as-is to perform recovery.

Tape Settings

Tape drives must be mounted before performing a backup. You should verify that the tape settings are valid, by clicking on 'Test Tape Backup', before saving them.

Tape Drives 1

Concurrent streams to tape drives

Media Management Vendor Library Parameters

Example: ENV=(NSR_COMPRESSION=TRUE, NSR_DATA_VOLUME_POOL=test_pool)

Tape Backup Type

● Backup Set
An Oracle proprietary format which has to be restored before use.

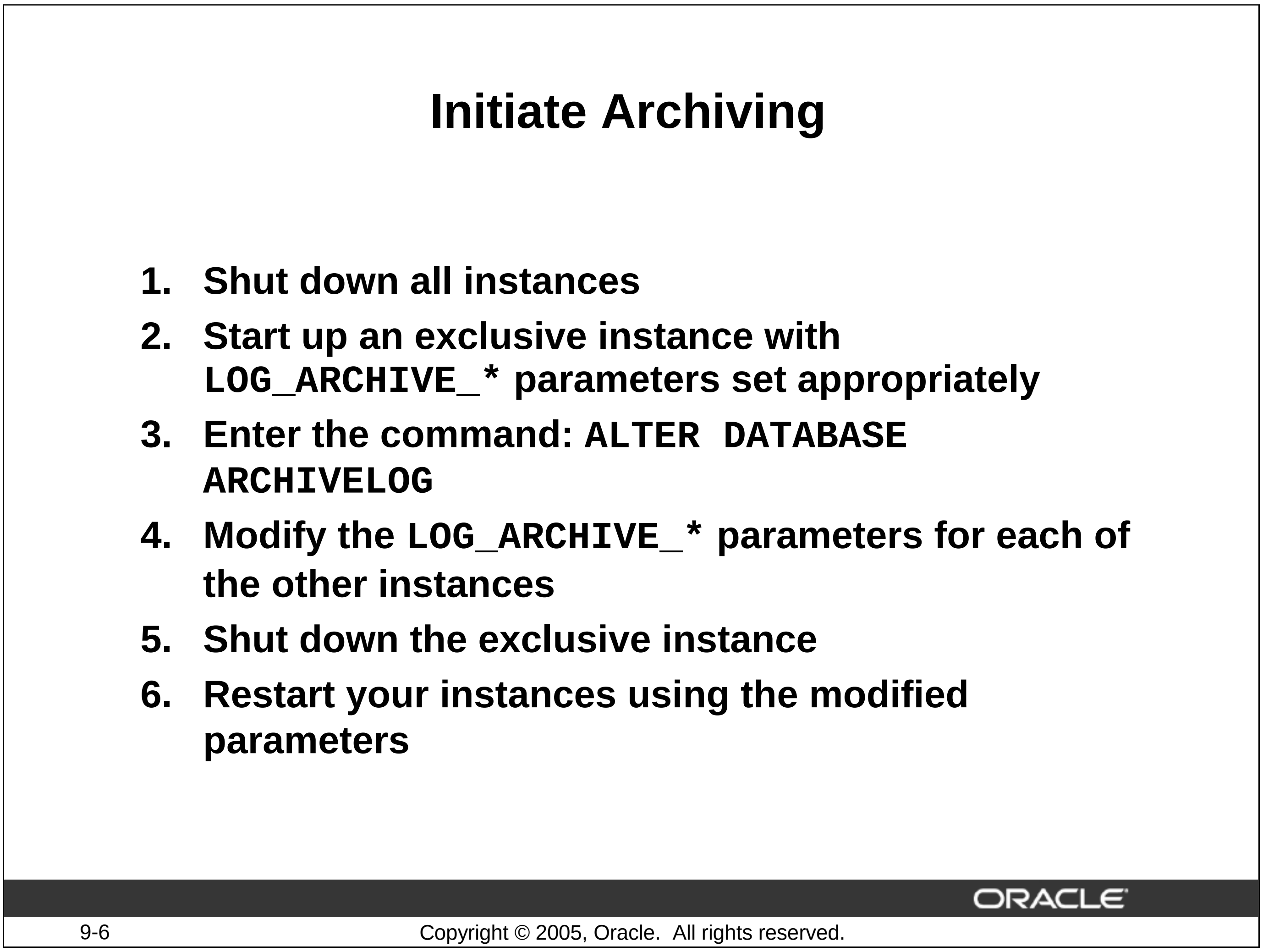
○ Compressed Backup Set
An Oracle proprietary format in compressed format which has to be restored before use.

Configure Backup Settings with EM

Persistent backup settings can be configured using Enterprise Manager. From the Database Control home page, click on the Maintenance folder tab, then click on Configure Backup Settings link. You can configure disk settings like the directory location of your disk backups, and level of parallelism. You can also choose the default backup type:

- Backup set
- Compressed backup set
- Image copy

You can also specify important tape related settings like the number of available tape drives and vendor-specific media management parameters.



Initiate Archiving

1. **Shut down all instances**
2. **Start up an exclusive instance with `LOG_ARCHIVE_*` parameters set appropriately**
3. **Enter the command: `ALTER DATABASE ARCHIVELOG`**
4. **Modify the `LOG_ARCHIVE_*` parameters for each of the other instances**
5. **Shut down the exclusive instance**
6. **Restart your instances using the modified parameters**

9-6

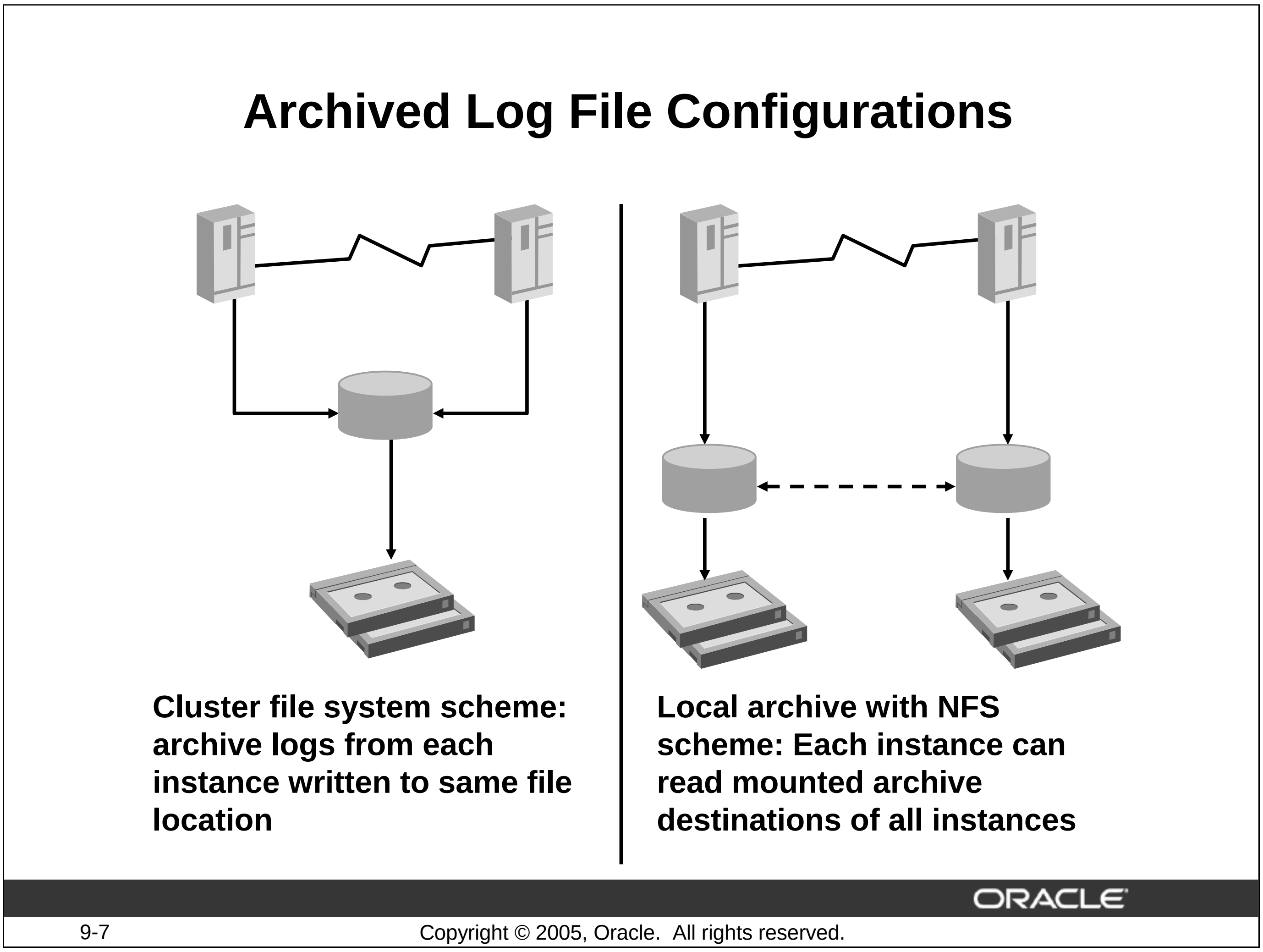
Initiate Archiving

To enable archive logging, your database must be mounted, but not open, by an exclusive instance. If you are using an SPFILE, you must first create SID-specific entries for this instance; otherwise, build a special-purpose text parameter file. The parameters that you must set for this exclusive instance include the following:

- `CLUSTER_DATABASE`: Set to `FALSE`
- `LOG_ARCHIVE_DEST_n`: Up to 10 values, depending on your archive strategy
- `LOG_ARCHIVE_FORMAT`: Include the `%t` or `%T` and `%R` parameters for thread identification
- `LOG_ARCHIVE_START`: Set to `TRUE`

You can now enable archiving for your database by following these steps:

1. Shut down all instances.
2. Start up your exclusive instance.
3. Enter the following command: `ALTER DATABASE ARCHIVELOG`
4. Modify the `LOG_ARCHIVE_*` parameters for each of the other instances.
5. Shut down the exclusive instance and reset its `CLUSTER_DATABASE` parameter.
6. Restart all of your instances using the modified parameters.



Archived Log File Configurations

Cluster file system scheme: archive logs from each instance written to same file location

Local archive with NFS scheme: Each instance can read mounted archive destinations of all instances

9-7

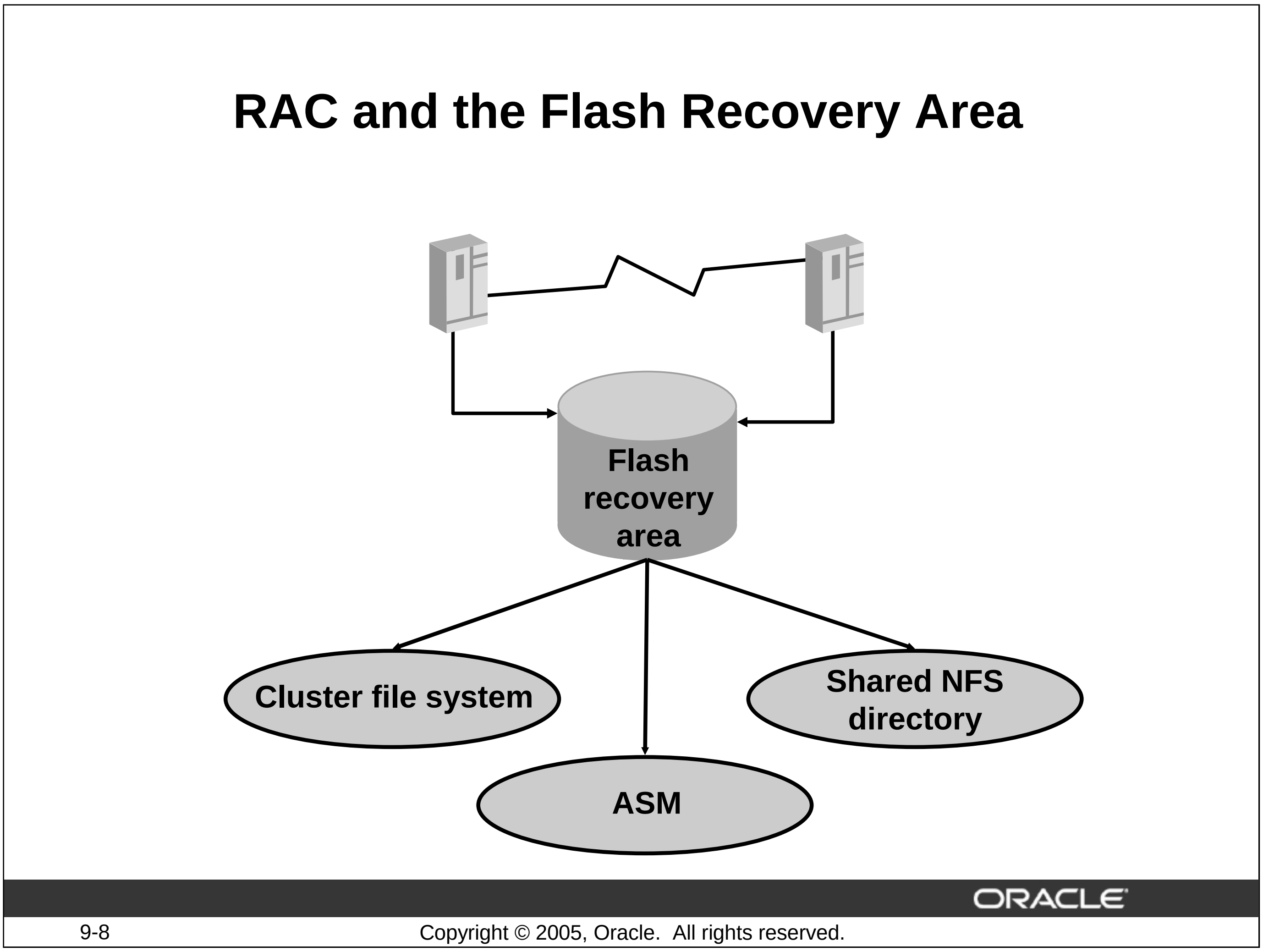
Archived Log File Configurations

During backup and recovery operations involving archived log files, the Oracle server determines the file destinations and names from the control file. The archived log file path names can also be stored in the optional recovery catalog if you are using RMAN. The archived log file path names do not include the node name, however, so RMAN expects to find the files it needs on the nodes where the channels are allocated.

If you are using a cluster file system, your instances can all write to the same archive log destination. This is known as the cluster file system scheme. Backup and recovery of the archive logs are easy because all logs are located in the same directory.

If a cluster file system is not available, then Oracle recommends that local archive log destinations be created for each instance with NFS-read mount points to all other instances. This is known as the local archive with NFS scheme. During backup, you can either back up the archive logs from each host or select one host to perform the backup for all archive logs. During recovery, one instance may access the logs from any host without having to first copy them to the local destination.

Using either scheme, you may wish to provide a second archive destination to avoid single points of failure.



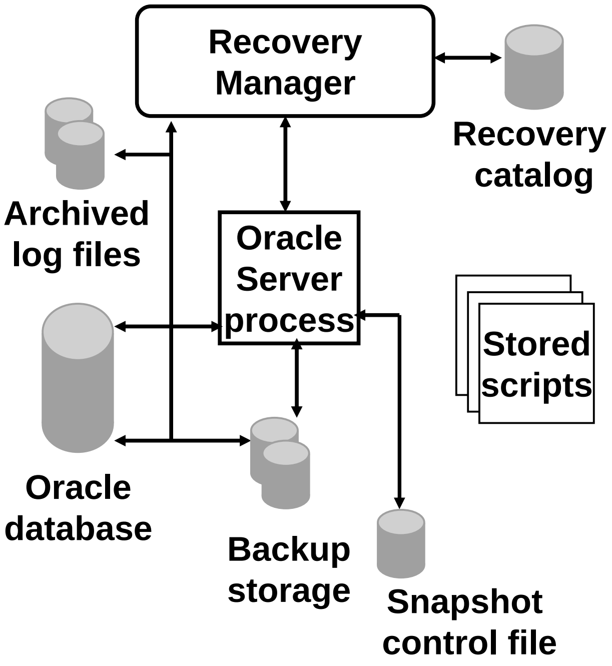
RAC and the Flash Recovery Area

RAC and the Flash Recovery Area

To use a flash recovery area in RAC, you must place it on an ASM disk, a Cluster File System, or on a shared directory that is configured through NFS for each RAC instance. In other words, the flash recovery area must be shared among all the instances of a RAC database.

Note: Set the initialization parameters `DB_RECOVERY_FILE_DEST` and `DB_RECOVERY_FILE_DEST_SIZE` to the same values on all instances.

Oracle Recovery Manager



RMAN provides the following benefits for Real Application Clusters:

- **Can read cluster files or raw partitions with no configuration changes**
- **Can access multiple archive log destinations**

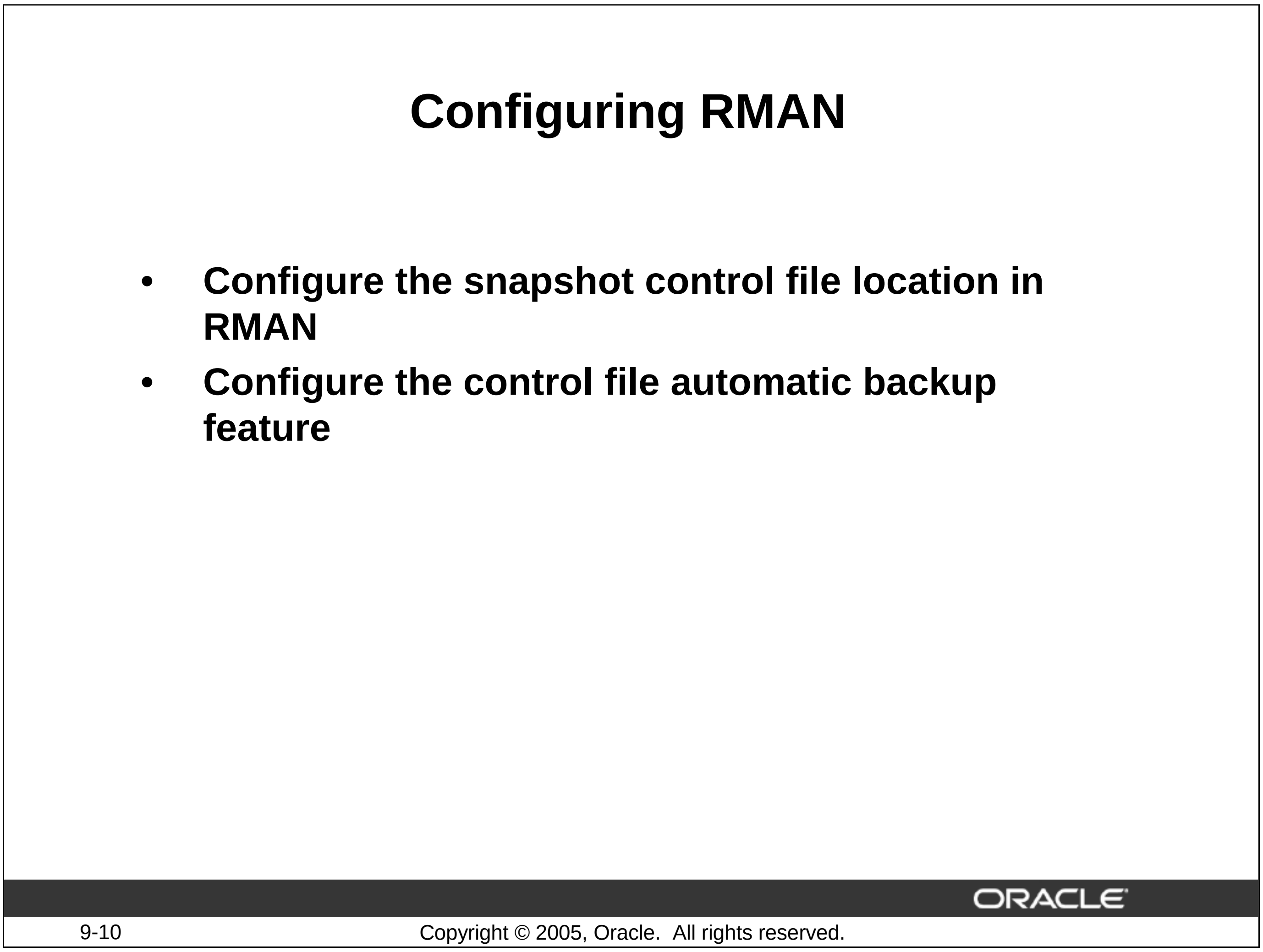
Oracle Recovery Manager

Oracle Recovery Manager (RMAN) can use stored scripts, interactive scripts, or an interactive GUI front end. When using RMAN with your RAC database, use stored scripts to initiate the backup and recovery processes from the most appropriate node.

If you use different Oracle Home locations for your Real Application Instances on each of your nodes, create a snapshot control file in a location that can be accessed by all your instances, either a cluster file or a shared raw device. Here is an example:

```
RMAN> CONFIGURE SNAPSHOT CONTROLFILE TO
'/oracle/db_files/snaps/snap_prod1.cf';
```

For recovery, you must ensure that each recovery node can access the archive log files from all instances using one of the archive schemes discussed earlier, or make the archived logs available to the recovering instance by copying them from another location.



Configuring RMAN

- Configure the snapshot control file location in RMAN
- Configure the control file automatic backup feature

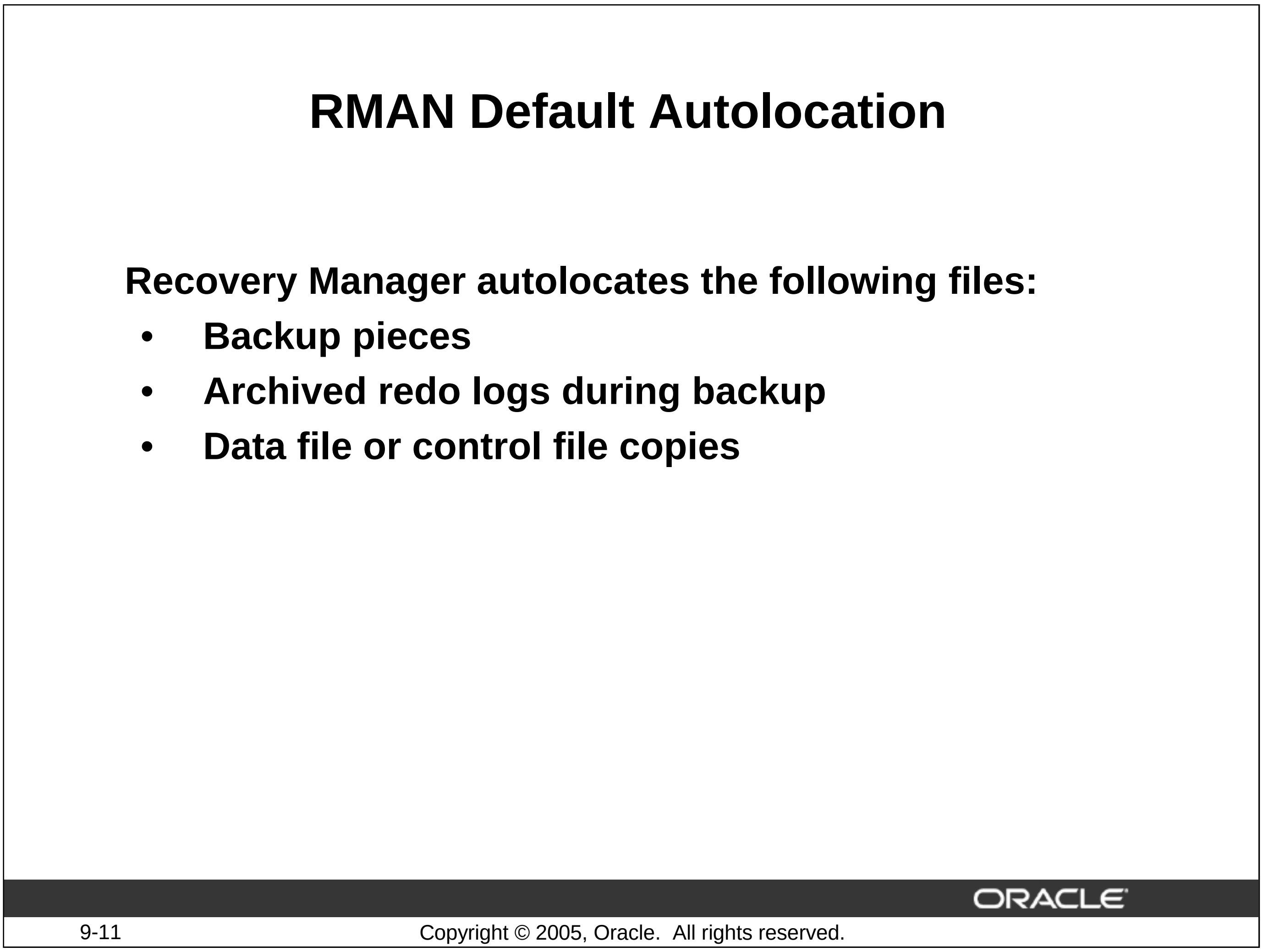
9-10

Snapshot Control File Management

RMAN creates snapshot copies of your control file as part of its backup operations. In a RAC database, the snapshot control file is created on the node that is making the backup. You need to configure a default path and file name for these snapshot control files that is valid on every node from which you might initiate an RMAN backup. You can use a shared file system location, including a raw device, if you want.

Automatic Control File Backups

RMAN optionally creates control file backups automatically after `BACKUP` and `COPY` commands. These backups can be used if the recovery catalog and current control file are lost. RMAN performs the control file autobackup on the first allocated channel. When you allocate multiple channels with different parameters (especially if you allocate a channel with the `CONNECT` command), you must determine which channel will perform the automatic backup. Always allocate the channel for the connected node first.



RMAN Default Autolocation

Recovery Manager autolocates the following files:

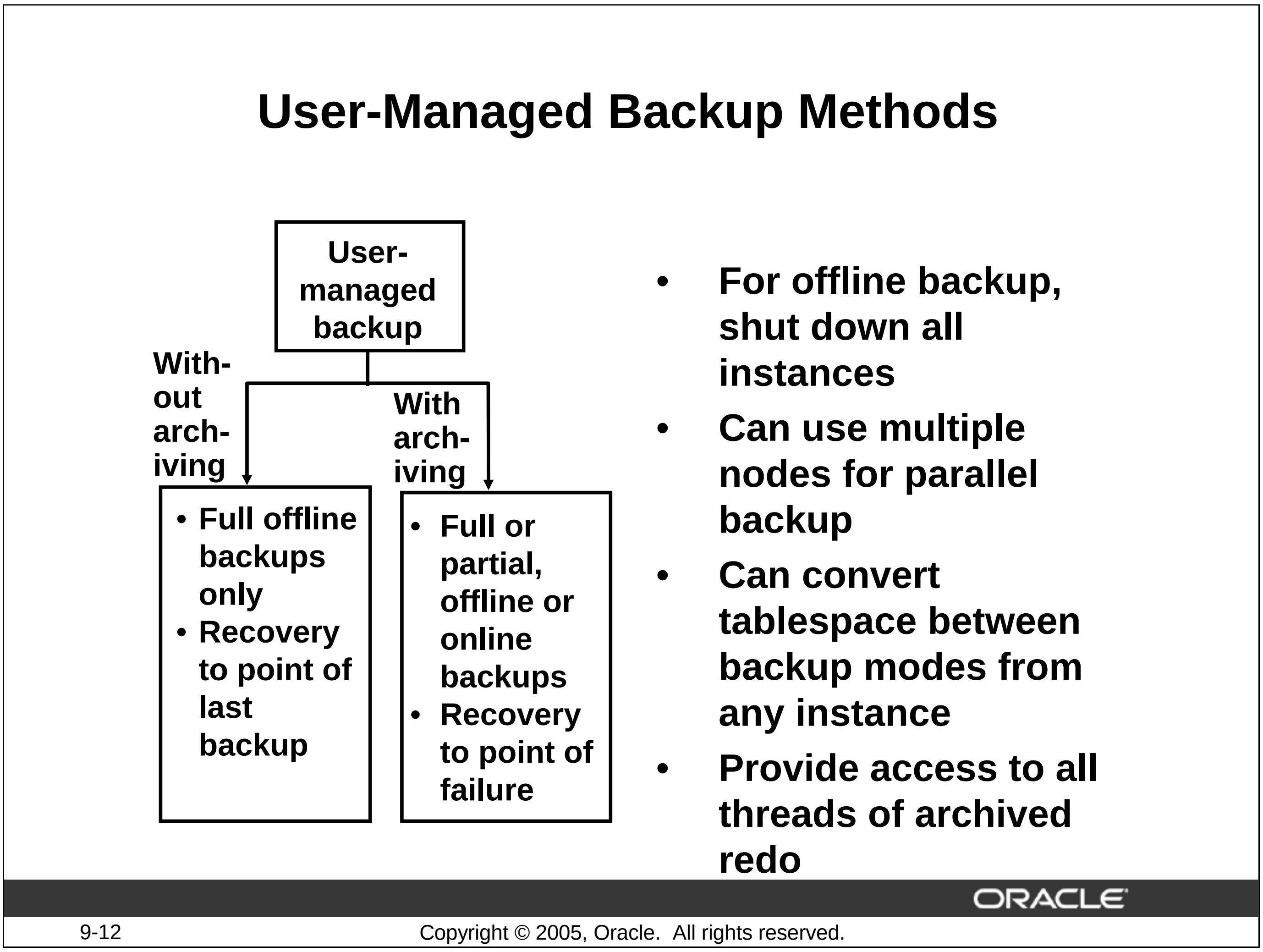
- **Backup pieces**
- **Archived redo logs during backup**
- **Data file or control file copies**

RMAN Default Autolocation

Recovery Manager automatically discovers which nodes of a RAC configuration can access the files that you want to back up or restore. Recovery Manager autolocates the following files:

- Backup pieces during backup or restore
- Archived redo logs during backup
- Data file or control file copies during backup or restore

In previous releases, you had to manually enable this option with `SET AUTOLOCATE`, and the option applied to backup pieces only.



User-Managed Backup Methods

User-managed backup

Without archiving

- Full offline backups only
- Recovery to point of last backup

With archiving

- Full or partial, offline or online backups
- Recovery to point of failure

- For offline backup, shut down all instances
- Can use multiple nodes for parallel backup
- Can convert tablespace between backup modes from any instance
- Provide access to all threads of archived redo

9-12

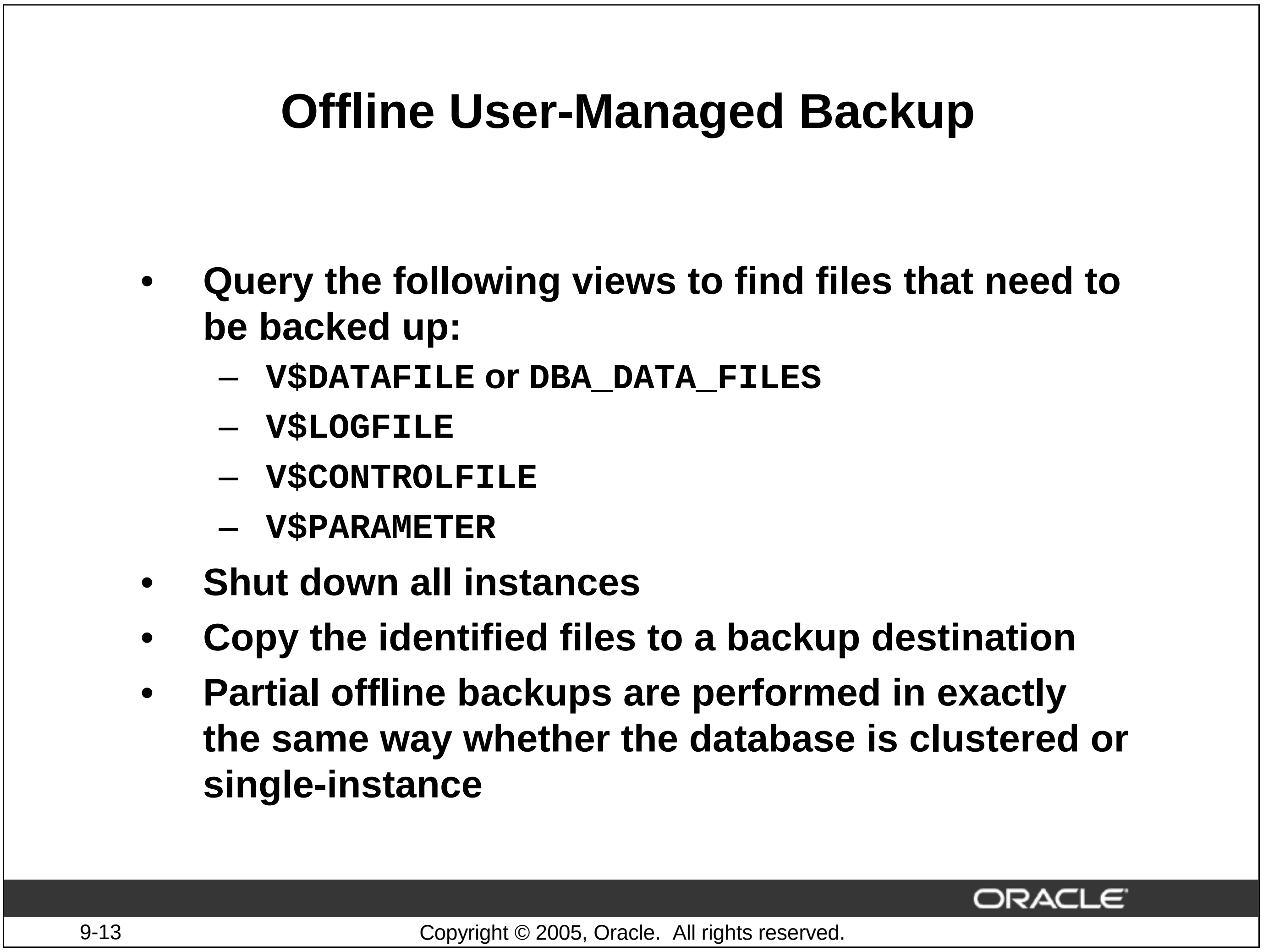
User-Managed Backup Methods

The user-managed backup and recovery options and methods for RAC databases are similar to the procedures that are used in a noncluster database. These include full offline backups and, if you are running in `ARCHIVELOG` mode, online backups.

There are some additional issues that you need to consider when backing up your RAC database, among which are the following:

- You must shut down every instance for an offline database backup.
- You can employ more than one node to back up different data files in parallel.
- You need to issue the `ALTER TABLESPACE` commands (`BEGIN BACKUP` and `END BACKUP`) on only one instance. It does not have to be on the same node where you perform the disk backup operations.

You must provide for the backup and recovery operations to access the archive log files that are independently written by each instance.



Offline User-Managed Backup

- Query the following views to find files that need to be backed up:
  - **`V$DATAFILE`** or **`DBA_DATA_FILES`**
  - **`V$LOGFILE`**
  - **`V$CONTROLFILE`**
  - **`V$PARAMETER`**
- Shut down all instances
- Copy the identified files to a backup destination
- Partial offline backups are performed in exactly the same way whether the database is clustered or single-instance

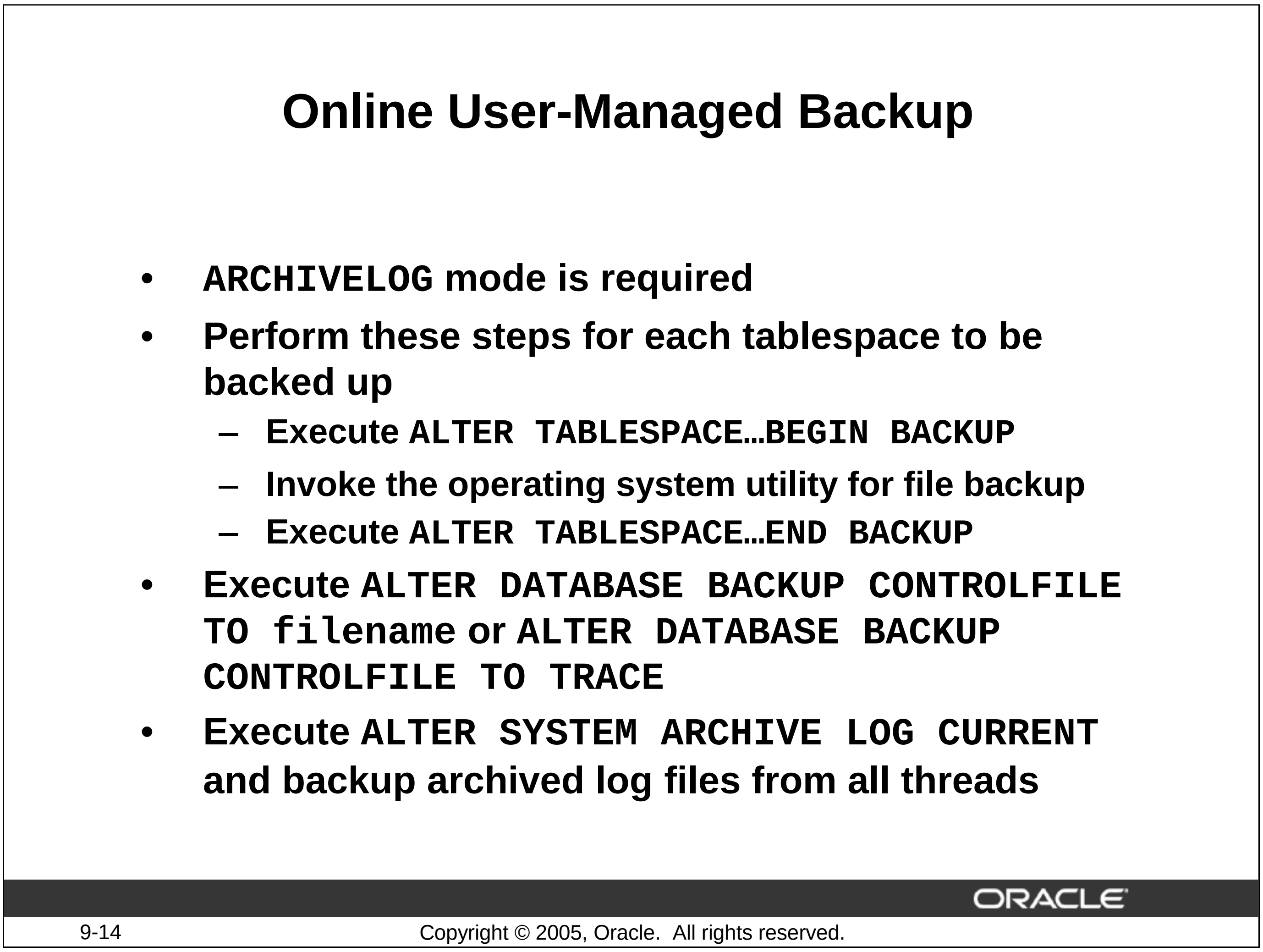
9-13

Offline User-Managed Backup

The full offline user-managed backup procedure for a RAC database is almost identical to the procedure for a single-instance configuration. The only difference is that you must shut down all of the instances, not just one, before you begin the actual backup. The main steps in the backup procedure include:

1. Query the `V$DATAFILE` view to obtain the names and locations of the data.
2. Query the `V$LOGFILE` view to obtain the names and locations of the online redo log files.
3. Query the `V$CONTROLFILE` view to obtain the names and locations of the control files.
4. Query the `V$PARAMETER` view to obtain the name of the SPFILE (if one is used).
5. Shut down all instances that are currently accessing the database.
6. After shutting down all instances, use an operating system utility to save all the data files, online redo log files, at least one copy of the control file, and the SPFILE.
7. Restart the instances.

The easiest way to perform this type of backup is to use a script. However, the script must confirm that all instances are shut down before starting to copy any of the files.



Online User-Managed Backup

- **`ARCHIVELOG` mode is required**
- **Perform these steps for each tablespace to be backed up**
  - **Execute `ALTER TABLESPACE…BEGIN BACKUP`**
  - **Invoke the operating system utility for file backup**
  - **Execute `ALTER TABLESPACE…END BACKUP`**
- **Execute `ALTER DATABASE BACKUP CONTROLFILE TO filename` or `ALTER DATABASE BACKUP CONTROLFILE TO TRACE`**
- **Execute `ALTER SYSTEM ARCHIVE LOG CURRENT` and backup archived log files from all threads**

Online User-Managed Backup

Online backups enable you to back up all or part of the database while it is running. The procedure is essentially the same as for performing an online backup of a nonclustered database. Online backups can be performed only when the database is running in `ARCHIVELOG` mode. The main steps in the backup procedure include:

1. Execute the `ALTER TABLESPACE … BEGIN BACKUP` command. This should be done for each tablespace before you back up any of its data files. You can set all tablespaces in backup mode simultaneously, or you can set them individually if you back up one at a time.
2. Issue the operating system commands to back up the data files for the tablespace.
3. Execute the `ALTER TABLESPACE … END BACKUP` command.
4. Back up the control files with the `ALTER DATABASE BACKUP CONTROLFILE TO` *`file_name`* command. For additional safety, back up the control file to a trace file with the `ALTER DATABASE BACKUP CONTROLFILE TO TRACE` command.
5. Execute an `ALTER SYSTEM ARCHIVE LOG CURRENT` command to archive all redo threads, including any unarchived logs from closed threads, and back up these along with any other archive log files that have not been previously backed up.

Channel Connections to Cluster Instances

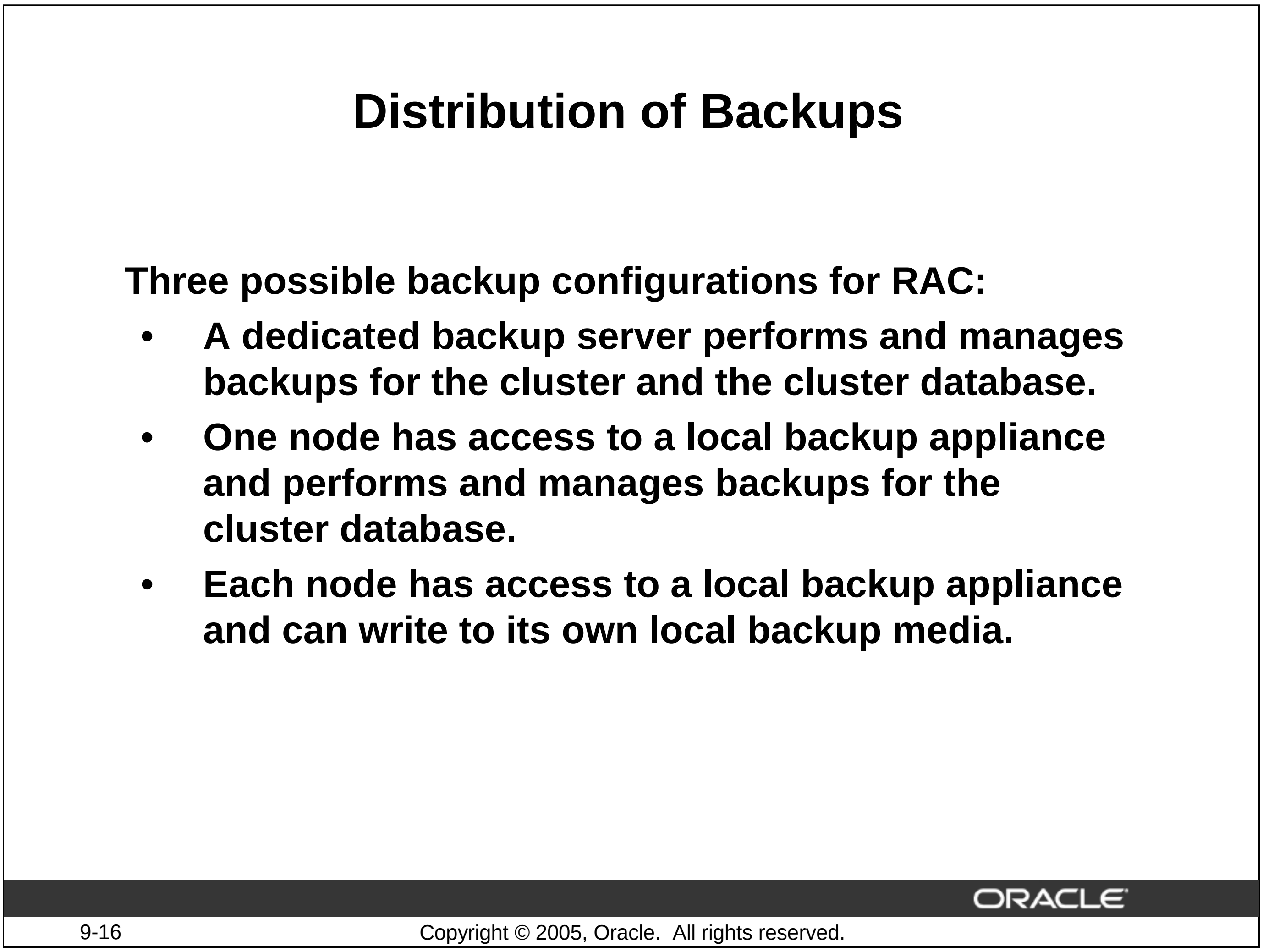
- When backing up, each allocated channel can connect to a different instance in the cluster.
- Each channel connection must resolve to only one instance.
- Instances to which the channels connect must be either all mounted or all open.

```
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT='sys/rac@RACDB1';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT='sys/rac@RACDB2';
CONFIGURE CHANNEL 3 DEVICE TYPE sbt CONNECT='sys/rac@RACDB3';
```

Channel Connections to Cluster Instances

When making backups using channels connected to different instances, each allocated channel can connect to a different instance in the cluster, and each channel connection must resolve to only one instance. The above example shows you a possible automatic channels configuration.

During a backup, the instances to which the channels connect must be either all mounted or all open. For example, if the `RACDB1` instance has the database mounted while the `RACDB2` and `RACDB3` instances have the database open, then the backup fails.



Distribution of Backups

Three possible backup configurations for RAC:

- A dedicated backup server performs and manages backups for the cluster and the cluster database.
- One node has access to a local backup appliance and performs and manages backups for the cluster database.
- Each node has access to a local backup appliance and can write to its own local backup media.

Distribution of Backups

When configuring the backup options for RAC, you have three possible configurations:

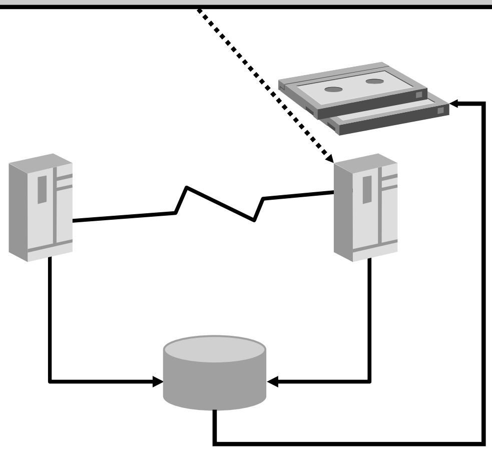
- **Network backup server:** A dedicated backup server performs and manages backups for the cluster and the cluster database. None of the nodes have local backup appliances.
- **One local drive:** One node has access to a local backup appliance and performs and manages backups for the cluster database. All nodes of the cluster should be on a cluster file system to be able to read all datafiles, archived redo logs, and `SPFILE`s. It is recommended that you do not use the non-cluster file system archiving scheme if you have backup media on only one local drive.
- **Multiple drives:** Each node has access to a local backup appliance and can write to its own local backup media.

In the cluster file system scheme, any node can access all the datafiles, archived redo logs, and `SPFILE`s. In the non-cluster file system scheme, you must write the backup script so that the backup is distributed to the correct drive and path for each node. For example, node 1 can back up the archived redo logs whose path names begin with `/arc_dest_1`, node 2 can back up the archived redo logs whose path names begin with `/arc_dest_2`, and node 3 can back up the archived redo logs whose path names begin with `/arc_dest_3`.

One Local Drive CFS Backup Scheme

```
RMAN> CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
RMAN> CONFIGURE DEFAULT DEVICE TYPE TO sbt;
```

```
RMAN> BACKUP DATABASE PLUS ARCHIVELOG DELETE INPUT;
```



9-17

One Local Drive CFS Backup Scheme

In a cluster file system backup scheme, each node in the cluster has read access to all the datafiles, archived redo logs, and SPFILEs. This includes Automated Storage Management (ASM), cluster file systems, and Network Attached Storage (NAS).

When backing up to only one local drive in the cluster file system backup scheme, it is assumed that only one node in the cluster has a local backup appliance such as a tape drive. In this case, run the following one-time configuration commands:

```
         CONFIGURE DEVICE TYPE sbt PARALLELISM 1;
         CONFIGURE DEFAULT DEVICE TYPE TO sbt;
```

Because any node performing the backup has read/write access to the archived redo logs written by the other nodes, the backup script for any node is simple:

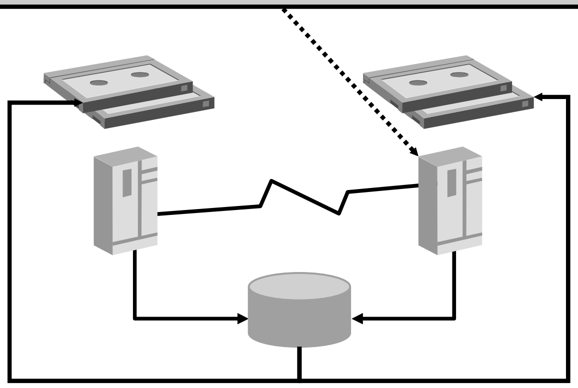
```
BACKUP DATABASE PLUS ARCHIVELOG DELETE INPUT;
```

In this case, the tape drive receives all datafiles, archived redo logs, and SPFILEs.

Multiple Drives CFS Backup Scheme

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT 'usr1/pwd1@n1';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT 'usr2/pwd2@n2';
CONFIGURE CHANNEL 3 DEVICE TYPE sbt CONNECT 'usr3/pwd3@n3';
```

```
BACKUP DATABASE PLUS ARCHIVELOG DELETE INPUT;
```



Multiple Drives CFS Backup Scheme

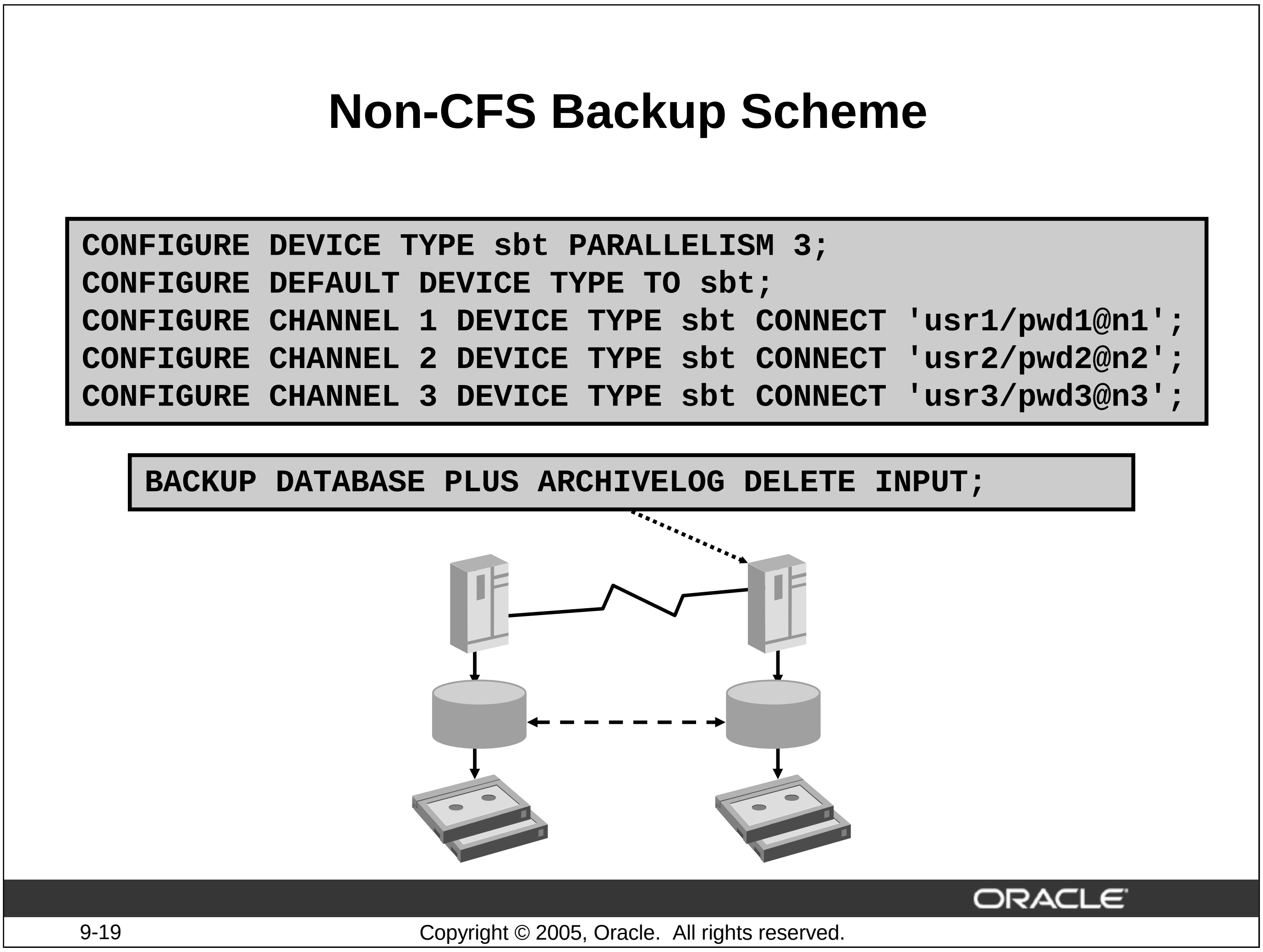
When backing up to multiple drives in the cluster file system backup scheme, it is assumed that each node in the cluster has its own local tape drive. Perform the following one-time configuration so that one channel is configured for each node in the cluster. This is a one-time configuration step. For example, enter the following at the RMAN prompt:

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT 'user1/passwd1@node1';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT 'user2/passwd2@node2';
CONFIGURE CHANNEL 3 DEVICE TYPE sbt CONNECT 'user3/passwd3@node3';
```

Similarly, you can perform this configuration for a device type of `DISK`. The following backup script, which you can run from any node in the cluster, distributes the datafiles, archived redo logs, and SPFILE backups among the backup drives:

```
BACKUP DATABASE PLUS ARCHIVELOG DELETE INPUT;
```

For example, if the database contains 10 datafiles and 100 archived redo logs are on disk, then the node 1 backup drive can back up datafiles 1, 3, and 7 and logs 1-33. Node 2 can back up datafiles 2, 5, and 10 and logs 34-66. The node 3 backup drive can back up datafiles 4, 6, 8 and 9 as well as archived redo logs 67-100.



Non-CFS Backup Scheme

```
CONFIGURE DEVICE TYPE sbt PARALLELISM 3;
CONFIGURE DEFAULT DEVICE TYPE TO sbt;
CONFIGURE CHANNEL 1 DEVICE TYPE sbt CONNECT 'usr1/pwd1@n1';
CONFIGURE CHANNEL 2 DEVICE TYPE sbt CONNECT 'usr2/pwd2@n2';
CONFIGURE CHANNEL 3 DEVICE TYPE sbt CONNECT 'usr3/pwd3@n3';
```

```
BACKUP DATABASE PLUS ARCHIVELOG DELETE INPUT;
```

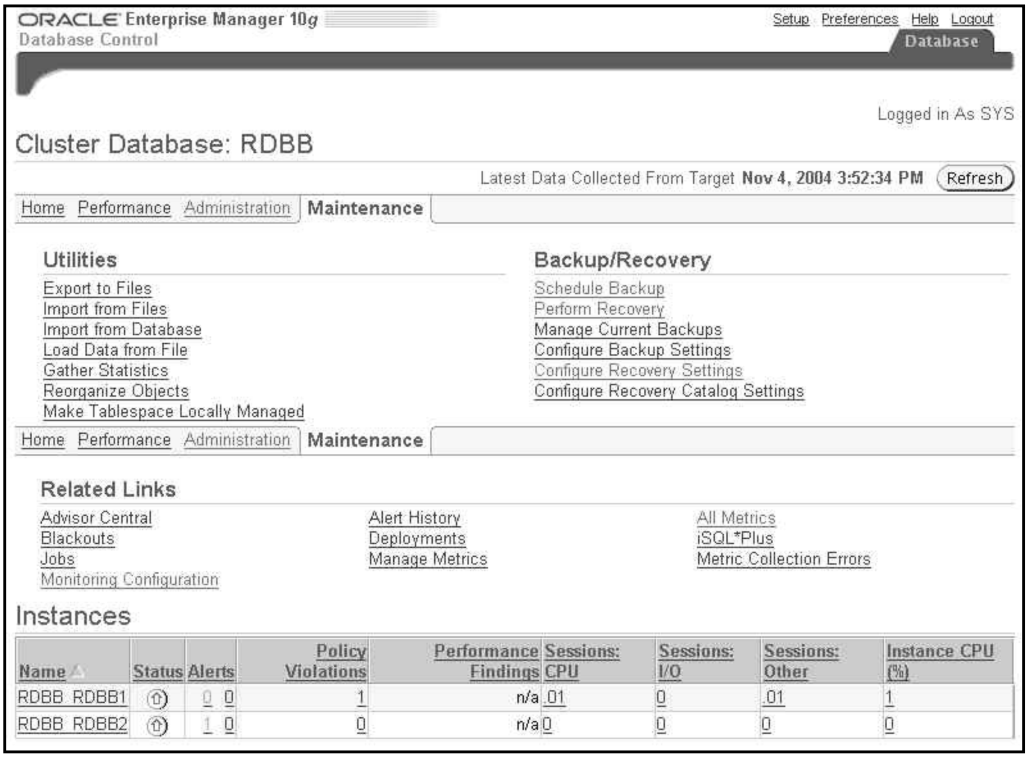
9-19

Non-Cluster File System Backup Scheme

In a non-cluster file system environment, each node can back up only its own local archived redo logs. For example, node 1 cannot access the archived redo logs on node 2 or node 3 unless you configure the network file system for remote access. To configure NFS, distribute the backup to multiple drives. However, if you configure NFS for backups, then you can only back up to one drive.

When backing up to multiple drives in a non-cluster file system backup scheme, it is assumed that each node in the cluster has its own local tape drive. You can perform a similar one-time configuration that the one shown above to configure one channel for each node in the cluster. Similarly, you can perform this configuration for a device type of `DISK`. Develop a production backup script for whole database backups that you can run from any node. With the `BACKUP` example, the datafiles backups, archived redo logs, and `SPFILE` backups are distributed among the different tape drives. However, channel 1 can only read the logs archived locally on `/arc_dest_1`. This is because the autolocation feature restricts channel 1 to only back up the archived redo logs in the `/arc_dest_1` directory. Because node 2 can only read files in the `/arc_dest_2` directory, channel 2 can only back up the archived redo logs in the `/arc_dest_2` directory, and so on. The important point is that all logs are backed up, but they are distributed among the different drives.

RAC Backup and Recovery Using EM



ORACLE Enterprise Manager 10*g*
Database Control

Setup Preferences Help Logout

Database

Logged in As SYS

Cluster Database: RDBB

Latest Data Collected From Target **Nov 4, 2004 3:52:34 PM** Refresh

Home Performance Administration **Maintenance**

Utilities

- Export to Files
- Import from Files
- Import from Database
- Load Data from File
- Gather Statistics
- Reorganize Objects
- Make Tablespace Locally Managed

Backup/Recovery

- Schedule Backup
- Perform Recovery
- Manage Current Backups
- Configure Backup Settings
- Configure Recovery Settings
- Configure Recovery Catalog Settings

Home Performance Administration **Maintenance**

Related Links

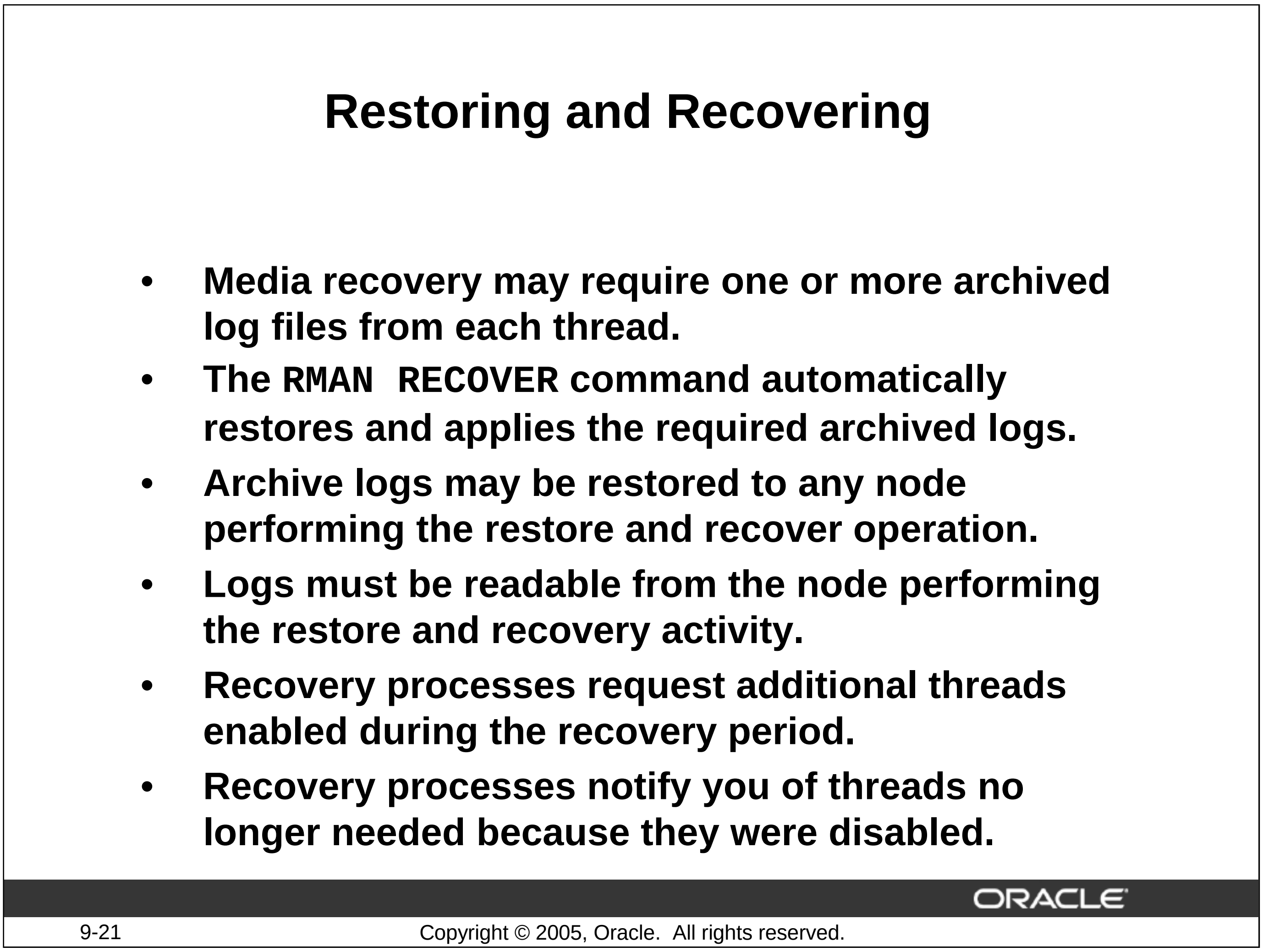
- Advisor Central
- Blackouts
- Jobs
- Monitoring Configuration
- Alert History
- Deployments
- Manage Metrics
- All Metrics
- iSQL*Plus
- Metric Collection Errors

Instances

Name △	Status	Alerts	Policy Violations	Performance Findings	Sessions: CPU	Sessions: I/O	Sessions: Other	Instance CPU (%)
RDBB_RDBB1	⇧	0 0	1	n/a	.01	0	.01	1
RDBB_RDBB2	⇧	1 0	0	n/a	0	0	0	0

RAC Backup and Recovery Using EM

The **Cluster Database Maintenance** page can be reached from the **Cluster Database Home** page by clicking the **Maintenance** tab link. From this page, you can perform a range of backup and recovery operations using RMAN, such as scheduling backups, performing recovery when necessary, and configuring backup and recovery settings. Also, you have links for executing different utilities to import or export data, load data, gather statistics on objects, reorganize objects, and convert a tablespace to be locally managed. As with other pages, the **Related Links** and **Instances** sections are available for you to manage other aspects of your cluster database.



Restoring and Recovering

- Media recovery may require one or more archived log files from each thread.
- The `RMAN RECOVER` command automatically restores and applies the required archived logs.
- Archive logs may be restored to any node performing the restore and recover operation.
- Logs must be readable from the node performing the restore and recovery activity.
- Recovery processes request additional threads enabled during the recovery period.
- Recovery processes notify you of threads no longer needed because they were disabled.

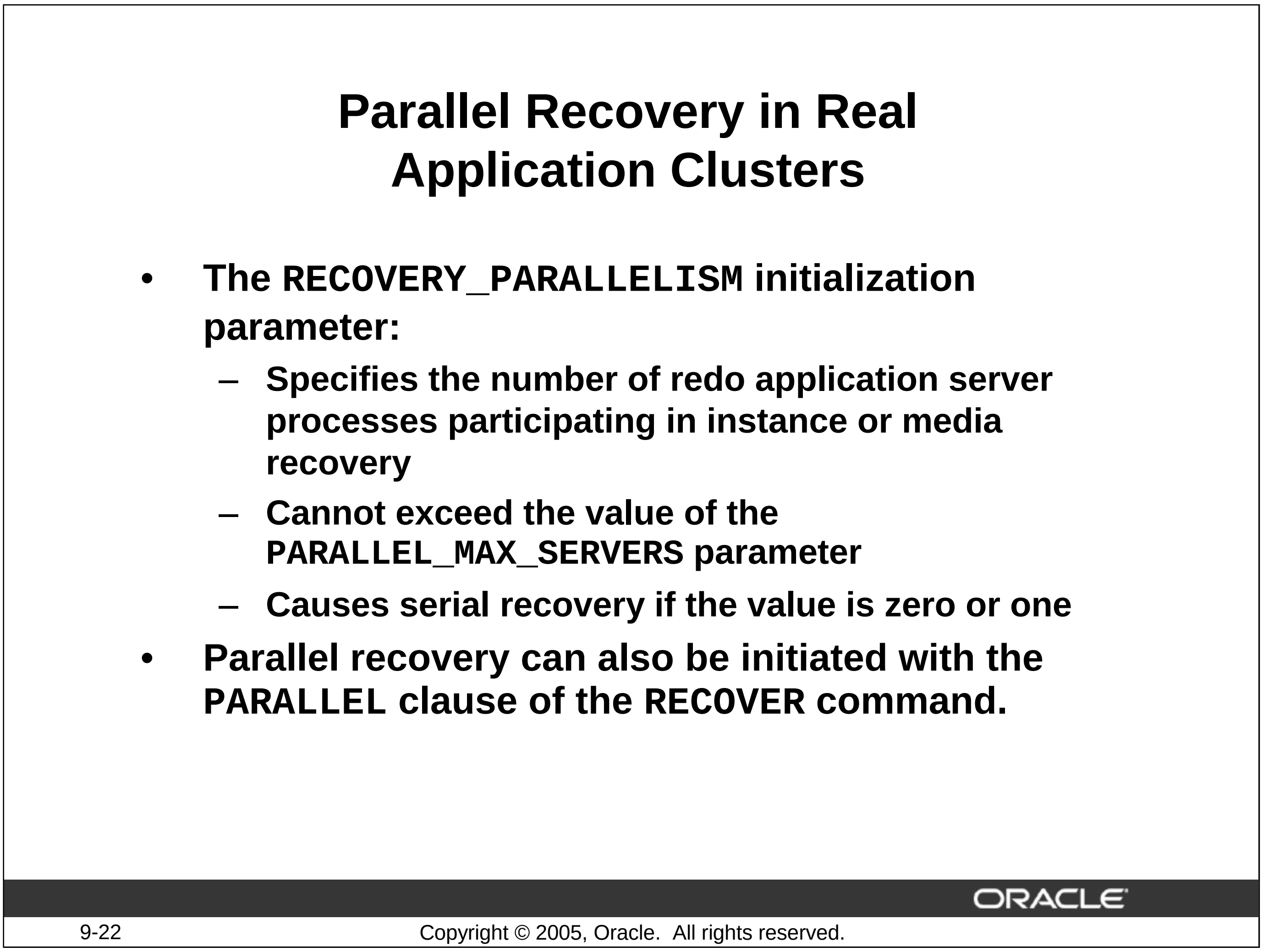
9-21

Restoring and Recovering

Media recovery of a database that is accessed by RAC may require at least one archived log file for each thread. However, if a thread’s online redo log contains enough recovery information, restoring archived log files for any thread is unnecessary.

If you use RMAN for media recovery and you share archive log directories, you can change the destination of the automatic restoration of archive logs with the SET clause to restore the files to a local directory of the node where you begin recovery. If you backed up the archive logs from each node without using a central media management system, you must first restore all the log files from the remote nodes and move them to the host from which you will start recovery with RMAN. However, if you backed up each node’s log files using a central media management system, you can use RMAN’s `AUTOLOCATE` feature. This enables you to recover a database using the local tape drive on the remote node.

If recovery reaches a time when an additional thread was enabled, the recovery process requests the archived log file for that thread. If you are using a backup control file, when all archive log files are exhausted you may need to redirect the recovery process to the online redo log files to complete recovery. If recovery reaches a time when a thread was disabled, the process informs you that the log file for that thread is no longer needed.



Parallel Recovery in Real Application Clusters

- The `RECOVERY_PARALLELISM` initialization parameter:
  - Specifies the number of redo application server processes participating in instance or media recovery
  - Cannot exceed the value of the `PARALLEL_MAX_SERVERS` parameter
  - Causes serial recovery if the value is zero or one
- Parallel recovery can also be initiated with the `PARALLEL` clause of the `RECOVER` command.

9-22

Parallel Recovery in RAC

Parallel recovery uses multiple CPUs and I/O parallelism to reduce the time that is required to perform thread or media recovery. Parallel recovery is most effective at reducing recovery time while concurrently recovering several data files on several disks. You can use parallel instance and crash recovery as well as parallel media recovery in RAC. You can use RMAN or other Oracle tools, such as SQL*Plus, to perform parallel recovery in RAC.

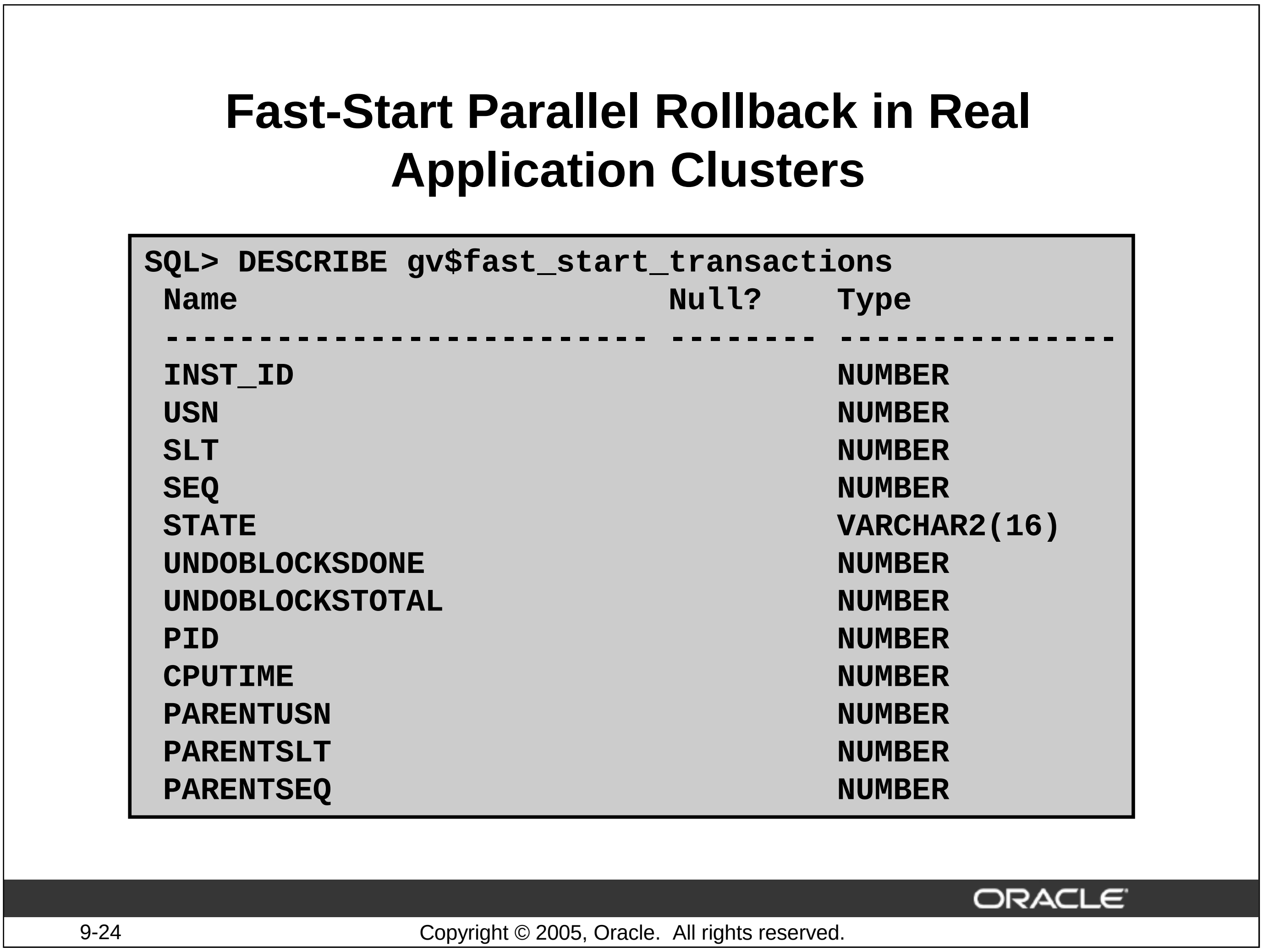
With RMAN’s `RESTORE` and `RECOVER` statements, The Oracle server automatically parallelizes recovery:

- Restoring data files: The number of allocated channels determines the maximum degree of parallelism.
- Applying incremental backups: The maximum degree of parallelism is determined by the number of allocated channels.

To perform parallel recovery with other tools, either set the `RECOVERY_PARALLELISM` initialization parameter or include the `PARALLEL` clause in your `RECOVER` command.

Parallel Recovery in Real Application Clusters

- **Parallel recovery uses one process to read the log files and multiple processes to apply redo to the data files.**
- **Oracle automatically starts the recovery processes—you do not need to use more than one session.**
- **The processes may be on one instance or spread across all instances.**



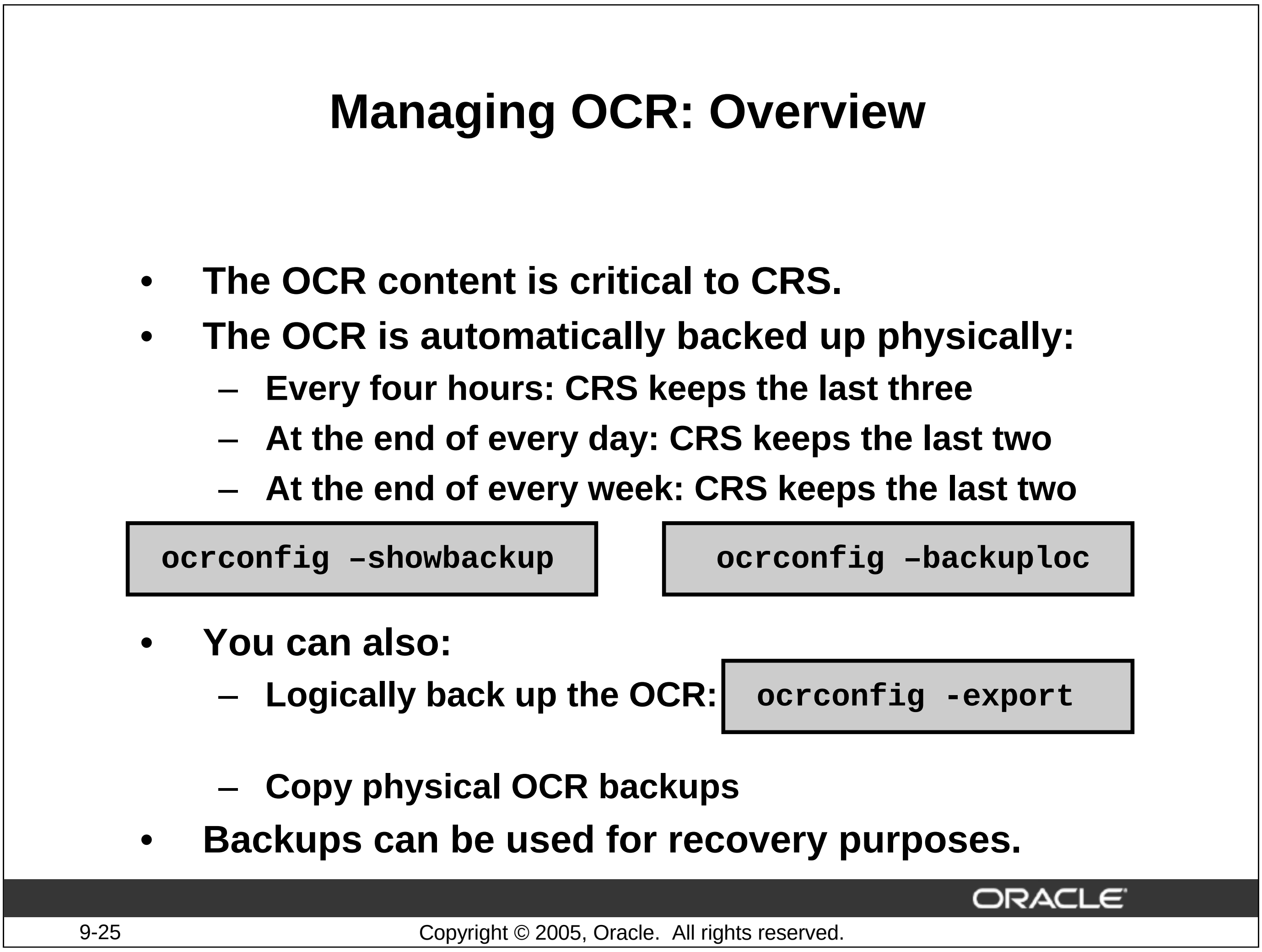
Fast-Start Parallel Rollback in Real Application Clusters

```
SQL> DESCRIBE gv$fast_start_transactions
 Name                       Null?    Type
 -------------------------- -------- ---------------
 INST_ID                             NUMBER
 USN                                 NUMBER
 SLT                                 NUMBER
 SEQ                                 NUMBER
 STATE                               VARCHAR2(16)
 UNDOBLOCKSDONE                      NUMBER
 UNDOBLOCKSTOTAL                     NUMBER
 PID                                 NUMBER
 CPUTIME                             NUMBER
 PARENTUSN                           NUMBER
 PARENTSLT                           NUMBER
 PARENTSEQ                           NUMBER
```

Fast-Start Parallel Rollback in RAC

If you have long-running transactions and your nodes have multiple CPUs, you should consider setting the initialization parameter `FAST_START_PARALLEL_ROLLBACK` to a nondefault value. This enables the rollback step of the recovery process to be performed in parallel. Since each instance is responsible for its own recovery processes, you need to monitor and tune the parallel rollback using the contents of `V$FAST_START_SERVERS` and `V$FAST_START_TRANSACTIONS` on each instance or the global views, `GV$FAST_START_SERVERS` and `GV$FAST_START_TRANSACTIONS`, from one of the active instances.

Although fast-start parallel rollback does not perform rollback activity across instances, it can improve the processing of rollback segments for a RAC database.



Managing OCR: Overview

- The OCR content is critical to CRS.
- The OCR is automatically backed up physically:
  - Every four hours: CRS keeps the last three
  - At the end of every day: CRS keeps the last two
  - At the end of every week: CRS keeps the last two

```
ocrconfig –showbackup
```

```
ocrconfig –backuploc
```

- You can also:
  - Logically back up the OCR: `ocrconfig -export`
  - Copy physical OCR backups
- Backups can be used for recovery purposes.

9-25

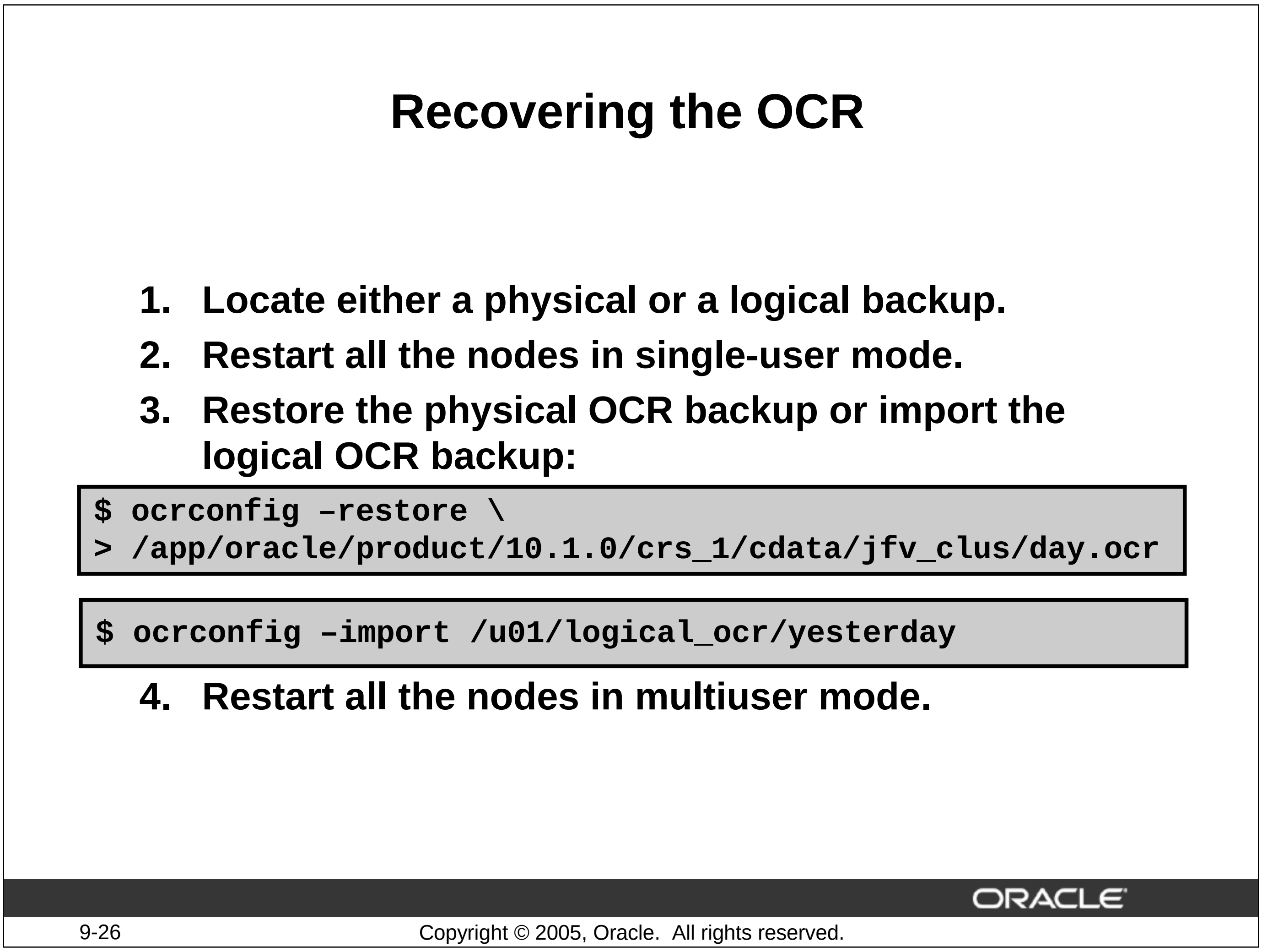
Managing the OCR: Overview

The OCR contains important cluster and database configuration information for RAC and CRS. One CRS instance in the cluster automatically creates OCR backups every four hours, and CRS retains the last three copies. The CRS instance also creates an OCR backup at the beginning of each day and of each week, and retains the last two copies. Although you cannot customize the backup frequencies or the number of retained copies, you have the possibility to identify the name and location of the automatically retained copies by using the `ocrconfig -showbackup` command.

The default target location of each automatically generated OCR backup file is the `<CRS Home>/cdata/<cluster name>` directory. It is recommended to change this location to one that is shared by all nodes in the cluster by using the `ocrconfig -backuploc` command. This command takes one argument that is the full path directory name of the new location.

Because of the importance of OCR information, it is also recommended to manually create copies of the automatically generated physical backups, and use the `ocrconfig -export` command to generate OCR logical backups. You need to specify a file name as the argument of the command, and it generates a binary file that you should not try to edit.

Note: If you try to export the OCR while an OCR client is running, then you get an error.



Recovering the OCR

1. Locate either a physical or a logical backup.
2. Restart all the nodes in single-user mode.
3. Restore the physical OCR backup or import the logical OCR backup:

```
$ ocrconfig –restore \
> /app/oracle/product/10.1.0/crs_1/cdata/jfv_clus/day.ocr
```

```
$ ocrconfig –import /u01/logical_ocr/yesterday
```

4. Restart all the nodes in multiuser mode.

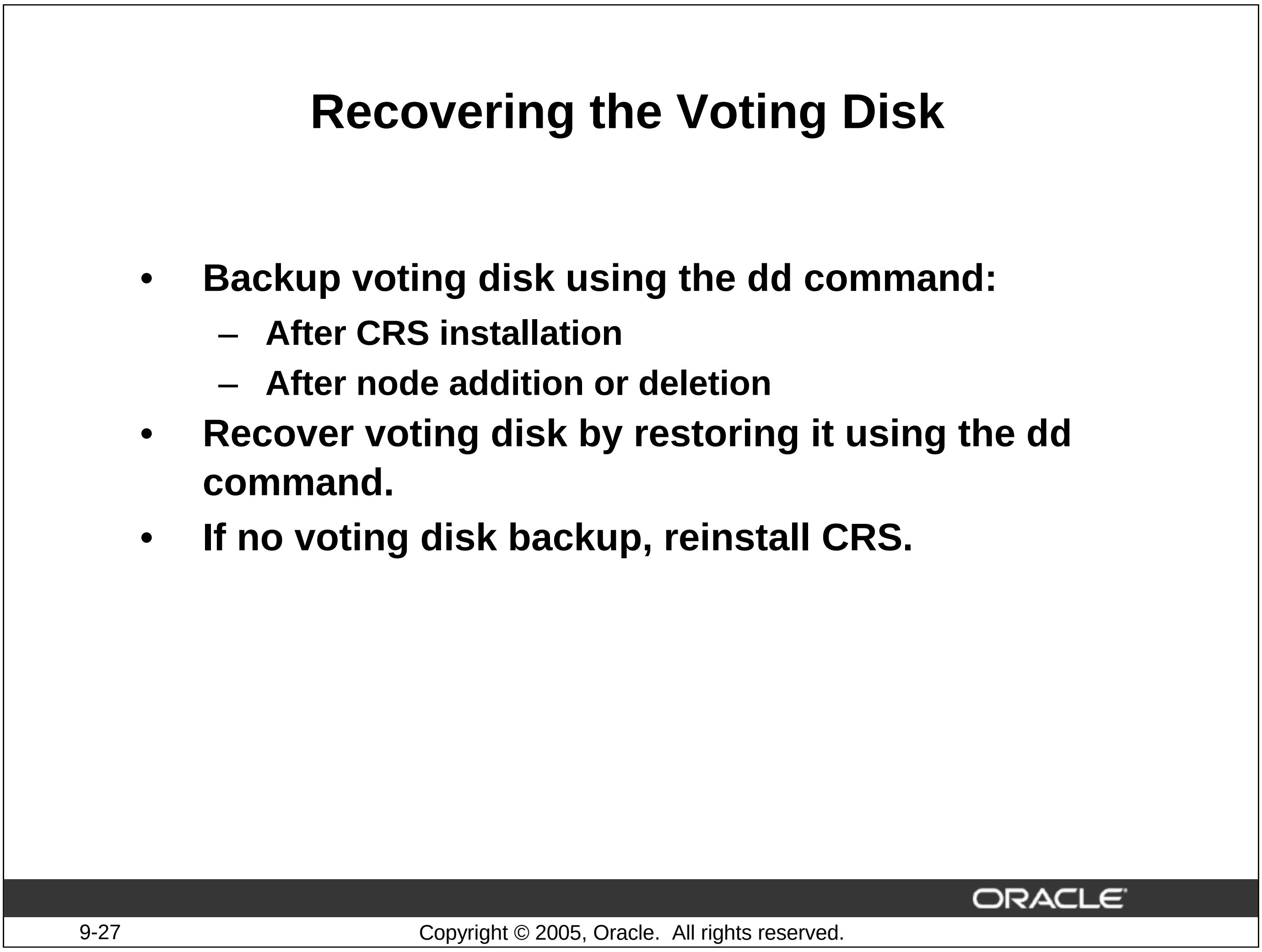
9-26

Recovering the OCR

If you need to recover the OCR, you can follow this procedure:

1. Find an OCR backup from a time before the inconsistency occurred.
2. If you have a backup of the OCR from before the time of the inconsistency, restart all the nodes in single-user mode or runlevel 1.
3. After all nodes are restarted in single-user mode, restore or import the OCR with the `ocrconfig` utility as shown in the slide.
4. Restart all the nodes in multiuser mode.

Note: It is highly recommended to store the OCR on RAID disk arrays.



Recovering the Voting Disk

- Backup voting disk using the dd command:
  - After CRS installation
  - After node addition or deletion
- Recover voting disk by restoring it using the dd command.
- If no voting disk backup, reinstall CRS.

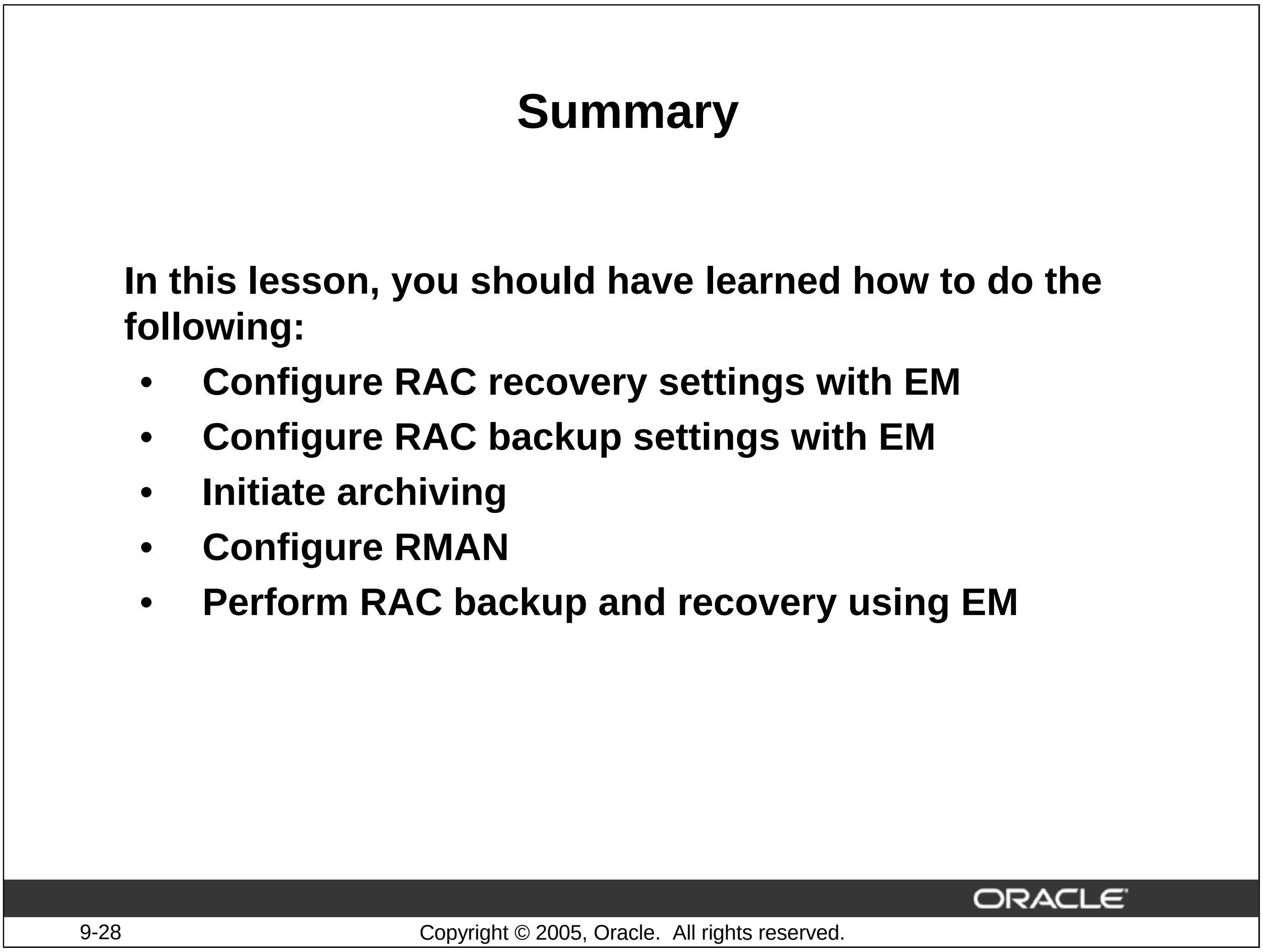
9-27

Recovering the Voting Disk

Basically, the voting disk should be stored on redundant devices to avoid its loss. However, if you loose your voting disk, it is possible to restore it using the `dd` command if you have a backup.

It is good practice to backup your voting disk right after CRS installation, and whenever you add or remove nodes to your cluster.

If you loose your voting disk, and you do not have any backup, then you must reinstall CRS.

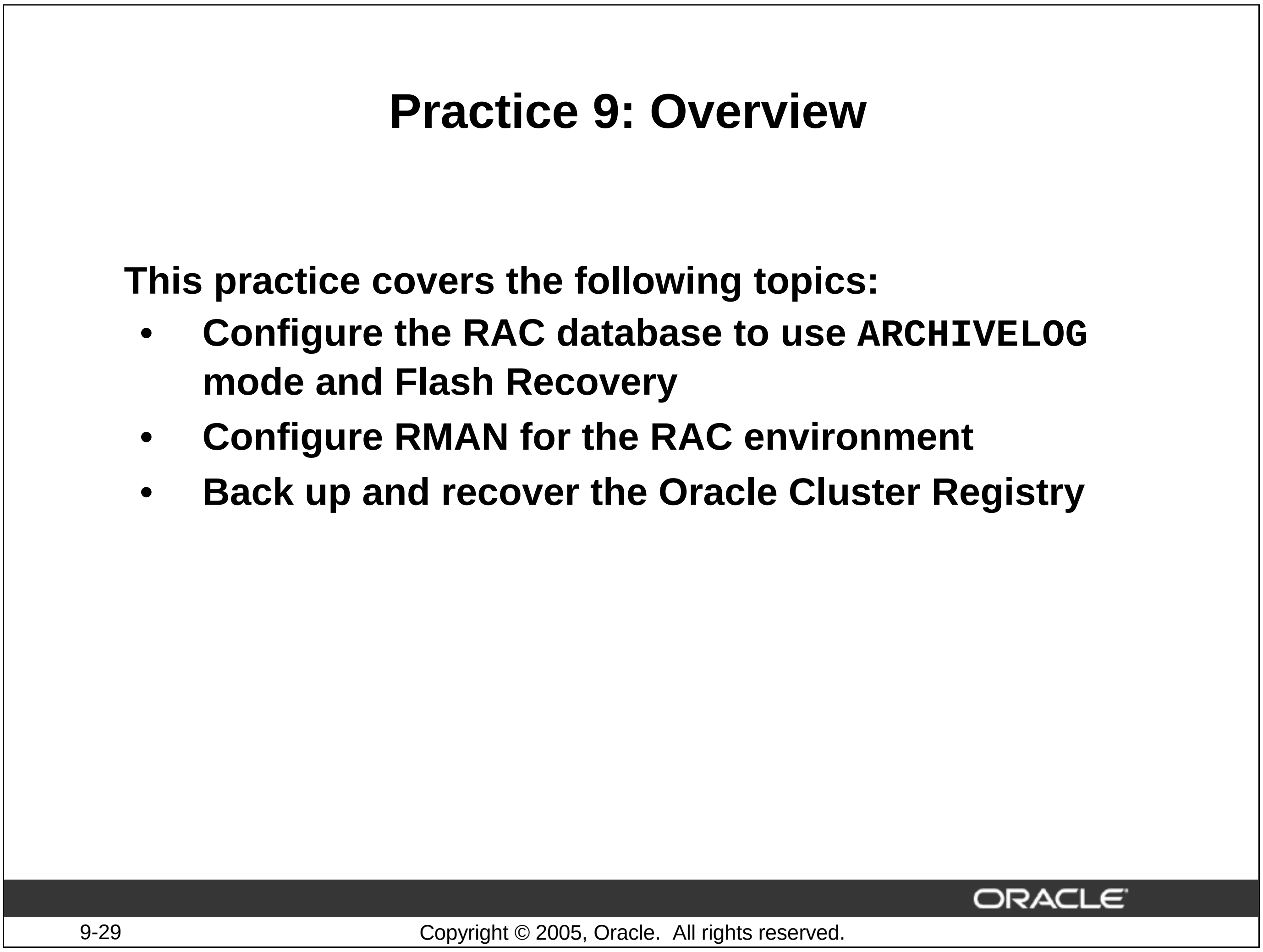


Summary

In this lesson, you should have learned how to do the following:

- Configure RAC recovery settings with EM
- Configure RAC backup settings with EM
- Initiate archiving
- Configure RMAN
- Perform RAC backup and recovery using EM

9-28



Practice 9: Overview

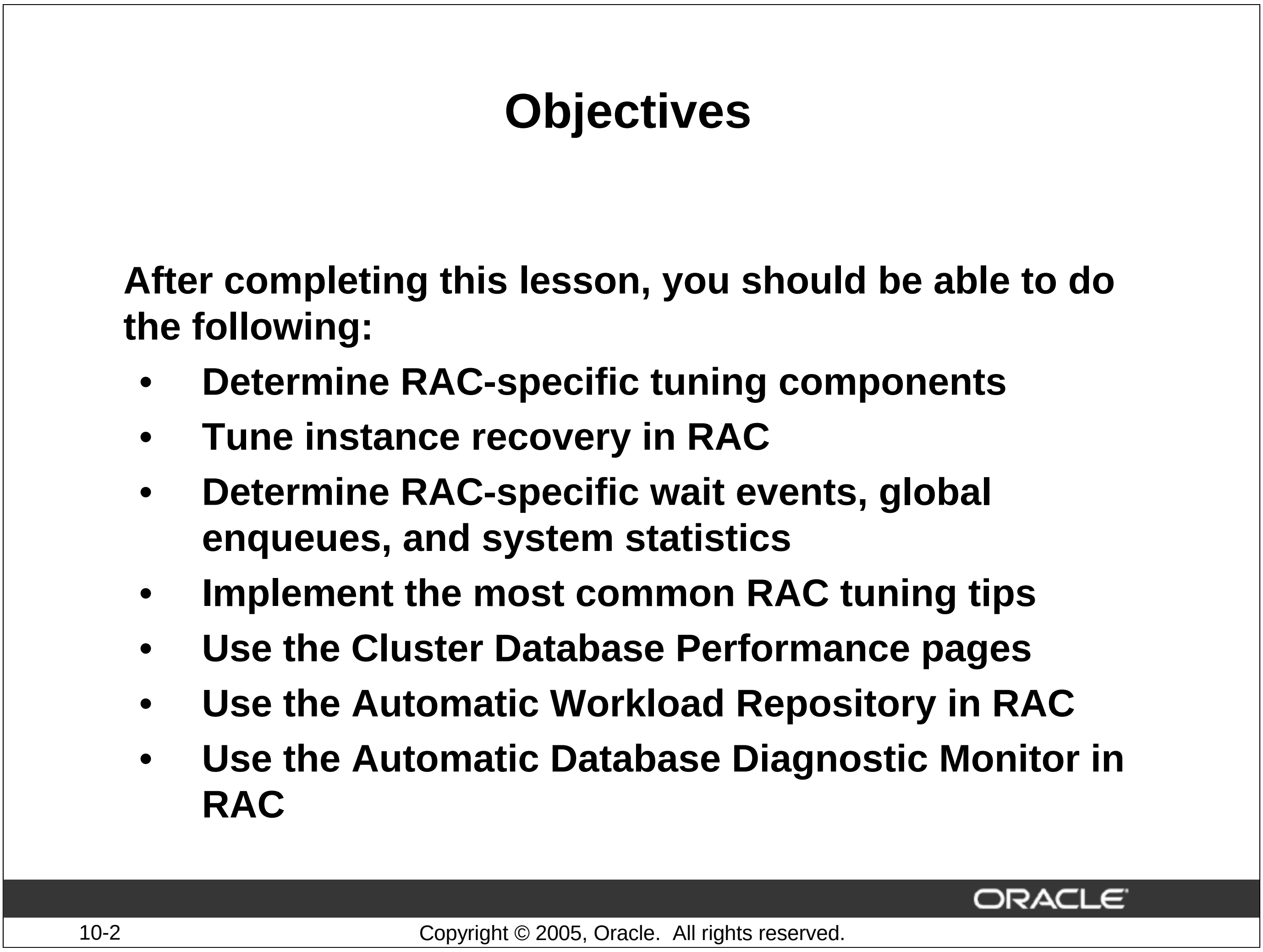
This practice covers the following topics:

- Configure the RAC database to use `ARCHIVELOG` mode and Flash Recovery
- Configure RMAN for the RAC environment
- Back up and recover the Oracle Cluster Registry

9-29

10

RAC Performance Tuning

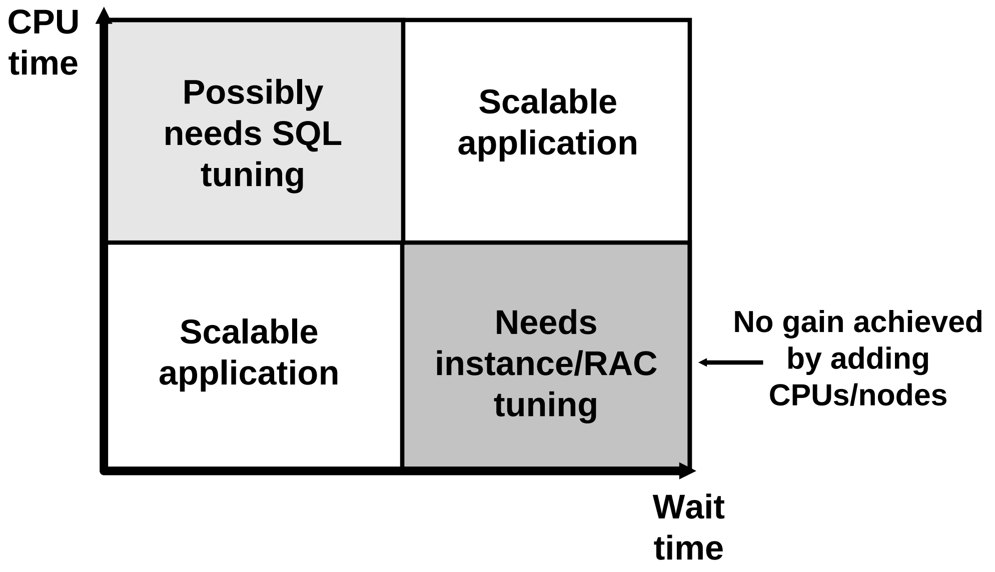


Objectives

After completing this lesson, you should be able to do the following:

- Determine RAC-specific tuning components
- Tune instance recovery in RAC
- Determine RAC-specific wait events, global enqueues, and system statistics
- Implement the most common RAC tuning tips
- Use the Cluster Database Performance pages
- Use the Automatic Workload Repository in RAC
- Use the Automatic Database Diagnostic Monitor in RAC

CPU and Wait Time Tuning Dimensions



CPU time

Possibly needs SQL tuning

Scalable application

Scalable application

Needs instance/RAC tuning

No gain achieved by adding CPUs/nodes

Wait time

CPU and Wait Time Tuning Dimensions

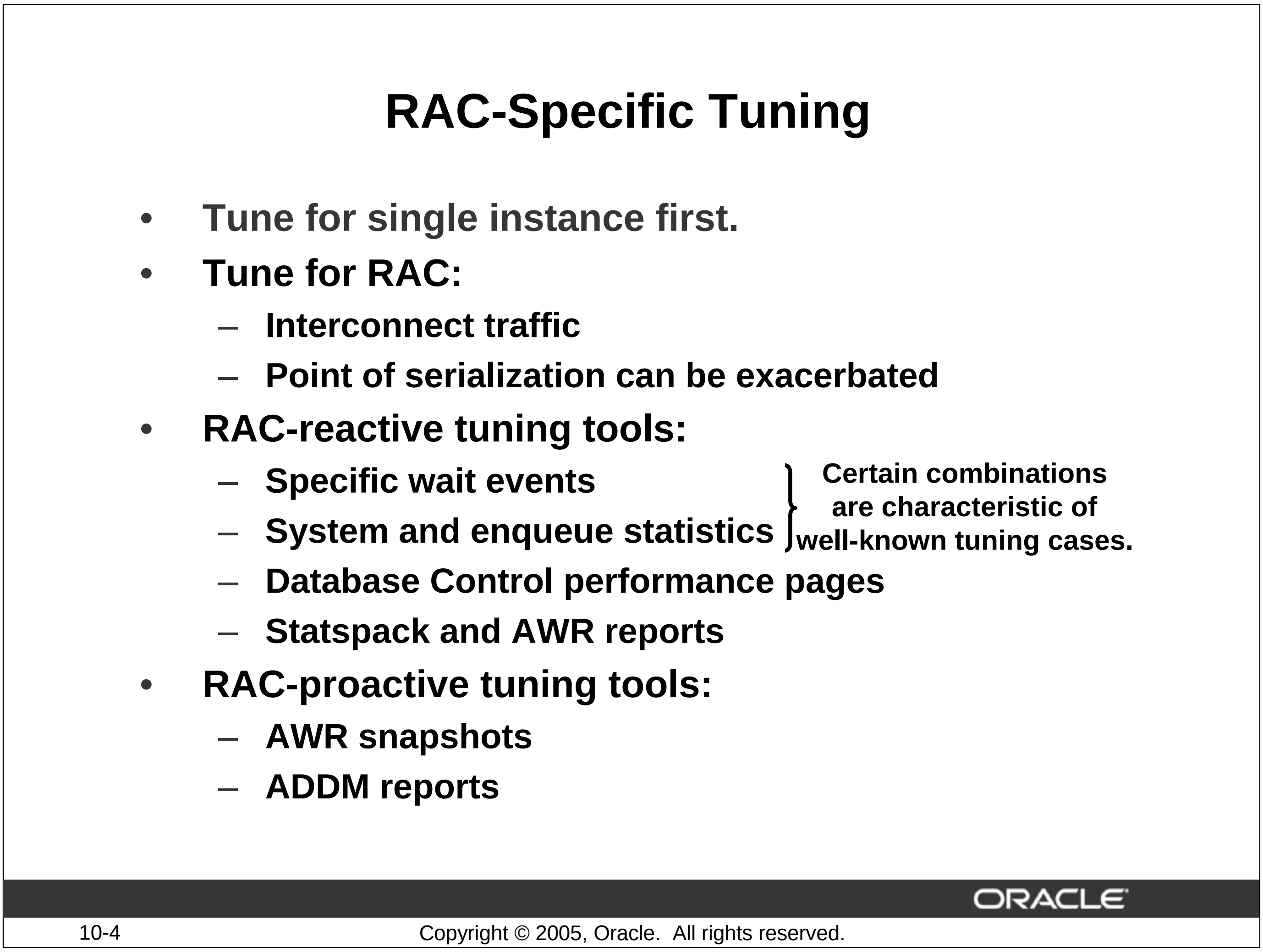
When tuning your system, it is important that you compare the CPU time with the wait time of your system. Comparing CPU time with wait time helps to determine how much of the response time is spent on useful work and how much on waiting for resources potentially held by other processes.

As a general rule, the systems where CPU time is dominant usually need less tuning than the ones where wait time is dominant. On the other hand, heavy CPU usage can be caused by badly written SQL statements.

Although the proportion of CPU time to wait time always tends to decrease as load on the system increases, steep increases in wait time are a sign of contention and must be addressed for good scalability.

Adding more CPUs to a node, or nodes to a cluster, would provide very limited benefit under contention. Conversely, a system where the proportion of CPU time does not decrease significantly as load increases can scale better, and would most likely benefit from adding CPUs or Real Application Clusters (RAC) instances if needed.

Note: Automatic Workload Repository (AWR) reports display CPU time together with wait time in the **Top 5 Event** section, if the CPU time portion is among the top five events.



RAC-Specific Tuning

- Tune for single instance first.
- Tune for RAC:
  - Interconnect traffic
  - Point of serialization can be exacerbated
- RAC-reactive tuning tools:
  - Specific wait events
  - System and enqueue statistics

    Certain combinations are characteristic of well-known tuning cases.
  - Database Control performance pages
  - Statspack and AWR reports
- RAC-proactive tuning tools:
  - AWR snapshots
  - ADDM reports

10-4

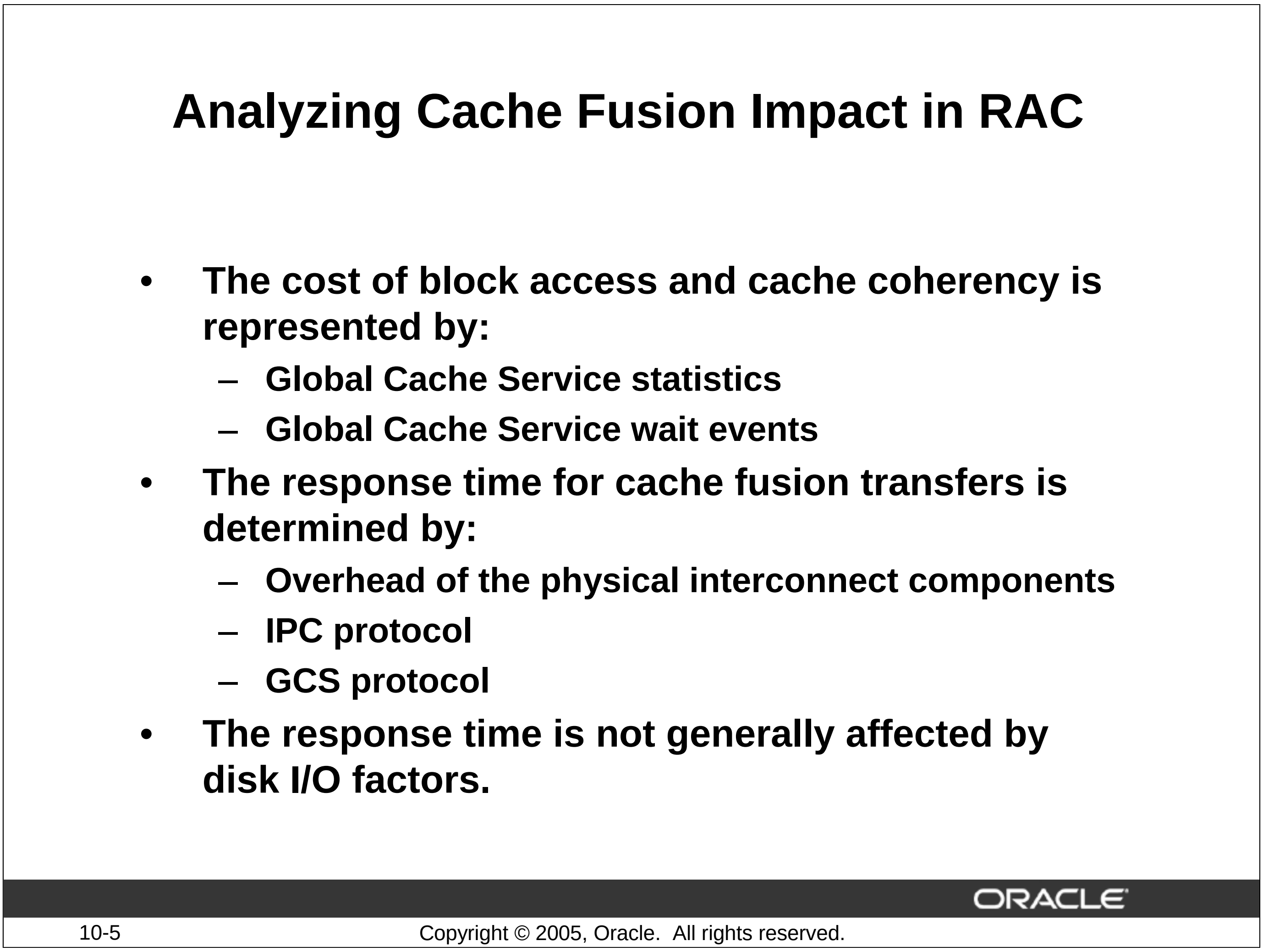
RAC-Specific Tuning

Although there are specific tuning areas for RAC, such as interconnect traffic, you get most benefits by tuning your system like a single-instance system. At least, this must be your starting point.

Obviously, if you have serialization issues in a single-instance environment, these may be exacerbated with RAC.

As shown in the slide, you have basically the same tuning tools with RAC as with a single-instance system. However, certain combinations of specific wait events and statistics are well-known RAC tuning cases.

In the remaining of this lesson, you see some of those specific combinations, as well as the RAC-specific information that you can get from the Database Control performance pages, and Statspack and AWR reports. Finally, you see the RAC-specific information that you can get from the Automatic Database Diagnostic Monitor (ADDM).



Analyzing Cache Fusion Impact in RAC

- **The cost of block access and cache coherency is represented by:**
  - **Global Cache Service statistics**
  - **Global Cache Service wait events**
- **The response time for cache fusion transfers is determined by:**
  - **Overhead of the physical interconnect components**
  - **IPC protocol**
  - **GCS protocol**
- **The response time is not generally affected by disk I/O factors.**

ORACLE

Analyzing Cache Fusion Impact in RAC

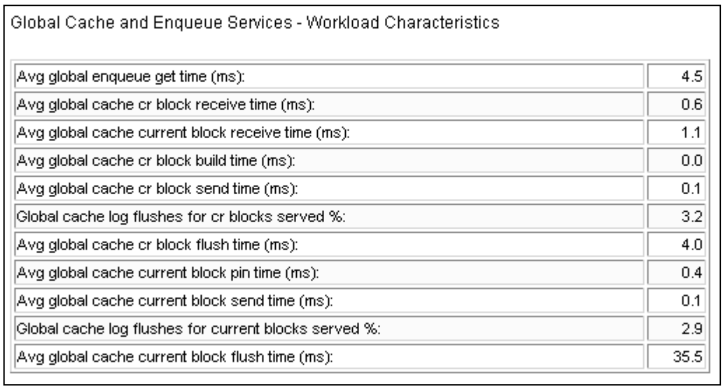
The effect of accessing blocks in the global cache and maintaining cache coherency is represented by:

- The Global Cache Service statistics for current and cr blocks; for example, gc current blocks received, gc cr blocks received, and so on.
- The Global Cache Service wait events for gc current block 3-way, gc cr grant 2-way, and so on.

The response time for cache fusion transfers is determined by the messaging time and processing time imposed by the physical interconnect components, the IPC protocol, and the GCS protocol. It is not affected by disk input/output (I/O) factors other than occasional log writes. The cache fusion protocol does not require I/O to data files in order to guarantee cache coherency, and RAC inherently does not cause any more I/O to disk than a nonclustered instance.

Typical Latencies for RAC Operations

AWR Report Latency Name	Lower Bound	Typical	Upper Bound
Average time to process cr block request	**0.1**	**1**	**10**
Avg `global` cache `cr` block `receive time (ms)`	**0.3**	**4**	**12**
Average time to process current block request	**0.1**	**3**	**23**
Avg `global` cache `current` block `receive time(ms)`	**0.3**	**8**	**30**



Global Cache and Enqueue Services - Workload Characteristics

Avg global enqueue get time (ms):	4.5
Avg global cache cr block receive time (ms):	0.6
Avg global cache current block receive time (ms):	1.1
Avg global cache cr block build time (ms):	0.0
Avg global cache cr block send time (ms):	0.1
Global cache log flushes for cr blocks served %:	3.2
Avg global cache cr block flush time (ms):	4.0
Avg global cache current block pin time (ms):	0.4
Avg global cache current block send time (ms):	0.1
Global cache log flushes for current blocks served %:	2.9
Avg global cache current block flush time (ms):	35.5

10-6

Typical Latencies for RAC Operations

In a RAC AWR report, there is a table in the RAC Statistics section containing average times (latencies) for some Global Cache Services and Global Enqueue Services operations. This table is shown in the slide and is called *Global Cache and Enqueue Services: Workload Characteristics*. Those latencies should be monitored over time, and significant increases in their values should be investigated. The table presents some typical values, based on empirical observations. Factors that may cause variations to those latencies include:

- Utilization of the IPC protocol. User-mode IPC protocols are faster.
- Scheduling delays, when the system is under high CPU utilization
- Log flushes for current blocks served

Other RAC latencies in AWR reports are mostly derived from `V$GES_STATISTICS` and may be useful for debugging purposes, but do not require frequent monitoring.

Note: The time to process consistent read (CR) block request in the cache corresponds to `(build time + flush time + send time)`, and the time to process current block request in the cache corresponds to`(pin time + flush time + send time)`.

Wait Events for RAC

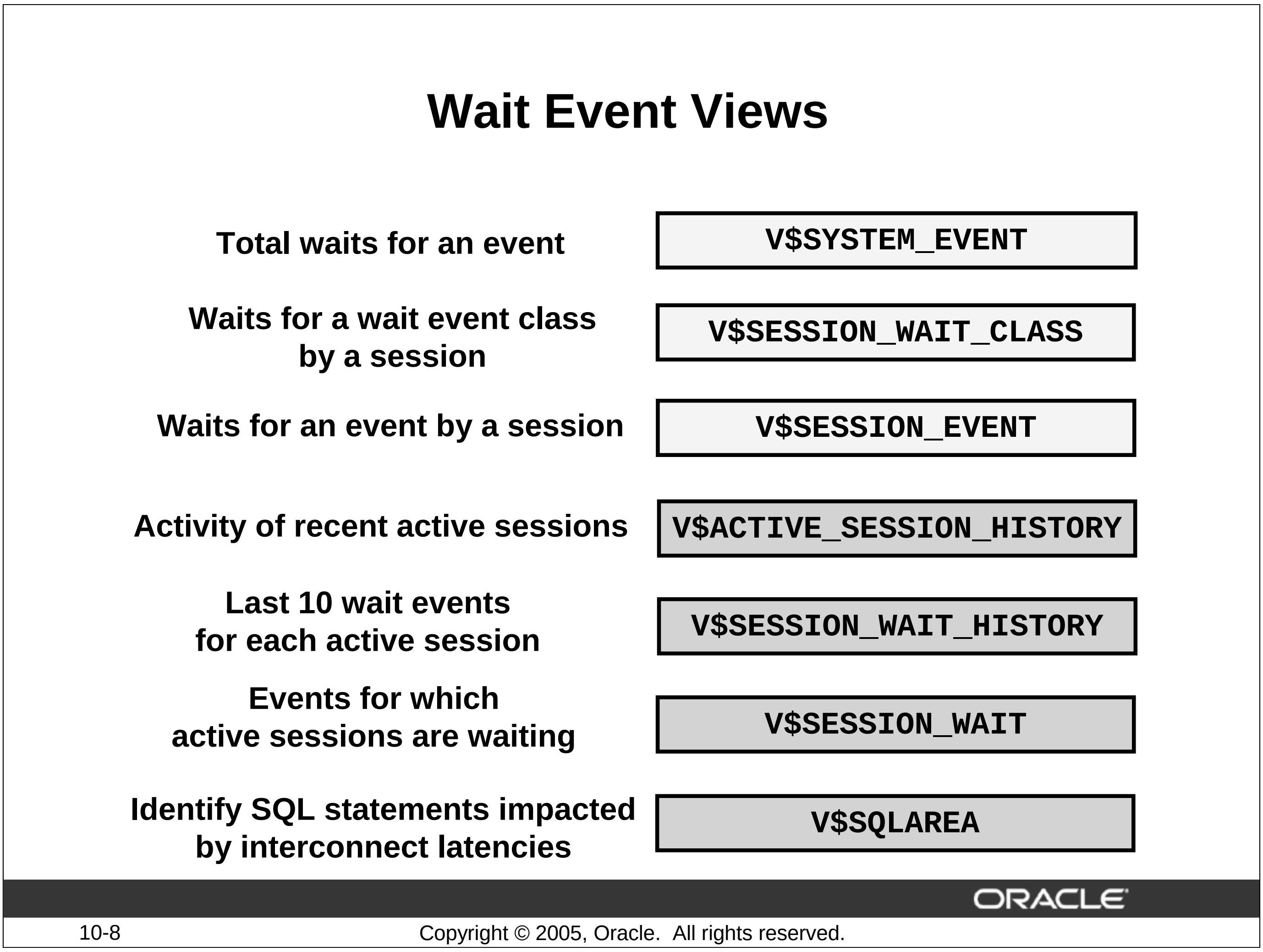
- Wait events help to analyze what sessions are waiting for.
- Wait times are attributed to events that reflect the outcome of a request:
  - Placeholders while waiting
  - Precise events when waited
- Global cache waits are summarized in a broader category called Cluster Wait Class.
- These wait events are used in the ADDM to enable cache fusion diagnostics.

Wait Events for RAC

Analyzing what sessions are waiting for is an important method to determine where time is spent. In RAC, the wait time is attributed to an event that reflects the exact outcome of a request. For example, when a session on an instance is looking for a block in the global cache, it does not know whether it will receive the data cached by another instance or whether it will receive a message to read from disk. The wait events for the global cache convey precise information and wait for global cache blocks or messages. They are mainly categorized by the following:

- Summarized in a broader category called Cluster Wait Class
- Temporarily represented by a placeholder event that is active while waiting for a block
- Attributed to precise events when the outcome of the request is known

The wait events for RAC convey information valuable for performance analysis. They are used in the ADDM to enable precise diagnostics of the impact of cache fusion.



Wait Event Views

Total waits for an event	V$SYSTEM_EVENT
Waits for a wait event class by a session	V$SESSION_WAIT_CLASS
Waits for an event by a session	V$SESSION_EVENT
Activity of recent active sessions	V$ACTIVE_SESSION_HISTORY
Last 10 wait events for each active session	V$SESSION_WAIT_HISTORY
Events for which active sessions are waiting	V$SESSION_WAIT
Identify SQL statements impacted by interconnect latencies	V$SQLAREA

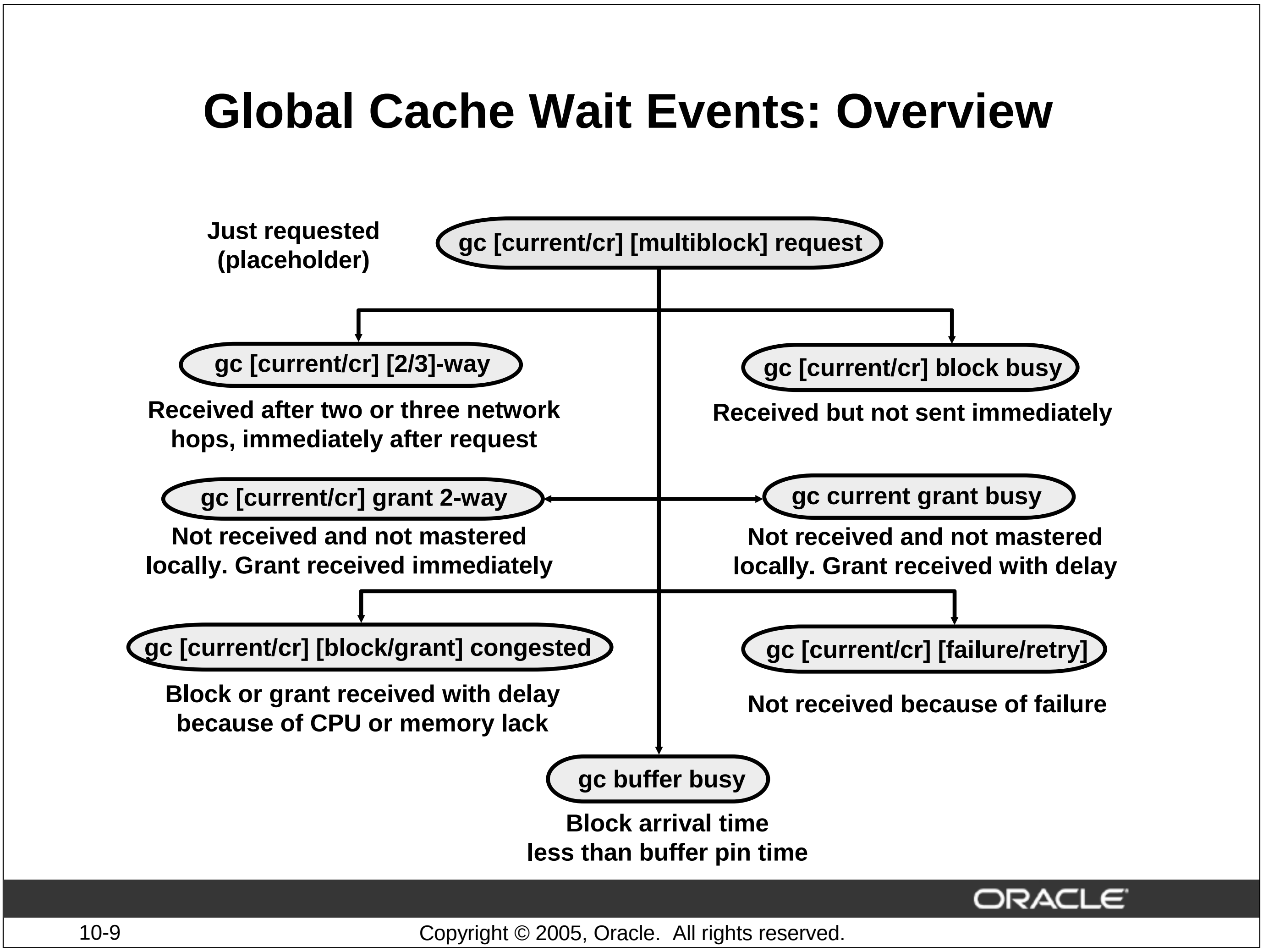
10-8

Wait Event Views

When it takes some time to acquire resources because of the total path length and latency for requests, processes sleep to avoid spinning for indeterminate periods of time. When the process decides to wait, it wakes up either after a specified timer value expires (timeout) or when the event it is waiting for occurs and the process is posted. The wait events are recorded and aggregated in the views shown in the slide. The first three are aggregations of wait times, timeouts, and the number of times waited for a particular event whereas the rest enable the monitoring of waiting sessions in real time, including a history of recent events waited for.

The individual events distinguish themselves by their names and the parameters that they assume. For most of the global cache wait events, the parameters include file number, block number, the block class, and access mode dispositions, such as mode held and requested. The wait times for events presented and aggregated in these views are very useful when debugging response time performance issues. Note that the time waited is cumulative, and that the event with the highest score is not necessarily a problem. However, if the available CPU power cannot be maxed out, or response times for an application are too high, the top wait events provide valuable performance diagnostics.

Note: Use the `CLUSTER_WAIT_TIME` column in `V$SQLAREA` to identify SQL statements impacted by interconnect latencies, or run an ADDM report on the corresponding AWR snapshot.



Global Cache Wait Events: Overview

Just requested
(placeholder)

gc [current/cr] [multiblock] request

gc [current/cr] [2/3]-way

Received after two or three network hops, immediately after request

gc [current/cr] block busy

Received but not sent immediately

gc [current/cr] grant 2-way

Not received and not mastered locally. Grant received immediately

gc current grant busy

Not received and not mastered locally. Grant received with delay

gc [current/cr] [block/grant] congested

Block or grant received with delay because of CPU or memory lack

gc [current/cr] [failure/retry]

Not received because of failure

gc buffer busy

Block arrival time
less than buffer pin time

Global Cache Wait Events: Overview

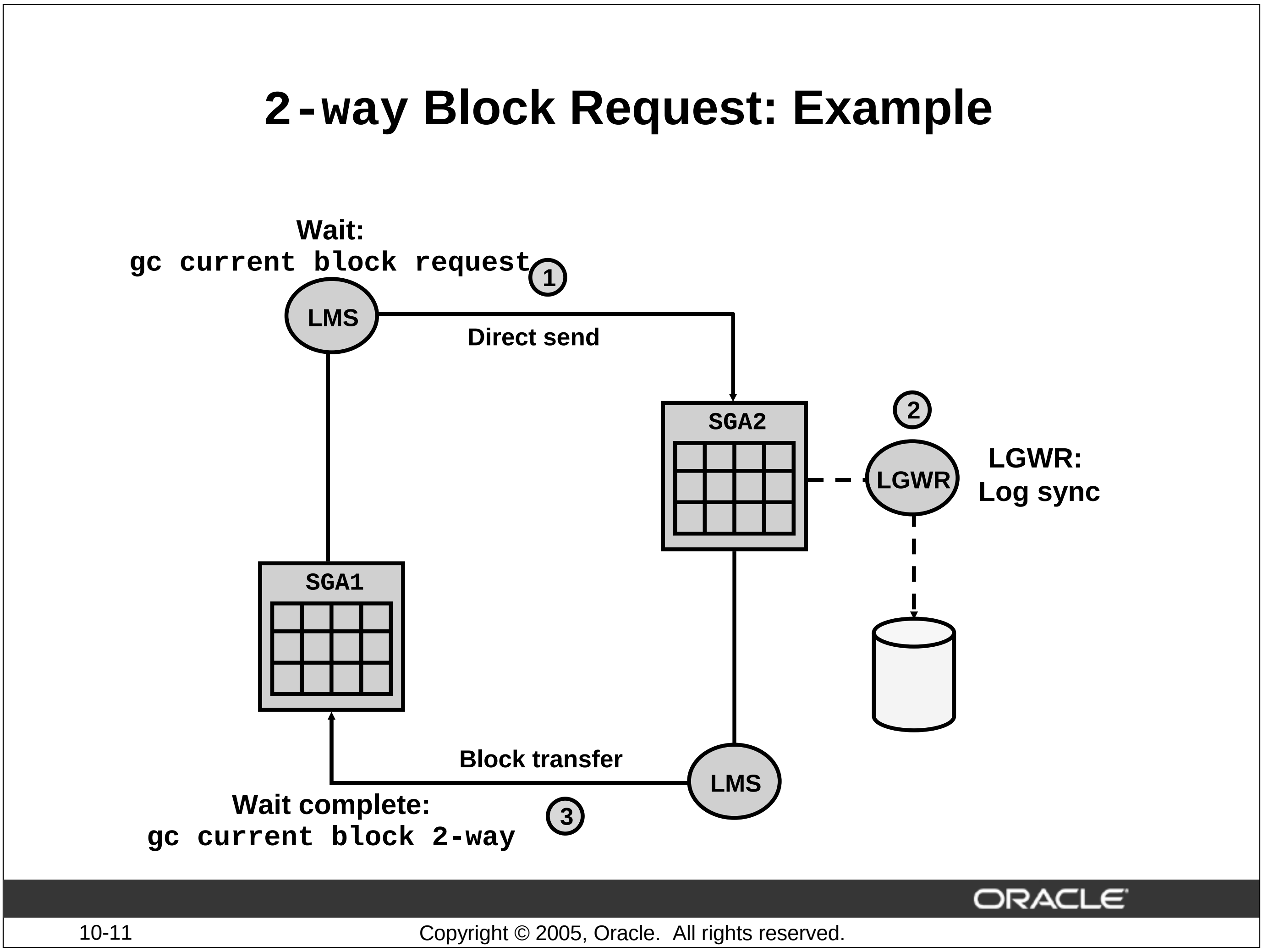
The main Global Cache wait events for Oracle Database 10*g* are described briefly in the slide:

- `gc current/cr request`: These wait events are relevant only while a gc request for a cr or current buffer is in progress. They act as placeholders until the request completes.
- `gc [current/cr] [2/3]-way`: A current or cr block was requested and received after two or three network hops. The request was processed immediately: it was not busy or congested.
- `gc [current/cr] block busy`: A current or cr block was requested and received, but was not sent immediately by LMS because some special condition that delayed the sending was found.
- `gc [current/cr] grant 2-way`: A current or cr block was requested and a grant message received. The grant was given without any significant delays. If the block is not in its local cache, a current or cr grant is followed by a disk read on the requesting instance.
- `gc current grant busy`: A current block was requested and a grant message received. The busy hint implies that the request was blocked because others were ahead of it or it could not be handled immediately.

Global Cache Wait Events: Overview (continued)

- `gc [current/cr] [block/grant] congested`: A current or cr block was requested and a block or grant message received. The congested hint implies that the request spent more than 1 ms in internal queues.
- `gc [current/cr] [failure/retry]`: A block was requested and a failure status was received or some other exceptional event has occurred.
- `gc buffer busy`: If the time between buffer accesses becomes less than the time the buffer is pinned in memory, the buffer containing a block is said to become busy and as a result interested users may have to wait for it to be unpinned.

Note: For more information, refer to the *Oracle Database Reference* guide.



2-way Block Request: Example

10-11

2-way Block Request: Example

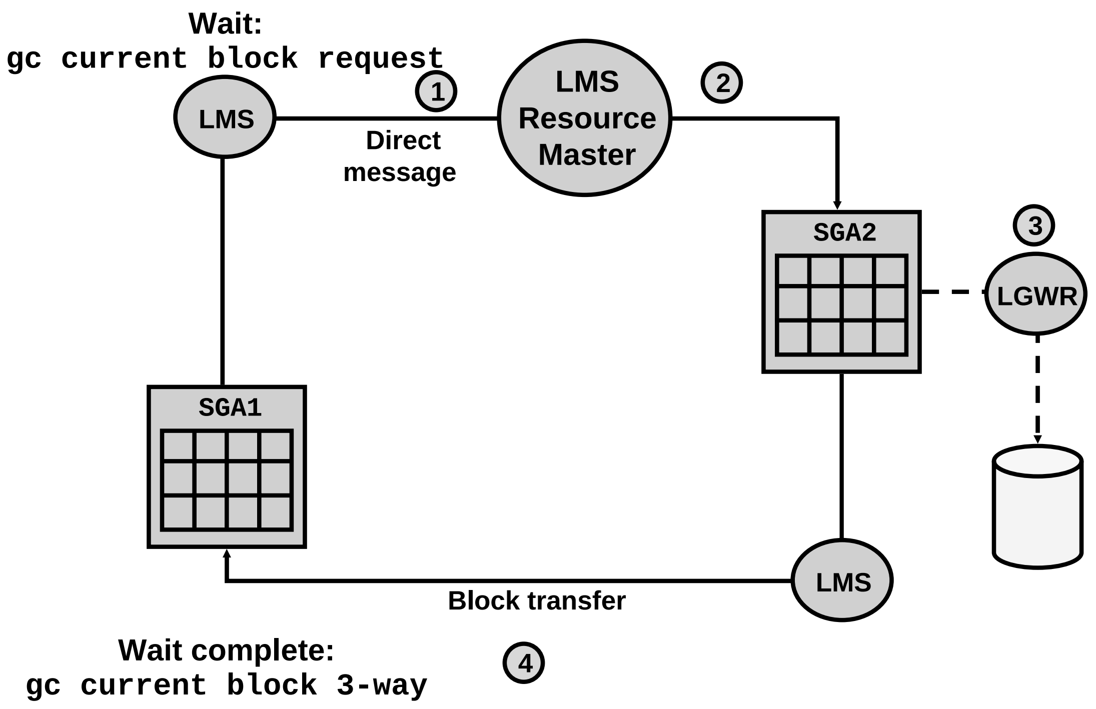
This slide shows you what typically happens when the master instance requests a block that is not cached locally. Here it is supposed that the master instance is called `SGA1`, and `SGA2` contains the requested block. The scenario is as follows:

1. `SGA1` sends a direct request to `SGA2`. So `SGA1` waits on the `gc current block request` event.
2. When `SGA2` receives the request, its local `LGWR` process may need to flush some recovery information to its local redo log files. For example, if the cached block is frequently changed, and the changes have not been logged yet, `LMS` would have to ask `LGWR` to flush the log before it can ship the block. This may add a delay to the serving of the block and may show up in the requesting node as a busy wait.
3. Then, `SGA2` sends the requested block to `SGA1`. When the block arrives in `SGA1`, the wait event is complete, and is reflected as `gc current block 2-way`.

Note: Using the notation `R`= time at requestor, `W`=wire time and transfer delay, and `S`= time at server, the total time for a round-trip would be:

`R`(send) + `W`(small msg) + `S`(process msg,process block,send) + `W`(block) + `R`(receive block)

3-way Block Request: Example



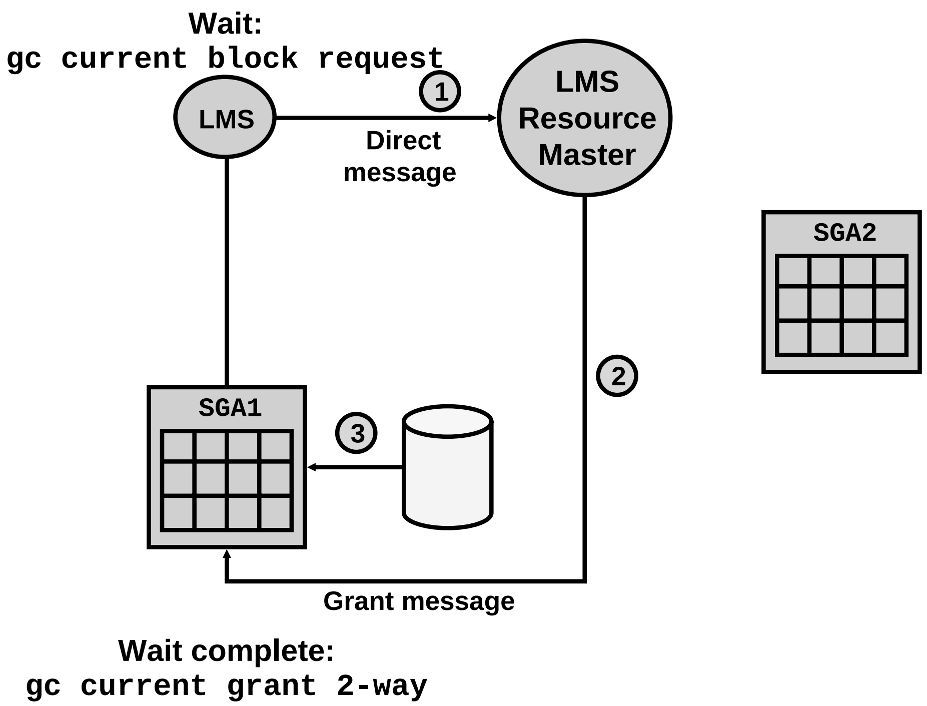
3-way Block Request: Example

This is a modified scenario for a cluster with more than two nodes. It is very similar to the previous one. However, the master for this block is on a node that is different from that of the requestor, and where the block is cached. Thus, the request must be forwarded. There is an additional delay for one message and the processing at the master node:

R(send) + W(small msg) + S(process msg,send) + W(small msg) + S(process msg,process block,send) + W(block) + R(receive block)

While a remote read is pending, any process on the requesting instance that is trying to write or read the data cached in the buffer has to wait for a `gc buffer busy`. The buffer remains globally busy until the block arrives.

2-way Grant: Example



Wait:
`gc current block request`

LMS

1

Direct message

LMS Resource Master

SGA2

2

SGA1

3

Grant message

Wait complete:
`gc current grant 2-way`

2-way Grant: Example

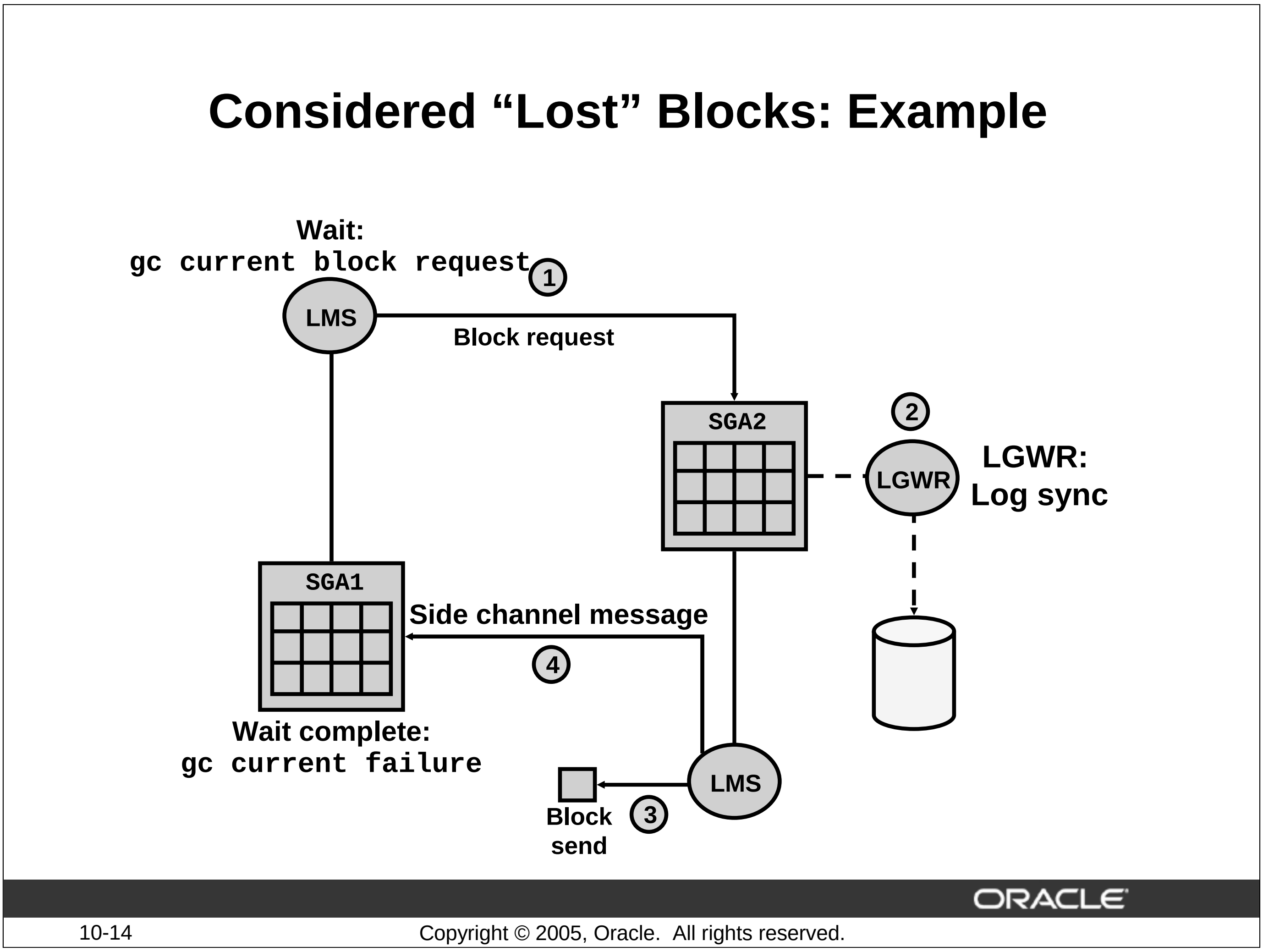
In this scenario, a grant message is sent by the master because the requested block is not cached in any instance.

If the local instance is the resource master, the grant happens immediately. If not, the grant is always 2-way, regardless of the number of instances in the cluster.

The grant messages are small. For every block read from the disk, a grant has to be received before the IO is initiated, which adds the latency of the grant round-trip to the disk latency:

`R`(send) + `W`(small msg) + `S`(process msg,send) + `W`(small msg) + `R`(receive block)

The round-trip looks similar to a 2-way block round-trip, with the difference that the wire time is determined by a small message, and the processing does not involve the buffer cache.



Considered “Lost” Blocks: Example

Wait:
`gc current block request`

1

LMS

Block request

SGA2

2

LGWR

LGWR:
Log sync

SGA1

Side channel message

4

Wait complete:
`gc current failure`

LMS

Block send

3

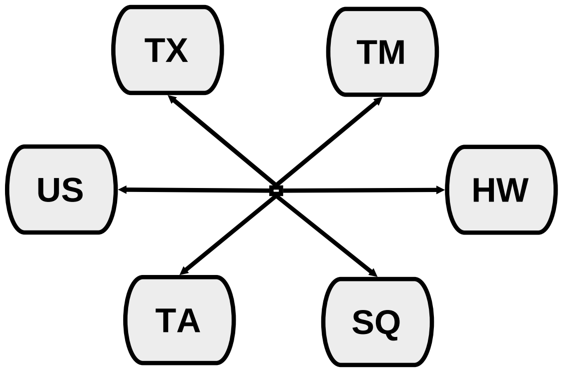
Considered “Lost” Blocks: Example

In this scenario, the side channel message arrives before the block itself. Under normal circumstances, this should never occur. Most of the time, this is an indicator of switch problems or the absence of a private interconnect. This is often related to operating system (OS) or network configuration issues.

Note: You can try to avoid these occurrences by reducing the value of the `DB_FILE_MULTIBLOCK_READ_COUNT` initialization parameter to less than `16`.

Global Enqueue Waits: Overview

- **Enqueues are synchronous.**
- **Enqueues are global resources in RAC.**
- **The most frequent waits are for:**



- **The waits may constitute serious serialization points.**

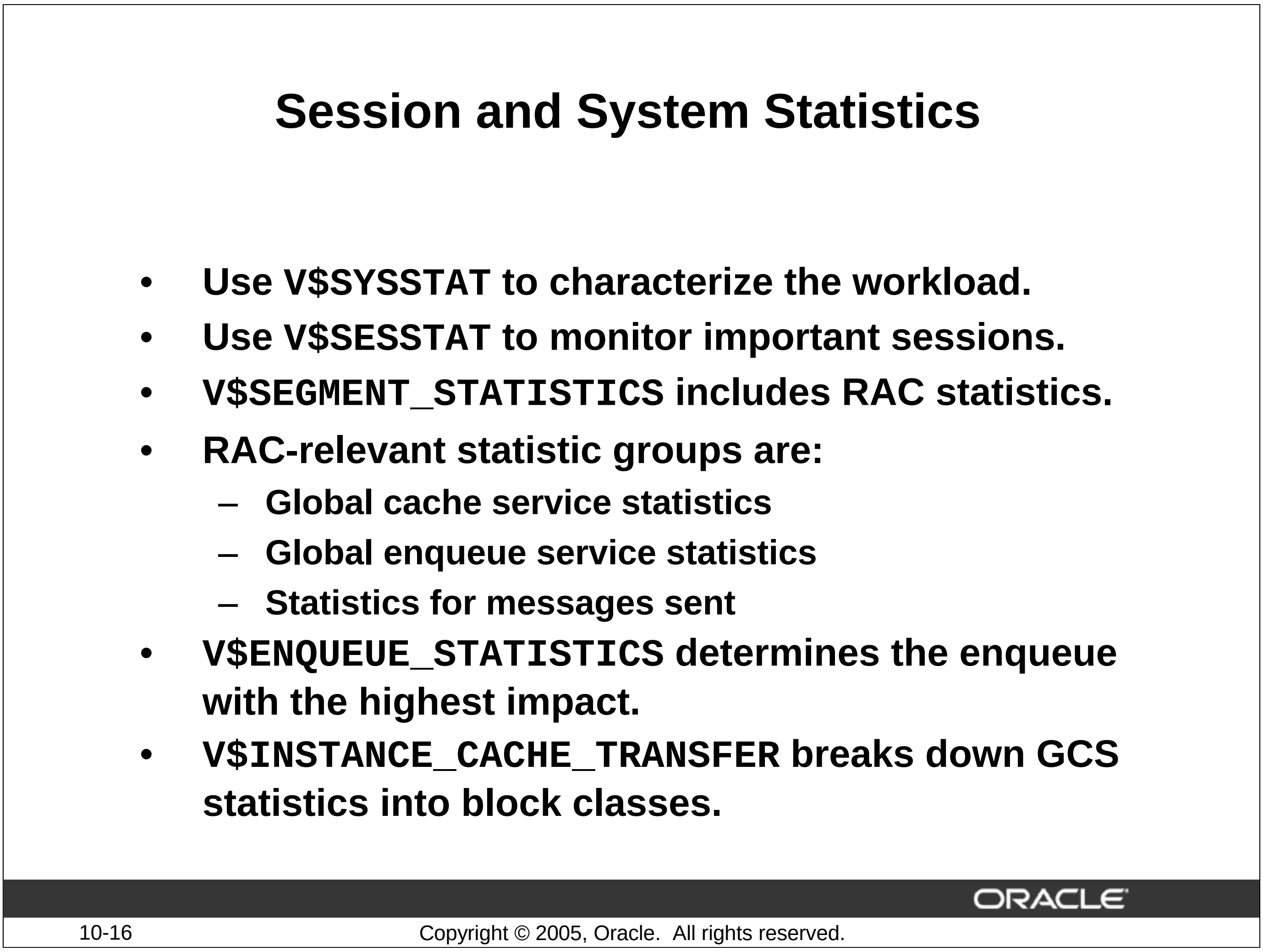
Global Enqueue Waits: Overview

An enqueue wait is not RAC specific, but involves a global lock operation when RAC is enabled. Most of the global requests for enqueues are synchronous, and foreground processes wait for them. Therefore, contention on enqueues in RAC is more visible than in single-instance environments. Most waits for enqueues occur for enqueues of the following types:

- TX: Transaction enqueue; used for transaction demarcation and tracking
- TM: Table or partition enqueue; used to protect table definitions during DML operations
- HW: High-Water Mark enqueue; acquired to synchronize a new block operation
- SQ: Sequence enqueue; used to serialize incrementing of an Oracle sequence number
- US: Undo Segment enqueue; mainly used by the Automatic Undo Management (AUM) feature
- TA: Enqueue used mainly for transaction recovery as part of instance recovery

In all of the cases above, the waits are synchronous and may constitute serious serialization points that can be exacerbated in a RAC environment.

Note: In Oracle Database 10*g*, the enqueue wait events specify the resource name and a reason for the wait. For example: *TX Enqueue index block split*. This makes diagnostics of enqueue waits easier.



Session and System Statistics

- Use V$SYSSTAT to characterize the workload.
- Use V$SESSTAT to monitor important sessions.
- `V$SEGMENT_STATISTICS` includes RAC statistics.
- RAC-relevant statistic groups are:
  - Global cache service statistics
  - Global enqueue service statistics
  - Statistics for messages sent
- `V$ENQUEUE_STATISTICS` determines the enqueue with the highest impact.
- `V$INSTANCE_CACHE_TRANSFER` breaks down GCS statistics into block classes.

Session and System Statistics

Using system statistics based on `V$SYSSTAT` enables characterization of the database activity based on averages. It is the basis for many metrics and ratios used in various tools and methods, such as AWR, Statspack, and Database Control.

In order to drill down to individual sessions or groups of sessions, `V$SESSTAT` is useful when the important session identifiers to monitor are known. Its usefulness is enhanced if an application fills in the `MODULE` and `ACTION` columns in `V$SESSION`.

`V$SEGMENT_STATISTICS` is useful for RAC because it also tracks the number of CR and current blocks received by the object.

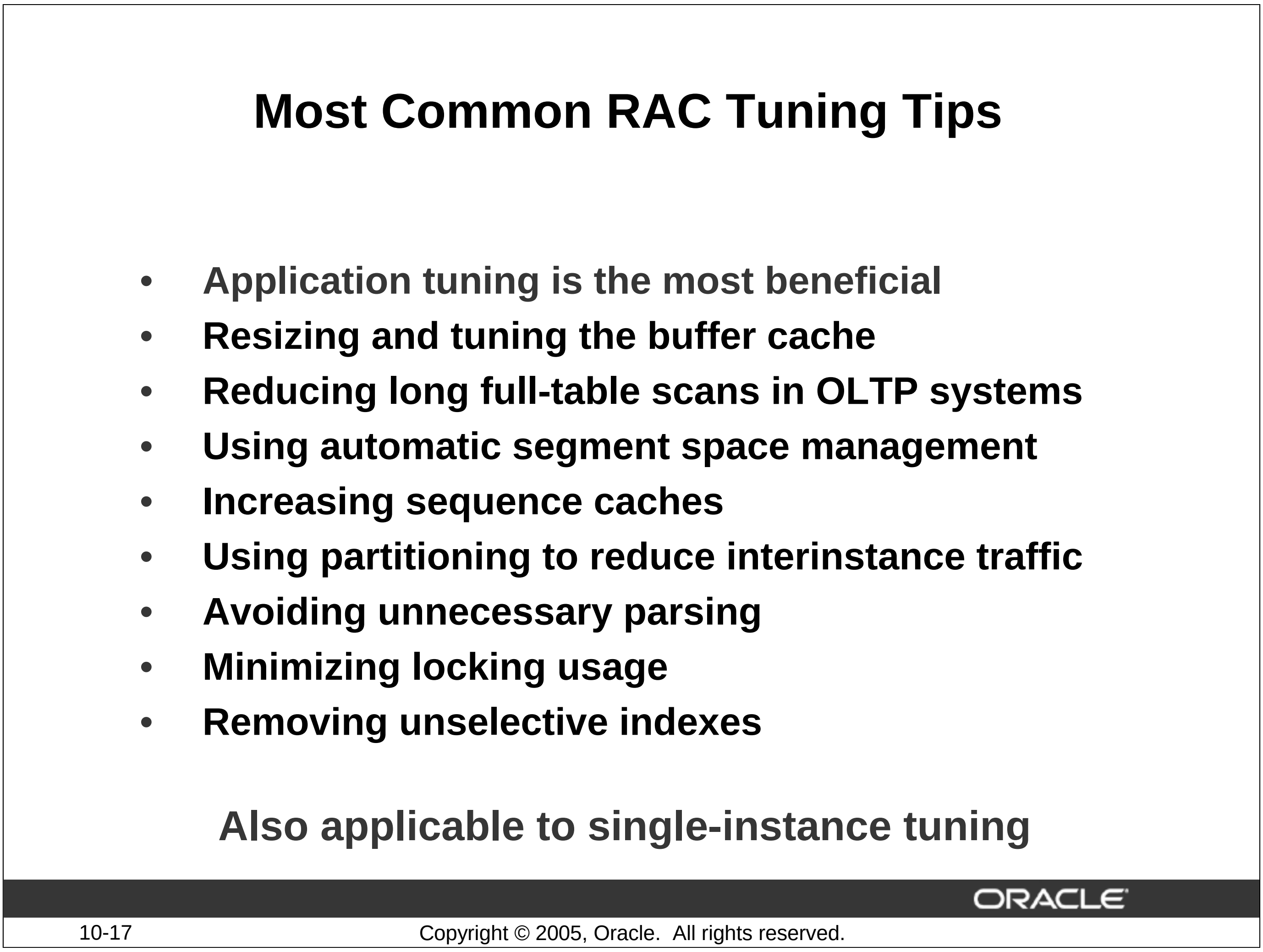
The RAC-relevant statistics can be grouped into:

- Global cache service statistics: *gc cr blocks received*, *gc cr block receive time*, and so on
- Global enqueue service statistics: *global enqueue gets*, and so on
- Statistics for messages sent: *gcs messages sent* and *ges messages sent*

`V$ENQUEUE_STATISTICS` can be queried to determine which enqueue has the highest impact on database service times and eventually response times.

`V$INSTANCE_CACHE_TRANSFER` indicates how many current and CR blocks per block class are received from each instance, including how many transfers incurred a delay.

Note: For more information about statistics, refer to the *Oracle Database Reference* guide.



Most Common RAC Tuning Tips

- Application tuning is the most beneficial
- Resizing and tuning the buffer cache
- Reducing long full-table scans in OLTP systems
- Using automatic segment space management
- Increasing sequence caches
- Using partitioning to reduce interinstance traffic
- Avoiding unnecessary parsing
- Minimizing locking usage
- Removing unselective indexes

Also applicable to single-instance tuning

10-17

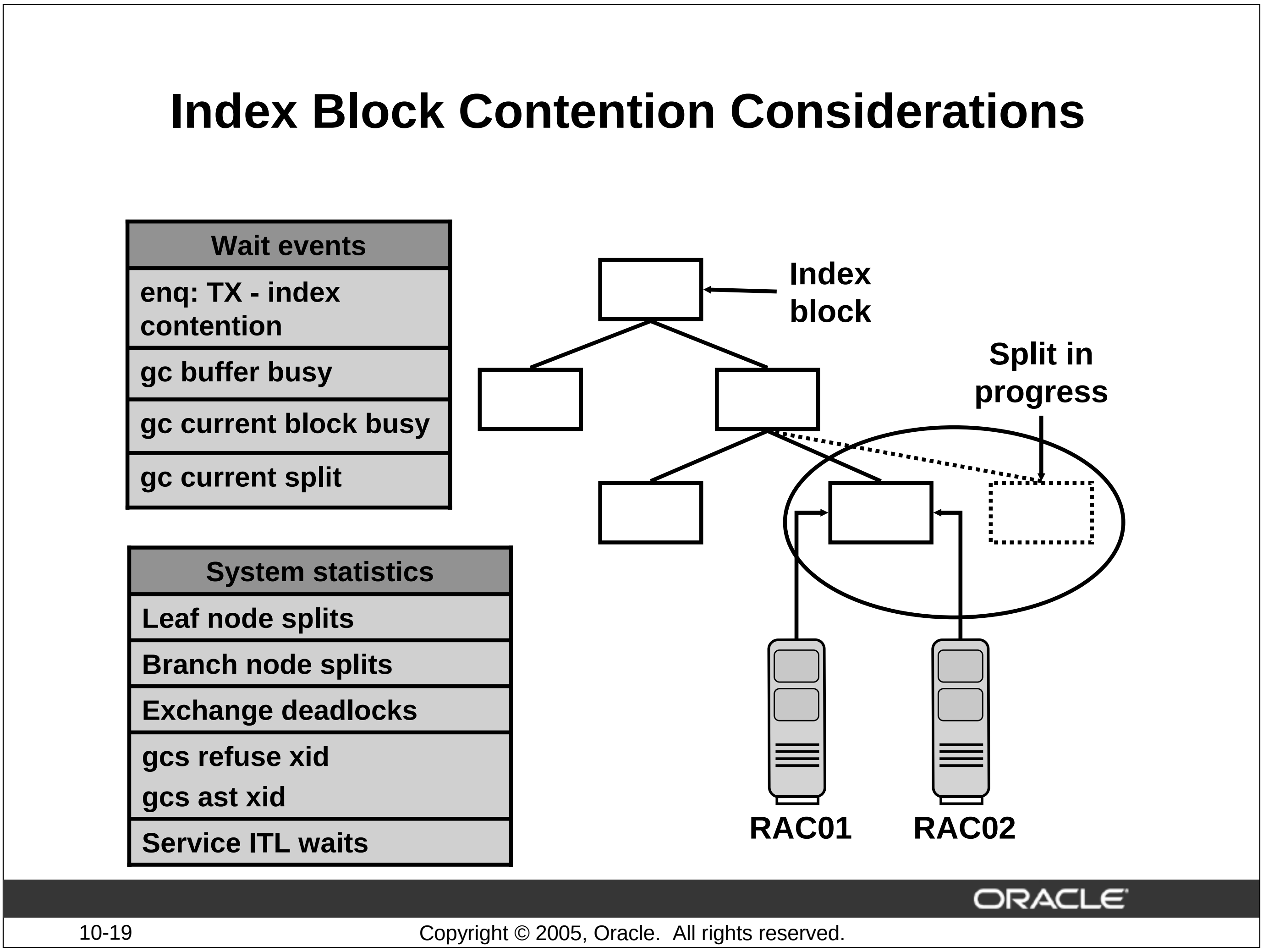
Most Common RAC Tuning Tips

In any database system, RAC or single instance, the most significant performance gains are usually obtained from traditional application tuning techniques. The benefits of those techniques are even more remarkable in a RAC database. In addition to traditional application tuning, some of the techniques that are particularly important for RAC include the following:

- Try to avoid long full-table scans to minimize GCS requests. The overhead caused by the global CR requests in this scenario is because of the fact that when queries result in local cache misses, an attempt is first made to find the data in another cache, based on the assumption that the chance that another instance has cached the block is high.
- Automatic segment space management can provide instance affinity to table blocks.
- Increasing sequence caches improves instance affinity to index keys deriving their values from sequences. That technique may result in significant performance gains for multi-instance insert intensive applications.
- Range or list partitioning may be very effective in conjunction with data-dependent routing, if the workload can be directed to modify a particular range of values from a particular instance.
- Hash partitioning may help to reduce buffer busy contention by making buffer access distribution patterns sparser, enabling more buffers to be available for concurrent access.

Most Common RAC Tuning Tips (continued)

- In RAC, library cache and row cache operations are globally coordinated. So, excessive parsing means additional interconnect traffic. Library cache locks are heavily used, in particular by applications using PL/SQL or Advanced Queuing. Library cache locks are acquired in exclusive mode whenever a package or procedure has to be recompiled.
- Because transaction locks are globally coordinated, they also deserve special attention in RAC. For example, using tables instead of Oracle sequences to generate unique numbers is not recommended because it may cause severe contention even for a single instance system.
- Indexes that are not selective do not improve query performance, but can degrade DML performance. In RAC, unselective index blocks may be subject to interinstance contention, increasing the frequency of cache transfers for indexes belonging to `INSERT` intensive tables.



Index Block Contention Considerations

Wait events
enq: TX - index contention
gc buffer busy
gc current block busy
gc current split

System statistics
Leaf node splits
Branch node splits
Exchange deadlocks
gcs refuse xid gcs ast xid
Service ITL waits

10-19

Index Block Contention Considerations

In application systems where the loading or batch processing of data is a dominant business function, there may be performance issues affecting response times because of the high volume of data inserted into indexes. Depending on the access frequency and the number of processes concurrently inserting data, indexes can become hot spots and contention can be exacerbated by:

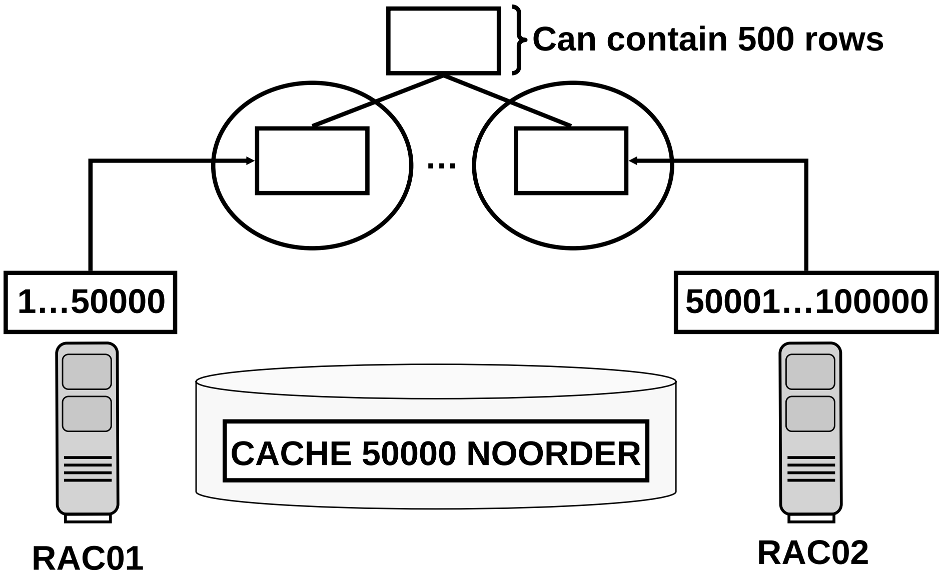
- Ordered monotonically increasing key values in the index (right-growing trees)
- Frequent leaf block splits
- Low tree depth: all leaf block access go through the root block

A leaf or branch block split can become an important serialization point if the particular leaf block or branch of the tree is concurrently accessed.

The tables in the slide sum up the most common symptoms associated with the splitting of index blocks, listing wait events and statistics that are commonly elevated when index block splits are prevalent. As a general recommendation, to alleviate the performance impact of globally hot index blocks and leaf block splits, a more uniform, less skewed distribution of the concurrency in the index tree should be the primary objective. This can be achieved by:

- Global index hash partitioning
- Increasing the sequence cache, if the key value is derived from a sequence

Oracle Sequences and Index Contention



Oracle Sequences and Index Contention

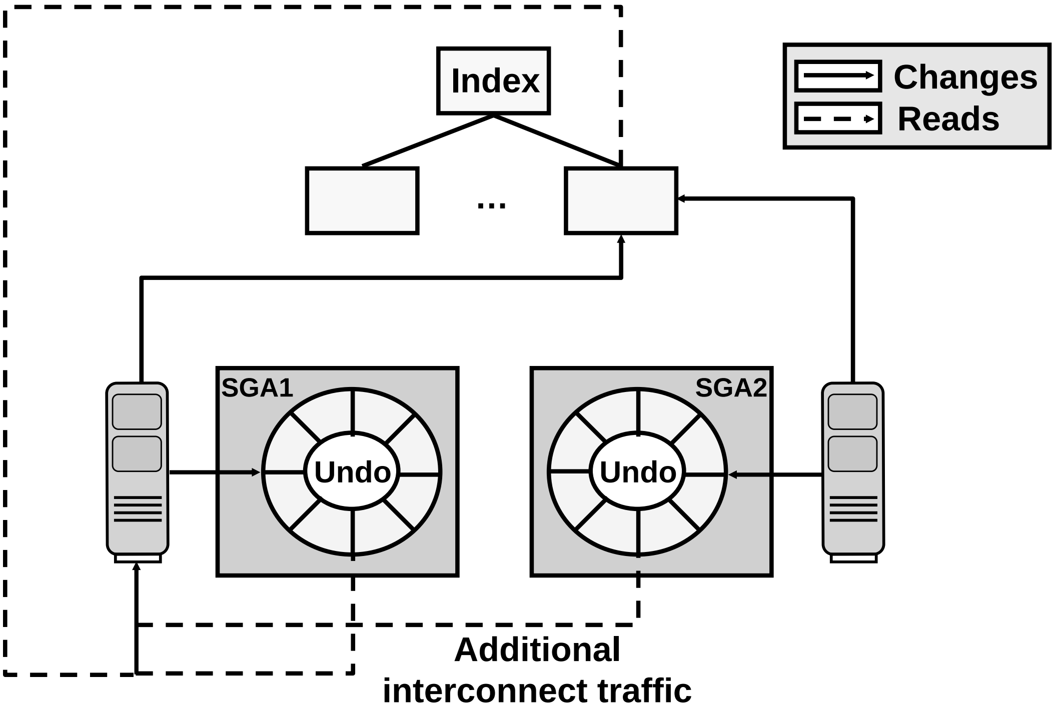
Indexes with key values generated by sequences tend to be subject to leaf block contention when the insert rate is high. That is because the index leaf block holding the highest key value is changed for every row inserted, as the values are monotonically ascending. In RAC, this may lead to a high rate of current and CR blocks transferred between nodes.

One of the simplest techniques that can be used to limit this overhead is to increase the sequence cache, if you are using Oracle sequences. As the difference between sequence values generated by different instances increases, successive index block splits tend to create instance affinity to index leaf blocks. For example, suppose that an index key value is generated by a `CACHE NOORDER` sequence and each index leaf block can hold 500 rows. If the sequence cache is set to 50000, while instance 1 inserts values 1, 2, 3, and so on, instance 2 concurrently inserts 50001, 50002, and so on. After some block splits, each instance writes to a different part of the index tree.

So, what is the ideal value for a sequence cache to avoid interinstance leaf index block contention, yet minimizing possible gaps? One of the main variables to consider is the insert rate: the higher it is, the higher must be the sequence cache. However, creating a simulation to evaluate the gains for a specific configuration is recommended.

Note: By default, the cache value is 20. Typically, 20 is too small for the example above.

Undo Block Considerations



Undo Block Considerations

Excessive undo block shipment and contention for undo buffers usually happens when index blocks containing active transactions from multiple instances are read frequently.

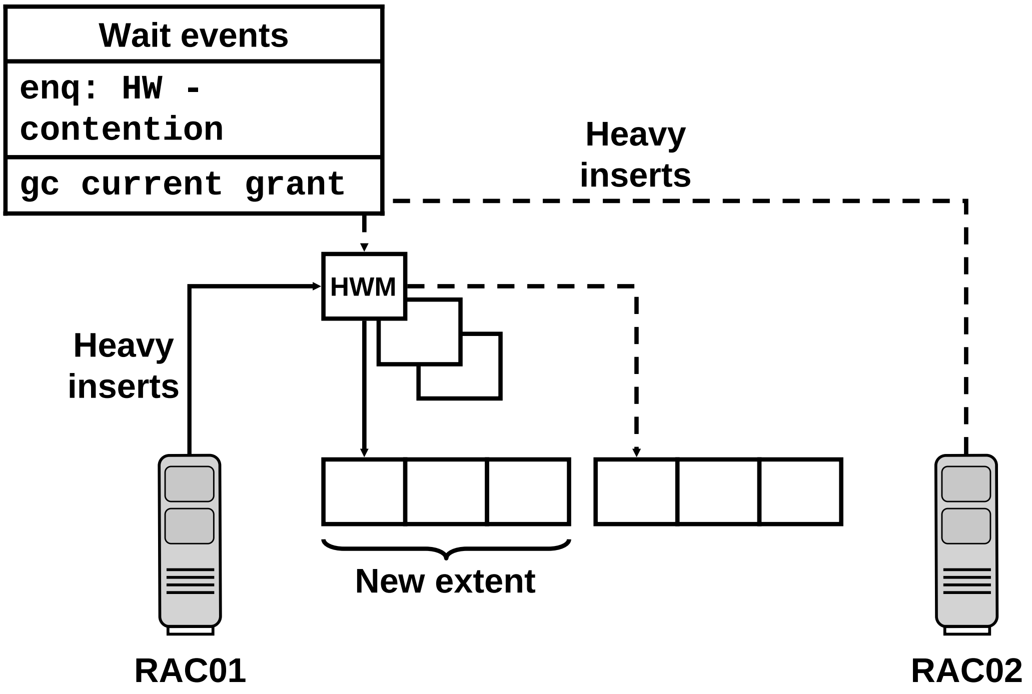
When a SELECT statement needs to read a block with active transactions, it has to undo the changes to create a CR version. If the active transactions in the block belong to more than one instance, there is a need to combine local and remote undo information for the consistent read. Depending on the amount of index blocks changed by multiple instances and the duration of the transactions, undo block shipment may become a bottleneck.

Usually this happens in applications that read recently inserted data very frequently, but commit infrequently. Techniques that can be used to reduce such situations include the following:

- Shorter transactions reduce the likelihood that any given index block in the cache contains uncommitted data, thereby reducing the need to access undo information for consistent read.
- As explained earlier, increasing sequence cache sizes can reduce interinstance concurrent access to index leaf blocks. CR versions of index blocks modified by only one instance can be fabricated without the need of remote undo information.

Note: In RAC, the problem is exacerbated by the fact that a subset of the undo information has to be obtained from remote instances.

High-Water Mark Considerations



High-Water Mark Considerations

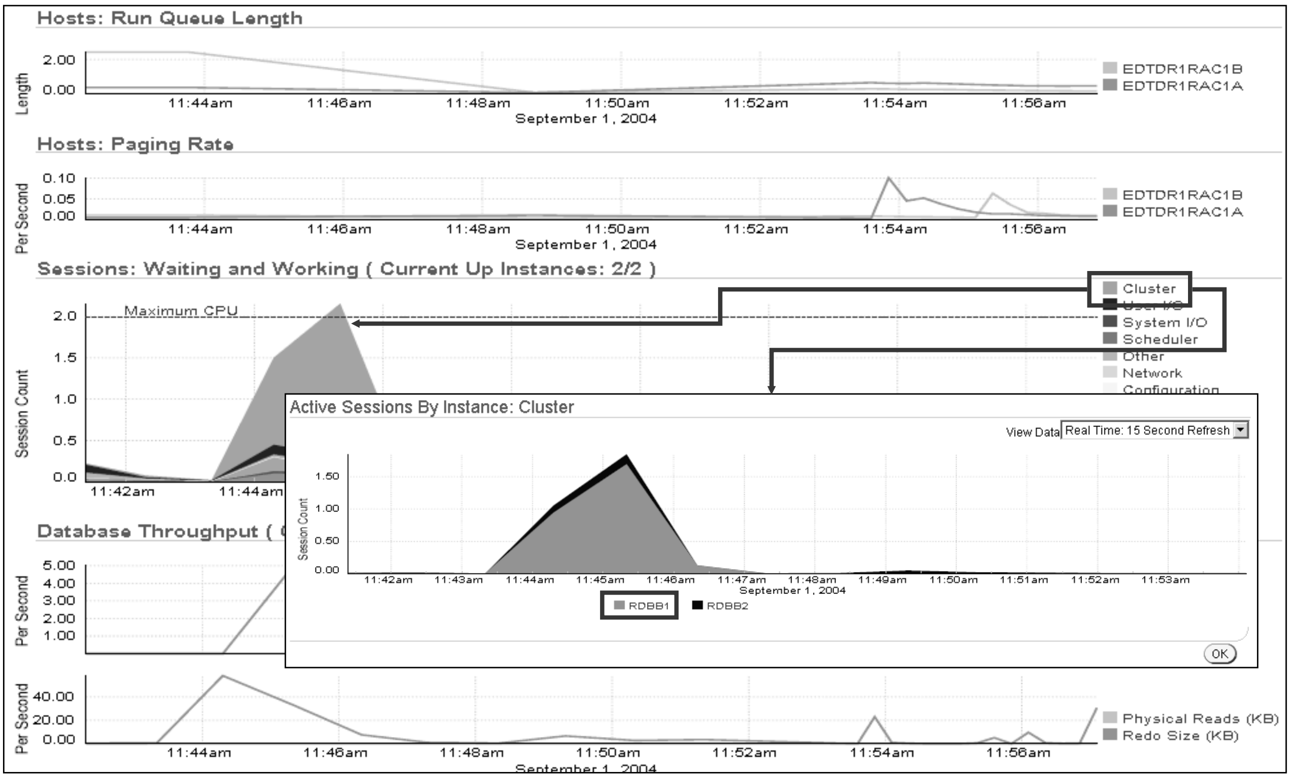
A certain combination of wait events and statistics presents itself in applications where the insertion of data is a dominant business function and new blocks have to be allocated frequently to a segment. If data is inserted at a high rate, new blocks may have to be made available after unfruitful searches for free space. This has to happen while holding the High-Water Mark (HWM) enqueue.

Therefore, the most common symptoms for this scenario include:

- A high percentage of wait time for `enq: HW – contention`
- A high percentage of wait time for `gc current grant` events

The former is a consequence of the serialization on the HWM enqueue, and the latter is because of the fact that current access to the new data blocks is required for the new block operation. In a RAC environment, the length of this space management operation is proportional to the time it takes to acquire the HW enqueue and the time it takes to acquire global locks for all the new blocks. This time is small under normal circumstances because there is never any access conflict for the new blocks. Therefore, this scenario may be observed in applications with business functions requiring a lot of data loading, and the main recommendation to alleviate the symptoms is to define uniform and large extent sizes for the locally managed and automatic space managed segments that are subject to high volume inserts.

Cluster Database Performance Page



Cluster Performance Page

To access the **Cluster Performance** page, click the **Performance** tab on the **Cluster Home** page. This page displays an overview of CPU, memory, and disk I/O utilization for each node in the cluster. You can choose an automatic refresh interval of 15 seconds or a manual refresh interval. At the bottom of the page, each node name is presented as a link, which you can click to access more detailed performance information about that node.

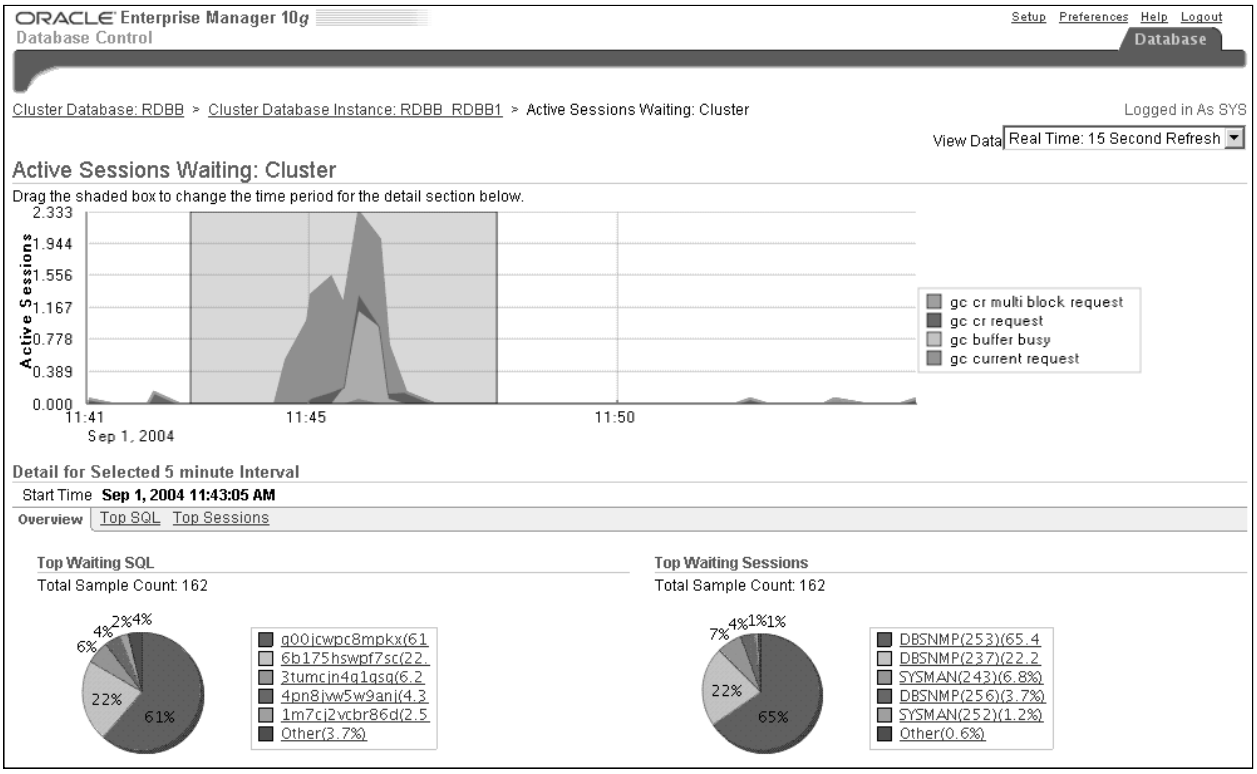
On this page, you can see performance charts for the entire cluster database. The charts displayed on this page are **Run Queue Length**, **Paging Rate**, **Sessions**, and **Database Throughput**. The **Run Queue Length** and **Paging Rate** graphics are host-related charts, and they break down the data by each node in the cluster. The **Sessions** and **Database Throughput** charts consolidate data across all instances that are currently open.

The slide illustrates a possible drill down from the **Sessions** graphic. If the cluster impact is high, you can click the **Cluster** wait class link that you can click to drill down to the **Active Sessions by Instance** graphic.

On the **Active Sessions by Instance** graphic, you can see for each node its corresponding session count over a period of time.

Note: The **Cluster Performance** page also contains a link to the **Cluster Cache Coherency** page.

Cluster Database Performance Page



ORACLE Enterprise Manager 10*g*
Database Control

Setup Preferences Help Logout

Database

Cluster Database: RDBB > Cluster Database Instance: RDBB_RDBB1 > Active Sessions Waiting: Cluster

Logged in As SYS

View Data Real Time: 15 Second Refresh

Active Sessions Waiting: Cluster

Drag the shaded box to change the time period for the detail section below.

Detail for Selected 5 minute Interval

Start Time **Sep 1, 2004 11:43:05 AM**

Overview Top SQL Top Sessions

Top Waiting SQL

Total Sample Count: 162

- g00jcwpc8mpkx(61
- 6b175hswpf7sc(22.
- 3tumcjn4q1qsq(6.2
- 4pn8jvw5w9anj(4.3
- 1m7cj2vcbr86d(2.5
- Other(3.7%)

Top Waiting Sessions

Total Sample Count: 162

- DBSNMP(253)(65.4
- DBSNMP(237)(22.2
- SYSMAN(243)(6.8%)
- DBSNMP(256)(3.7%)
- SYSMAN(252)(1.2%)
- Other(0.6%)

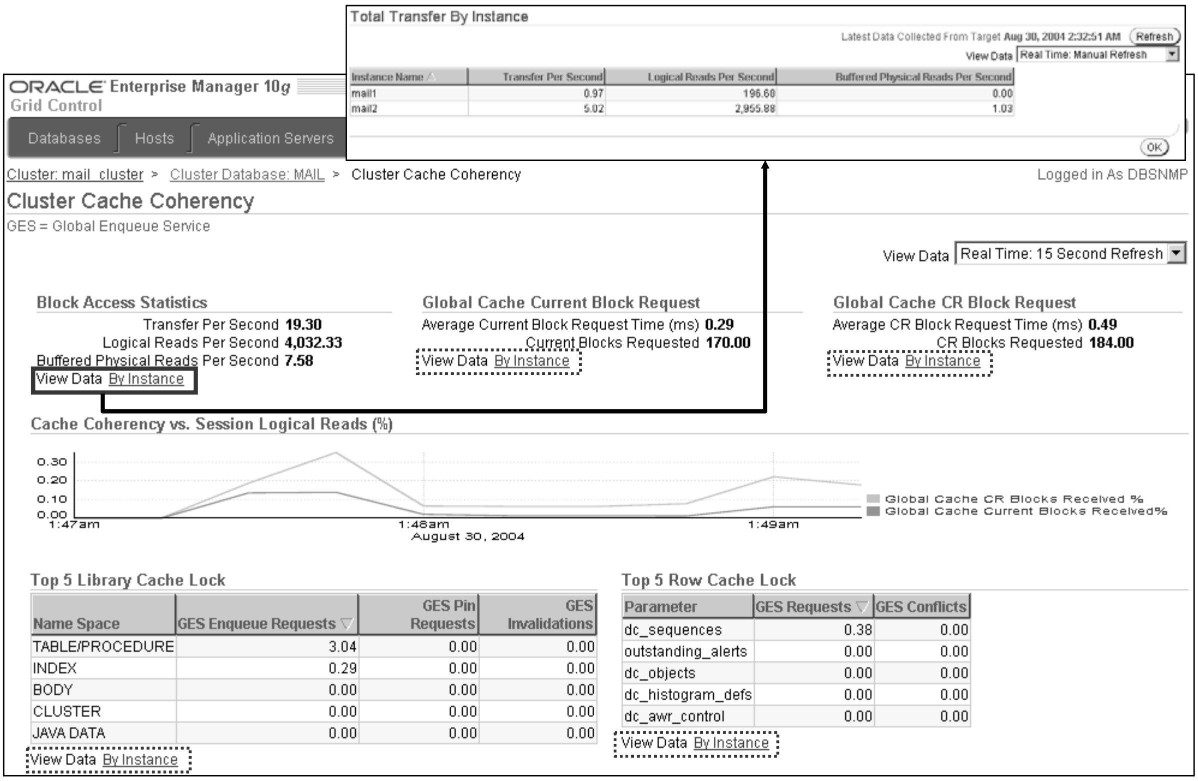
Cluster Performance Page (continued)

On the **Active Sessions by Instance** graphic page, you can click one of the node links to drill down to the **Active Sessions Waiting** graphic presented in the slide.

The **Active Sessions Waiting: Cluster** page shows you the top SQL statements and the top sessions consuming significant resources from the **Cluster** wait class.

You can further drill down to a specific SQL statement or session that has the highest impact on that particular instance.

Cluster Cache Coherency Page



Total Transfer By Instance

Latest Data Collected From Target **Aug 30, 2004 2:32:51 AM** (Refresh)

View Data: Real Time: Manual Refresh

Instance Name △	Transfer Per Second	Logical Reads Per Second	Buffered Physical Reads Per Second
mail1	0.97	196.60	0.00
mail2	5.02	2,955.88	1.03

(OK)

Cluster: mail_cluster > Cluster Database: MAIL > Cluster Cache Coherency

Logged in As DBSNMP

Cluster Cache Coherency

GES = Global Enqueue Service

View Data: Real Time: 15 Second Refresh

Block Access Statistics

Transfer Per Second **19.30**
Logical Reads Per Second **4,032.33**
Buffered Physical Reads Per Second **7.58**
View Data By Instance

Global Cache Current Block Request

Average Current Block Request Time (ms) **0.29**
Current Blocks Requested **170.00**
View Data By Instance

Global Cache CR Block Request

Average CR Block Request Time (ms) **0.49**
CR Blocks Requested **184.00**
View Data By Instance

Cache Coherency vs. Session Logical Reads (%)

0.30
0.20
0.10
0.00
1:47am
1:48am
August 30, 2004
1:49am
Global Cache CR Blocks Received %
Global Cache Current Blocks Received%

Top 5 Library Cache Lock

Name Space	GES Enqueue Requests ▽	GES Pin Requests	GES Invalidations
TABLE/PROCEDURE	3.04	0.00	0.00
INDEX	0.29	0.00	0.00
BODY	0.00	0.00	0.00
CLUSTER	0.00	0.00	0.00
JAVA DATA	0.00	0.00	0.00

View Data By Instance

Top 5 Row Cache Lock

Parameter	GES Requests ▽	GES Conflicts
dc_sequences	0.38	0.00
outstanding_alerts	0.00	0.00
dc_objects	0.00	0.00
dc_histogram_defs	0.00	0.00
dc_awr_control	0.00	0.00

View Data By Instance

Cluster Cache Coherency

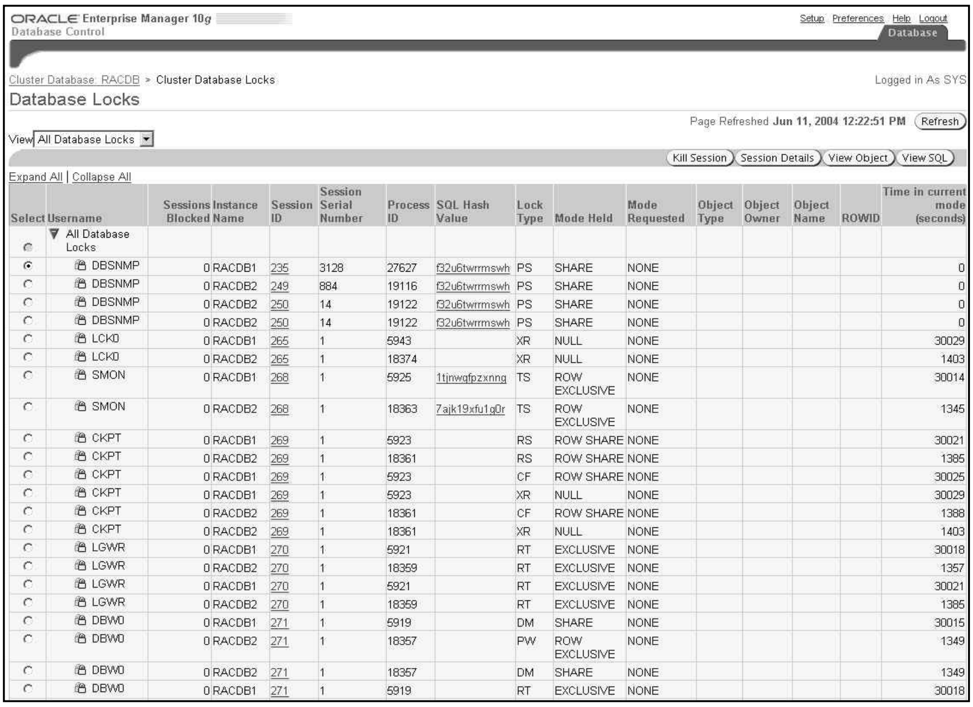
Cluster cache coherency metrics help you to identify processing trends and optimize performance for your RAC environment. The **Cluster Cache Coherency** page enables you to view cache coherency metrics for the entire cluster database, grouped into the following categories:

- Block Access Statistics
- Global Cache Current Block Request
- Global Cache CR Block Request
- Top 5 Library Cache Lock
- Top 5 Row Cache Lock

For each of these categories, click **By Instance** to view a table of the metrics in that group for each instance of the cluster database. This page also displays the **Cache Coherency vs. Session Logical Reads** chart, which graphs the percentage of Global Cache CR Blocks Received, Current Blocks Received, against logical reads for the session.

If your request latency value is too high, the `DB_FILE_MULTIBLOCK_READ_COUNT` value may be too high. This is because a requesting process can issue more than one request for a block depending on the setting of this parameter, causing the requesting process to wait longer. High request latency may also be caused by a large number of incoming requests to the LMS process for Global Cache Services.

Database Locks Page



ORACLE Enterprise Manager 10g
Database Control
Setup Preferences Help Logout
Database

Cluster Database: RACDB > Cluster Database Locks

Logged in As SYS

Database Locks

Page Refreshed **Jun 11, 2004 12:22:51 PM** (Refresh)

View: All Database Locks

Kill Session
Session Details
View Object
View SQL
Expand All | Collapse All

Select	Username	Sessions Blocked	Instance Name	Session ID	Session Serial Number	Process ID	SQL Hash Value	Lock Type	Mode Held	Mode Requested	Object Type	Object Owner	Object Name	ROWID	Time in current mode (seconds)
○	▼ All Database Locks														
●	DBSNMP	0	RACDB1	235	3128	27627	f32u6twrrmswh	PS	SHARE	NONE					0
○	DBSNMP	0	RACDB2	249	884	19116	f32u6twrrmswh	PS	SHARE	NONE					0
○	DBSNMP	0	RACDB2	250	14	19122	f32u6twrrmswh	PS	SHARE	NONE					0
○	DBSNMP	0	RACDB2	250	14	19122	f32u6twrrmswh	PS	SHARE	NONE					0
○	LCK0	0	RACDB1	265	1	5943		XR	NULL	NONE					30029
○	LCK0	0	RACDB2	265	1	18374		XR	NULL	NONE					1403
○	SMON	0	RACDB1	268	1	5925	1tjnwqfpzxnng	TS	ROW EXCLUSIVE	NONE					30014
○	SMON	0	RACDB2	268	1	18363	7ajk19xfu1g0r	TS	ROW EXCLUSIVE	NONE					1345
○	CKPT	0	RACDB1	269	1	5923		RS	ROW SHARE	NONE					30021
○	CKPT	0	RACDB2	269	1	18361		RS	ROW SHARE	NONE					1385
○	CKPT	0	RACDB1	269	1	5923		CF	ROW SHARE	NONE					30025
○	CKPT	0	RACDB1	269	1	5923		XR	NULL	NONE					30029
○	CKPT	0	RACDB2	269	1	18361		CF	ROW SHARE	NONE					1388
○	CKPT	0	RACDB2	269	1	18361		XR	NULL	NONE					1403
○	LGWR	0	RACDB1	270	1	5921		RT	EXCLUSIVE	NONE					30018
○	LGWR	0	RACDB2	270	1	18359		RT	EXCLUSIVE	NONE					1357
○	LGWR	0	RACDB1	270	1	5921		RT	EXCLUSIVE	NONE					30021
○	LGWR	0	RACDB2	270	1	18359		RT	EXCLUSIVE	NONE					1385
○	DBW0	0	RACDB1	271	1	5919		DM	SHARE	NONE					30015
○	DBW0	0	RACDB2	271	1	18357		PW	ROW EXCLUSIVE	NONE					1349
○	DBW0	0	RACDB2	271	1	18357		DM	SHARE	NONE					1349
○	DBW0	0	RACDB1	271	1	5919		RT	EXCLUSIVE	NONE					30018

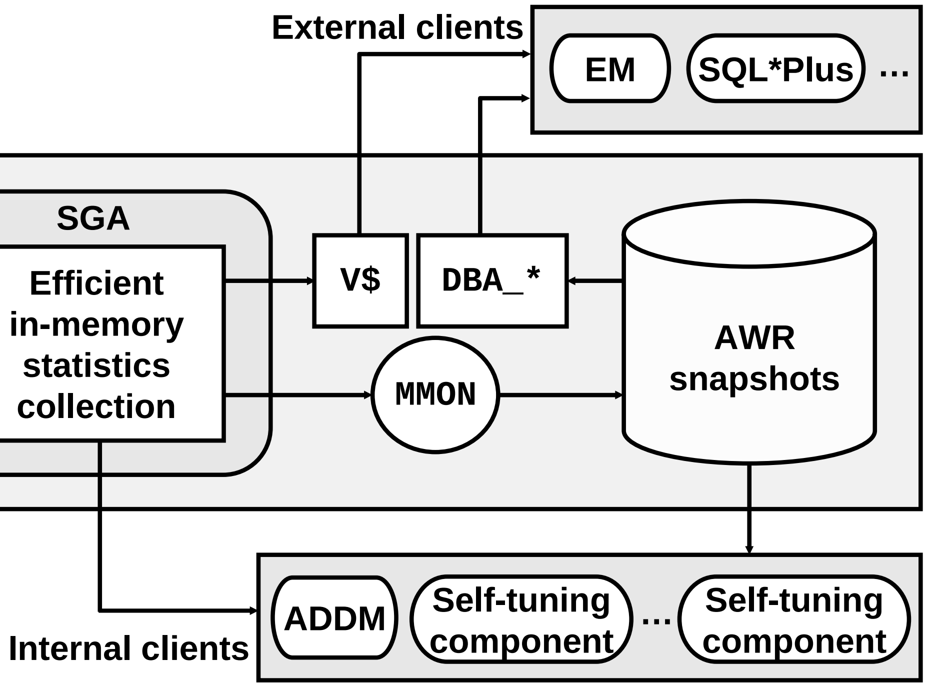
Database Locks Page

Locks are mechanisms that prevent destructive interaction between transactions accessing the same resource, either user objects such as tables and rows or system objects not visible to users, such as shared data structures in memory and data dictionary rows. Locks are also used to control concurrent access to data. Locks guarantee data integrity while enabling maximum concurrent access to data by unlimited users.

Use the **Database Locks** page to view a table showing User Locks, Blocking Locks, or the complete list of all database locks. You can access the **Database Locks** page from the Monitoring section of the **Database Performance Page** by clicking **Database Locks**.

The **Database Locks** page displays the table of locks listed by User Locks, Blocking Locks, or All Database Locks. Click the expand or collapse icon beside the lock type to view the type of locks that you want to display, or you can use the **Expand All** or **Collapse All** links to expand or collapse the entire list of locks. You can filter the type of locks that appear on the page by choosing the lock type from the drop-down list at the top of the **Database Locks** page. To view details about a specific session, click the **Select** field for that row, and click **Session Details**. To terminate a session, click the **Select** field for that session, and then click **Kill Session**. To view objects related to a session, click **View Objects**. For cluster databases, the Database Locks table displays the Instance Name column and shows databasewide locks.

Automatic Workload Repository: Overview



Automatic Workload Repository: Overview

AWR is the infrastructure that provides services to Oracle Database 10*g* components to collect, maintain, and utilize statistics for problem detection and self-tuning purposes.

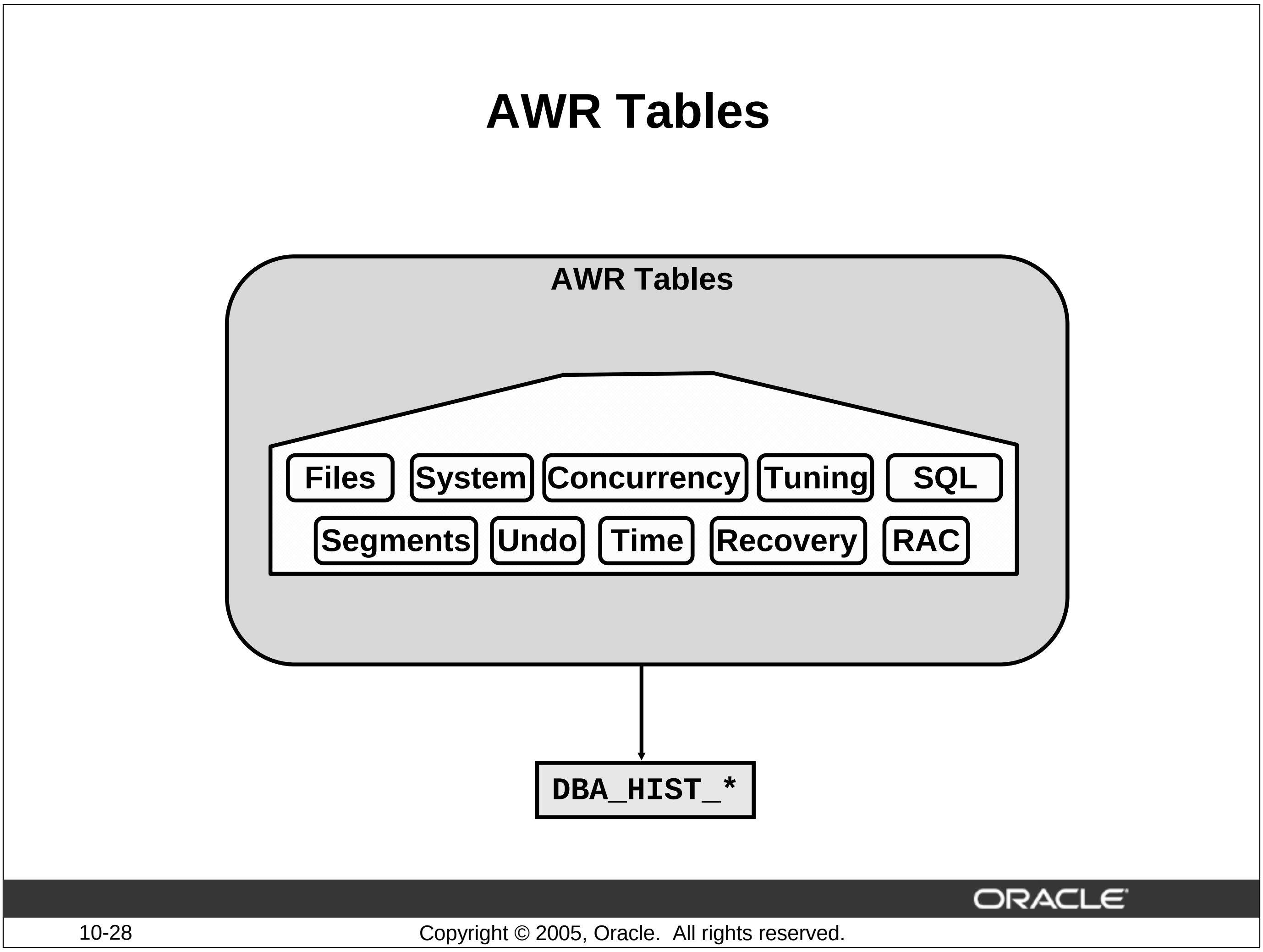
The AWR infrastructure consists of two major parts:

- An in-memory statistics collection facility that is used by Oracle Database 10*g* components to collect statistics. These statistics are stored in memory for performance reasons. Statistics stored in memory are accessible through dynamic performance (V$) views.
- AWR snapshots represent the persistent portion of the facility. They are accessible through data dictionary views and Database Control.

Statistics are stored in persistent storage for several reasons:

- The statistics need to survive instance crashes.
- Some analyses need historical data for baseline comparisons.
- Memory overflow: When old statistics are replaced by new ones because of memory shortage, the replaced data can be stored for later use.

The memory version of the statistics is transferred to the disk on a regular basis by a new background process called Manageability Monitor (MMON). With AWR, the Oracle database provides a way to capture historical statistics data automatically, without the intervention of DBAs.



AWR Tables

AWR Tables

Files

System

Concurrency

Tuning

SQL

Segments

Undo

Time

Recovery

RAC

DBA_HIST_*

10-28

AWR Tables

AWR contains two types of tables:

- Metadata tables: Are used to control, process, and describe AWR tables. For example, the Oracle database uses metadata tables to determine when to perform snapshots, and what to capture to the disk. Also, metadata tables contain the mapping between the `snapshot_id` and the corresponding wall clock time.
- Historical statistic tables: Store historical statistical information about the Oracle database. Each snapshot is a capture of the in-memory database statistics data at a certain point in time.

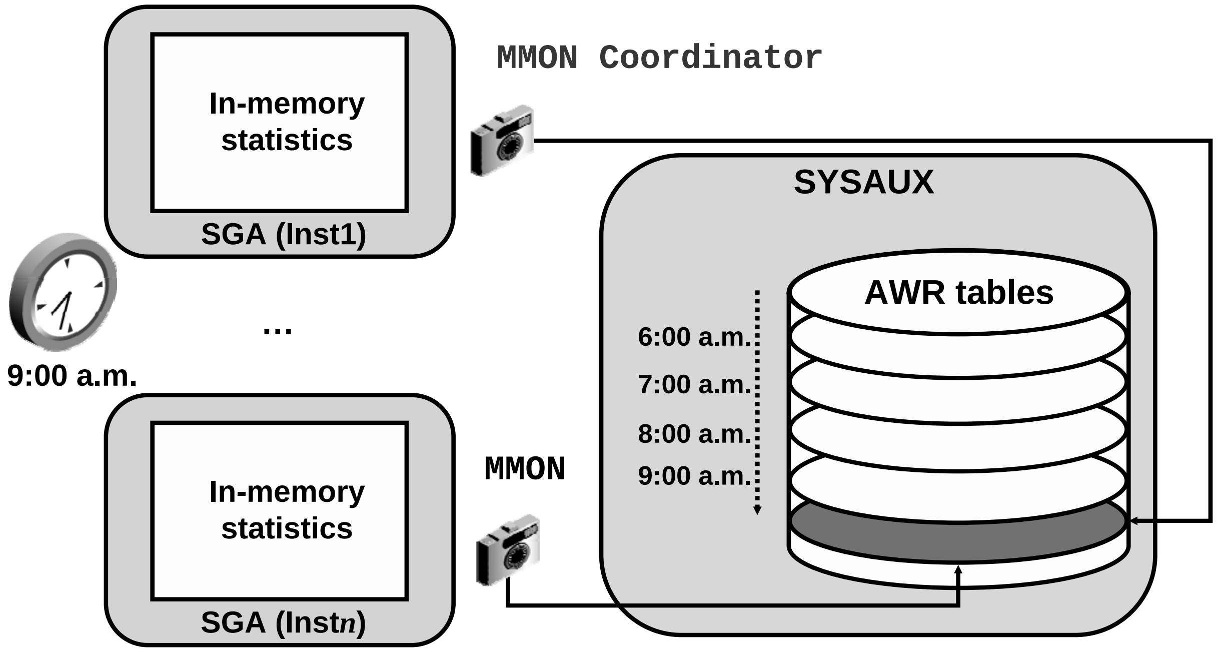
All names of AWR tables are prefixed with `WRx$`, where `x` specifies the kind of table:

- `WRM$` tables store metadata information for AWR.
- `WRH$` tables store historical data or snapshots.

You can use dictionary views to query AWR data. Any view related to historical information in AWR has the `DBA_HIST_`suffix.

AWR uses partitioning for efficient querying and purging of data. The snapshot tables are organized into the following categories: File Statistics, General System Statistics, Concurrency Statistics, Instance Tuning Statistics, SQL Statistics, Segment Statistics, Undo Statistics, Time-Model Statistics, Recovery Statistics, and RAC Statistics.

AWR Snapshots in RAC



MMON Coordinator

In-memory statistics

SGA (Inst1)

...

9:00 a.m.

In-memory statistics

SGA (Inst*n*)

MMON

SYSAUX

AWR tables

6:00 a.m.

7:00 a.m.

8:00 a.m.

9:00 a.m.

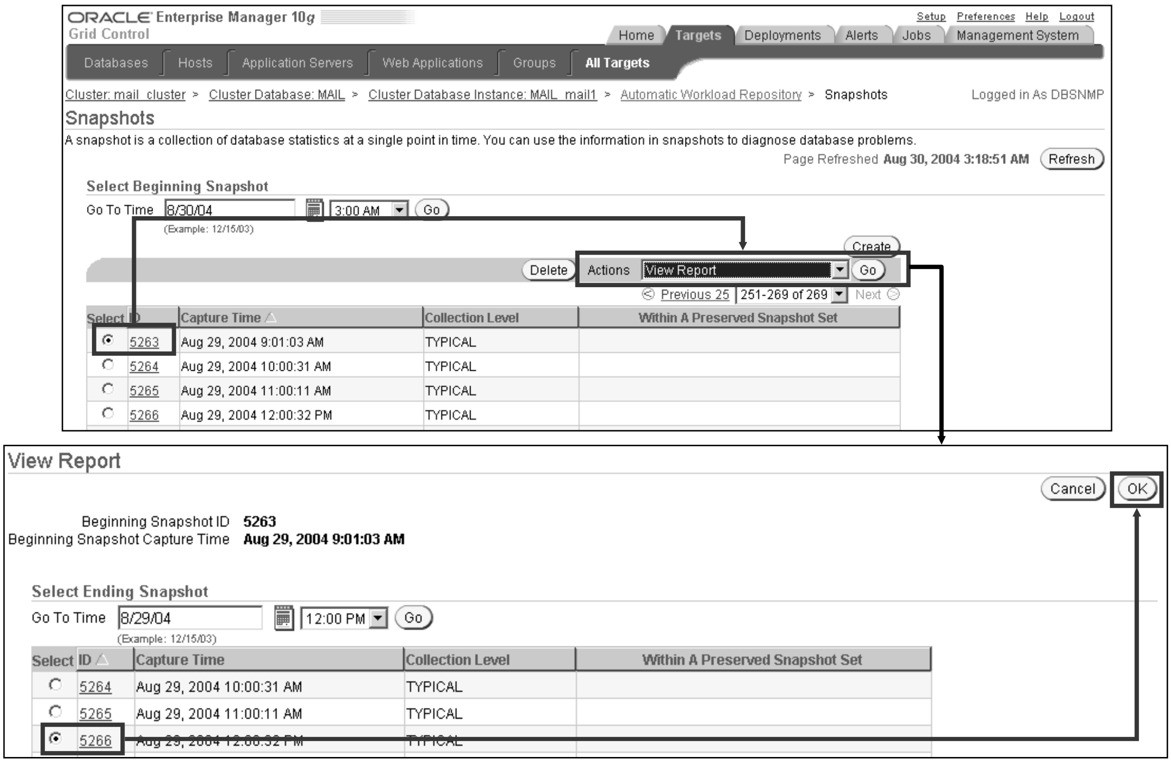
AWR Snapshots in RAC

In RAC environments, each AWR snapshot captures data from all active instances within the cluster. The data for each snapshot set that is captured for all active instances is from roughly the same point in time. In addition, the data for each instance is stored separately and is identified with an instance identifier. For example, the `buffer_busy_wait` statistic shows the number of buffer waits on each instance. AWR does not store data that is aggregated from across the entire cluster. In other words, the data is stored for each individual instance.

The statistics snapshots generated by AWR can be evaluated by producing reports displaying summary data such as load and cluster profiles based on regular statistics and wait events gathered on each instance.

AWR functions in a similar way as Statspack. The difference is that AWR automatically collects and maintains performance statistics for problem detection and self-tuning purposes. Unlike in Statspack, in AWR there is only one `snapshot_id` per snapshot across instances.

Generating and Viewing AWR Reports



Cluster: mail_cluster > Cluster Database: MAIL > Cluster Database Instance: MAIL_mail1 > Automatic Workload Repository > Snapshots

Logged in As DBSNMP

Snapshots

A snapshot is a collection of database statistics at a single point in time. You can use the information in snapshots to diagnose database problems.

Page Refreshed Aug 30, 2004 3:18:51 AM

Select Beginning Snapshot

Go To Time 8/30/04 3:00 AM
(Example: 12/15/03)

Actions View Report

251-269 of 269

Select	ID	Capture Time	Collection Level	Within A Preserved Snapshot Set
●	5263	Aug 29, 2004 9:01:03 AM	TYPICAL	
○	5264	Aug 29, 2004 10:00:31 AM	TYPICAL	
○	5265	Aug 29, 2004 11:00:11 AM	TYPICAL	
○	5266	Aug 29, 2004 12:00:32 PM	TYPICAL	

View Report

Beginning Snapshot ID 5263
Beginning Snapshot Capture Time Aug 29, 2004 9:01:03 AM

Select Ending Snapshot

Go To Time 8/29/04 12:00 PM
(Example: 12/15/03)

Select	ID	Capture Time	Collection Level	Within A Preserved Snapshot Set
○	5264	Aug 29, 2004 10:00:31 AM	TYPICAL	
○	5265	Aug 29, 2004 11:00:11 AM	TYPICAL	
●	5266	Aug 29, 2004 12:00:32 PM	TYPICAL	

Generating and Viewing AWR Reports

Although an AWR snapshot contains data for all instances of your RAC environment, an AWR report is only relevant to one particular instance. So, to generate an AWR report when using Database Control, you must go to the **Administration** tab of the corresponding instance's home page. From there, click the **Automatic Workload Repository** link in the **Workload** section. When on the **Automatic Workload Repository** page, click the link corresponding to the **Snapshots** field. This takes you to the **Snapshots** page shown in the slide.

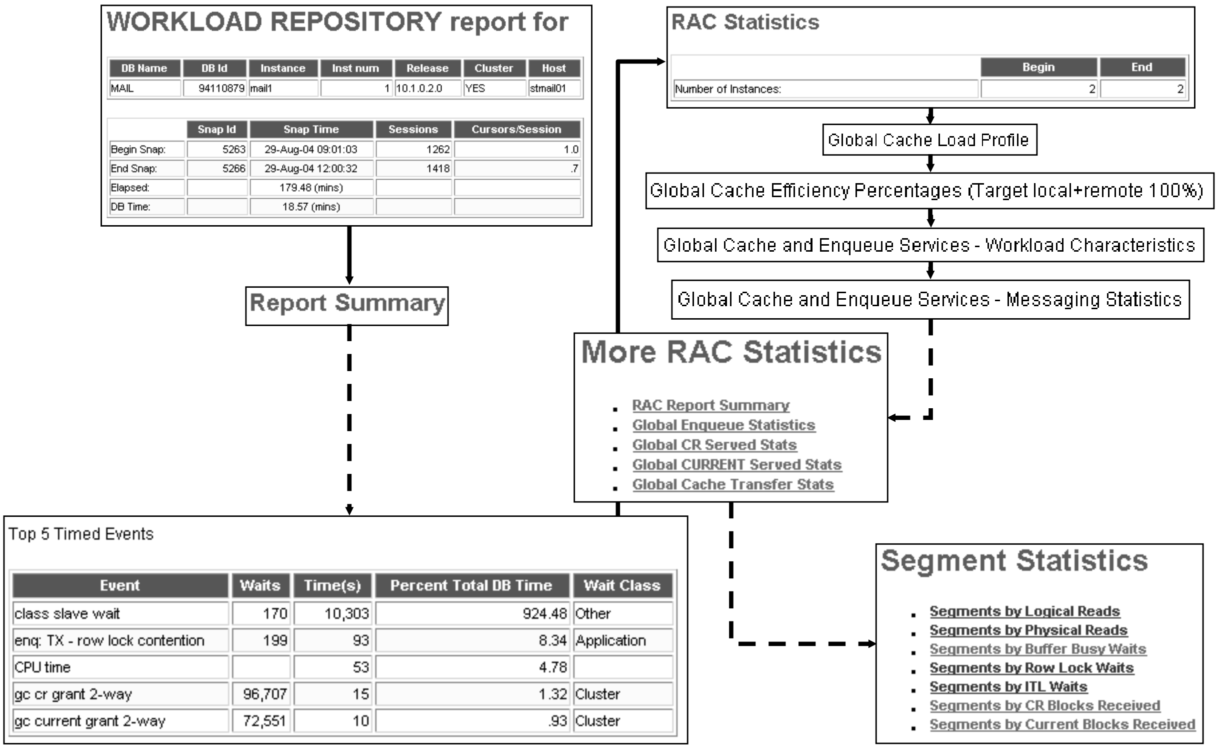
Select a beginning snapshot identifier by using the **Select** column of the **Select Beginning Snapshot** table, and select **View Report** in the **Actions** list. Then, click the **Go** button next to the **Actions** field.

When on the **View Report** page, select the ending snapshot identifier by using the **Select** column of the **Select Ending Snapshot** table. When done, click **OK**.

This takes you to the **Snapshot Details** page, which contains the corresponding AWR report.

Note: The report generation interface is also provided in the form of an `awrrpt.sql` SQL*Plus script. The script generates either an HTML report or a text file. The script should be run when having the `SELECT ANY DICTIONARY` privilege. This script can be found in the corresponding `$ORACLE_HOME/rdbms/admin` directory.

AWR Reports and RAC: Overview



WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Release	Cluster	Host
MAIL	94110879	mail1	1	10.1.0.2.0	YES	stmail01

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	5263	29-Aug-04 09:01:03	1262	1.0
End Snap:	5266	29-Aug-04 12:00:32	1418	.7
Elapsed:		179.48 (mins)		
DB Time:		18.57 (mins)		

Report Summary

Top 5 Timed Events

Event	Waits	Time(s)	Percent Total DB Time	Wait Class
class slave wait	170	10,303	924.48	Other
enq: TX - row lock contention	199	93	8.34	Application
CPU time		53	4.78	
gc cr grant 2-way	96,707	15	1.32	Cluster
gc current grant 2-way	72,551	10	.93	Cluster

RAC Statistics

	Begin	End
Number of Instances:	2	2

Global Cache Load Profile

Global Cache Efficiency Percentages (Target local+remote 100%)

Global Cache and Enqueue Services - Workload Characteristics

Global Cache and Enqueue Services - Messaging Statistics

More RAC Statistics

- RAC Report Summary
- Global Enqueue Statistics
- Global CR Served Stats
- Global CURRENT Served Stats
- Global Cache Transfer Stats

Segment Statistics

- Segments by Logical Reads
- Segments by Physical Reads
- Segments by Buffer Busy Waits
- Segments by Row Lock Waits
- Segments by ITL Waits
- Segments by CR Blocks Received
- Segments by Current Blocks Received

AWR Reports and RAC

The RAC-related statistics in an AWR report are organized in different sections. A RAC statistics section appears after the Top 5 Timed Events. This section contains:

- The number of instances open at the time of the begin snapshot and the end snapshot to indicate whether instances joined or left between the two snapshots
- The Global Cache Load Profile, which essentially lists the number of blocks and messages that are sent and received, as well as the number of fusion writes
- The Global Cache Efficiency Percentages, which indicate the percentage of buffer gets broken up into buffers received from the disk, local cache, and remote caches. Ideally, the percentage of disk buffer access should be close to zero.
- GCS and GES Workload Characteristics, which gives you an overview of the more important numbers first. Because the global enqueue convert statistics have been consolidated with the global enqueue get statistics, the report only prints the average global enqueue get time. The round-trip times for CR and current block transfers follow, as well as the individual sender-side statistics for CR and current blocks. The average log flush times are computed by dividing the total log flush time by the number of actual log flushes. Also, the report prints the percentage of blocks served that actually incurred a log flush.

AWR Reports and RAC (continued)

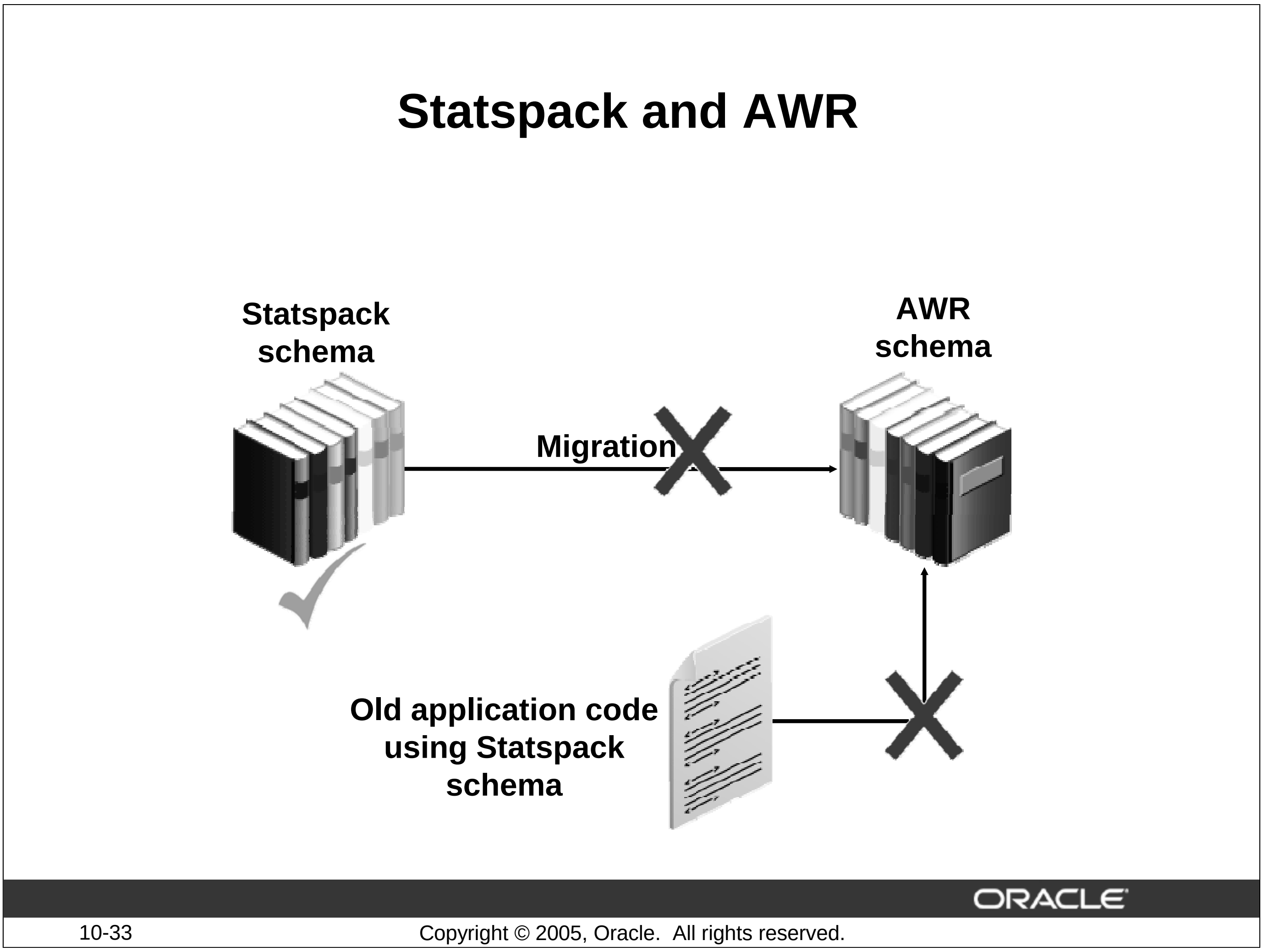
- GCS and GES Messaging Statistics. The most important statistic here is the *average message sent queue time on ksxp*, which gives a good indicator of how well the IPC works. Average numbers should be less than 1 ms.

Additional RAC statistics are then organized in the following sections:

- The Global Enqueue Statistics section contains data extracted from `V$GES_STATISTICS`.
- The Global CR Served Stats section contains data from `V$CR_BLOCK_SERVER`.
- The Global CURRENT Served Stats section contains data from `V$CURRENT_BLOCK_SERVER`.
- The Global Cache Transfer Stats section contains data from `V$INSTANCE_CACHE_TRANSFER`.

The Segment statistics section also includes the GC Buffer Busy Waits, CR Blocks Received, and CUR Blocks Received information for relevant segments.

Note: For more information about wait events and statistics, refer to the *Oracle Database Reference* guide.



Statspack and AWR

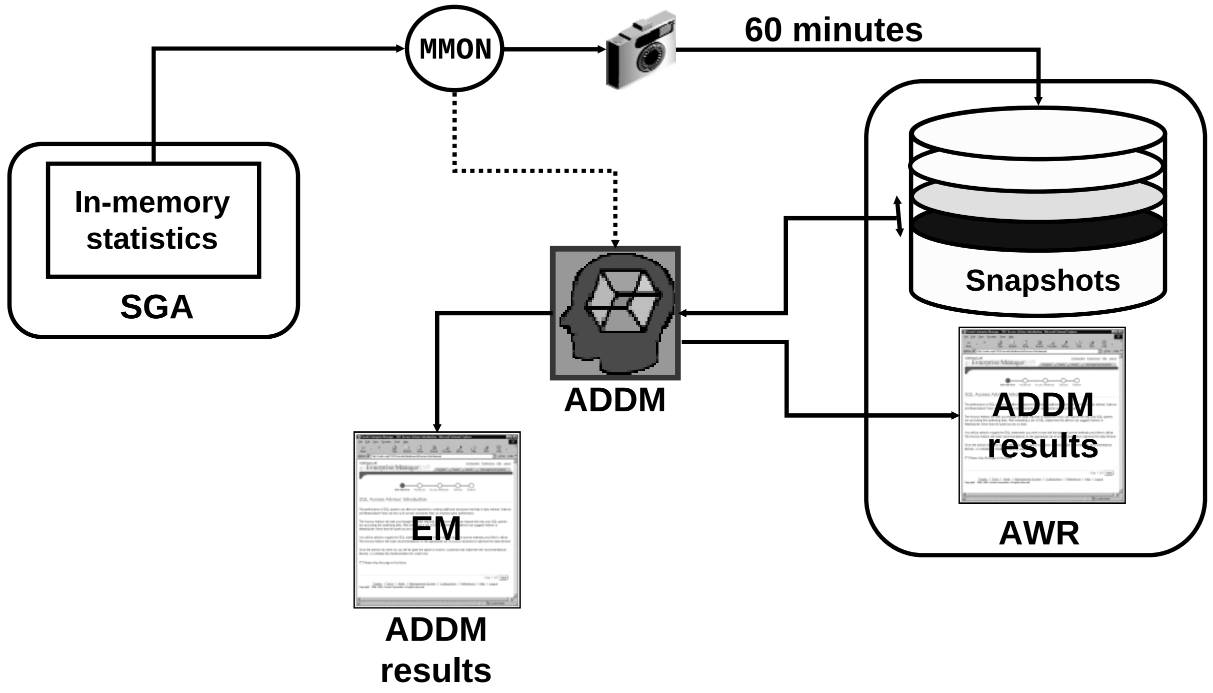
Statspack and AWR

In the past, historical data could be obtained manually by using Statspack. You can continue to use Statspack in Oracle Database 10*g*. But if you want to use the workload repository instead, you have to change your application code.

Statspack users should switch to the workload repository in Oracle Database 10*g*.

There is no supported path to migrate Statspack data into the workload repository. Also, there is no view created on top of the workload repository to simulate the Statspack schema.

Automatic Database Diagnostic Monitor



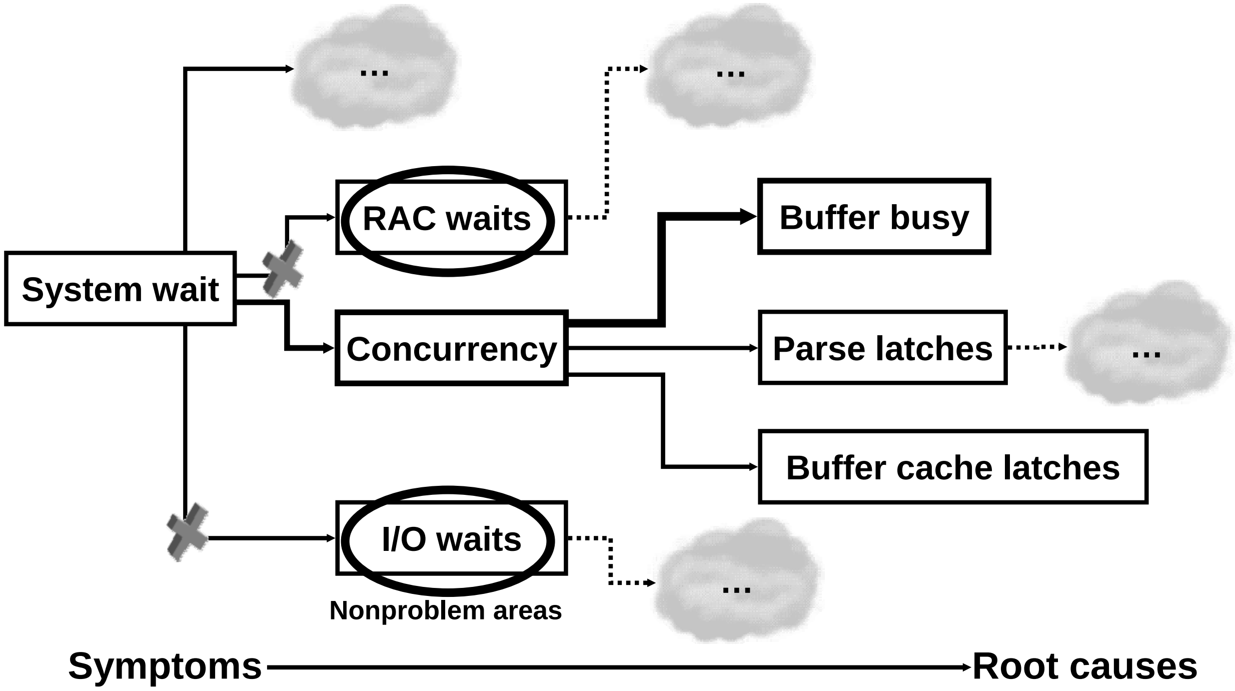
Automatic Database Diagnostic Monitor

By default, the database automatically captures statistical information every 60 minutes from the SGA and stores it inside AWR in the form of snapshots. These snapshots are stored on the disk and are similar to Statspack snapshots. However, they contain more precise information than Statspack snapshots.

Additionally, the ADDM is scheduled to run automatically by the new MMON process on every database instance to detect problems proactively. Each time a snapshot is taken, ADDM is triggered to perform an analysis of the period corresponding to the last two snapshots. Such a capability proactively monitors the instance and detects bottlenecks before they become a significant problem.

The results of each ADDM analysis are stored inside AWR (`WRI$` tables) and are accessible also through the Enterprise Manager console.

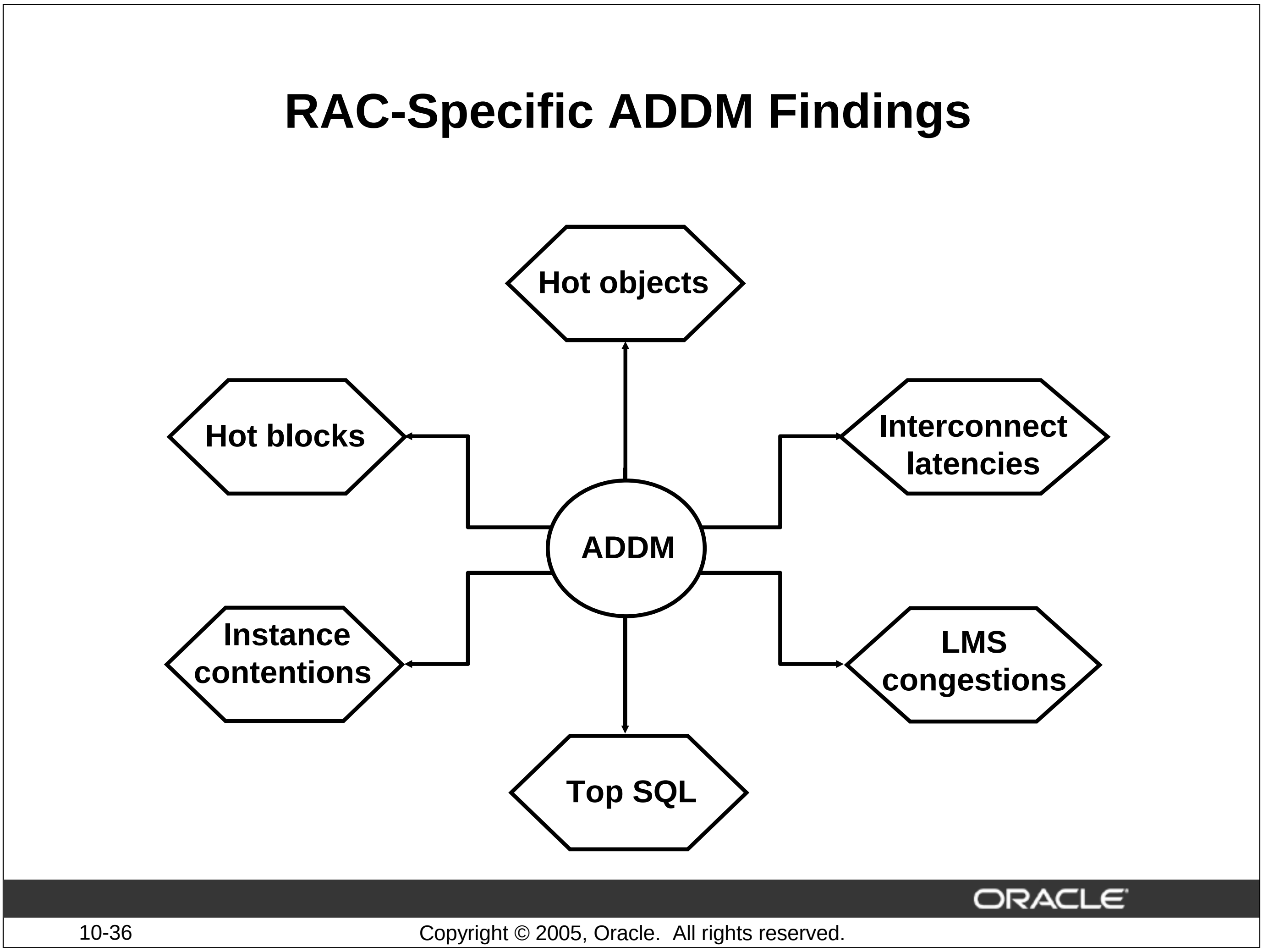
ADDM Problem Classification



ADDM Problem Classification

Internally, the ADDM uses a tree structure to represent all possible tuning issues. The tree is based on the new wait and time statistics model that is used by the Oracle database. The root of this tree represents the symptoms. Going down to the leaves, the ADDM identifies root causes. The ADDM walks down the tree by using time-based thresholds for each node.

If the time-based threshold is not exceeded for a particular node, the ADDM prunes the corresponding subtree. This enables the ADDM to identify nonproblem areas. This tree structure enables the ADDM to efficiently prune the search space to quickly identify the problems.



RAC-Specific ADDM Findings

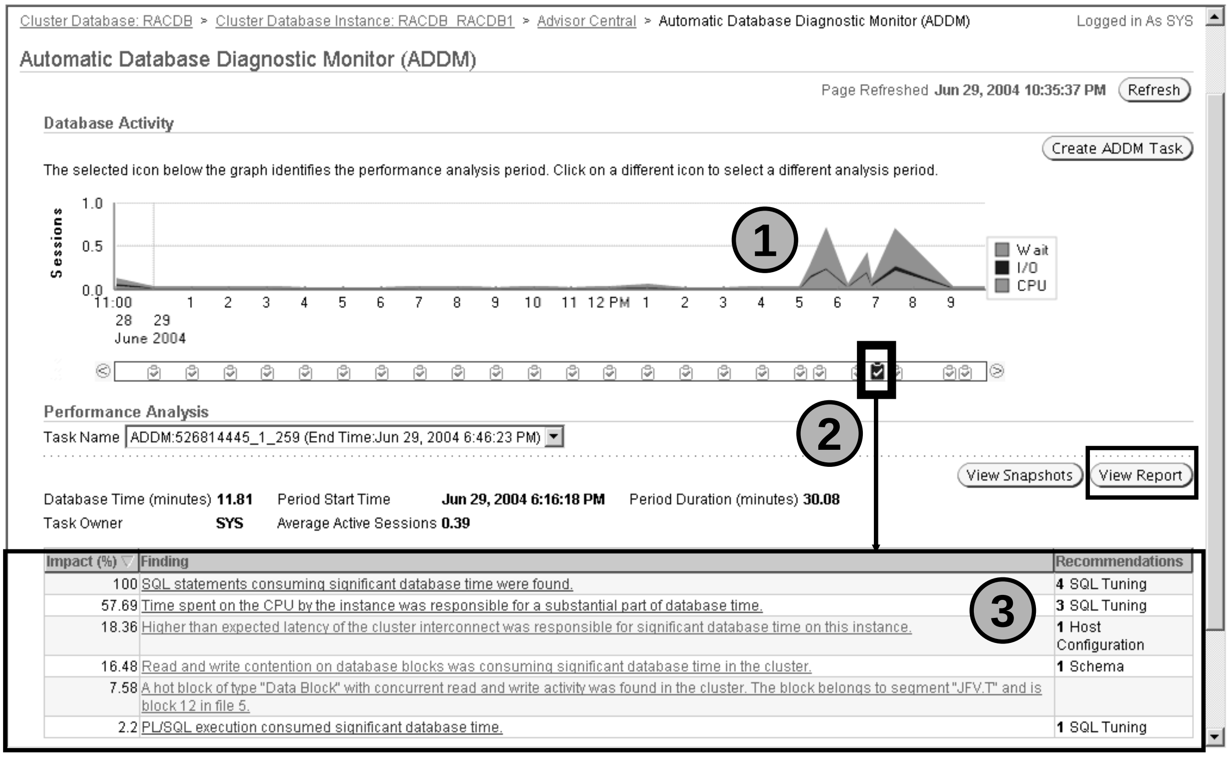
10-36

RAC-Specific ADDM Findings

RAC-specific ADDM findings include:

- Hot block (with block details) with high read/write contention within an instance and across the cluster
- Hot object with high read/write contention within an instance and across the cluster
- Cluster interconnect latency issues in a RAC environment
- LMS congestion issues: LMS processes are not able to keep up with lock requests.
- Top SQL that encounters interinstance messaging
- Contention on other instances: Basically multiple instances are updating the same set of blocks concurrently.

ADDM Analysis: Results



Cluster Database: RACDB > Cluster Database Instance: RACDB_RACDB1 > Advisor Central > Automatic Database Diagnostic Monitor (ADDM) Logged in As SYS

Automatic Database Diagnostic Monitor (ADDM)

Page Refreshed **Jun 29, 2004 10:35:37 PM**

Database Activity

The selected icon below the graph identifies the performance analysis period. Click on a different icon to select a different analysis period.

Performance Analysis

Task Name ADDM:526814445_1_259 (End Time:Jun 29, 2004 6:46:23 PM)

Database Time (minutes) **11.81** Period Start Time **Jun 29, 2004 6:16:18 PM** Period Duration (minutes) **30.08**

Task Owner **SYS** Average Active Sessions **0.39**

Impact (%) ▽	Finding	Recommendations
100	SQL statements consuming significant database time were found.	**4** SQL Tuning
57.69	Time spent on the CPU by the instance was responsible for a substantial part of database time.	**3** SQL Tuning
18.36	Higher than expected latency of the cluster interconnect was responsible for significant database time on this instance.	**1** Host Configuration
16.48	Read and write contention on database blocks was consuming significant database time in the cluster.	**1** Schema
7.58	A hot block of type "Data Block" with concurrent read and write activity was found in the cluster. The block belongs to segment "JFV.T" and is block 12 in file 5.	
2.2	PL/SQL execution consumed significant database time.	**1** SQL Tuning

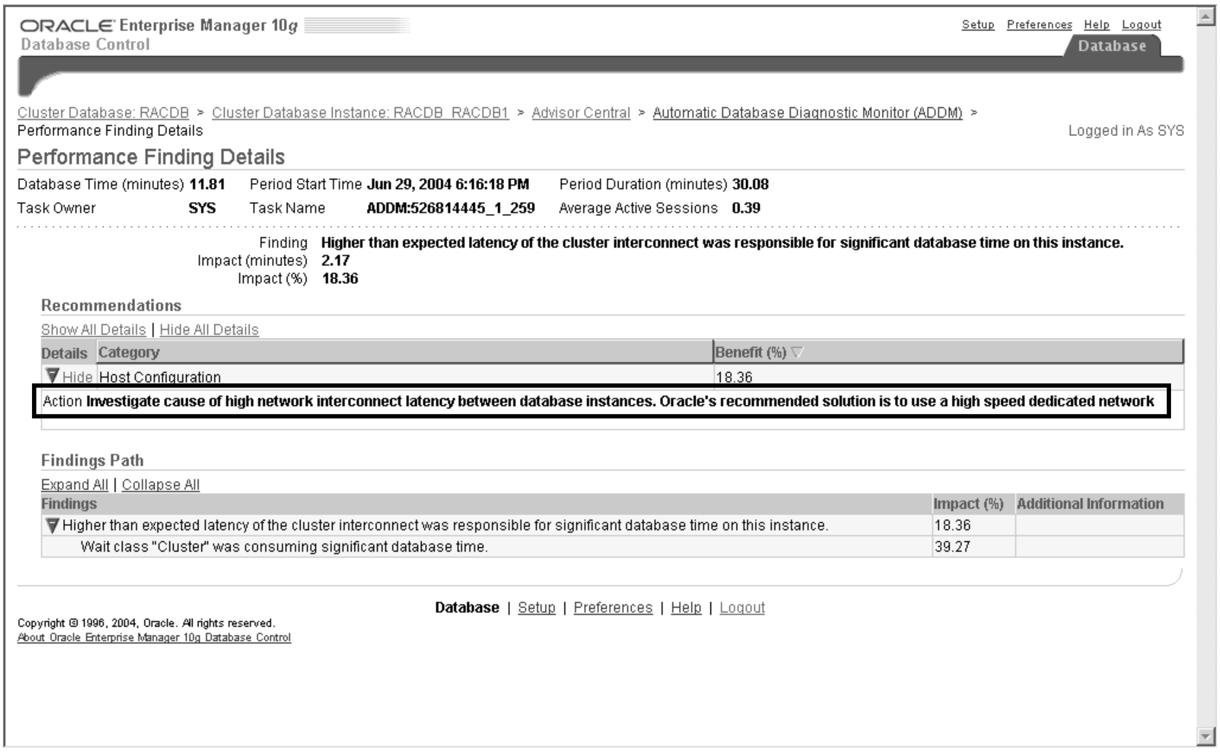
ADDM Analysis: Results

On the **Automatic Database Diagnostic Monitor (ADDM)** page, you can see the detailed findings for the latest ADDM run. The **Database Time** represents the sum of the nonidle time spent by sessions in the database for the analysis period. A specific **Impact** percentage is given for each finding. The impact represents the time consumed by the corresponding issue as compared with the database time for the analysis period. Here is a corresponding description of the numbers shown in the slide:

1. The graphic shows that the number of average active users increased dramatically at this point. Also, the major problem is a wait problem.
2. The icon shows that the ADDM output displayed at the bottom of the page corresponds to this point in time. You can go into the past (to view previous analysis) by clicking the other icons.
3. The findings give you a short summary of what the ADDM found as performance areas in the instance that could be tuned. By clicking a particular issue, you are directed to the **Performance Finding Details** page.

You can click the **View Report** button to get details about the performance analysis in the form of a text report. By clicking a particular issue, you are directed to the **Performance Finding Details** page.

ADDM Recommendations



Cluster Database: RACDB > Cluster Database Instance: RACDB_RACDB1 > Advisor Central > Automatic Database Diagnostic Monitor (ADDM) >
Performance Finding Details

Logged in As SYS

Performance Finding Details

Database Time (minutes)	**11.81**	Period Start Time	**Jun 29, 2004 6:16:18 PM**	Period Duration (minutes)	**30.08**
Task Owner	**SYS**	Task Name	**ADDM:526814445_1_259**	Average Active Sessions	**0.39**

Finding **Higher than expected latency of the cluster interconnect was responsible for significant database time on this instance.**
Impact (minutes) **2.17**
Impact (%) **18.36**

Recommendations

Show All Details | Hide All Details

Details	Category	Benefit (%) ▽
▼ Hide	Host Configuration	18.36

Action **Investigate cause of high network interconnect latency between database instances. Oracle's recommended solution is to use a high speed dedicated network**

Findings Path

Expand All | Collapse All

Findings	Impact (%)	Additional Information
▼ Higher than expected latency of the cluster interconnect was responsible for significant database time on this instance.	18.36	
Wait class "Cluster" was consuming significant database time.	39.27	

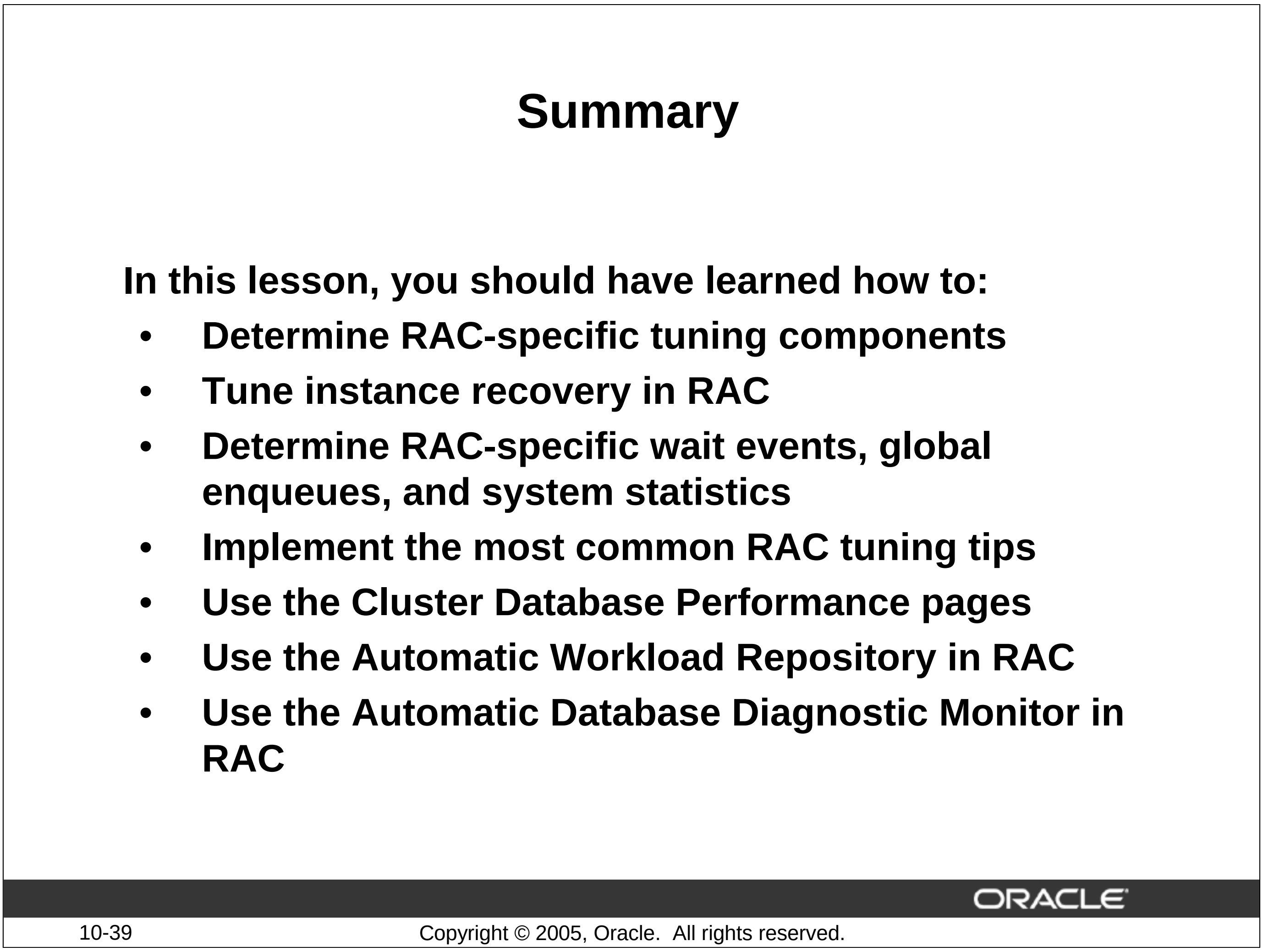
Copyright © 1996, 2004, Oracle. All rights reserved.
About Oracle Enterprise Manager 10g Database Control

ADDM Recommendations

On the **Performance Finding Details** page, you are given recommendations to solve the corresponding issue. Recommendations are divided into many categories such as **Schema**, **SQL Tuning**, **DB configuration**, and so on. The **Benefit(%)** column gives you the maximum reduction in database elapse time if the recommendation is implemented.

The ADDM considers a variety of changes to a system, and its recommendations can include:

- Hardware changes: Adding CPUs or changing the I/O subsystem configuration
- Database configuration: Changing initialization parameter settings
- Schema changes: Hash-partitioning a table or index, or using automatic segment space management
- Application changes: Using the cache option for sequences or using bind variables
- Using other advisors: Running the SQL Tuning Advisor on high-load SQL or running the Segment Advisor on hot objects

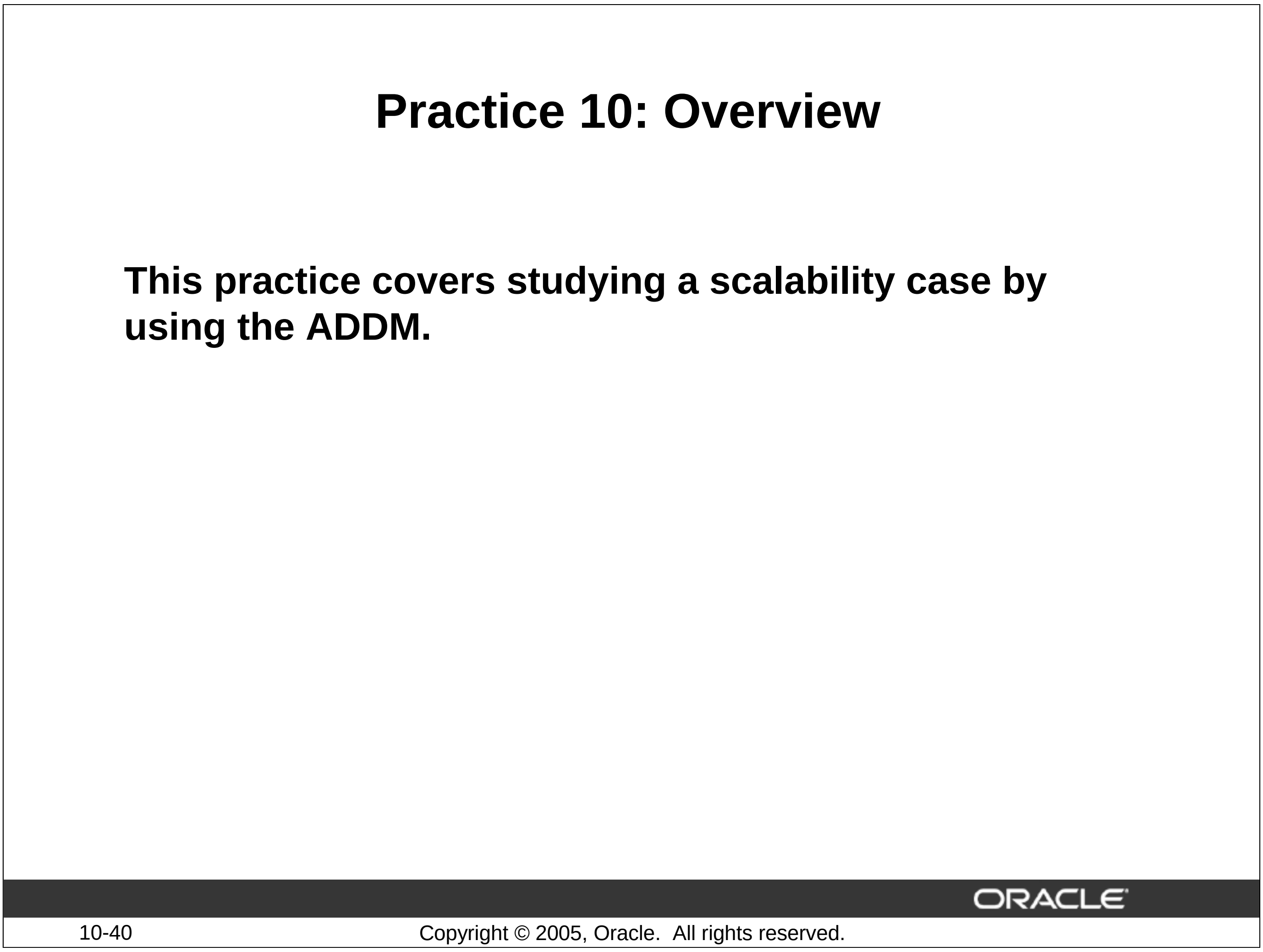


Summary

In this lesson, you should have learned how to:

- Determine RAC-specific tuning components
- Tune instance recovery in RAC
- Determine RAC-specific wait events, global enqueues, and system statistics
- Implement the most common RAC tuning tips
- Use the Cluster Database Performance pages
- Use the Automatic Workload Repository in RAC
- Use the Automatic Database Diagnostic Monitor in RAC

10-39



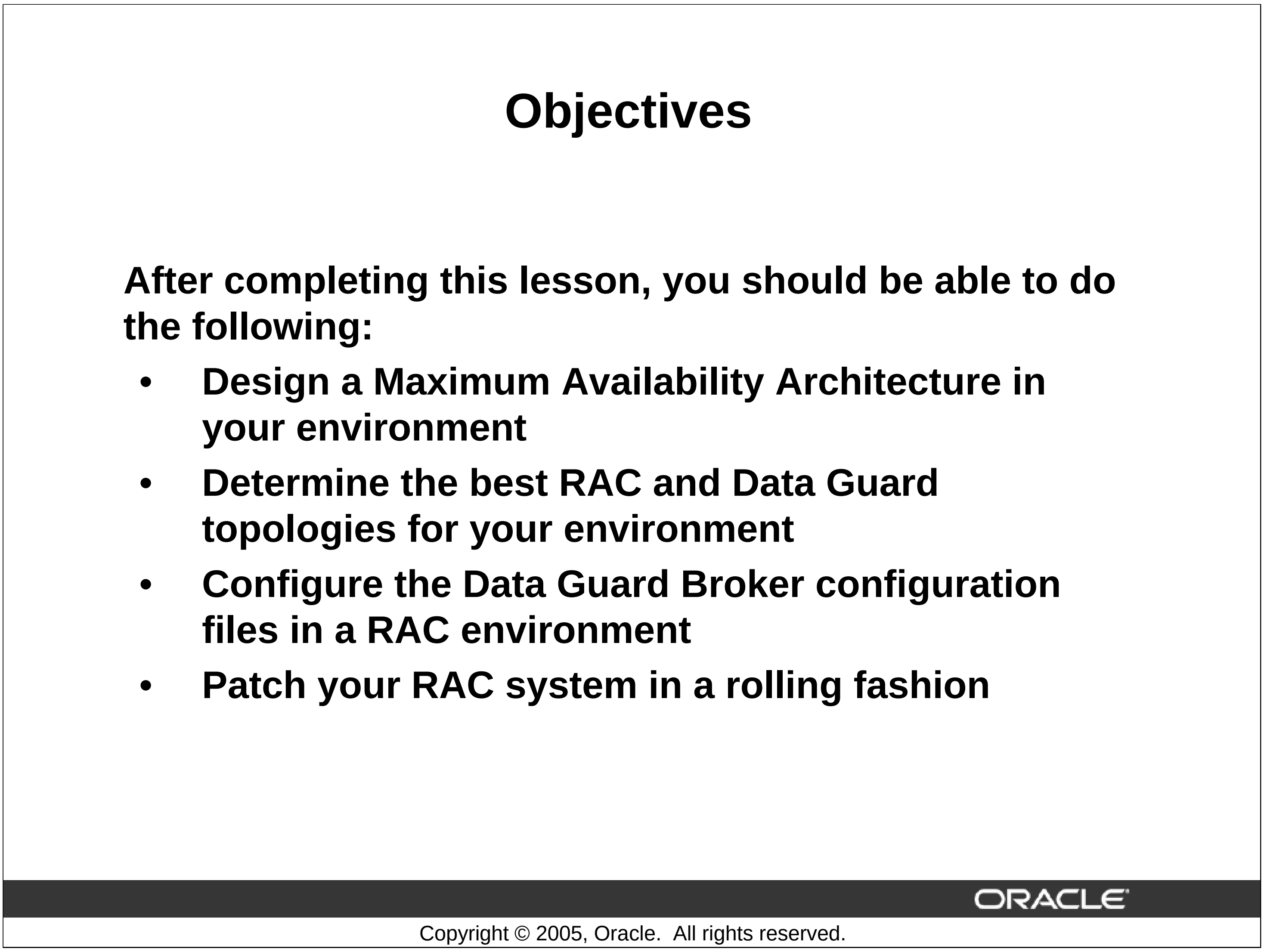
Practice 10: Overview

This practice covers studying a scalability case by using the ADDM.

10-40

11

Design for High Availability



Objectives

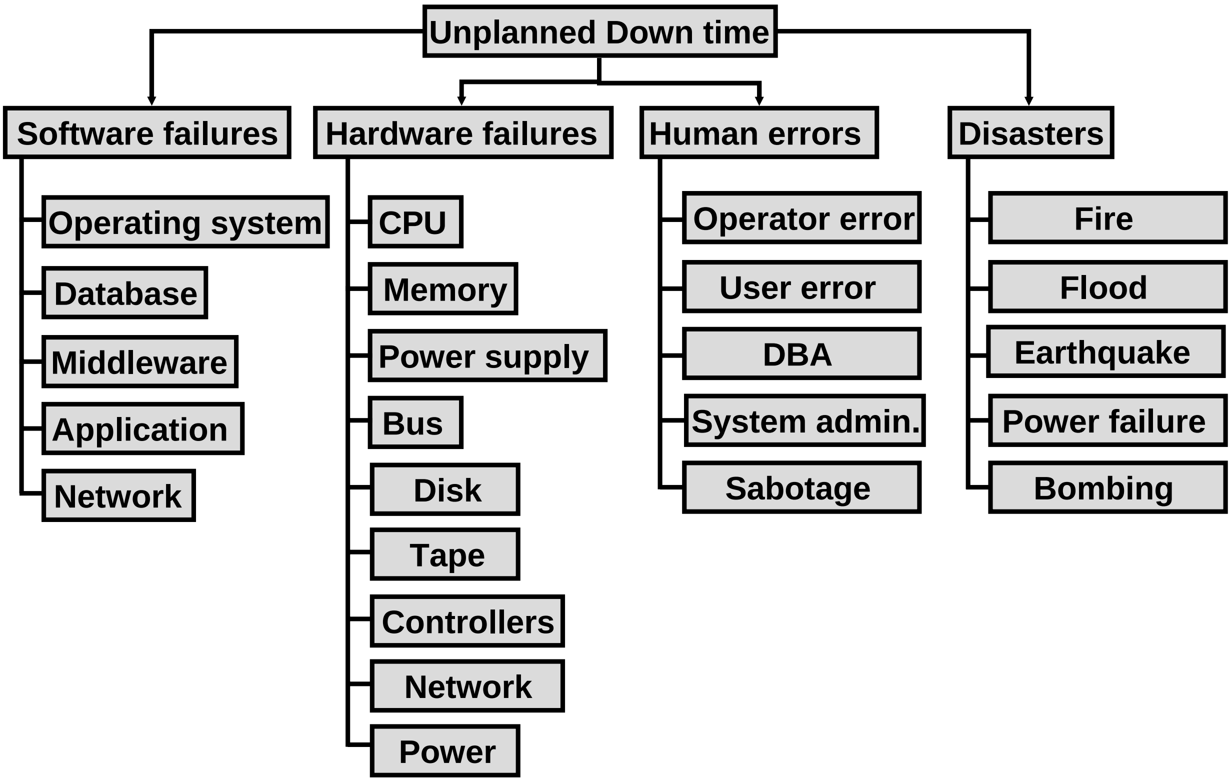
After completing this lesson, you should be able to do the following:

- Design a Maximum Availability Architecture in your environment
- Determine the best RAC and Data Guard topologies for your environment
- Configure the Data Guard Broker configuration files in a RAC environment
- Patch your RAC system in a rolling fashion

Objectives

The goal of this lesson is to provide an overview of the various high-availability architectures that you can implement with RAC. It is out of the scope of this lesson to give you detailed information on how to set up these architectures. For more information, refer to the corresponding documentation or courses.

Causes of Unplanned Down Time



Causes of Unplanned Down Time

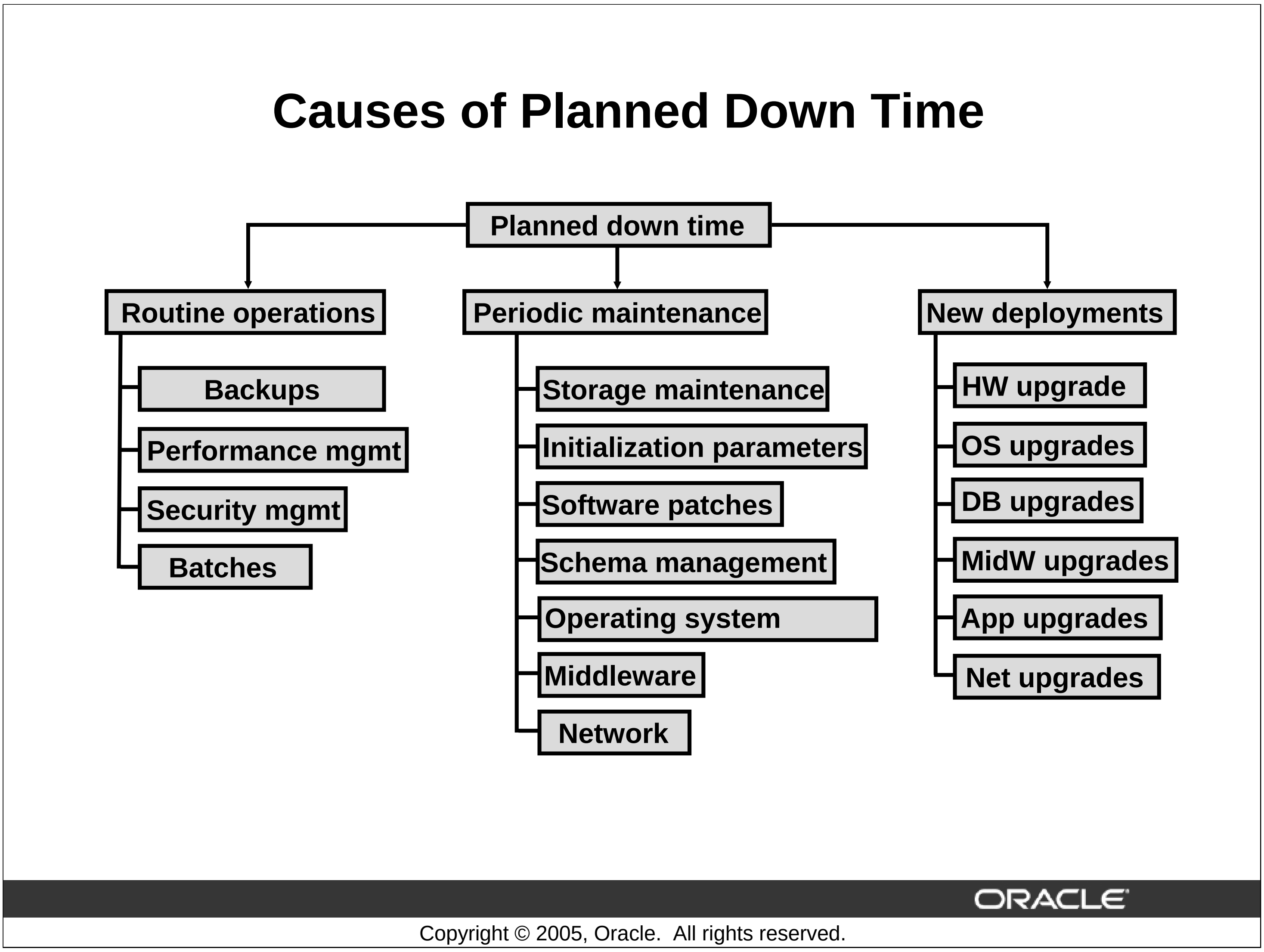
One of the true challenges in designing a highly available solution is examining and addressing all the possible causes of down time. It is important to consider causes of both unplanned and planned down time. The above schema, which is a taxonomy of unplanned failures, classifies failures as software failures, hardware failures, human error, and disasters. Under each category heading is a list of possible causes of failures related to that category.

Software failures include operating system, database, middleware, application, and network failures. A failure of any one of these components can cause a system fault.

Hardware failures include system, peripheral, network, and power failures.

Human error, which is a leading cause of failures, includes errors by an operator, user, database administrator, or system administrator. Another type of human error that can cause unplanned down time is sabotage.

The final category is disasters. Although infrequent, these can have extreme impacts on enterprises, because of their prolonged effect on operations. Possible causes of disasters include fires, floods, earthquakes, power failures, and bombings. A well-designed high-availability solution accounts for all these factors in preventing unplanned down time.



Causes of Planned Down Time

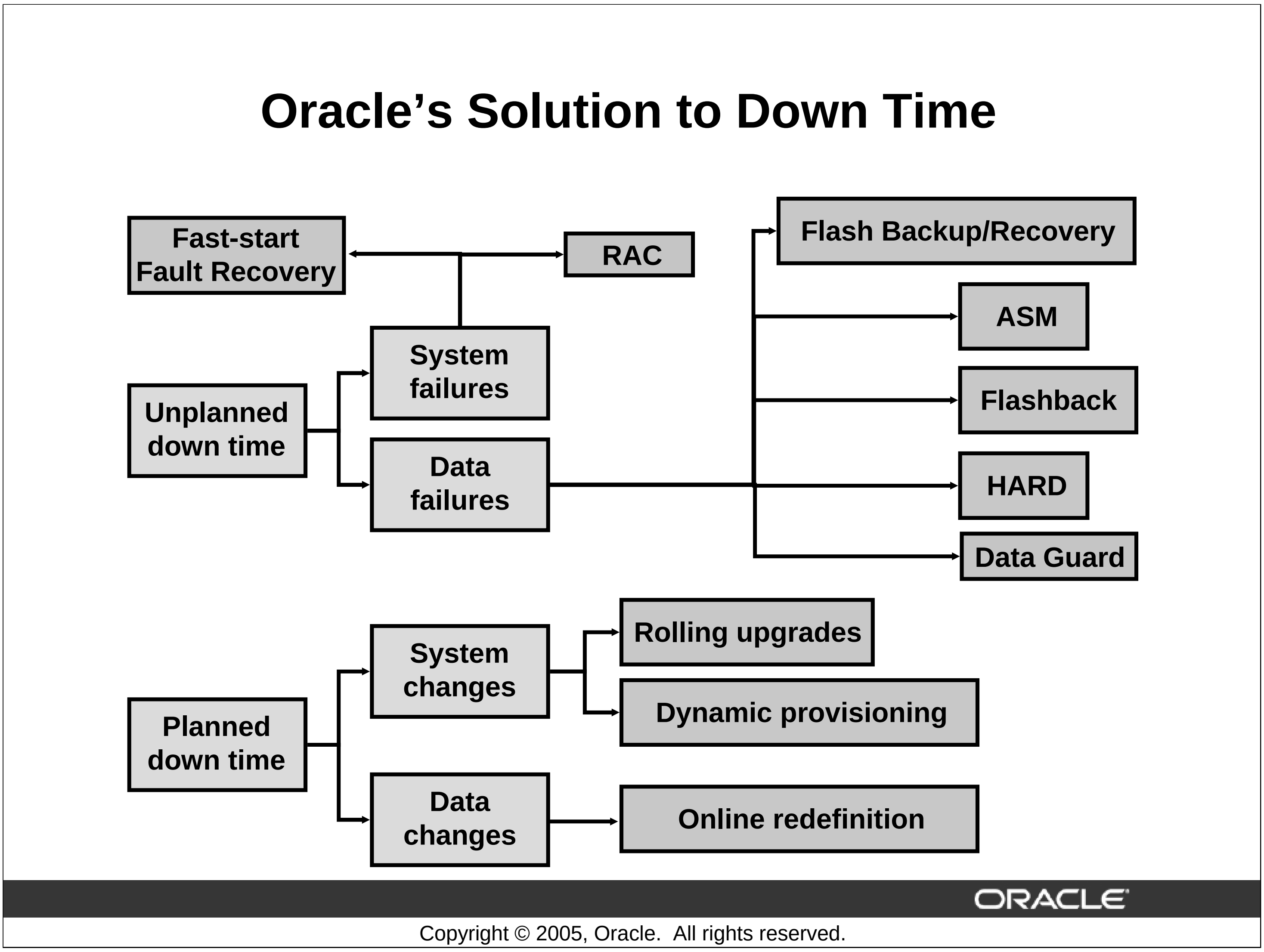
- Planned down time
  - Routine operations
    - Backups
    - Performance mgmt
    - Security mgmt
    - Batches
  - Periodic maintenance
    - Storage maintenance
    - Initialization parameters
    - Software patches
    - Schema management
    - Operating system
    - Middleware
    - Network
  - New deployments
    - HW upgrade
    - OS upgrades
    - DB upgrades
    - MidW upgrades
    - App upgrades
    - Net upgrades

Causes of Planned Down Time

Planned down time can be just as disruptive to operations, especially in global enterprises that support users in multiple time zones, up to 24-hours per day. In these cases, it is important to design a system to minimize planned interruptions. As shown by the schema in the slide above, causes of planned down time include routine operations, periodic maintenance, and new deployments. Routine operations are frequent maintenance tasks that include backups, performance management, user and security management, and batch operations.

Periodic maintenance, such as installing a patch or reconfiguring the system, is occasionally necessary to update the database, application, operating system middleware, or network.

New deployments describe major upgrades to the hardware, operating system, database, application, middleware, or network. It is important to consider not only the time to perform the upgrade, but also the effect the changes may have on the overall application.



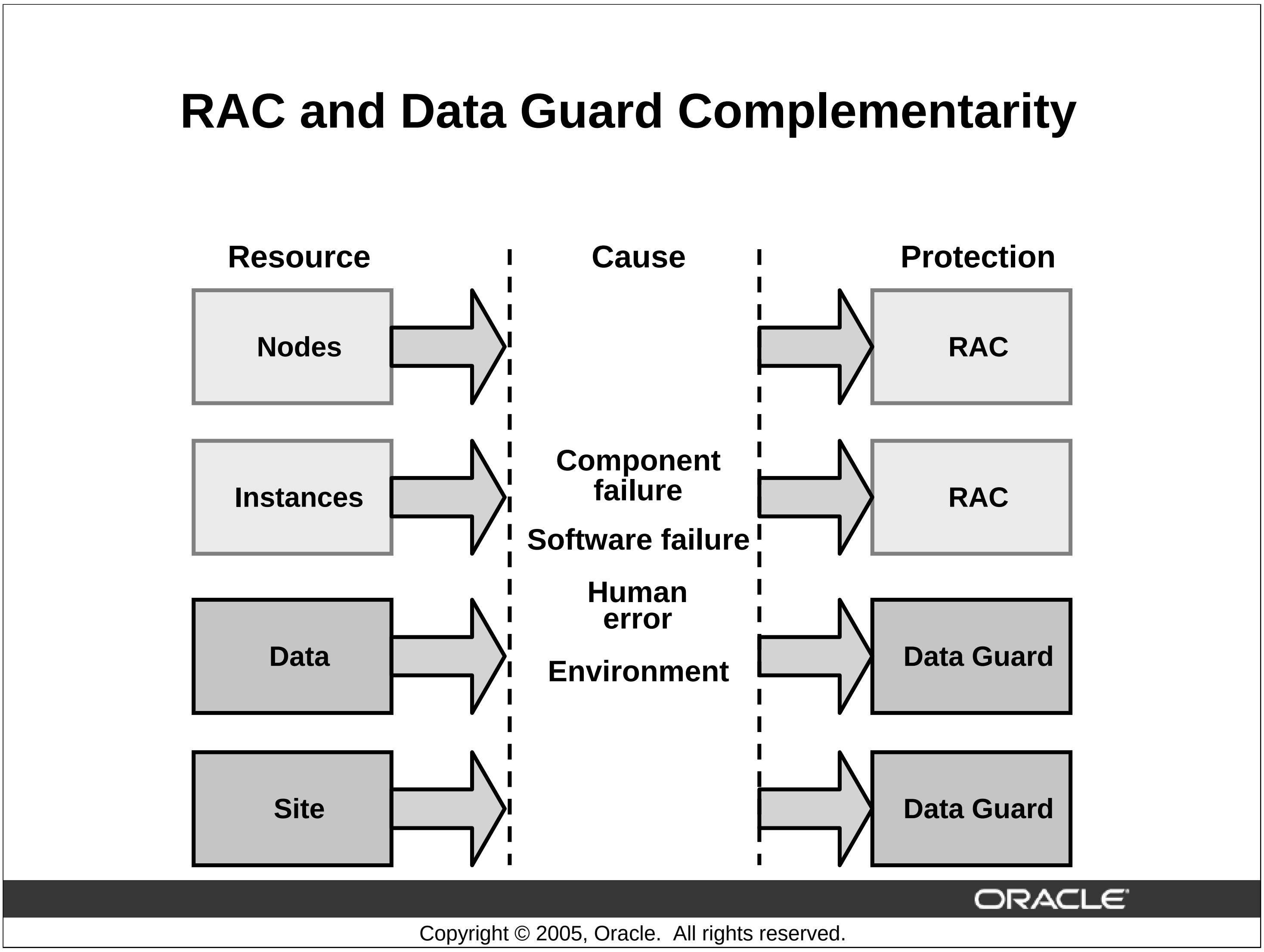
Oracle's Solution to Down Time

Oracle's Solution to Down Time

Unplanned down time is primarily the result of computer failures or data failures. Planned down time is primarily due to data changes or system changes:

- RAC provides optimal performance, scalability, and availability gains.
- Fast-start Fault Recovery enables you to bound the database crash/recovery time. The database self-tunes checkpoint processing to safeguard the desired recovery time objective.
- ASM provides a higher level of availability using online provisioning of database storage.
- Flashback provides a family of human error correction technology.
- Oracle Hardware Assisted Resilient Data (HARD) is a comprehensive program designed to prevent data corruptions before they happen.
- Recovery Manager (RMAN) automates database backup and recovery by using the flash recovery area.
- Data Guard must be the foundation of any Oracle database disaster-recovery plan.
- With online redefinition, the Oracle database supports many maintenance operations without disrupting database operations, or users updating or accessing data.
- The Oracle database continues to broaden support for dynamic reconfiguration enabling it to adapt to changes in demand and hardware with no disruption of service.
- The Oracle database supports the application of patches to the nodes of a RAC system, as well as database software upgrades, in a rolling fashion.

Note: The above list is not complete.



RAC and Data Guard Complementarity

Resource

Nodes

Instances

Data

Site

Cause

Component failure

Software failure

Human error

Environment

Protection

RAC

RAC

Data Guard

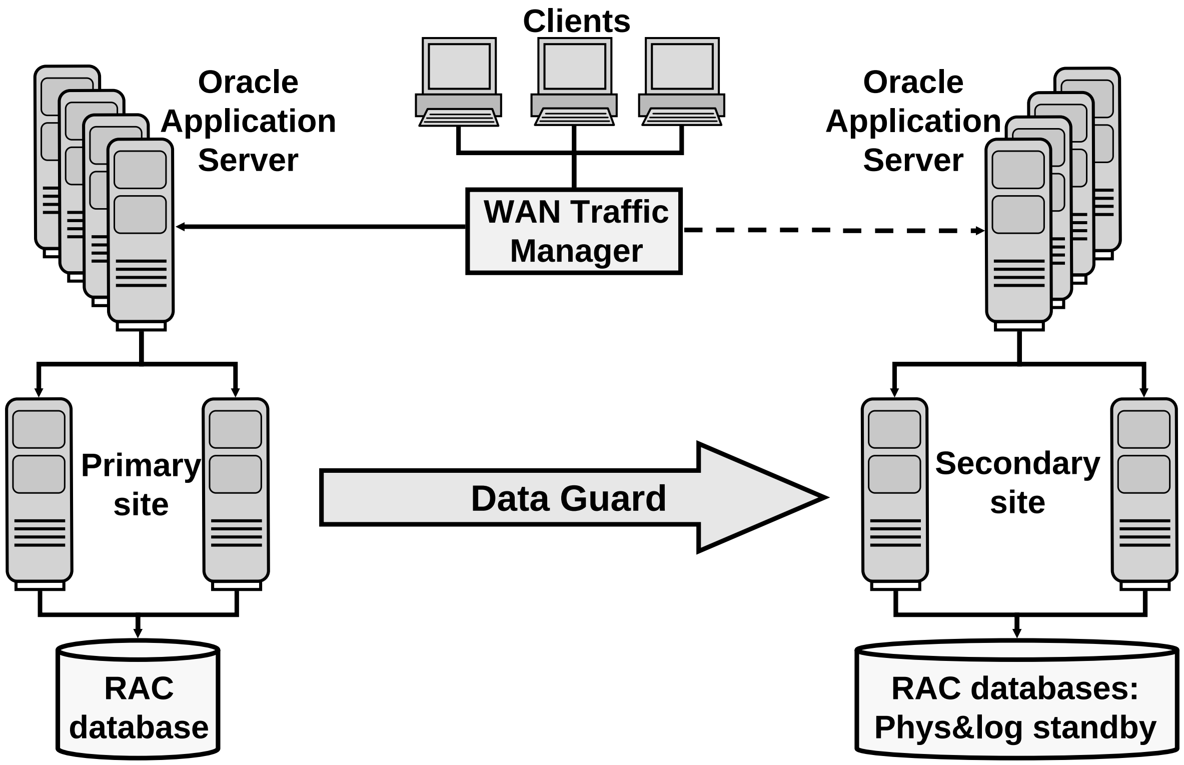
Data Guard

RAC and Data Guard Complementarity

RAC and Data Guard together provide the benefits of system-level, site-level, and data-level protection, resulting in high levels of availability and disaster recovery without loss of data:

- RAC addresses system failures by providing rapid and automatic recovery from failures, such as node failures and instance crashes.
- Data Guard addresses site failures and data protection through transactionally consistent primary and standby databases that do not share disks, enabling recovery from site disasters and data corruption.

Maximum Availability Architecture

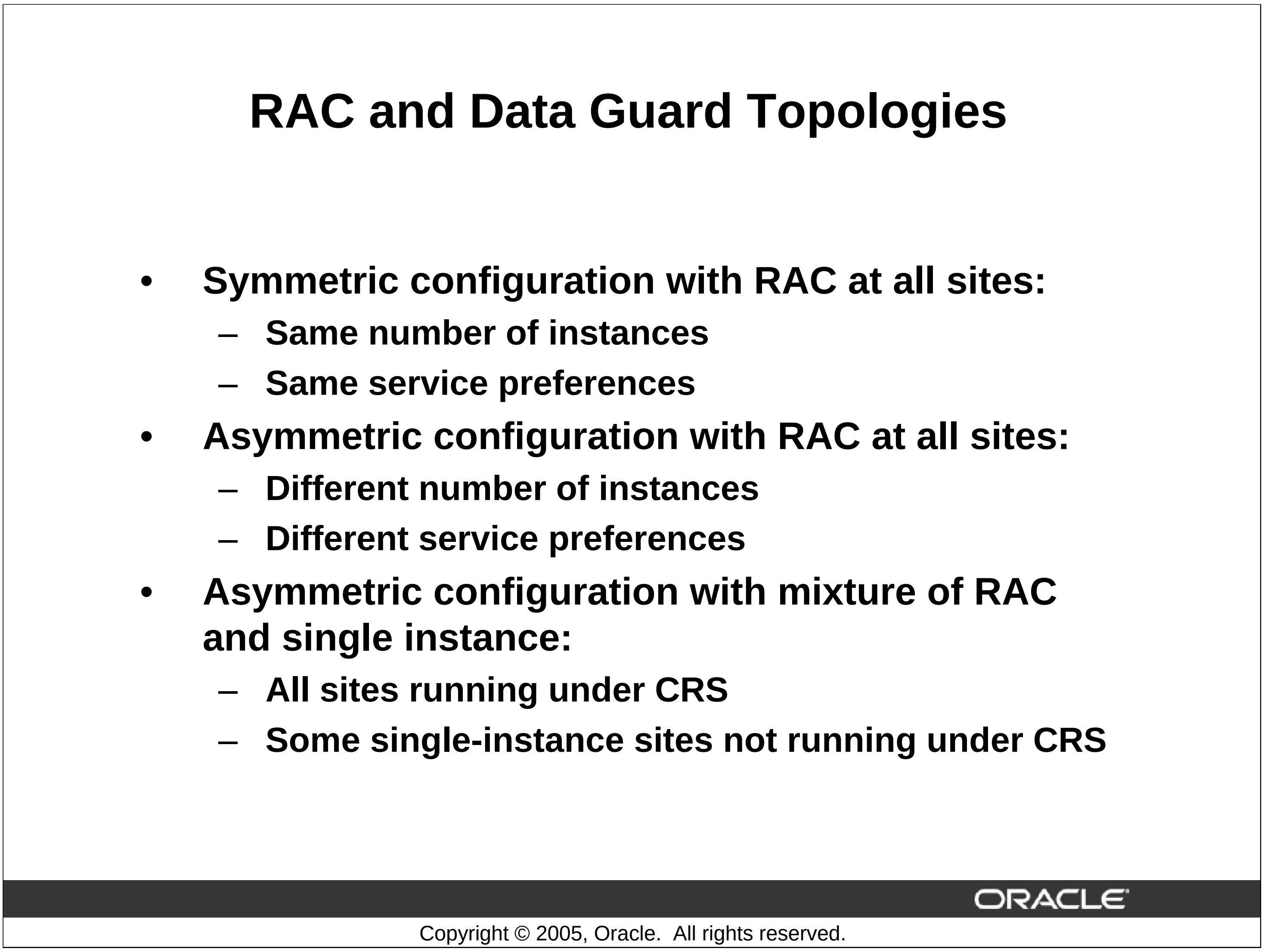

Clients
Oracle Application Server
WAN Traffic Manager
Oracle Application Server
Primary site
Data Guard
Secondary site
RAC database
RAC databases: Phys&log standby

Maximum Availability Architecture (MAA)

RAC and Data Guard provide the basis of the database MAA solution. MAA provides the most comprehensive architecture for reducing down time for scheduled outages and preventing, detecting, and recovering from unscheduled outages. The recommended MAA has two identical sites. The primary site contains the RAC database, and the secondary site contains both a physical standby database and a logical standby database on RAC. Identical site configuration is recommended to ensure that performance is not sacrificed after a failover or switchover. Symmetric sites also enable processes and procedures to be kept the same between sites, making operational tasks easier to maintain and execute.

The graphic illustrates identically configured sites. Each site consists of redundant components and redundant routing mechanisms, so that requests are always serviceable even in the event of a failure. Most outages are resolved locally. Client requests are always routed to the site playing the production role.

After a failover or switchover operation occurs due to a serious outage, client requests are routed to another site that assumes the production role. Each site contains a set of application servers or mid-tier servers. The site playing the production role contains a production database using RAC to protect from host and instance failures. The site playing the standby role contains one standby database, and one logical standby database managed by Data Guard. Data Guard switchover and failover functions allow the roles to be traded between sites.
Note: For more information, see the following Web site:
http://otn.oracle.com/deploy/availability/htdocs/maa.htm



RAC and Data Guard Topologies

- Symmetric configuration with RAC at all sites:
  - Same number of instances
  - Same service preferences
- Asymmetric configuration with RAC at all sites:
  - Different number of instances
  - Different service preferences
- Asymmetric configuration with mixture of RAC and single instance:
  - All sites running under CRS
  - Some single-instance sites not running under CRS

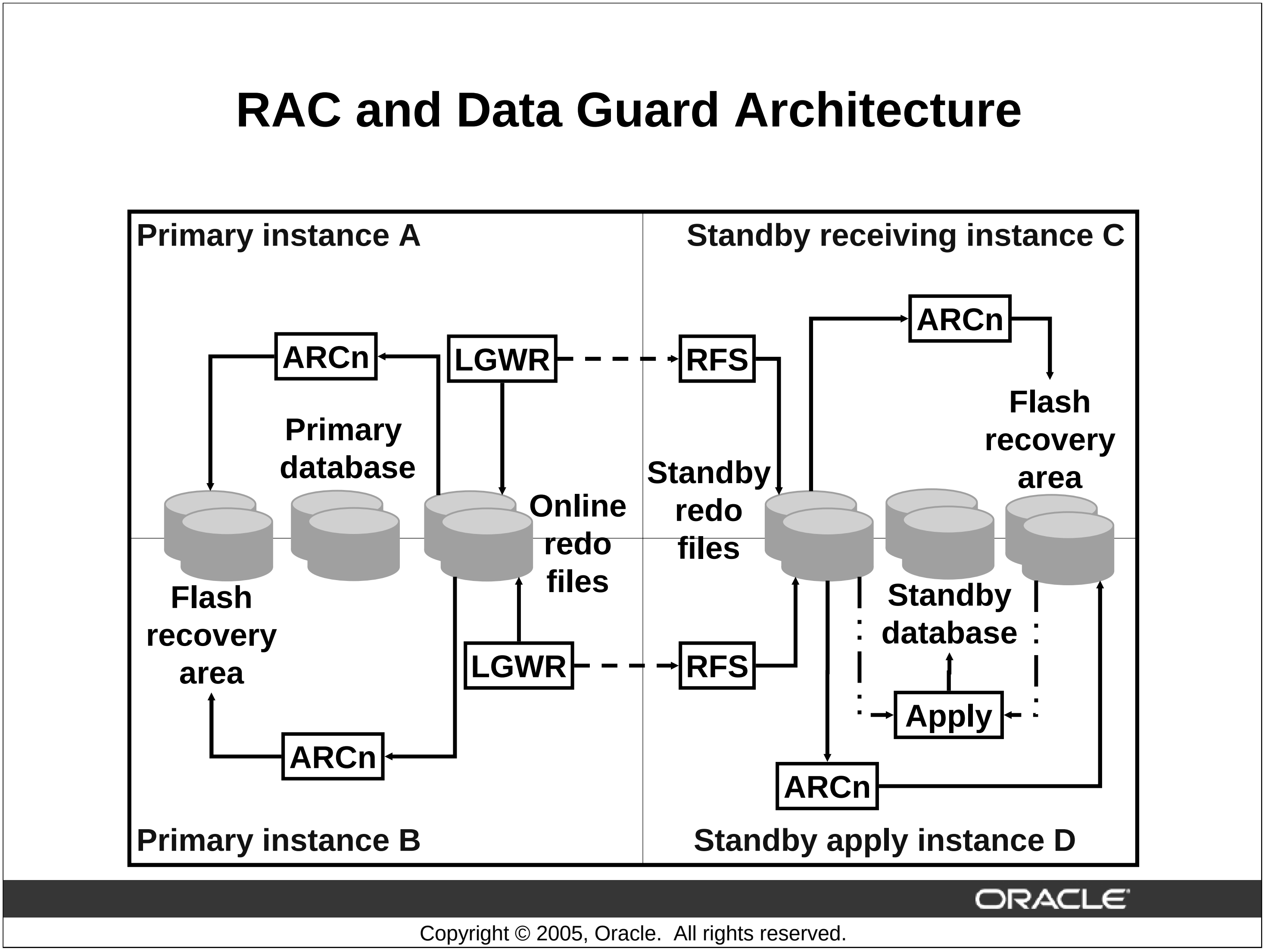
RAC and Data Guard Topologies

You can configure a standby database to protect a primary database in a RAC environment. Basically, all kind of combinations are supported. For example, it is possible to have your primary database running under RAC, and your standby database running as a single-instance database. It is also possible to have both the primary and standby databases running under RAC.

The slide above explains the distinction between symmetric environments and asymmetric ones.

If you want to create a symmetric environment running RAC, then all databases need to have the same number of instances and the same service preferences. As the DBA, you need to make sure that this is the case by manually configuring them in a symmetric way.

However, if you want to benefit from the tight integration of CRS and Data Guard Broker, make sure that both the primary site and the secondary site are running under CRS, and that both sites have the same services defined.



RAC and Data Guard Architecture

RAC and Data Guard Architecture

Although it is perfectly possible to use a RAC to single-instance Data Guard (DG) configuration, you also have the possibility to use a RAC-to-RAC DG configuration. In this mode, although multiple standby instances can receive redo from the primary database, only one standby instance can apply the redo stream generated by the primary instances.

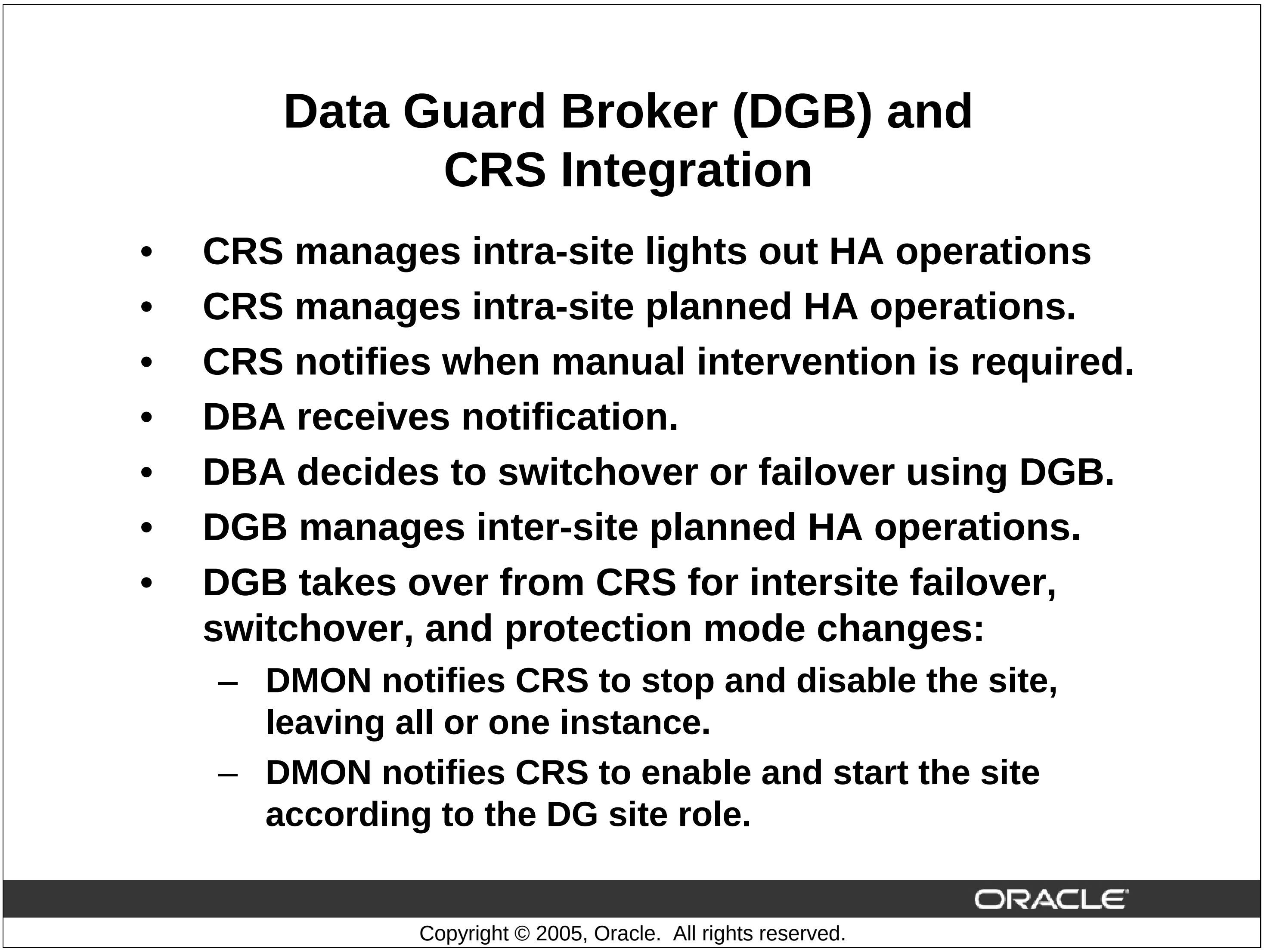
A RAC-to-RAC DG configuration can be set up in different ways, and the slide shows you one possibility with a symmetric configuration where each primary instance sends its redo stream to a corresponding standby instance using standby redo log files. It is also possible for each primary instance to send its redo stream to only one standby instance that can also apply this stream to the standby database. However, you can get performance benefits by using the configuration shown in the slide above. For example, assume that the redo generation rate on the primary is too great for a single receiving instance on the standby side to handle. Suppose further that the primary database is using the `SYNC` redo transport mode. If a single receiving instance on the standby cannot keep up with the primary, then the primary's progress is going to be throttled by the standby. If the load is spread across multiple receiving instances on the standby, then this is less likely to occur.

If the standby can keep up with the primary, another approach is to use only one standby instance to receive and apply the complete redo stream. For example, you can set up the primary instances to remotely archive to the same Oracle Net service name.

RAC and Data Guard Architecture (continued)

Then, you can configure one of the standby nodes to handle that service. This instance then both receives and applies redo from the primary. If you need to do maintenance on that node, then you can stop the service on that node and start it on another node. This approach allows for the primary instances to be more independent of the standby configuration because they are not configured to send redo to a particular instance.

Note: For more information, refer to the *Oracle Data Guard Concepts and Administration* guide.



Data Guard Broker (DGB) and CRS Integration

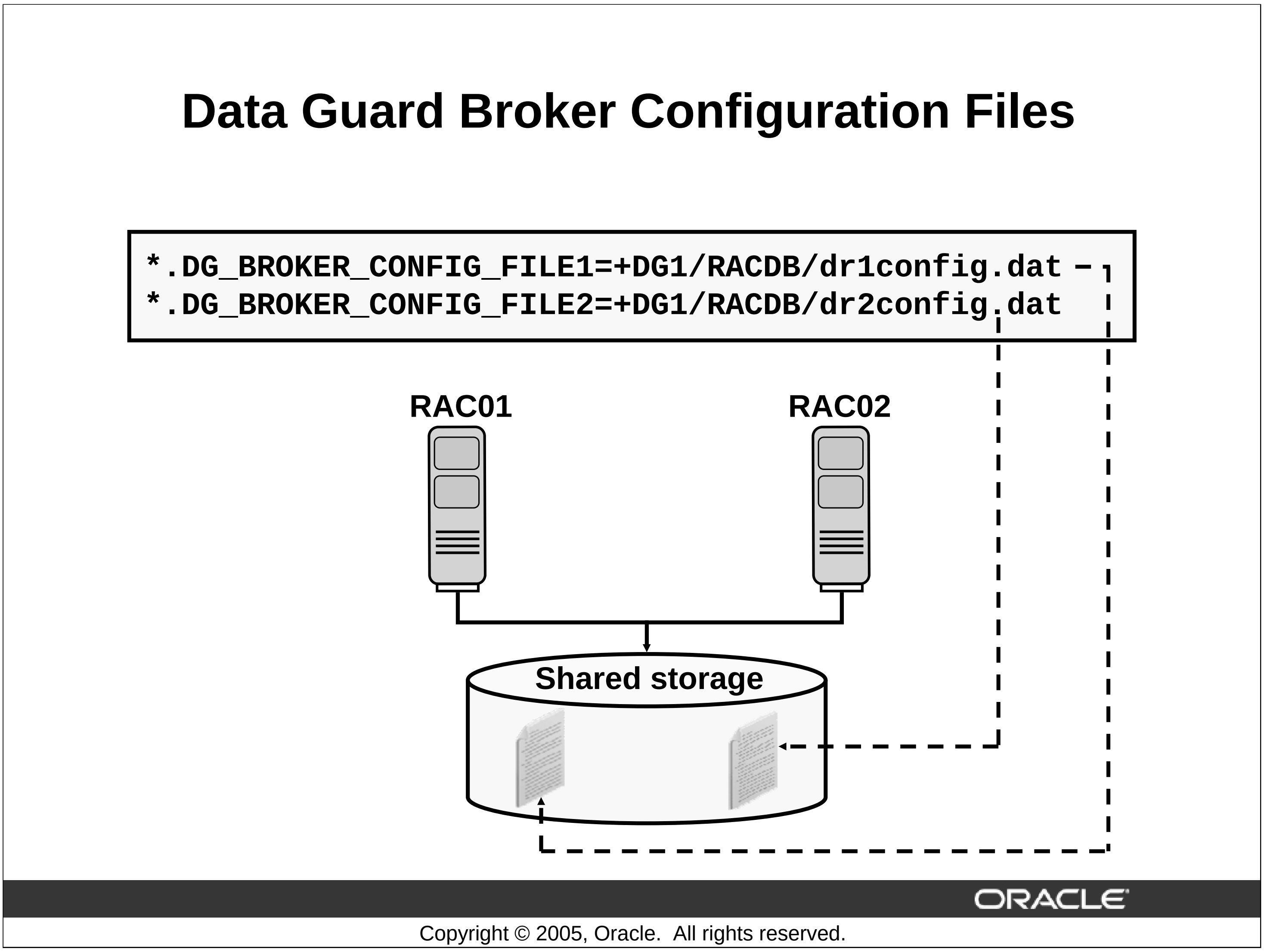
- CRS manages intra-site lights out HA operations
- CRS manages intra-site planned HA operations.
- CRS notifies when manual intervention is required.
- DBA receives notification.
- DBA decides to switchover or failover using DGB.
- DGB manages inter-site planned HA operations.
- DGB takes over from CRS for intersite failover, switchover, and protection mode changes:
  - DMON notifies CRS to stop and disable the site, leaving all or one instance.
  - DMON notifies CRS to enable and start the site according to the DG site role.

Data Guard Broker (DGB) and CRS Integration

DGB is tightly integrated with CRS. CRS manages individual instances to provide unattended high availability of a given clustered database. DGB manages individual databases (clustered or otherwise) in a Data Guard configuration to provide disaster recovery in the event that CRS is unable to maintain availability of the primary database.

For example, CRS posts `NOT_RESTARTING` events for the database group and service groups that cannot be recovered. These events are available through Enterprise Manager, ONS, and server-side callouts. As a DBA, when you receive those events, you might decide to repair and restart the primary site, or to invoke DGB to failover.

DGB and CRS work together to temporarily suspend service availability on the primary database, accomplish the actual role change for both databases during which CRS works with the DGB to properly restart the instances as necessary, then to resume service availability on the new primary database. The broker manages the underlying Data Guard configuration and its database roles while CRS manages service availability that depends upon those roles. Applications that rely upon CRS for managing service availability will see only a temporary suspension of service as the role change occurs within the Data Guard configuration.



Data Guard Broker Configuration Files

```
*.DG_BROKER_CONFIG_FILE1=+DG1/RACDB/dr1config.dat
*.DG_BROKER_CONFIG_FILE2=+DG1/RACDB/dr2config.dat
```

Data Guard Broker Configuration Files

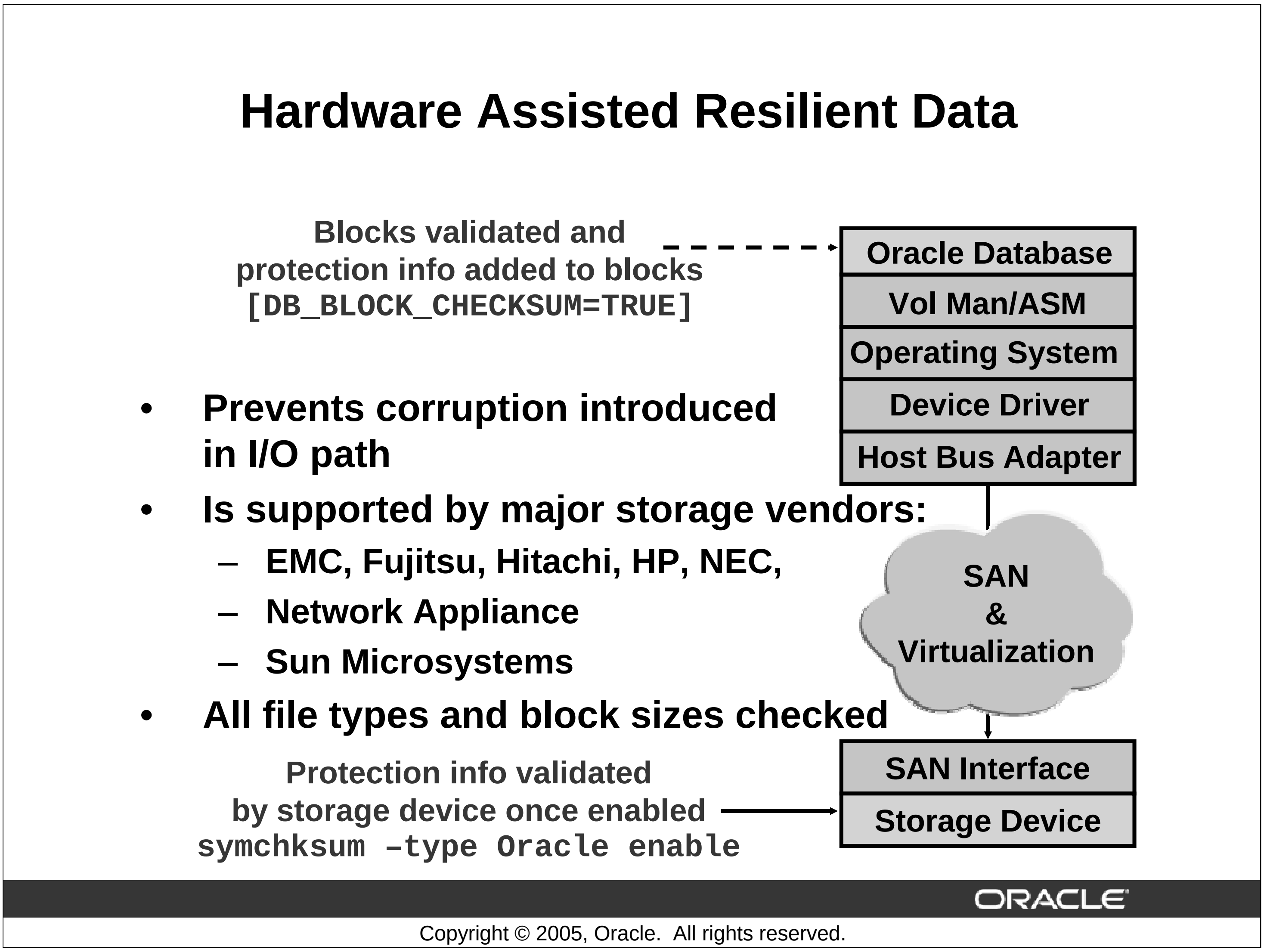
Two copies of the Data Guard Broker (DGB) configuration file are maintained for each database so as to always have a record of the last known valid state of the configuration. When the broker is started for the first time, the configuration files are automatically created and named using a default path name and file name that is operating system specific.

When using a RAC environment, the DGB configuration files must be shared by all instances of the same database. You can override the default path name and file name by setting the following initialization parameters for that database: `DG_BROKER_CONFIG_FILE1`, `DG_BROKER_CONFIG_FILE2`.

You have three possible options to share those files:

- Cluster file system
- Raw devices
- ASM

The above example illustrates a case where those files are stored in an ASM disk group called `DG1`. It is assumed that you have already created a directory called `RACDB` in `DG1`.



Hardware Assisted Resilient Data

Blocks validated and
protection info added to blocks
`[DB_BLOCK_CHECKSUM=TRUE]`

- Prevents corruption introduced in I/O path
- Is supported by major storage vendors:
  - EMC, Fujitsu, Hitachi, HP, NEC,
  - Network Appliance
  - Sun Microsystems
- All file types and block sizes checked

Protection info validated
by storage device once enabled
`symchksum –type Oracle enable`

Hardware Assisted Resilient Data

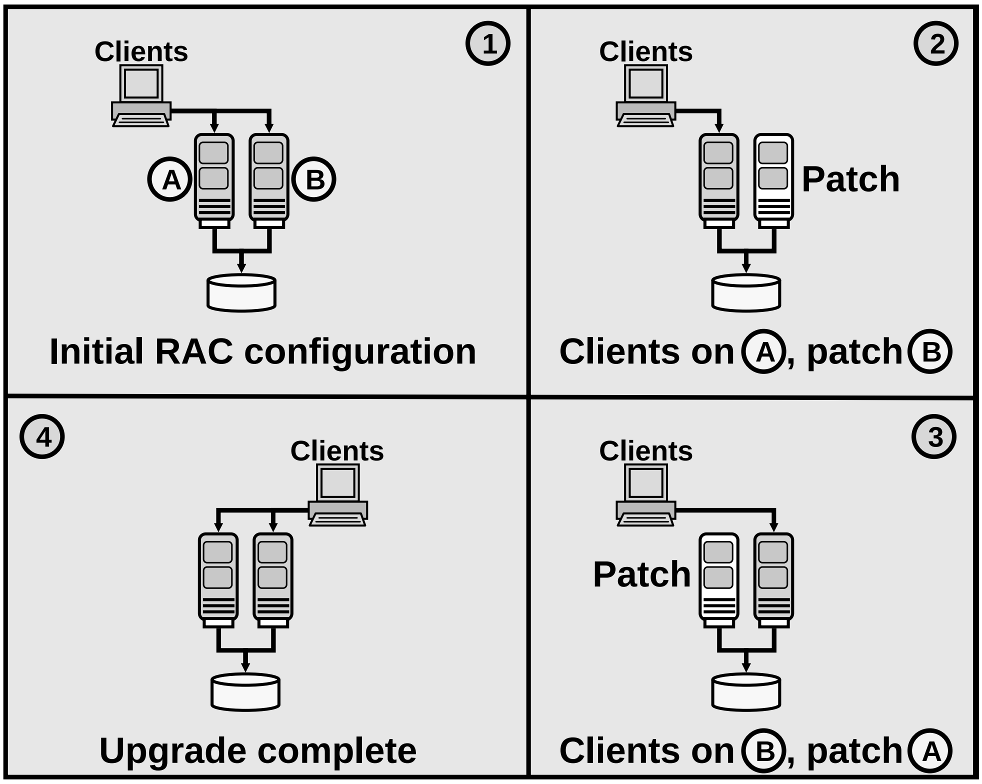
One problem that can cause lengthy outages is data corruption. Today, the primary means for detecting corruptions caused by hardware or software outside of Oracle, such as an I/O subsystem, is the Oracle checksum. However, after a block is passed to the operating system, through the volume manager and out to disk, Oracle itself can no longer provide any checking that the block being written is still correct.

With disk technologies expanding in complexity, with configurations such as Storage Area Networks (SANs) becoming more popular, the number of layers between the host processor and the physical spindle continues to increase. With more layers, the chance of any problem increases. With the HARD initiative, it is possible to enable the verification of database block checksum information by the storage device. Verifying that the block is still the same at the end of the write as it was in the beginning gives you an additional level of security.

By default, the Oracle database automatically add checksum information to its blocks. These checksums can be verified by the storage device if you enabled this possibility. In case a block is found to be corrupted by the storage device, it will log an I/O corruption, or it will cancel the I/O and report the error back to the instance.

Note: The way you enable the checksum validation at the storage device side is vendor specific. The above example was used with EMC Symmetrix storage.

Rolling Patch Upgrade Using RAC



Oracle patch upgrades

Operating system upgrades

Hardware upgrades

Rolling Patch Upgrade Using RAC

This is supported but only for single patches that are marked as rolling upgrade compatible. Rolling RAC patching allows the interoperation of a patched node and an unpatched node simultaneously. This means only one node is out of commission while it is patched.

Using the `OPATCH` tool to apply a rolling RAC patch, you are prompted to stop the instances on the node to be patched. First, the local node is patched, then you are asked for the next node to patch from a list. As each node is patched, the user is prompted when it is safe to restart the patched node. The cycle of prompting for a node, of stopping the instances on the node, of patching the node, and of restarting the instances continues until you stop the cycle, or until all nodes are patched.

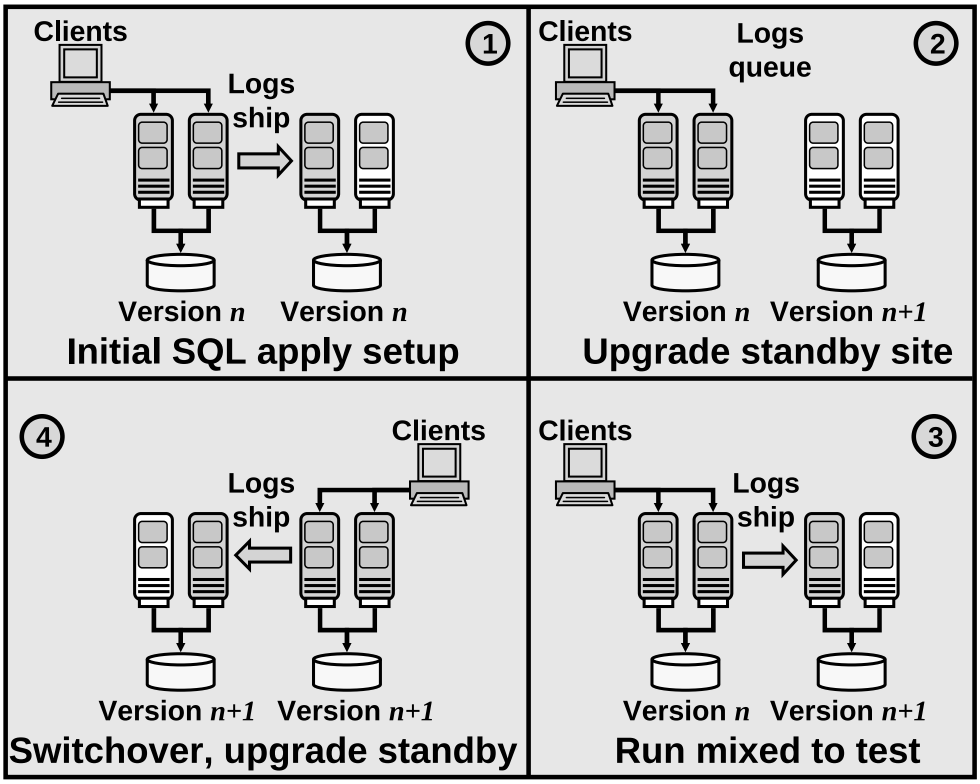
After you download the patch to your node, you need to unzip it before you can apply it. You can determine if the patch is flagged as rolling upgradable by checking the *Patch_number*`/etc/config/inventory` file. Near the end of the file you must see the following mark:

```
<online_rac_installable>true</online_rac_installable>
```

It is important to stress that although rolling patch upgrade allows you to test the patch before propagating it to the other nodes, it is preferable to test patches on test environment rather than directly on your production system.

Note: Some components cannot be changed a node at a time. The classic example is the data dictionary. Because, there is only a single data dictionary, all instances need to be shut down.

Rolling Release Upgrade Using SQL Apply



Patch set upgrades

Major release upgrades

Cluster software and hardware upgrades

Rolling Release Upgrade Using SQL Apply

It is possible to do a rolling upgrade using logical standby databases. For example, using SQL Apply and logical standby databases, you are able to upgrade the Oracle database software from patchset release 10.1.0.n to the next database 10.1.0.(n+1) patchset release.

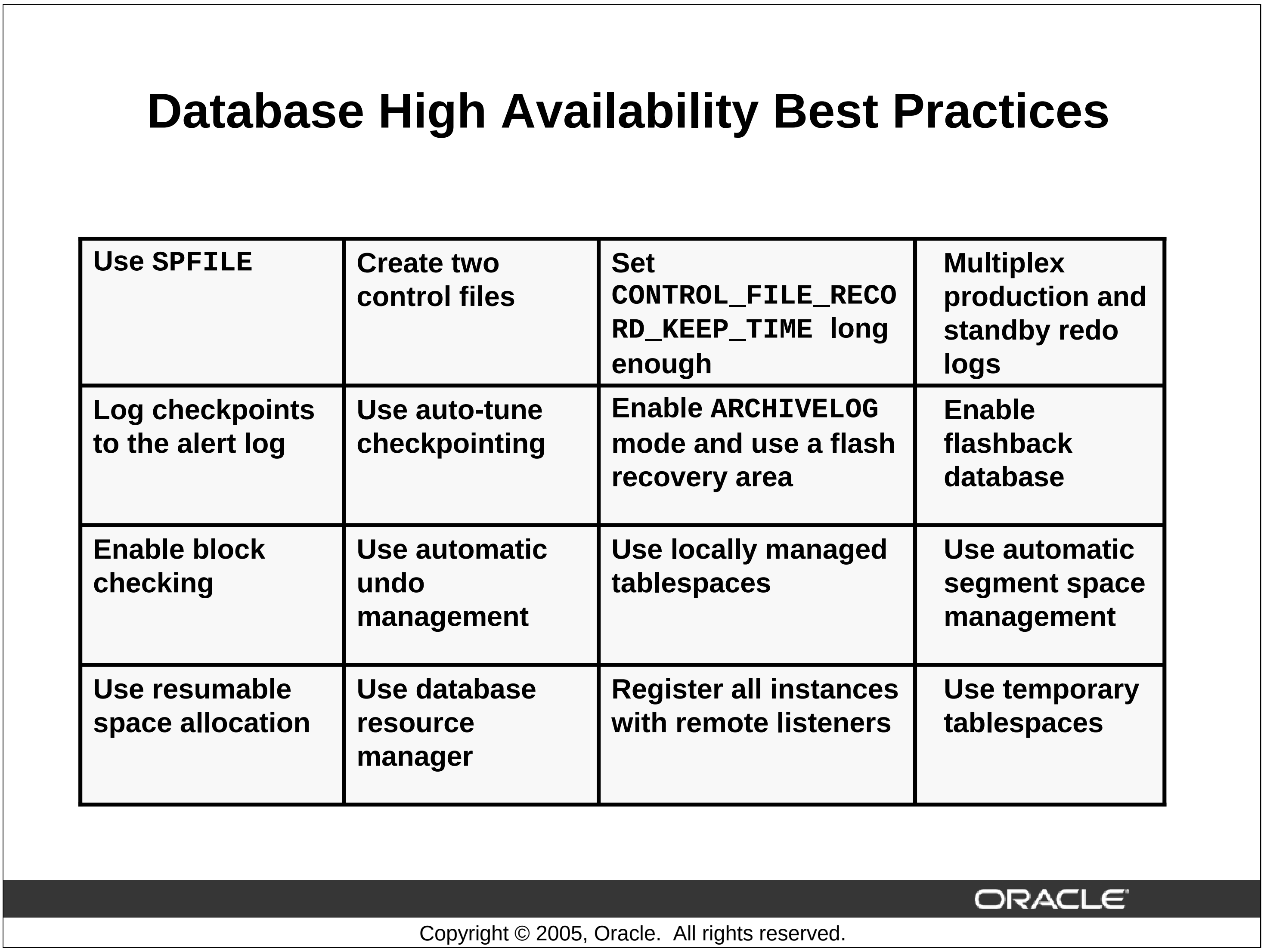
The first step in the slide, shows the Data Guard configuration before the upgrade begins, with the primary and logical standby databases both running the same Oracle software version.

At step two, you stop SQL Apply and upgrade the Oracle database software on the logical standby database to version n+1. During the upgrade, redo data accumulates on the primary system.

With step three, you restart SQL Apply and the redo data that was accumulating on the primary system is automatically transmitted and applied on the newly upgraded logical standby database. The Data Guard configuration can run the mixed versions for an arbitrary period.

In the last step, you perform a switchover. Then, activate the user applications and services on the new primary database. Before you can enable again SQL Apply, you need to upgrade the new standby site. This is because the new standby site does not understand new redo information. Finally, raise the compatibility level on each database.

Note: SQL Apply does not support all data types. So, this can prevent you to use this method.



Database High Availability Best Practices

Use `SPFILE`	Create two control files	Set `CONTROL_FILE_RECORD_KEEP_TIME` long enough	Multiplex production and standby redo logs
Log checkpoints to the alert log	Use auto-tune checkpointing	Enable `ARCHIVELOG` mode and use a flash recovery area	Enable flashback database
Enable block checking	Use automatic undo management	Use locally managed tablespaces	Use automatic segment space management
Use resumable space allocation	Use database resource manager	Register all instances with remote listeners	Use temporary tablespaces

Database High Availability Best Practices

The above table gives you a short summary of the recommended practices that apply to single-instance databases, RAC databases, and Data Guard standby databases.

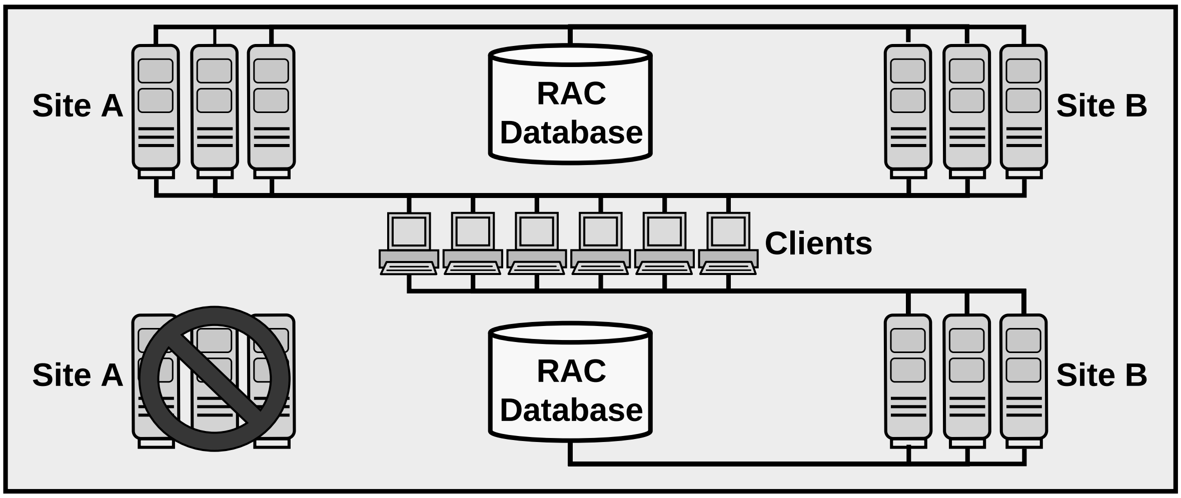
These practices affect the performance, availability, and MTTR of your system. Some of these practices may reduce performance, but they are necessary to reduce or avoid outages. The minimal performance impact is outweighed by the reduced risk of corruption or the performance improvement for recovery.

Note: For more information on how to set up the above features, refer to the following documents:

- *Administrator's Guide*
- *Data Guard Concepts and Administration*
- *Net Services Administrator’s Guide*

Extended RAC: Overview

- Full utilization of resources, no matter where they are located



- Fast recovery from site failure

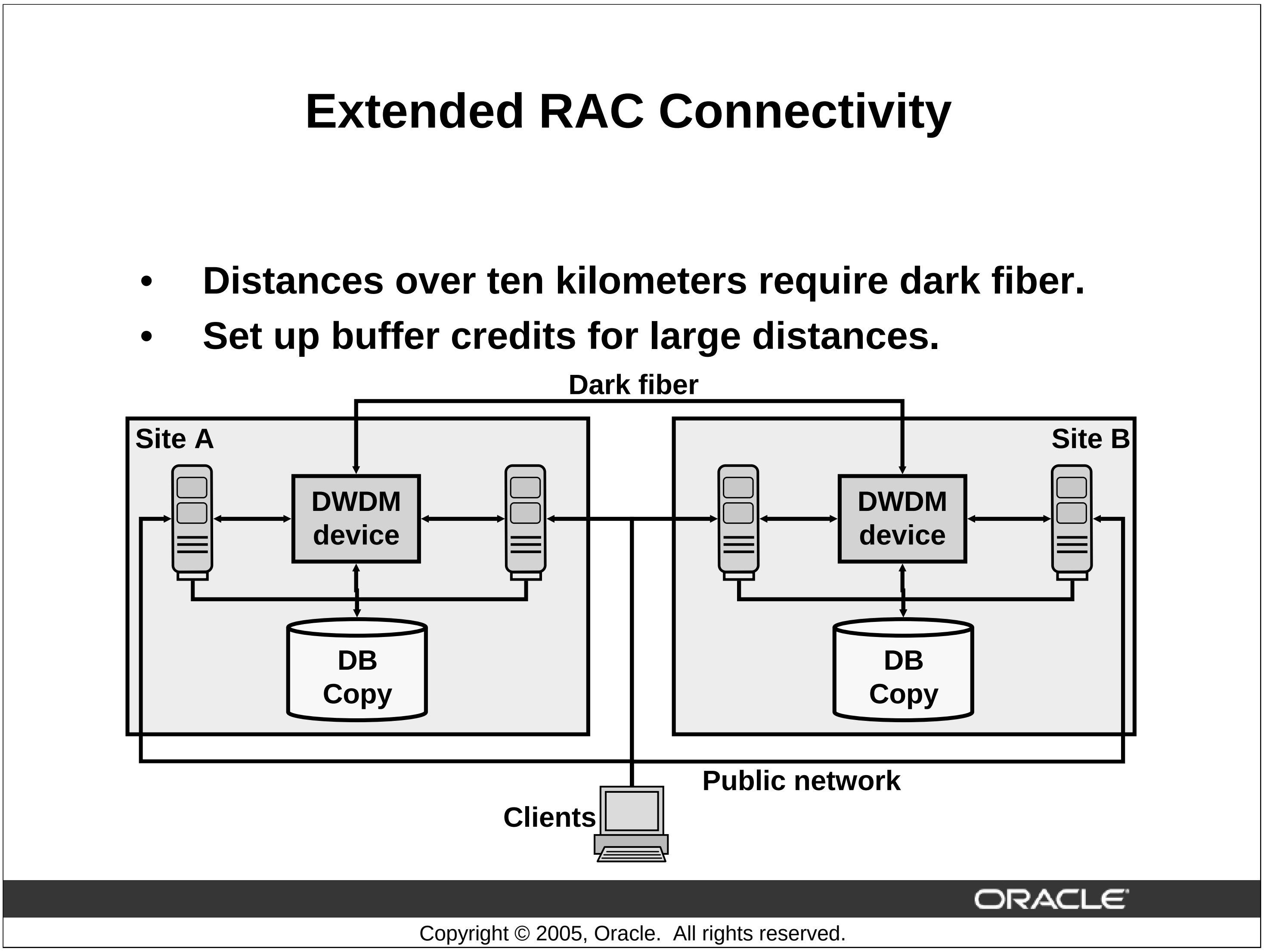
Extended RAC: Overview

Typically, RAC databases share a single set of storage and are located on servers in the same data center.

With extended RAC, you can use disk mirroring and Dense Wavelength Division Multiplexing (DWDM) equipment to extend the reach of the cluster. This configuration allows two data centers, separated by up to 100 kilometers, to share the same RAC database with multiple RAC instances spread across the two sites.

As shown in the slide above, this RAC topology is very interesting, because the clients' work gets distributed automatically across all nodes independently of their location, and in case one site goes down, the clients' work continues to be executed on the remaining site. The types of failures that extended RAC can cover are mainly failures of an entire data center due to a limited geographic disaster. Fire, flooding, and site power failure are just a few examples of limited geographic disasters that can result in the failure of an entire data center.

Note: Extended RAC does not use special software other than the normal RAC installation.



Extended RAC Connectivity

- Distances over ten kilometers require dark fiber.
- Set up buffer credits for large distances.

Extended RAC Connectivity

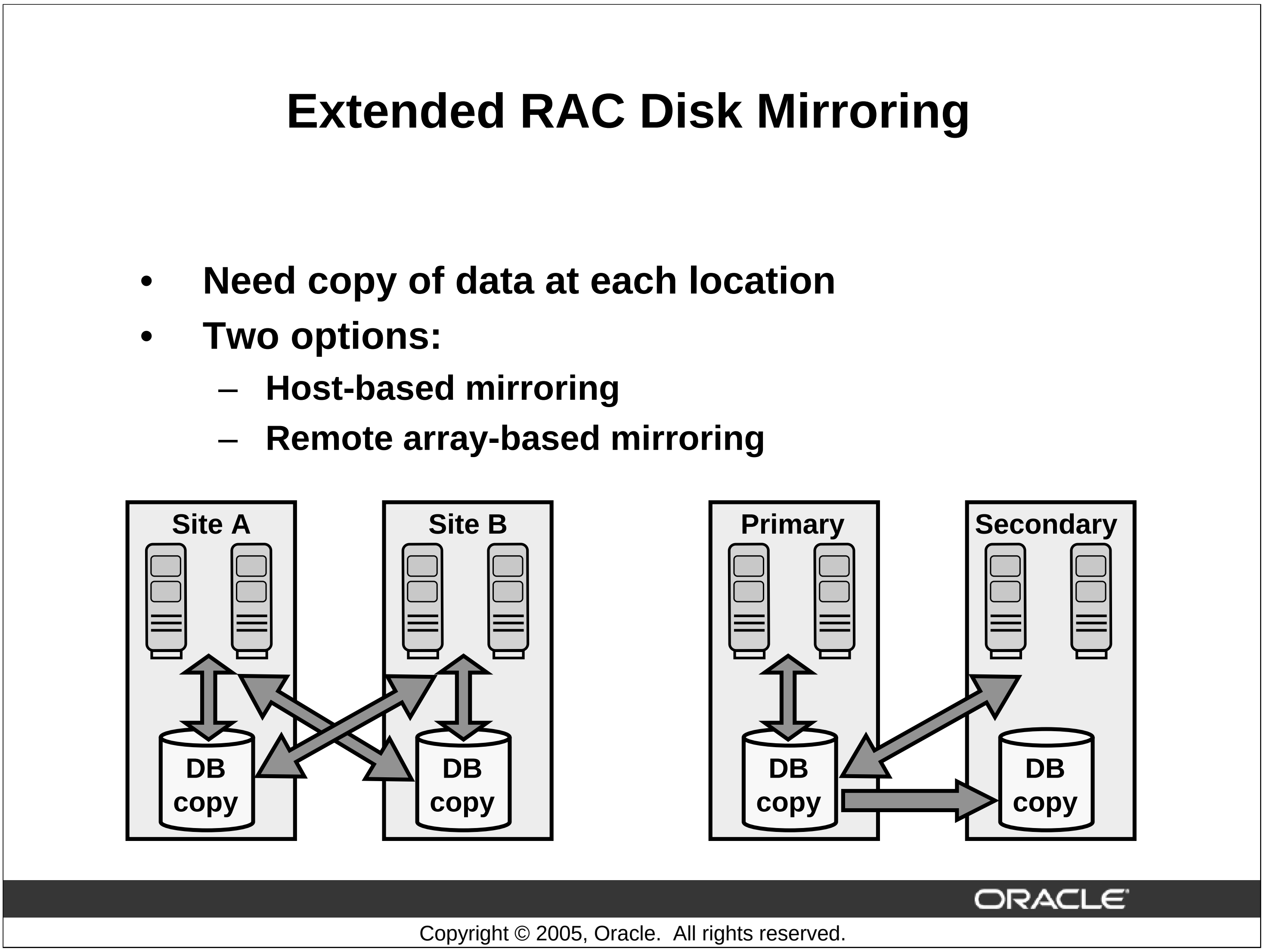
In order to extend a RAC cluster to another site separated from your data center by more than ten kilometers, it is required to use DWDM over dark fiber to get good performance results.

DWDM is a technology that uses multiple lasers, and transmits several wavelengths of light simultaneously over a single optical fiber. DWDM enables the existing infrastructure of a single fiber cable to be dramatically increased. DWDM systems can support more than 150 wavelengths, each carrying up to 10Gbps. Such systems provide more than a terabit per second of data transmission on one optical strand that is thinner than a human hair.

As shown in the slide above, each site should have its own DWDM device connected together by a dark fiber optical strand. All traffic between the two sites is sent through the DWDM and carried on dark fiber. This includes mirrored disk writes, network and heartbeat traffic, and memory-to-memory data passage. Also shown on the graphic are the set of disks at each site. Each site maintains a copy of the RAC database.

It is important to note that depending on the site’s distance, you should tune and determine the minimum value of buffer credits in order to maintain the maximum link bandwidth. Buffer credit is a mechanism defined by the Fiber Channel standard that establishes the maximum amount of data that can be sent at any one time.

Note: Dark fiber is a single fiber optic cable or strand mainly sold by telecom providers.



Extended RAC Disk Mirroring

- Need copy of data at each location
- Two options:
  – Host-based mirroring
  – Remote array-based mirroring

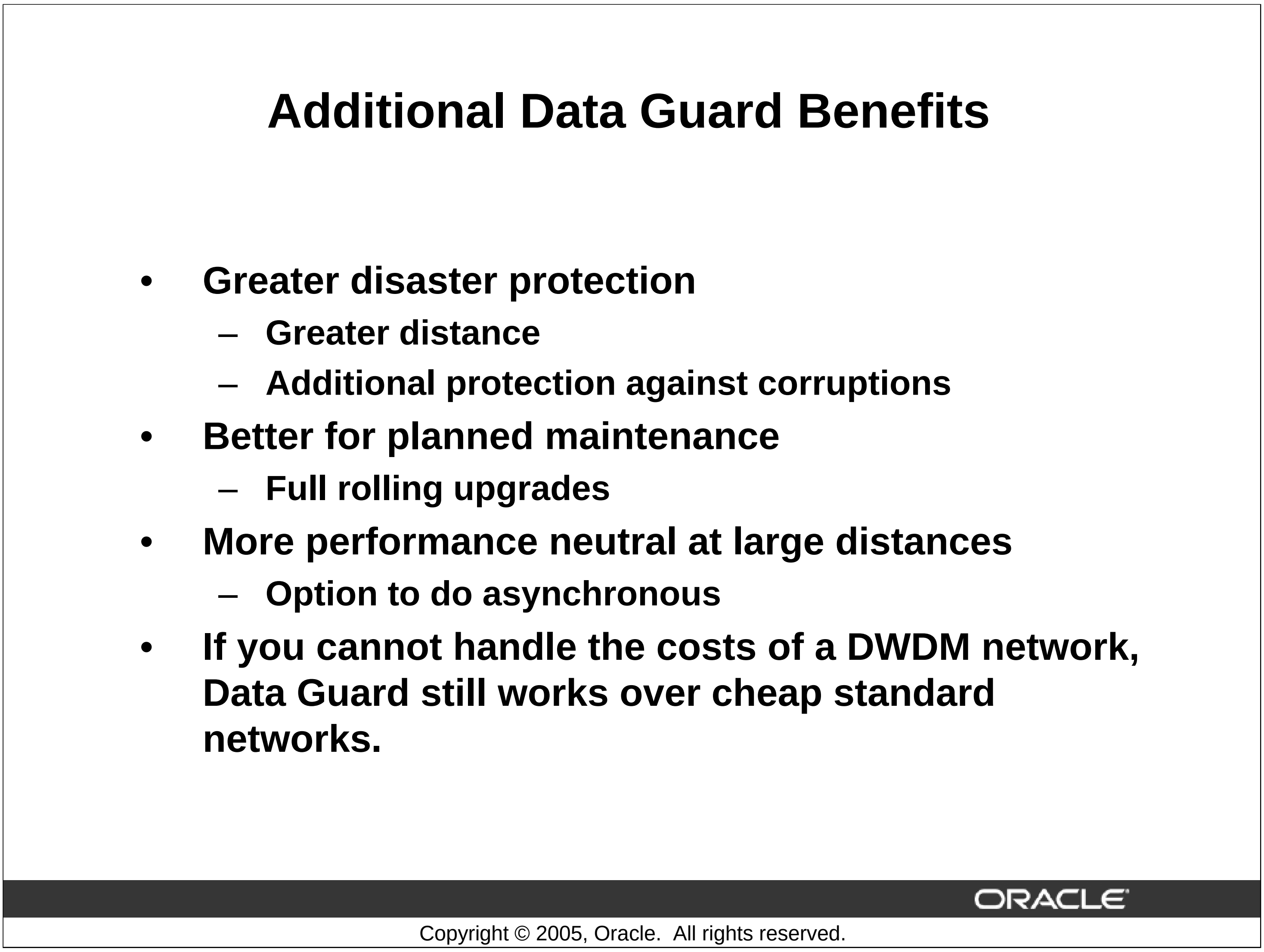
Extended RAC Disk Mirroring

Although there is only one RAC database, each data center has its own set of storage which is synchronously mirrored using either a cluster-aware host-based Logical Volume Manager (LVM) solution, such as SLVM with MirrorDiskUX, or an array-based mirroring solution, such as EMC SRDF.

With host-based mirroring, shown to the left of the slide, the disks appear as one set, and all I/Os get sent to both sets of disks. This solution requires closely integrated clusterware and LVM, which does not exist with the Oracle Database 10*g* clusterware.

With array-based mirroring, shown to the right, all I/Os are sent to one site, and are then mirrored to the other. This alternative is the only option if you only have the Oracle Database 10*g* clusterware. In fact, this solution is like a primary/secondary site setup. If the primary site fails, all access to primary disks is lost. An outage may be incurred before one can switch to the secondary site.

Note: With extended RAC, designing the cluster in a manner that ensures the cluster can achieve quorum after a site failure is a critical issue. For more information, regarding this topic refer to the *Oracle Technology Network* site.



Additional Data Guard Benefits

- Greater disaster protection
  - Greater distance
  - Additional protection against corruptions
- Better for planned maintenance
  - Full rolling upgrades
- More performance neutral at large distances
  - Option to do asynchronous
- If you cannot handle the costs of a DWDM network, Data Guard still works over cheap standard networks.

Additional Data Guard Benefits

Data Guard provides a greater Disaster Protection:

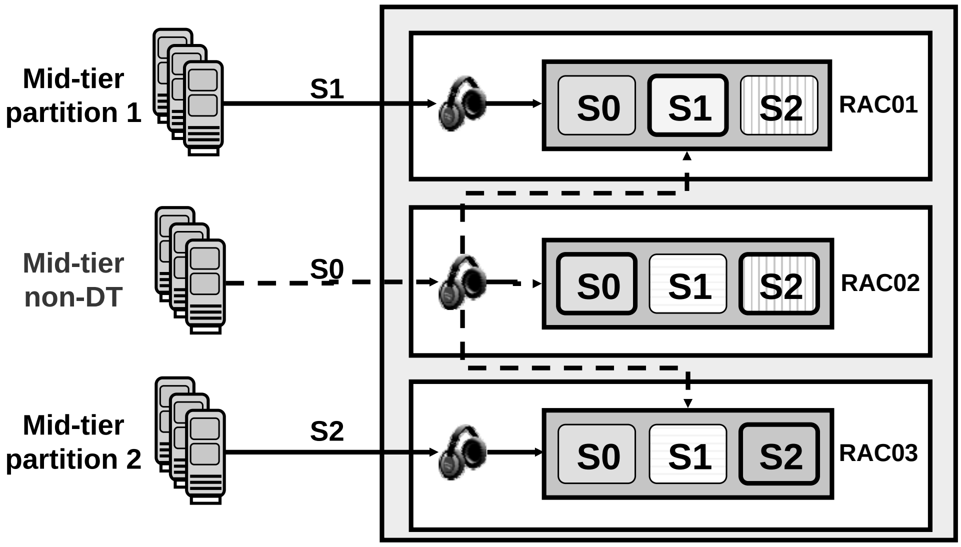
- Distance over 100 kilometers without performance hit
- Additional protection against corruptions because it uses a separate database
- Optional delay to protect against user errors

Data Guard also provides better planned maintenance capabilities by supporting full rolling upgrades.

Also, if you cannot handle the costs of a DWDM network, then Data Guard still works over cheap standard networks.

Using Distributed Transactions with RAC

- **Scope of application: XA or MS DTC**
- **All transaction branches occur on same instance**



Using Distributed Transactions with RAC

When using RAC with distributed transactions (Microsoft Distributed Transaction Coordinator or XA), it is possible for two application components in the same transaction to connect to different nodes on a RAC cluster. This situation can occur on systems with automatic load balancing where the application cannot control which database nodes a distributed transaction branch gets processed. It is important that working in a tightly coupled transaction remain on the same node as separating them may lead to deadlocks or problems with the two-phase commit.

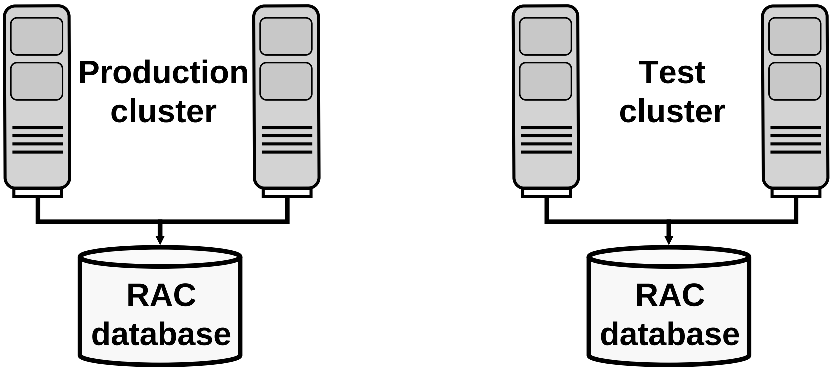
Each distributed transaction's operations must have an affinity to a single database node within a RAC cluster. By using node affinity, it is possible to use RAC reliably with distributed transactions.

The above graphic presents a possible solution. Assume you have three RAC nodes, `RAC01`, `RAC02`, and `RAC03`, where each one is capable of servicing any nondistributed transaction coming from a middle-tier. For distributed transactions from other middle tiers, they are partitioned statically via Oracle Net aliases across one of these three nodes. Thus, each node publishes itself as an `S0` service for nondistributed transactions. In addition, `RAC01` and `RAC02` publish themselves as a singleton service: `S1`, and `S2` respectively.

Each mid-tier client has an address list in its Oracle Net alias that assigns common distributed transaction branches to the same RAC node. In case one of these database nodes fails, CRS starts the corresponding service on one of the available instances.

Using a Test Environment

- The most common cause of down time is change.
- Test your changes on a separate test cluster before changing your production environment.



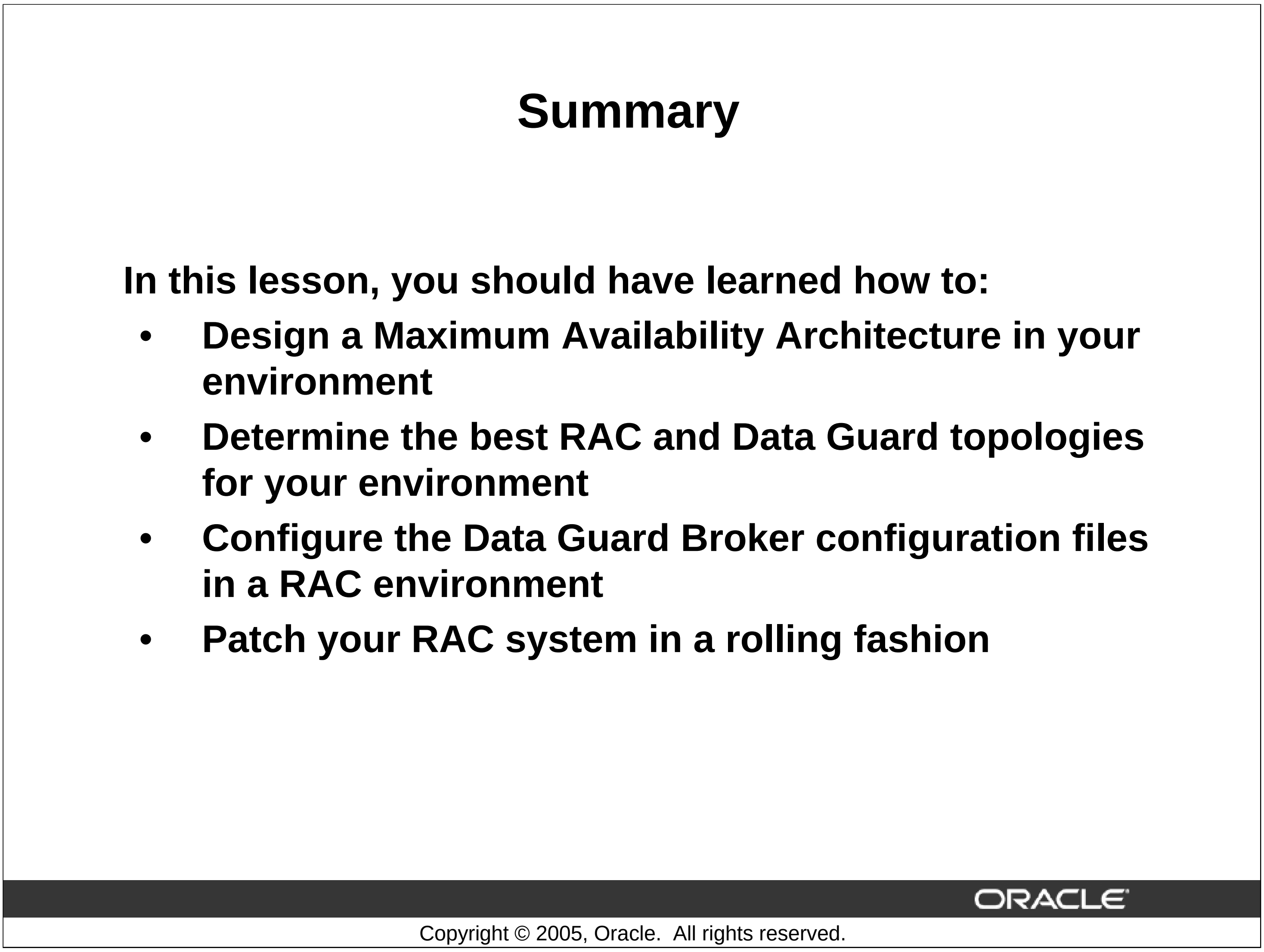
Using a Test Environment

Change is the most likely cause of down time in a production environment. A proper test environment can catch more than 90 percent of the changes that could lead to a down time of the production environment, and is invaluable for quick test and resolution of issues in production.

When your production is RAC, your test environment should be a separate RAC cluster with all the identical software components and versions.

Without a test cluster, your production environment will not be highly available.

Note: Not using a test environment is one of the most common errors seen by Oracle Support Services.



Summary

In this lesson, you should have learned how to:

- Design a Maximum Availability Architecture in your environment
- Determine the best RAC and Data Guard topologies for your environment
- Configure the Data Guard Broker configuration files in a RAC environment
- Patch your RAC system in a rolling fashion