Oracle® Real Application Clusters

Deployment and Performance Guide

10*g* Release 1 (10.1)

Part No. B10768-02

June 2004

ORACLE®

Oracle Real Application Clusters Deployment and Performance Guide 10*g* Release 1 (10.1)

Part No. B10768-02

Primary Authors: David Austin, Mark Bauer

Contributing Authors: Carol Colrain, Javier Seen

Contents

3 Monitoring Performance

4 Monitoring Performance with Oracle Enterprise Manager

5 Application-Specific Deployment Topics

A Services Deployment Example

Index

Send Us Your Comments

Oracle Real Application Clusters Deployment and Performance Guide 10*g* Release 1 (10.1)

Part No. B10768-02

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: infodev_us@oracle.com
- FAX: (650) 506-7227. Attn: Server Technologies Documentation Manager
- Postal service:

  Oracle Corporation
  Server Technologies Documentation Manager
  500 Oracle Parkway, Mailstop 4op11
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

The *Oracle Real Application Clusters Deployment and Performance Guide* explains the deployment considerations for implementing applications on Oracle Real Application Clusters (RAC) 10*g* databases. This manual also provides post-deployment information about monitoring RAC database performance. This preface contains the following topics:

Intended Audience

The *Oracle Real Application Clusters Deployment and Performance Guide* is for database administrators who perform the following tasks:

- Plan and deploy applications on RAC databases
- Monitor the performance of RAC databases

All single-instance Oracle database deployment and performance methodologies apply to RAC. Therefore, you should be familiar with the information in *Oracle Database Performance Tuning Guide*, and *Oracle Data Warehousing Guide*.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Structure

Chapter 1, "Introduction to Deployment and Performance"

This chapter explains the main considerations for deploying applications on RAC databases and for taking advantage of the high availability performance features of RAC.

Chapter 2, "Design and Deployment Techniques"

This chapter describes database deployment techniques for RAC environments that are in addition to those required for single-instance Oracle database deployments.

Chapter 3, "Monitoring Performance"

This chapter provides a few tips about how to monitor RAC performance.

Chapter 4, "Monitoring Performance with Oracle Enterprise Manager"

This chapter presents the RAC-specific Oracle Enterprise Manager performance monitoring features.

Chapter 5, "Application-Specific Deployment Topics"

This chapter provides a few guidelines for the deployment of online transaction processing (OLTP), data warehouse, and general purpose or hybrid applications in RAC environments.

Appendix A, "Services Deployment Example"

This appendix contains an example of configuring services for high availability and workload management.

Related Documents

For more information, see these Oracle resources:

- *Oracle Real Application Clusters Installation and Configuration Guide*
- *Oracle Real Application Clusters Quick Installation Guide for Oracle Database Standard Edition for Windows*
- *Oracle Real Application Clusters Administrator's Guide*
- *Oracle Enterprise Manager Concepts*
- *Oracle Database Performance Tuning Guide*
- *Oracle Data Warehousing Guide*

- *Oracle Database Concepts*
- *Oracle Database Net Services Administrator's Guide*
- *Oracle Database New Features*
- *Oracle Database Reference*
- *Oracle Database Platform Guide for Windows*

Note that the following documents are on the Oracle product CD-ROM:

- *Oracle Real Application Clusters Installation and Configuration Guide*
- *Oracle Real Application Clusters Quick Installation Guide for Oracle Database Standard Edition for Windows*
- *Oracle Enterprise Manager Grid Control Installation and Basic Configuration*
- *Oracle Database Administrator's Reference 10g Release 1 (10.1) for UNIX Systems: AIX-Based Systems, HP-UX, hp Tru64 UNIX, Linux, and the Solaris Operating System (SPARC)*

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://otn.oracle.com/membership/`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://otn.oracle.com/documentation/`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.

Convention	Meaning	Example
`UPPERCASE monospace (fixed-width) font`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.
`lowercase monospace (fixed-width) font`	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to start SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run *`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
`[ ]`	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
`{ }`	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`
`|`	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`
`...`	Horizontal ellipsis points indicate either: ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code	`CREATE TABLE ... AS `*`subquery`*`;` `SELECT `*`col1`*`, `*`col2`*`, ... , `*`coln`*` FROM employees;`

Convention	Meaning	Example
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	SQL> SELECT NAME FROM V$DATAFILE; NAME ---------- /fsl/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*
UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;
lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;

What's New in Deployment and Performance?

The following topic describes the new features for Oracle Real Application Clusters (RAC) in deployment and performance for this release:

- Oracle Database 10g New Features in RAC Deployment and Performance

  See Also: *Oracle Database New Features* for a complete description of the new features in Oracle Database 10*g*

Oracle Database 10*g* New Features in RAC Deployment and Performance

- Services and Automatic Workload Management

  Application workloads can be defined as services so that they can be individually managed and controlled. You can create a service for each separate application or for major components within a complex application. Once created, you can define where and when the service runs. Your entire database workload can be separated into a few services, each of which can be managed independently, reducing your need to manage individual users or sessions for many tasks.

- Cluster Ready Services

  Oracle Real Application Clusters (RAC) 10g introduces a complete, integrated clusterware management solution on all Oracle Database 10g platforms. This clusterware is called Cluster Ready Services (CRS) and replaces third party clusterware on most platforms.

  CRS also provides a platform for services on RAC and you can use services to maximize the value of your cluster's processing resources. Each service can be assigned to one or more instances for normal startup (preferred), depending on its processing requirements. Additionally, you can define one or more alternate (available) instances that a service can use should one of its assigned (preferred) instances become unavailable.

  See Also: Chapter 1, "Introduction to Deployment and Performance" for more information about CRS and services in RAC 10*g*

- Oracle Enterprise Manager Enhancements for RAC Environments

  The Oracle Enterprise Manager Database Control is installed and configured by the Database Configuration Assistant (DBCA) to enable you to manage your RAC

database with its instance targets, listener targets, host targets, and cluster target. You can also install Enterprise Manager onto other machines either inside or outside your cluster and use Oracle Enterprise Manager Grid Control. Enterprise Manager Grid Control enables you to manage multiple RAC databases and multiple cluster targets.

See Also: Chapter 4, "Monitoring Performance with Oracle Enterprise Manager" for more information

1

Introduction to Deployment and Performance

This chapter introduces Oracle Real Application Clusters (RAC) application deployment and performance by explaining the main points to remember when you deploy applications on RAC. This chapter includes the following topics:

- Real Application Clusters Documentation Overview
- Overview of Deploying Applications on Real Application Clusters
- Implementing Oracle Features with Real Application Clusters

Real Application Clusters Documentation Overview

This section describes the RAC documentation set. This book, the *Oracle Real Application Clusters Deployment and Performance Guide*, highlights the main deployment topics for RAC by briefly describing Cluster Ready Services (CRS), storage, database creation, and services deployment in RAC. Design and deployment topics in this book describe service topologies and workload management in RAC. Specifically, this book describes how the Automatic Workload Repository tracks and reports service levels and how you can use service level thresholds and alerts to improve high availability in your RAC environment. There is also a services deployment example in the appendix of this book that you can use to learn more about how to deploy and manage services in RAC environments.

The *Oracle Real Application Clusters Deployment and Performance Guide* provides information about how to monitor and tune performance in RAC environments using both Oracle Enterprise Manager and using information in the Automated Workload Repository and Oracle performance views. This book also highlights some application-specific deployment techniques for online transaction processing and data warehousing environments. In addition to this book, the *Oracle Real Application Clusters Administrator's Guide* is on the Server Documentation CD and the *Oracle Real Application Clusters Installation and Configuration Guide* is on your platform CD as described under the following headings:

- Oracle Real Application Clusters Administrator's Guide
- Oracle Real Application Clusters Installation and Configuration Guide and Oracle Real Application Clusters Quick Installation Guide for Oracle Database Standard Edition for Windows

Oracle Real Application Clusters Administrator's Guide

The *Oracle Real Application Clusters Administrator's Guide* provides RAC-specific administration information. Some of the topics described in that book include the use of Oracle Enterprise Manager in RAC environments. The book also describes how to

administer services and storage, and how to use RAC scalability features to add and delete instances and nodes in RAC environments. The Oracle Real Application Clusters Administrator's Guide also discusses how to use Recovery Manager (RMAN), and how to perform backup and recovery in RAC.

The *Oracle Real Application Clusters Administrator's Guide* also describes how to use the Server Control (SRVCTL) utility to start and stop the database and instances, manage configuration information, and to delete or move instances and services. You can also use the appendix to resolve various RAC tools error and informational messages. A troubleshooting section describes how to interpret the content of various RAC-specific log files.

Oracle Real Application Clusters Installation and Configuration Guide and Oracle Real Application Clusters Quick Installation Guide for Oracle Database Standard Edition for Windows

The platform-specific Oracle Database 10*g* CD contains a copy of the *Oracle Real Application Clusters Installation and Configuration Guide* in both HTML and PDF formats. That book contains the pre-installation, installation, and post-installation information for all UNIX- and Windows-based platforms on which RAC operates. If you are installing Oracle Database 10*g* Standard Edition with RAC on a Windows-based system, refer to the *Oracle Real Application Clusters Quick Installation Guide for Oracle Database Standard Edition for Windows*.

Note: Additional information for this release may be available in the Oracle Database 10*g* README or Release Notes.

Overview of Deploying Applications on Real Application Clusters

To optimally deploy applications on RAC, remember the following few points:

- Storage for RAC datafiles must be shared storage—When you install RAC, use a Cluster File System for datafile storage when available.
- Create your database with the Database Configuration Assistant (DBCA).
- Define services for your environment with the DBCA and administer them with Oracle Enterprise Manager and the Server Control (SRVCTL) Utility.
- Use the Server Parameter File (SPFILE)—The SPFILE should be located on either a cluster file system file or on a shared raw device.
- Use Automatic Undo Management.
- Use Automatic Segment-Space Management.
- Automatic Database Diagnostic Monitor (ADDM) to reduce the effort required to tune Oracle systems.

See Also: *Oracle Real Application Clusters Installation and Configuration Guide* for more information about configuring these features for Oracle Real Application Clusters 10*g*

Implementing Oracle Features with Real Application Clusters

The Oracle features described in this section enhance the performance of your RAC environment. The features discussed in this section are:

Cluster File Systems in Real Application Clusters

Depending on your hardware platform, you can store Oracle homes and Oracle datafiles on a cluster file system. Cluster file systems are simpler to configure and manage than raw device storage. Cluster file systems also offer scalable, low latency, highly resilient storage that significantly reduces costs.

Storage Management Features and Real Application Clusters

The advanced storage features of Oracle Automatic Storage Management greatly enhance manageability for RAC just as with single instance Oracle. Other storage features include Oracle-managed files, automatic segment-space management, and automatic undo management. Refer to the Oracle database documentation for more information about using storage management features.

See Also:

- *Oracle Database 2 Day DBA*
- *Oracle Database Administrator's Guide*
- *Oracle Database Application Developer's Guide - Object-Relational Features*

Services in Oracle Database 10*g*

With Oracle Database 10*g*, application workloads can be defined as services so that they can be individually managed and controlled. You can create a service for each separate application or for major components within a complex application. Once created, you can define where and when the service runs. Your entire database workload can be separated into a few services, each of which can be managed independently, reducing your need to manage individual users or sessions for many tasks.

In a RAC database, you can use services to maximize the value of your cluster's processing resources. Each service can be assigned to one or more instances for normal startup (preferred), depending on its processing requirements. Additionally, you can define one or more alternate (available) instances that a service can use should one of its assigned (preferred) instances become unavailable.

On both cluster and non-cluster environments, performance metrics can be tracked by service using the Automatic Workload Repository (AWR). Thresholds on performance metrics can be set to automatically generate alerts should these thresholds be crossed. Services can be mapped to Resource Manager consumer groups to provide more fine-grained resource allocation controls such as placing limits on CPU consumption. Other Oracle tools and facilities such as Job Scheduler, Parallel Query, and Oracle Streams Advanced Queuing can also use services to manage their workloads.

Cluster Ready Services and High Availability in Real Application Clusters

This section introduces the following high availability features:

- Cluster Ready Services
- Cluster Ready Services and High Availability

Cluster Ready Services

Oracle Real Application Clusters 10*g* introduces a complete, integrated clusterware management solution on all Oracle Database 10*g* platforms. This clusterware functionality provides all the features required to manage your cluster database including node membership, group services, global resource management, and high availability functions.

The clusterware facility is called Cluster Ready Services (CRS) and you install it as part of the RAC installation process. Oracle database features such as Oracle 10*g* services use the underlying CRS mechanisms to provide their capabilities. Oracle also continues to support select third-party clusterware products on specified platforms.

Cluster Ready Services and High Availability

High availability configurations have redundant hardware and software that maintain operations by avoiding single points-of-failure. When outages occur, CRS relocates the processing performed by the inoperative component to a backup component. Oracle's recovery processes quickly re-master resources, recover partial or failed transactions, and rapidly restore the system.

You can combine many Oracle products and features to create highly reliable computing environments. Doing this requires capacity and redundancy planning. In addition, consider your overall system costs and your return on investment. There are also other practical considerations such as selecting the appropriate hardware and deciding whether to use idle machines that are part of your high availability configuration.

Additional Oracle High Availability Features and Solutions

This section describes the following additional high availability solutions:

- Connection Load Balancing in Real Application Clusters
- Recovery Manager (RMAN) in Real Application Clusters
- Data Guard
- Primary/Secondary Instance Configurations in Earlier Releases

Connection Load Balancing in Real Application Clusters

The connection load balancing feature automatically distributes connections among active instances. Connection load balancing does this based on the workload of each node and instance in a cluster. RAC and Cache Fusion combined with connection load balancing supports all types of applications without application or data partitioning.

Recovery Manager (RMAN) in Real Application Clusters

Recovery Manager (RMAN) is an Oracle tool that you can use to backup, copy, restore, and recover datafiles, control files, SPFILEs, and archived redo logs. You can invoke RMAN as a command line utility or use in Oracle Enterprise Manager.

A best practice is to configure RMAN so that all instances can access all the archive log threads throughout your cluster database. In the event of media recovery, the recovering instance requires access to all of the archived redo log threads. Therefore, simplify media recovery administration by ensuring that a recovering instance can access a local copy of the archive log threads from all of the instances in your cluster database.

> **See Also:** *Oracle Real Application Clusters Administrator's Guide* for details about configuring RMAN for use with RAC and *Oracle Database Backup and Recovery Advanced User's Guide* for detailed information about RMAN

Data Guard

Oracle Data Guard works with standby databases to protect your data against errors, failures, and corruptions that might otherwise destroy your database. Data Guard protects critical data by automating the creation, management, and monitoring aspects of standby database environments. Oracle Data Guard automates the otherwise manual process of maintaining a transactional consistent copy of an Oracle database to recover from the loss of or damage to the production database.

Primary/Secondary Instance Configurations in Earlier Releases

If you are upgrading from a pre-Oracle 10*g* Primary/Secondary configuration, then the Database Upgrade Assistant (DBUA) creates a service on your database with one preferred instance and one available instance.

2

Design and Deployment Techniques

This chapter briefly describes database design and deployment techniques for Oracle Real Application Clusters (RAC) environments. It also describes general high availability topics such as deploying services and how Cluster Ready Services (CRS) manages services within RAC. The topics in this chapter are:

- Service Configuration Recommendations for High Availability
- General Database Deployment Topics for Real Application Clusters

Service Configuration Recommendations for High Availability

This section describes the following high availability service configuration recommendations:

- Service Topologies and Managing Workloads in Real Application Clusters Environments
- Recommended Real Application Clusters Service Configurations
- Automatic Workload Repository
- Setting Service Levels and Thresholds

Service Topologies and Managing Workloads in Real Application Clusters Environments

Services are the basis for workload management in RAC. Clients and mid-tier applications make connection requests by specifying a global service name. Because RAC can reallocate services among instances in response to planned and unplanned outages, services greatly extend the availability and scalability of RAC environments.

> **See Also:** *Oracle Enterprise Manager Concepts* for more information about administering services with Enterprise Manager

Recommended Real Application Clusters Service Configurations

The recommended service configuration is to uniformly distribute service assignments across all available nodes. This simplifies your configuration and provides optimal high availability. Another approach is to non-uniformly configure services. In other words, workload sharing configurations can resemble many different topologies.

For example, assume that you have a five-node cluster with two instances, A and B, serving as the preferred instances for CRM. This same cluster could have instances C, D, and E as the preferred instances for AP. Instances A and B are the available

instances for AP if one or more of AP's preferred instances become unavailable. Instances C, D, and E are the available instances for CRM if one or more of the CRM preferred instances becomes unavailable.

This configuration enables each service to use a group of instances that acts as both the preferred instances and as the available recovery instances. After an outage, a client recovers its connections on another instance in the same group.

In this configuration, during normal operations RAC routes application sessions by service to separate groups of instances. If a preferred instance becomes unavailable, then CRS relocates connections among the remaining RAC instances that offer that service.

Workload managed configurations achieve the highest availability and performance by transparently maintaining affinity based on service. Planned and unplanned outages on one domain can be isolated from other domains and the affected service is recovered or upgraded in isolation.

Automatic Workload Repository

The Automatic Workload Repository tracks service level statistics as metrics. Server generated alerts can be placed on these metrics when they exceed or fail to meet certain thresholds. You can then respond, for example, by changing the priority of a job, stopping overloaded processes, or by modifying a service level requirement. This enables you to maintain continued service availability despite service level changes. You can configure service levels to have priorities relative to other services, and you can also configure:

- The measurement of service quality
- Event notification and alert mechanisms to monitor service quality changes
- Recovery scenarios for responses to service quality changes

The Automatic Workload Repository ensures that the CRS workload management framework and resource manager have persistent and global representations of performance data. This information helps Oracle schedule job classes by service and to assign priorities to consumer groups. If necessary, you can rebalance workloads manually with the `DBMS_SERVICE.disconnect_session_by_service_name` PL/SQL procedure. You can use this procedure to disconnect a series of sessions and leave the service running.

> **See Also:** *Oracle Database Performance Tuning Guide* for details about the Automatic Workload Repository and*PL/SQL Packages and Types Reference* for details about Oracle packages

Setting Service Levels and Thresholds

Enterprise Manager and local listeners subscribe to events that indicate changes in service levels. You can set service level metric thresholds with either Enterprise Manager or with Oracle-supplied packages.

You can see historical values for metrics in the `V$SERVICEMETRIC_HISTORY` view. Information about a service from the application level is available in the `V$SESSION` and `V$SQL` views. Service levels, or thresholds, are the baseline operational levels, and events indicate violations of these baselines. You can also examine `GV$SVCMETRIC` for timings such as resource consumption. Use the `V$ACTIVE_SERVICES` and `GV$ACTIVE_SERVICES` views to identify which services are running on which instances.

See Also: *Oracle Real Application Clusters Administrator's Guide* for more information about how to configure services in RAC environments

How Cluster Ready Services Manages Service Relocation

When an instance goes offline due to a planned outage or failure CRS relocates the service to another available instances. CRS relocates the service and re-establishes the connection without service interruption. This occurs as long as the underlying service components on which the relocation relies are enabled for relocation and restart.

General Database Deployment Topics for Real Application Clusters

This section describes a few topics that you might consider when deploying databases for RAC. Your RAC database performance will not be compromised if you do not employ these techniques. If you have an effective single-instance design, then your application will run well on a RAC database.

Tablespace Use in Real Application Clusters

In addition to using locally managed tablespaces, you can further simplify space administration by using automatic segment-space management. Automatic segment-space management distributes instance workloads among each instance's subset of blocks for inserts. This improves RAC performance because it minimizes block transfers. To deploy automatic undo management in a RAC environment, each instance must have its own undo tablespace.

Object Creation and Performance in Real Application Clusters

As a general rule, only use DDL statements for maintenance tasks and avoid executing DDL statements during peak system operation periods. In most systems, the amount of new object creation and other DDL statements should be limited. Just as in single-instance Oracle databases, excessive object creation and deletion can increase performance overhead.

Node Addition and Deletion and the SYSAUX Tablespace in Real Application Clusters

If you add nodes to your RAC database environment, then you may need to increase the size of the `SYSAUX` tablespace. Conversely, if you remove nodes from your cluster database, then you may be able to reduce the size of your `SYSAUX` tablespace.

See Also: *Oracle Real Application Clusters Installation and Configuration Guide* for guidelines about sizing the `SYSAUX` tablespace for multiple instances.

Distributed Transactions and Oracle Real Application Clusters

When a transaction starts on an Oracle RAC database instance, all the operations of that transaction must be completed on that instance. This is also true in distributed transaction environments using protocols such as X/Open XA distributed transaction processing or the Microsoft Distributed Transaction Coordinator (DTC).

In all cases, all branches of a distributed transaction running on an Oracle RAC database must be executed on the same instance. Running different branches on different instances can cause deadlocks or problems with the two-phase commit protocol. Connection pool facilities at the application tier that load balance across

multiple connections to an Oracle RAC database must ensure that all of the operations of each transaction execute on only one Oracle RAC database instance.

> **See Also:** *Oracle Database Application Developer's Guide - Fundamentals* for more information about distributed transactions in Real Application Clusters

3

Monitoring Performance

This chapter describes how to monitor Oracle Real Application Clusters (RAC) performance and includes the following topics:

- Overview of Monitoring Real Application Clusters Databases
- Performance Views in Real Application Clusters
- Real Application Clusters Performance Statistics
- Automatic Workload Repository in Real Application Clusters Environments
- Monitoring RAC Statistics and Events

> **See Also:** Chapter 4, "Monitoring Performance with Oracle Enterprise Manager" for information about monitoring RAC performance with Oracle Enterprise Manager

Overview of Monitoring Real Application Clusters Databases

All single-instance Oracle database tuning practices apply to applications running on RAC databases. Therefore, implement the single-instance tuning methodologies described in *Oracle Database Performance Tuning Guide*.

Verifying the Interconnect Settings for Real Application Clusters

The interconnect and internode communication protocols can affect Cache Fusion performance. In addition, the interconnect bandwidth, its latency, and the efficiency of the IPC protocol determine the speed with which Cache Fusion processes block transfers.

Influencing Interconnect Processing

Once your interconnect is operative, you cannot significantly influence its performance. However, you can influence an interconnect protocol's efficiency by adjusting the IPC buffer sizes.

> **See Also:** Your vendor-specific interconnect documentation for more information about adjusting IPC buffer sizes

Although you should rarely need to set the `CLUSTER_INTERCONNECTS` parameter, you can use it to assign a private network IP address or NIC as in the following example:

```
CLUSTER_INTERCONNECTS=10.0.0.1
```

If you are using an operating system-specific vendor IPC protocol, then the trace information may not reveal the IP address. However, Oracle uses the correct network interface based on the use of vendor-specific IPC libraries.

Note: You can also use the `oifcfg` command as described in the ***Oracle Real Application Clusters Administrator's Guide*** to assign private network or private IP addresses.

Performance Views in Real Application Clusters

Each instance has a set of instance-specific views. You can also query global dynamic performance views to retrieve performance information from all of the qualified instances. Global dynamic performance view names are prefixed with *GV$*. A global view contains all columns from its respective instance-specific view as well as the `INST_ID` column. The instance number is also appended to the names of the archived redo log threads to create a unique identifier for each instance's archived redo logs.

See Also: *Oracle Database Reference* for restrictions on GV$ views and complete descriptions of related parameters and views

Creating Real Application Clusters Data Dictionary Views with CATCLUST.SQL

If you did not create your RAC database with the Database Configuration Assistant (DBCA), then you must run the `CATCLUST.SQL` script to create RAC-related views and tables. You must have `SYSDBA` privileges to run this script.

See Also: *Oracle Real Application Clusters Installation and Configuration Guide* for more information about creating your RAC database

Real Application Clusters Performance Statistics

This section provides an overview of the V$ and GV$ views that provide statistics that you can use evaluate block transfers in your cluster. Use these statistics to analyze interconnect block transfer rates as well as the overall performance of your RAC database.

The Content of Real Application Clusters Statistics

RAC-specific statistics appear as message request counters or as timed statistics. Message request counters include statistics showing the number of certain types of block mode conversions. Timed statistics reveal the total or average time waited for read and write I/O for particular types of operations.

Automatic Workload Repository in Real Application Clusters Environments

In RAC environments, each Automatic Workload Repository (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 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.

Monitoring RAC Statistics and Events

This section explains wait events and statistics specific to RAC and how to interpret them when assessing performance data generated by the Automatic Workload Repository, Statspack, or by ad-hoc queries of the dynamic performance views.

> **See Also:** *Oracle Database Performance Tuning Guide* for more information about wait event analysis and the `spdoc.txt` file for details about the Statspack utility.

RAC statistics and events in AWR and Statspack reports

The statistics snapshots generated by AWR and Statspack 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.

Most of the relevant data is summarized on the RAC Statistics Page. This information includes:

- Global cache load profile
- Global cache efficiency percentages—workload characteristics
- Global cache and Enqueue Service (GES)—messaging statistics

Additional RAC-related sections appear later in the report:

- Global enqueue statistics
- Global CR statistics
- Global CURRENT served statistics
- Global cache transfer statistics.

Wait Events for RAC

Analyzing and interpreting 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 which 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 now convey precise information and waiting for global cache blocks or messages is:

- Summarized in a broader category called Cluster Wait Class
- Temporarily represented by a placeholder event which is active while waiting for a block, for example:
  - gc current block request
  - gc cr block request
- Attributed to precise events when the outcome of the request is known, for example:
  - gc current block 3-way
  - gc current block busy

- gc cr block grant 2-way

In summary, the wait events for RAC convey information valuable for performance analysis. They are used in Automatic Database Diagnostic Monitor (ADDM) to enable precise diagnostics of the impact of cache fusion.

Monitoring Performance by Analyzing GCS and GES Statistics

In order to determine the amount of work and cost related to inter-instance messaging and contention, examine block transfer rates, remote requests made by each transaction, the number and time waited for global cache events as described under the following headings:

- Analyzing Cache Fusion Impact in Real Application Clusters
- Analyzing Performance Using GCS and GES Statistics

Analyzing Cache Fusion Impact in Real Application Clusters

The effect of accessing blocks in the global cache and maintaining 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 and processing times imposed by the physical interconnect components, the IPC protocol and the GCS protocol. It is not affected by disk 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 non-clustered instance.

Analyzing Performance Using GCS and GES Statistics

This section describes how to monitor Global Cache Service performance by identifying data blocks and objects which are frequently used ("hot") by all instances. High concurrency on certain blocks may be identified by Global Cache Service wait events and times.

The following wait events indicate that the access to cached data blocks was held up because they were busy either in the remote or the local cache, respectively:

- gc current block busy
- gc current block 2-way busy
- gc current block 3-way busy
- gc cr block 2-way busy
- gc cr block 3-way busy

This means that the blocks were pinned or held up by sessions or delayed by a log write on a remote instance (for example, gc current, cr 2-way busy, or cr 3-way busy), or that a session on the same instance is already accessing a block which is in transition between instances and the current session needs to wait behind it (for example, gc current block busy).

The V$SESSION_WAIT view to identify objects and data blocks with contention. The gc wait events contain the file and block number for a block request in p1 and p2, respectively.

An additional segment statistic, gc buffer busy, has been added to quickly determine the "busy" objects without recourse to the query on V$SESSION_WAIT mentioned earlier.

The AWR infrastructure provides a view of active session history which can also be used to trace recent wait events and their arguments. It is therefore useful for hot block analysis.

Most of the reporting facilities used by AWR and Statspack contain the object statistics and cluster wait class category, so that sampling of the views mentioned earlier is largely unnecessary.

It is advisable to run ADDM on the snapshot data collected by the AWR infrastructure to obtain an overall evaluation of the impact of the global cache. The advisory will also identify the busy objects and SQL highest cluster wait time.

Analyzing Cache Fusion Transfer Impact Using GCS Statistics

This section describes how to monitor Global Cache Service performance by identifying objects read and modified frequently and the service times imposed by the remote access. Waiting for blocks to arrive may constitute a significant portion of the response time, in the same way that reading from disk could increase the block access delays, only that cache fusion transfers in most cases are faster than disk access latencies.

The following wait events indicate that the remotely cached blocks were shipped to the local instance without having been busy, pinned or requiring a log flush:

- gc current block 2-way
- gc current block 3-way
- gc cr block 2-way
- gc cr block 3-way

The object statistics for gc current blocks received and gc cr blocks received enable quick identification of the indexes and tables which are shared by the active instances. As mentioned earlier, creating an ADDM analysis will, in most cases, point you to the SQL statements and database objects that could be impacted by inter-instance contention.

> **Note:** You must run Statspack at level 7 to collect statistics related to block contention and segment block waits.

Any increases in the average wait times for the events mentioned earlier could be caused by the following:

- High load: CPU shortages, long run queues, scheduling delays
- Misconfiguration: using public instead of private interconnect for message and block traffic

If the average wait times are acceptable and no interconnect or load issues can be diagnosed, then the accumulated time waited can usually be attributed to a few SQL statements which need to be tuned to minimize the number of blocks accessed.

The column CLUSTER_WAIT_TIME in V$SQLAREA represents the wait time incurred by individual SQL statements for global cache events and will identify the SQL which may need to be tuned.

Analyzing Response Times Based on Wait Events

Most global cache wait events that show a high total time as reported in the AWR and Statspack reports or in the dynamic performance views are normal and may present themselves as the top database time consumers without actually indicating a problem. This section describes the most important and frequent wait events that you should be aware of when interpreting performance data.

If user response times increases and a high proportion of time waited is for global cache (gc), then the cause should be determined. Most reports include a breakdown of events sorted by percentage of the total time.

It is useful to start with an ADDM report, which would analyze the routinely collected performance statistics with respect to their impact and point to the objects and SQL contributing most to the time waited, and then move on to the more detailed reports produced by AWR and Statspack.

The most important wait events for RAC include various categories, such as:

- Block-oriented
  - gc current block 2-way
  - gc current block 3-way
  - gc cr block 2-way
  - gc cr block 3-way
- Message-oriented
  - gc current grant 2-way
  - gc cr grant 2-way
- Contention-oriented
  - gc current block busy
  - gc cr block busy
  - gc current buffer busy
- Load-oriented
  - gc current block congested
  - gc cr block congested

The block-oriented wait event statistics indicate that a block was received as either the result of a 2-way or a 3-way message, that is, the block was sent from either the resource master requiring 1 message and 1 transfer, or was forwarded to a third node from which it was sent, requiring 2 messages and 1 block transfer.

These events are usually the most frequent in the absence of block contention and the length of the wait is determined by the time it takes on the physical network, the time to process the request in the serving instances and the time it takes for the requesting process to wake up after the block arrives.

The average wait time and the total wait time should be considered when being alerted to performance issues where these particular waits have a high impact.

Usually, either interconnect or load issues or SQL execution against a large shared working set can be found to be the root cause.

The message-oriented wait event statistics indicate that no block was received because it was not cached in any instance. Instead a global grant was given, allowing the requesting instance to read the block from disk or modify it.

If the time consumed by these events is high, then it may be assumed that the frequently executed SQL causes a lot of disk I/O (in the event of the cr grant) or that the workload inserts a lot of data and needs to find and format new blocks frequently (in the event of the current grant).

The contention-oriented wait event statistics indicate that a block was received which was pinned by a session on another node, was deferred because a change had not yet been flushed to disk or because of high concurrency, and therefore could not be shipped immediately. A buffer may also be busy locally when a session has already initiated a cache fusion operation and is waiting for its completion when another session on the same node is trying to read or modify the same data. High service times for blocks exchanged in the global cache may exacerbate the contention, which can be caused by frequent concurrent read and write accesses to the same data.

The load-oriented wait events indicate that a delay in processing has occurred in the GCS, which is usually caused by high load, CPU saturation and would have to be solved by additional CPUs, load-balancing, offloading processing to different times or a new cluster node.For the events mentioned, the wait time encompasses the entire round trip from the time a session starts to wait after initiating a block request until the block arrives.

4

Monitoring Performance with Oracle Enterprise Manager

This chapter explains how to use Enterprise Manager to monitor Oracle Real Application Clusters (RAC) database performance. This topics in this chapter are:

- Overview of Oracle Enterprise Manager for Real Application Clusters
- Enterprise Manager Performance Pages for Real Application Clusters
- Service Relocation and High Availability Events

> **See Also:** *Oracle Enterprise Manager Concepts* for information about Enterprise Manager and the Enterprise Manager online help for more information about using Enterprise Manager

Overview of Oracle Enterprise Manager for Real Application Clusters

Enterprise Manager enables you to view current and past RAC database availability, performance metrics, and response times. Enterprise Manager can display this information at a high level for multiple cluster databases without requiring you to explicitly access each individual database.

Enterprise Manager supports monitoring at all levels of a cluster environment such as the cluster, the cluster database, and the cluster database instances. Enterprise Manager also responds to metrics from across the entire RAC database and publishes alerts when thresholds are exceeded. Enterprise Manager interprets both pre-defined or customized metrics. You can also copy customized metrics from one cluster database instance to another, or from one RAC database to another.

Enterprise Manager also uses data from the Automatic Workload Repository to display performance information and to initiate database alerts. Statistics that are collected by the Automatic Workload Repository are aggregated from all instances in a RAC database and displayed on a summary Enterprise Manager page.

Enterprise Manager provides performance information for:

- Top Sessions—The most time- and resource-consuming sessions, waits, and SQL statements at the cluster database instance level.
- Top Consumers—This includes Top Services, Top Modules, Top Actions by Module and Service, and Top Clients. This enables statistics aggregation, client identification support, and so on.
- Database Locks—Includes client identifiers, service modules, and actions.
- Cluster Cache Coherency—Performance information shows Global Cache Service (GCS) and Global Enqueue Service (GCS) activity

When you install RAC and create your RAC database with the Database Configuration Assistant (DBCA), the Enterprise Manager Database Control tools are pre-authorized to monitor your RAC environment. You can use Enterprise Manager Database Control to manage a single RAC database with its instance targets, listener targets, host targets, and cluster target. You can also install Enterprise Manager Grid Control onto servers either inside or outside your cluster and manage multiple RAC databases and multiple cluster targets.

Enterprise Manager Performance Pages for Real Application Clusters

Enterprise Manager displays both aggregate and instance-specific performance statistics using color-coded histograms. On any Enterprise Manager statistics display, highlight a statistic name on a statistics chart and Enterprise Managers re-displays the chart with colors for easier viewing. To obtain details about a highlighted statistic, click it and Enterprise Manager displays a detail page that provides more information about the statistic.

From the Cluster Database Home page you can access RAC-specific detail pages that show performance trends across one or more cluster databases. The following sections describe these pages in more detail:

- Using the Cluster Performance Page
- Using the Cluster Database Performance Page
- Using the Cluster Cache Coherency Page
- Using the Cluster Cache Coherency Instances Page

Using the Cluster Performance Page

The Cluster Performance Page displays usage statistics for all hosts or for individual hosts. This information enables you to add, suspend, or redistribute resources as needed.

Using the Cluster Database Performance Page

The Cluster Database Performance Page displays charts showing run queue length, paging rate, service time, and database throughput for each host or instance. This page also provides access to detailed information such as the Details for Wait Class Page for Service Time and the Top Sessions Page for Database Throughput.

Using the Cluster Cache Coherency Page

The Cluster Cache Coherency Page displays cache coherency metrics for the entire cluster database. This page groups metrics into the following categories:

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

The Cluster Cache Coherency Page can display these statistics for each instance. This page also displays the Cache Coherency vs. Session Logical Reads chart. This chart graphs the percentage of Global Cache CR Blocks Received, Current Blocks Received, Converts, and Gets against logical reads for the session.

Using the Cluster Cache Coherency Instances Page

The Cluster Cache Coherency Instances Page provides real-time monitoring of global cache statistics. The Cluster Cache Coherency Instances Page displays tables of metrics from the following groups for all cluster instances such as:

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

Service Relocation and High Availability Events

Enterprise Manager monitors events at the database and instance levels, and any available node can monitor database events. However, only one node at a time monitors the entire database while each node monitors events for its local instance.

5

Application-Specific Deployment Topics

This chapter discusses topics for deploying online (OLTP), data warehouse, and general purpose (hybrid) applications in Oracle Real Application Clusters (RAC) environments. The topics in this chapter are:

General Deployment Strategies for Real Application Clusters-Based Applications

All single-instance application development and deployment techniques apply to RAC. If your applications run well on a single-instance Oracle database, then they will run well on a RAC database.

Deploying OLTP Applications in Real Application Clusters

Cache Fusion makes RAC databases the optimal deployment servers for online transaction processing (OLTP) applications. This is because these types of applications require:

- High availability in the event of failures
- Scalability to accommodate increased system demands
- Load balancing according to demand fluctuations

The high availability features of Oracle and RAC can re-distribute and load balance workloads to surviving instances without interrupting processing. RAC also provides excellent scalability so that if you add or replace a node, then Oracle re-masters resources and re-distributes processing loads.

Flexible Implementation with Cache Fusion

To accommodate the frequently changing workloads of online transaction processing systems, RAC remains flexible and dynamic despite changes in system load and system availability. RAC addresses a wide range of service levels that, for example, fluctuate due to:

- Varying user demands
- Peak scalability issues like trading storms (bursts of high volumes of transactions)
- Varying availability of system resources

Deploying Data Warehouse Applications with Real Application Clusters

This section discusses how to deploy data warehouse systems in RAC environments by briefly describing the data warehouse features available in shared disk architectures. The topics in this section are:

- Speed-Up for Data Warehouse Applications on Real Application Clusters
- Parallel Execution in Data Warehouse Systems and RAC
- Using Parallel Instance Groups

Speed-Up for Data Warehouse Applications on Real Application Clusters

RAC is ideal for data warehouse applications because it augments the single instance benefits of Oracle. RAC does this by maximizing the processing available on all of the nodes that belong to a RAC database to provide speed-up for data warehouse systems.

The query optimizer considers parallel execution when determining the optimal execution plans. The default cost model for the query optimizer is **CPU+I/O** and the cost unit is **time**. In RAC, the query optimizer dynamically computes intelligent defaults for parallelism based on the number of processors in the nodes of the cluster. An evaluation of the costs of alternative access paths, table scans versus indexed access, for example, takes into account the degree of parallelism (DOP) available for the operation. This results in Oracle selecting the execution plans that are optimized for your RAC configuration.

Parallel Execution in Data Warehouse Systems and RAC

Oracle's parallel execution feature uses multiple processes to execute SQL statements on one or more CPUs. Parallel execution is available on both single-instance Oracle databases and RAC databases.

RAC takes full advantage of parallel execution by distributing parallel processing across all available instances. The number of processes that can participate in parallel operations depends on the DOP assigned to each table or index.

> **See Also:** *Oracle Database Performance Tuning Guide* for more information about the query optimizer

Using Parallel Instance Groups

You can control the instances that allocate parallel execution server processes with instance groups. To do this, assign each active instance to at least one or more instance groups. Then dynamically control which instances spawn parallel processes by activating a particular group of instances.

Note: An instance can belong to one or more groups. You can enter several instance group names with the `INSTANCE_GROUPS` parameter using a comma as a separator.

A

Services Deployment Example

This appendix provides a services deployment example for a Oracle Real Application Clusters (RAC) database. This appendix includes the following topics:

- Configuration Planning
- Using Services
- Manual Configuration for Workload Management
- Using Callouts for Fast Application Notification
- Configuring JDBC Fast Application Notification

Configuration Planning

This appendix contains an example of configuring and integrating Cluster Ready Services (CRS), a RAC database, and services to provide continuous application support. The basis for this example is a RAC database, called ORADB, running on a four-node cluster with one instance on each node. The instance names are RAC01, RAC02, RAC03, and RAC04. The database supports an application with five major components, ERP, CRM, SELF_SERVICE, which involve online transaction processing (OLTP) processing, and HOT_BATCH, and STD_BATCH, which are batch-oriented.

Configuration planning for high availability (HA) services involves defining which application, or parts of an application, you want to be manage with the services and which service features you want to enable.

Service Planning

To take advantage of all HA service capabilities, your plan needs to identify the service name, the primary user (client, application server, job scheduler, and so on), the preferred instances (where the service will start by default), and the available instances (where the service will run if a preferred instance becomes unavailable). You may also define the service priority (to rank importance of the services when competing for resources), and response time thresholds (to indicate when a service is not performing at its required rate).

This example includes all the options and Table A–1 summarizes the planned configuration for the ORADB database and its services, with columns Preferred Instances and Available Instances containing the HA information and columns Priority and Response Time containing the performance information.

Table A–1 Service Planning Work Sheet

Service	Usage	Preferred Instances	Available Instances	Priority	Response Time (sec) Warning / Critical
ERP	Client service	RAC01, RAC02	RAC03, RAC04	HIGH	0.5 / 0.75
CRM	Client service	RAC03, RAC04	RAC01, RAC02	STANDARD	0.5 / 1.0
SELF_SERVICE	Client service	RAC01, RAC02, RAC03, RAC04	-	STANDARD	1.0 / 1.5
HOT_BATCH	Job scheduler	RAC01	RAC02, RAC03, RAC04	HIGH	1.0 / 1.5
STD_BATCH	Job scheduler	RAC01, RAC02, RAC03, RAC04	-	LOW	3.0 / 5.0

The plan calls for the ERP service to run on instances RAC01 and RAC02 when the cluster and database starts normally. That is, ERP will become available on instances RAC01 and RAC02 as they start up. The other two instances, RAC03 and RAC04, are available for the ERP service should one of its preferred instances fail. So, for example, if RAC01 becomes unavailable, either RAC03 or RAC04 takes over running the ERP service on behalf of the failed instance while RAC02, its remaining preferred instance, continues to run the ERP service. If both RAC01 and RAC02 are disabled, the ERP service runs on RAC03 and RAC04 instead.

The plan for the CRM service is similar to that for ERP but with the instances taking on the opposite roles: RAC03 and RAC04 are instances where the CRM service should start and RAC01 and RAC02 are the instances that take over should one or both of the preferred instances fail.

Both the SELF_SERVICE and STD_BATCH services are planned to start on all four instances whereas the HOT_BATCH service starts only on RAC01. Because they are already assigned to four instances, the plan does not define any available instances for the SELF_SERVICE and STD_BATCH services. The plan allows the HOT_BATCH service to use any of the other instances should RAC01 become unavailable.

Table A-1 also lists the planned priorities and response times for the services. The plan assigns the highest priority to the ERP and HOT_BATCH services, which means they will have precedence over the other services if resources become scarce - for example, if only instances RAC03 and RAC04 are available. In such a case, the service with the lowest priority rating, STD_BATCH, may be terminated and, if necessary, the CRM or SELF_SERVICE services could be flagged for termination. The response times are thresholds for which notifications should be triggered if performance fails to meet the listed values.

Cluster Node and Network Interface Planning

When you have completed your logical configuration, you may want to prepare an interface worksheet to record the cluster node interface names and addresses. Most of your interfaces, particularly the public interfaces, will have equivalent host names and domain names. In cases where names are resolvable to IP addresses, you may have provided these names when using the Oracle Universal Installer (OUI) and the Database Configuration Assistant (DBCA). Similarly, you may have entered the name, if there is one, or the IP address of the private interconnects used for the cluster interconnect interface. As you complete your interface worksheet, you may want to record the names, when they exist, along with the IP addresses.

Your virtual IP addresses (VIPs) should not be fixed to any physical interface on your network and VIPs may or may not have a corresponding name. In NetCA or in tnsnames.ora, you can enter either the names or the IP addresses of the VIPs. For

vendor systems that support cluster aliases, you can replace the list of names or IP addresses with the corresponding cluster alias name or IP address. The cluster alias name for this example is clusalias. To execute some steps shown in this example, you will need to know the subnet mask for all of your VIPs used and location of your CRS home directory. In this example, the subnet mask is 255.255.255.0 for all VIPs and the CRS home directory is /private/oracle/crs, neither of which is included in the following table.

Table A–2 shows the interface worksheet for this example.

Table A–2 Example of a cluster node interface and address worksheet

Public physical node name | IP address | Physical interface name(s)	Public virtual IP name | IP address | Logical interface name(s)	Private interconnect IP address | Physical interface name
clusnode-1 | 139.184.101.201 | hme0 [, hme1]	clusnode-1vip | 139.184.201.1 | hme0:1 [, hme1:1]	172.16.0.1 | qfe0
clusnode-2 | 139.184.101.202 | hme0 [, hme1]	clusnode-2vip | 139.184.201.2 | hme0:1 [, hme1:1]	172.16.0.2 | qfe0
clusnode-3 | 139.184.101.203 | hme0 [, hme1]	clusnode-3vip | 139.184.201.3 | hme0:1 [, hme1:1]	172.16.0.3 | qfe0
clusnode-4 | 139.184.101.204 | hme0 [, hme1]	clusnode-4vip | 139.184.201.4 | hme0:1 [, hme1:1]	172.16.0.4 | qfe0

Manual Configuration for High Availability

The three steps in the first part of this example show you how to build the configuration, based on the information shown in Tables A–1and A–2. See RACAD Appendix B for a complete list of SRVCTL commands and syntax.

Note: You must be logged into the system as `root` on UNIX or administrator on Windows when adding the VIPs. All other SRVCTL operations are executed as the `oracle` owner and `dba` group.

Step 1. Add Node Applications

Most of your node applications configuration should have been completed during your Cluster Ready Services (CRS) installation. You can verify this by running `crs_stat` command, which should show a sequence of resource metadata and a listener resource on each active node in the cluster. If you need to add new node application manually, for example, suppose you added `clusnode-5` after your initial Oracle installation, you would use the following SRVCTL command logged in as `root` on UNIX or as `Administrator` on Windows:

```
srvctl add nodeapps -n clusnode-5 -o $ORACLE_HOME -A '139.184.201.5/255.255.255.0/hme0|hme1'
```

See Also: *Oracle Real Application Clusters Administrator's Guide*, Appendix B, "Server Control (SRCVCTL) Reference" for a list of SRVCTL commands and examples

Step 2. Add Database and Instance Applications

Define the database and each of your four instances as follows (note that, in this example, the `SPFILE` location is `$ORACLE_HOME/dbs/ORADB_spfile`):

```
srvctl add database -d ORADB -o $ORACLE_HOME -s $ORACLE_HOME/dbs/ORADB_spfile
srvctl add instance -d ORADB -i RAC01 -n clusnode-1
srvctl add instance -d ORADB -i RAC02 -n clusnode-2
srvctl add instance -d ORADB -i RAC03 -n clusnode-3
srvctl add instance -d ORADB -i RAC04 -n clusnode-4
```

Step 3. Add Service Applications

Add the service definitions as follows:

```
srvctl add service -d ORADB -s ERP -r RAC01,RAC02 -a RAC03,RAC04
srvctl add service -d ORADB -s CRM -r RAC03,RAC04 -a RAC01,RAC02
srvctl add service -d ORADB -s SELF_SERVICE -r RAC01,RAC02,RAC03,RAC04
srvctl add service -d ORADB -s HOT_BATCH -r RAC01 -a RAC02,RAC03,RAC04
srvctl add service -d ORADB -s STD_BATCH -r RAC01,RAC02,RAC03,RAC04
```

Using Services

In this section of the example, you can see how to set up your Oracle Net Services configuration files and other application-related resources to ensure your application uses the services you have configured.

Using Services with Client Applications

Applications and mid-tier connection pools select a service by using the TNS connection data. The service must match the service that has been created using `add service` with SRVCTL or DBCA. You can check the services that are currently running by querying the `V$ACTIVE_SERVICES` view.

You may use the virtual addresses for client communication to ensure that connections and SQL statements issued against a node that is down do not result in a TCP/IP time out. If your system offers a cluster alias, you may use the cluster alias for the connection only. However, you must not use host names as addresses. The address lists in the following examples use either virtual IP addresses or cluster alias.

TNS Connection Description for ERP Service

```
TNS Connection Description for ERP Service
ERP= (DESCRIPTION=
       (ADDRESS_LIST=
        (LOAD_BALANCE=yes)
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-1vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-2vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-3vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-4vip)(PORT=1521))
       )
       (CONNECT_DATA=(SERVICE_NAME=ERP)))
```

Alternatively, in the case of platforms supporting cluster aliases, the TNS alias can be simplified to:

```
ERP=(DESCRIPTION=
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusalias)(PORT=1521))
        (CONNECT_DATA=(SERVICE_NAME=ERP)))
```

TNS Connection Description for ERP Service with TAF BASIC

```
ERP= (DESCRIPTION=
     (LOAD_BALANCE=on)
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-1vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-2vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-3vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-4vip)(PORT=1521))
        (CONNECT_DATA=(SERVICE_NAME=ERP))
        (FAILOVER_MODE=(BACKUP=ERP)(TYPE=SELECT)(METHOD=BASIC)(RETRIES=180)(DELAY
=5))
      )
```

TNS Connection Description for ERP Service with TAF Preconnect

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

(BACKUP=ERP_PRECONNECT)(TYPE=SESSION)(METHOD=PRECONNECT)(RETRIES=180)(DELAY =5))
      )

ERP_PRECONNECT = (DESCRIPTION=
     (LOAD_BALANCE=on)
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-1vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-2vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-3vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-4vip)(PORT=1521))
        (CONNECT_DATA=(SERVICE_NAME=ERP_PRECONNECT))
        (FAILOVER_MODE=
           (BACKUP=ERP)(TYPE=SESSION)(METHOD=BASIC)(RETRIES=180)(DELAY =5))
      )
```

Thick JDBC Connection Description for ERP Service

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

url="jdbc:oracle:oci:@(DESCRIPTION=
     (LOAD_BALANCE=on)
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-1vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-2vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-3vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-4vip)(PORT=1521))
        (CONNECT_DATA=(SERVICE_NAME=ERP)))"

url="jdbc:oracle:oci:@(DESCRIPTION=
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusalias)(PORT=1521))
        (CONNECT_DATA=(SERVICE_NAME=ERP)))"
```

Thin JDBC Connection Description for ERP Service

```
url="jdbc:oracle:thin:@(DESCRIPTION=
     (LOAD_BALANCE=on)
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-1vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-2vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-3vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-4vip)(PORT=1521))
```

```
        (CONNECT_DATA=(SERVICE_NAME=ERP)))"

url="jdbc:oracle:thin:@(DESCRIPTION=
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusalias)(PORT=1521))
        (CONNECT_DATA=(SERVICE_NAME=ERP)))"
```

Listener Configuration for Services

You should cross-register your listeners using the REMOTE_LISTENERS initialization parameter so that all your listeners know about all of your services and the instances in which they run. The listeners should use server side load balancing, optionally based on session count for connection. The listeners must be listening on the VIPs and on the cluster aliases, when available. The listeners must not listen on the host name: listening on the host name results in disconnected sessions when VIPs automatically relocate to their owning nodes.

Sample listener.ora Entry

Each listener on each cluster node should have dual addressing, one pointing at the node VIP name (or address) and the other pointing at the host's physical IP address (or name).

```
# Listener name definition for host clusnode-1 (see Table A-2 for details):
#

LISTENER_CLUSNODE-1 =
     (ADDRESS = (PROTOCOL = TCP)(HOST = clusnode-1vip)(PORT = 1521))

SID_LIST_LISTENER_CLUSNODE-1 =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = $ORACLE_HOME)
      (PROGRAM = extproc)
    )
  )
```

Sample Remote Listener Entries

```
# TNS alias entry maps to REMOTE_LISTENER initialization parameter:
LISTENERS_ORADB=
    (ADDRESS_LIST =
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-1vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-2vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-3vip)(PORT=1521))
        (ADDRESS=(PROTOCOL=TCP)(HOST=clusnode-4vip)(PORT=1521)))
```

Oracle Instance Parameters

You must ensure that the LOCAL_LISTENER, REMOTE_LISTENER, and ACTIVE_INSTANCE_COUNT initialization parameter values are valid to use the VIPs for your services. The listener definition values should the same as those defined in the section "Using Services with Client Applications". Follow these guidelines to set the correct values:

```
local_listener=LISTENER_CLUSNODE-1 -- TNS entry listing the virtual IP address for
                                   -- node CLUSNODE-1
remote_listener=LISTENERS_ORADB    -- TNS entry listing the virtual IP addresses
                                   -- used by database ORADB
```

You must ensure that the ACTIVE_INSTANCE_COUNT parameter is left at its default value - this parameter must not be set.

Manual Configuration for Workload Management

The four steps in this next part of this example show you how to complete your service configuration to enable workload management, DBMS_SCHEDULER.CREATE_JOB execution time, resource consumption, and wait events. The first two steps are required to configure the service priorities and the job classes for the server side services in the Automatic Workload Repository (AWR). Steps three and four define service performance thresholds and enable the measurement of modules and actions within services.

Step 1. Add Service Priorities

Before mapping services to consumer groups, you must create the required consumer groups and their related resource plans, which can be priority based or ratio based. For this example, the site already has three consumer groups named high_priority, standard_priority, and low_priority. These consumer groups map to a database resource plan that reflects the intended resource consumption.

The following SQL*Plus commands call PL/SQL to map each service to the desired consumer group and then display the results by querying DBA_SCHEDULER_JOB_CLASSES:

```
REM Create the consumer groups
execute dbms_resource_manager.create_pending_area;
execute DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (CONSUMER_GROUP =>
'HIGH_PRIORITY', COMMENT => 'High priority consumer group');
execute DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (CONSUMER_GROUP =>
'STANDARD_PRIORITY', COMMENT => 'Standard priority consumer group');
execute DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (CONSUMER_GROUP =>
'LOW_PRIORITY', COMMENT => 'Low priority consumer group');

REM Create the service to consumer group mapping
execute DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING (ATTRIBUTE =>
DBMS_RESOURCE_MANAGER.SERVICE_NAME,VALUE => 'ERP',CONSUMER_GROUP =>
'HIGH_PRIORITY');
execute DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING (ATTRIBUTE =>
DBMS_RESOURCE_MANAGER.SERVICE_NAME,VALUE => 'CRM',CONSUMER_GROUP =>
'STANDARD_PRIORITY');
execute DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING (ATTRIBUTE =>
DBMS_RESOURCE_MANAGER.SERVICE_NAME,VALUE => 'SELF_SERVICE',CONSUMER_GROUP =>
'STANDARD_PRIORITY');
execute DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING (ATTRIBUTE =>
DBMS_RESOURCE_MANAGER.SERVICE_NAME,VALUE => 'HOT_BATCH',CONSUMER_GROUP =>
'HIGH_PRIORITY');
execute DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING (ATTRIBUTE =>
DBMS_RESOURCE_MANAGER.SERVICE_NAME,VALUE => 'STD_BATCH',CONSUMER_GROUP =>
'LOW_PRIORITY');
execute dbms_resource_manager.submit_pending_area;

REM View the resource manager mappings
col value format a30 trunc
col attribute format a20 trunc
col consumer_group format a20 trunc
SELECT ATTRIBUTE, VALUE, CONSUMER_GROUP from DBA_RSRC_GROUP_MAPPINGS;
```

The query output would look like:

```
ATTRIBUTE           VALUE                          CONSUMER_GROUP
------------------- ------------------------------ --------------------
SERVICE_NAME        ERP                            HIGH_PRIORITY
SERVICE_NAME        HOT_BATCH                      HIGH_PRIORITY
SERVICE_NAME        STD_BATCH                      LOW_PRIORITY
SERVICE_NAME        CRM                            STANDARD_PRIORITY
SERVICE_NAME        SELF_SERVICE                   STANDARD_PRIORITY
ORACLE_USER         SYS                            SYS_GROUP
ORACLE_USER         SYSTEM                         SYS_GROUP
```

You must ensure that the database user profiles include this mapping to prevent users from accessing services to which they are not entitled:

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

Step 2. Add Job Classes

The database employs two batch queues managed by the Job Scheduler, called HOT_BATCH and STD_BATCH. These queues correspond to job classes with services of the same name. The following PL/SQL code creates the job classes with assigned services:

```
REM For single instance, the services must be created explicitly:
REM execute dbms_service.create_service('HOT_BATCH', 'HOT_BATCH') ;
REM execute dbms_service.create_service('STD_BATCH', 'STD_BATCH') ;

REM Otherwise, for RAC, the instances were created with srvctl, so
REM job classes can be directly scheduled with the scheduler:
execute DBMS_SCHEDULER.CREATE_JOB_CLASS( JOB_CLASS_NAME => 'HOT_BATCH',
RESOURCE_CONSUMER_GROUP => NULL, SERVICE => 'HOT_BATCH', LOGGING_LEVEL =>
DBMS_SCHEDULER.LOGGING_RUNS, LOG_HISTORY => 30, COMMENTS => 'P1 batch');

execute DBMS_SCHEDULER.CREATE_JOB_CLASS( JOB_CLASS_NAME => 'STD_BATCH',
RESOURCE_CONSUMER_GROUP => NULL, SERVICE => 'STD_BATCH', LOGGING_LEVEL =>
DBMS_SCHEDULER.LOGGING_RUNS, LOG_HISTORY => 30, COMMENTS => 'P3 batch');

REM Verify the job class to service configuration
col service format a30 trunc
select JOB_CLASS_NAME, SERVICE from DBA_SCHEDULER_JOB_CLASSES;
```

The query output would look like:

```
JOB_CLASS_NAME                 SERVICE
------------------------------ ------------------------------
DEFAULT_JOB_CLASS
AUTO_TASKS_JOB_CLASS
HOT_BATCH                      HOT_BATCH
STD_BATCH                      STD_BATCH
```

The jobs executing in these job classes execute at instances offering the service.

Step 3. Add Service Performance Thresholds

Add thresholds for the ERP and HOT-BATCH services as listed in table A-1.

Note: The response target times are converted from seconds, shown in the planning worksheet (Table A–1), to microseconds, required by the DBMS_SERVER_ALERT.SET_THRESHOLD package in the following example.

The thresholds must be created for each RAC instance. Run the statements in this step in a SQL*Plus session:

```
REM ERP service, baseline at 0.25s, warning at 0.5s, critical at 0.75:
execute DBMS_SERVER_ALERT.SET_THRESHOLD(dbms_server_alert.elapsed_time_per_call,
dbms_server_alert.operator_ge,    '500000', dbms_server_alert.operator_ge,
'750000', 1, 5, 'RAC01', dbms_server_alert.object_type_service, 'ERP');
execute DBMS_SERVER_ALERT.SET_THRESHOLD(dbms_server_alert.elapsed_time_per_call,
dbms_server_alert.operator_ge, '500000', dbms_server_alert.operator_ge, '750000',
1, 5, 'RAC02', dbms_server_alert.object_type_service, 'ERP');
execute DBMS_SERVER_ALERT.SET_THRESHOLD(dbms_server_alert.elapsed_time_per_call,
dbms_server_alert.operator_ge, '500000', dbms_server_alert.operator_ge, '750000',
1, 5, 'RAC03', dbms_server_alert.object_type_service, 'ERP');
execute DBMS_SERVER_ALERT.SET_THRESHOLD(dbms_server_alert.elapsed_time_per_call,
dbms_server_alert.operator_ge, '500000', dbms_server_alert.operator_ge, '750000',
1, 5, 'RAC04', dbms_server_alert.object_type_service, 'ERP');

REM HOT_BATCH service, baseline at 0.5, warning at 1.0s, critical at 1.5s:
execute DBMS_SERVER_ALERT.SET_THRESHOLD(dbms_server_alert.elapsed_time_per_call,
dbms_server_alert.operator_ge, '1000000', dbms_server_alert.operator_ge,
'1500000', 1, 5, 'RAC01', dbms_server_alert.object_type_service, 'ERP');
execute DBMS_SERVER_ALERT.SET_THRESHOLD(dbms_server_alert.elapsed_time_per_call,
dbms_server_alert.operator_ge, '1000000', dbms_server_alert.operator_ge,
'1500000', 1, 5, 'RAC02', dbms_server_alert.object_type_service, 'ERP');
execute DBMS_SERVER_ALERT.SET_THRESHOLD(dbms_server_alert.elapsed_time_per_call,
dbms_server_alert.operator_ge, '1000000', dbms_server_alert.operator_ge,
'1500000', 1, 5, 'RAC03', dbms_server_alert.object_type_service, 'ERP');
execute DBMS_SERVER_ALERT.SET_THRESHOLD(dbms_server_alert.elapsed_time_per_call,
dbms_server_alert.operator_ge, '1000000', dbms_server_alert.operator_ge,
'1500000', 1, 5, 'RAC04', dbms_server_alert.object_type_service, 'ERP');

REM verify the threshold configuration
select METRICS_NAME, WARNING_VALUE, CRITICAL_VALUE, OBSERVATION_PERIOD from
dba_thresholds;
```

Step 4. Enable Service, Module, and Action Monitoring

You can enable performance data and tracing for important modules and actions within each service. The performance statistics are available in the V$SERV_MOD_ACT_STATS view. The following commands, executed in a SQL*Plus session, perform these actions:

1. Enable monitoring for the exceptions pay action in the module, payroll, under the ERP service
2. Enable monitoring for the all actions in the module, payroll, under the ERP service
3. Enable monitoring for the all actions in the module, posting, under the HOT_BATCH service
4. Confirm the configuration by querying DBA_ENABLED_AGGREGATIONS

```
execute DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(SERVICE_NAME => 'ERP', MODULE_NAME=>
'PAYROLL', ACTION_NAME => 'EXCEPTIONS PAY');
execute DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(SERVICE_NAME => 'ERP', MODULE_NAME
```

```
=> 'PAYROLL', ACTION_NAME => null);
execute DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(SERVICE_NAME => 'HOT_BATCH',
MODULE_NAME =>'POSTING', ACTION_NAME => null);

REM Verify the enabled service, module, action configuration
col AGGREGATION_TYPE format a20 trunc heading 'AGGREGATION'
col PRIMARY_ID format a20 trunc heading 'SERVICE'
col QUALIFIER_ID1 format a20 trunc heading 'MODULE'
col QUALIFIER_ID2 format a20 trunc heading 'ACTION'
select * from DBA_ENABLED_AGGREGATIONS ;
```

The query output would look like:

```
AGGREGATION          SERVICE              MODULE     ACTION
------------         -------------------- ---------- -------------
SERVICE_MODULE_ACTIO ERP                  PAYROLL    EXCEPTIONS PAY
SERVICE_MODULE_ACTIO ERP                  PAYROLL
SERVICE_MODULE_ACTIO HOT_BATCH            POSTING
```

Using Services with Job Scheduler

Use the `DBMS_SCHEDULER.CREATE_JOB` procedure to define jobs to execute under the job classes. In this example, the `MY_NAME.MY_PROC` procedure will run in the HOT_BATCH service because of the job class assignment defined earlier, in Step 2:

```
execute 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', ENABLED => TRUE,
AUTO_DROP => false, COMMENTS => 'my report on daily status');
```

Using Callouts for Fast Application Notification

Custom-written application callouts are programs or shell script wrappers that can be used to start and stop on- or off-cluster applications, or connection pools managed by middleware. They are immediately executed by RAC when a service or any part of the service starts, stops or fails to automatically restart. Other actions that can be encoded as callouts (besides restarting applications) include: logging fault tickets, e-mailing or paging administrators, and invoking third-party event systems or clusterware components.

Callouts are not a requirement to deploy RAC-HA on CRS, but Oracle strongly advises customers to build notification mechanisms using callouts. Unless your CRS home directory is shared across the network, you must deploy each new callout under `/private/oracle/crs/racg/usrco` directory on each RAC node.

The following example, a Bourne shell script, contains a number of callout options that are invoked whenever an HA event occurs. The callouts perform the following two actions: write an uptime status record to the log, and log a fault ticket (with the IT trouble ticket application) for all DOWN conditions.

```
#!/usr/bin/sh
#

# Description: Example wrapper script to enable RAC event logging and notification
#              to generic third-party systems.  The script showcases two possible
#              methods to enable local or remote logging/notification of RAC-
#              detected events.
#

# Note:  Unless your CRS home directory is on an NFS-mounted device, you should
```

```
#        copy this script to the racg/usrco directories in your CRS home
#        directory, for all RAC nodes protected by Oracle Integrated Clusterware.
#        This is a one-time setup.
#

# For additional details on callouts and what name=value pairs are passed by RAC,
# please review /private/oracle/crs/racg/usrco/README.

# Global variables:
#

AWK=/usr/bin/awk
MY_CRS_HOME=/private/oracle/crs

# Scan and parse arglist:
#

for ARGS in $*; do
   PROPERTY=`echo $ARGS | $AWK -F"=" '{print $1}'`
   VALUE=`echo $ARGS    | $AWK -F"=" '{print $2}'`

   #> map EVTTYPE to EVENT_TYP, NODE to HOST:
   case $PROPERTY in
     #> note: EVENT_TYP is one of: NODE, DATABASE, INSTANCE, SERVICE,
SERVICEMEMBER
     EVENT_TYP | event_typ)     NOTIFY_EVENT_TYP=$VALUE ;;
     VERSION | version)         NOTIFY_VERSION=$VALUE ;;
     SERVICE | service)         NOTIFY_SERVICE=$VALUE ;;
     DATABASE | database)       NOTIFY_DBNAME=$VALUE ;;
     INSTANCE | instance)       NOTIFY_INSTANCE=$VALUE ;;
     HOST | host)               NOTIFY_HOST=$VALUE ;;
     STATUS | status)           NOTIFY_STATUS=$VALUE ;;
     TIMESTAMP | timestamp)     NOTIFY_SVRLOGDATE=$VALUE ;;
   esac
done


# ###################################################
# [1] Notification Method 1:  On-cluster file logging
# ###################################################
# This section simply writes one-line entries for each event published by RAC,
# and the log is written to standard RAC log directory.  It will blindly record
# all RAC events, regardless of state (UP, DOWN or NOT_RESTARTING):

RACEVT_LOGFILE=$MY_CRS_HOME/racg/log/rac_${NOTIFY_SERVICE}_uptime.log

echo RAC\(v$NOTIFY_VERSION\):  $NOTIFY_STATUS event, type "$NOTIFY_EVENT_TYP", \
     `if [ -n "$NOTIFY_SERVICE" ]; then \
        echo "for service $NOTIFY_SERVICE"
      fi` \
     \[`if [ -n "$NOTIFY_INSTANCE" ]; then \
        echo "inst: $NOTIFY_INSTANCE"
        fi` \
       `if [ -n "$NOTIFY_DATABASE" ]; then \
        echo "db: $NOTIFY_DATABASE"
        fi` \
       `if [ -n "$NOTIFY_HOST" ]; then \
        echo "db: $NOTIFY_HOST"
        fi` \
     \] received on $NOTIFY_SVRLOGDATE >>  $RACEVT_LOGFILE
```

```
# ########################################################
# [2] Notification Method 2:  On-cluster program execution
# ########################################################
# Let's assume you have a custom client program in /tmp (say logTicket) to which
# you can pass certain arguments.  This program connects to a customer-service
# application that processes incident tickets for your IT department:
#    % /tmp/logTicket {serverside_timestamp} \
#                     {databasename} {servicename} \
#                     {instancename} {hostname}
#

# Let us also assume that a ticket would be logged only for NOT_RESTARTING events,
# as they are the ones that exceeded RAC-monitored timeouts and seriously need
# human intervention for full resolution.
#

# -------------------
# ONE SOLUTION TO [2]:
# -------------------
if [ $NOTIFY_STATUS  = "NOT_RESTARTING" -o $NOTIFY_STATUS = "not_restarting" ];
then
  /tmp/logTicket $NOTIFY_SVRLOGDATE $NOTIFY_DBNAME \
                        $NOTIFY_SERVICE \
                        $NOTIFY_INSTANCE $NOTIFY_HOST   >> $RACEVT_LOGFILE
fi
```

Configuring JDBC Fast Application Notification

To use Fast Application Notification, the application must use the JDBC Implicit Connection Cache. JDBC connection pools are integrated with the callout mechanism, providing the following benefits:

- Balancing connections across RAC when a service first starts up - rather than directing the minimum sessions defined for the connection pool to the first RAC instance that supports the service.
- Consuming additional RAC instances immediately a service is registered `UP` at additional instances.
- Cleaning up terminated connections immediately a service is declared `DOWN` at any instance, and immediately that nodes are declared `DOWN`
- Reporting an error to clients immediately the `NOT RESTARTING` status is detected, instead of making the client wait while the service repeatedly tries to restart.
- Distributing client work requests at runtime across the RAC instances supporting a service.

Configuring the JDBC Client Side

Refer to JDBC User's Guide and Reference for configuring the JDBC Implicit Connection Cache and Oracle Notification Service (ONS). If ONS is not configured correctly, the implicit connection cache creation fails and an appropriate exception occurs upon the first `getConnection()request`.

Set the `ConnectionFailoverEnabled` property before making the first `getConnection()` request to a DataSource. When Fast Connection Failover is enabled, the failover applies to every connection in the connection cache. If your application explicitly creates a connection cache using the Connection Cache Manager, you must first set `ConnectionFailoverEnabled`.

Configuring the RAC High Availability Server Side

The RAC event system needs to be configured to forward the HA events to every ONS, so that ONS clients at the mid-tier can receive and respond to the state changes.

Step 1 - Configure the ONS Daemon

The ONS daemon must be configured to broadcast the HA events from RAC to Oracle Application Server 10*g* clients. The ONS configuration file is located in `$ORACLE_HOME/opmn/conf/ons.config` and file should be built by OUI during installation Here is a sample RAC `ons.config` file:

```
localport=4100
loglevel=3
remoteport=4101
nodes=139.185.140.63:4101,139.185.140.64:4101,139.185.140.65:4101
```

where nodes is a list of all ONS daemons that talk to each other on RAC and Oracle Application Server. These values are given as a list of either host name or IP address plus port combinations in a comma separated list. Note that the port value that is given is the remote port that each ONS instance is listening on.

Step 2 - Check that the ONS Daemon is Running

The ONS daemon is running as a node application. To check node applications use the command: `srvctl status nodeapps`. Your results should be similar to the following:

```
NAME=ora.clusnode-4.ons
TYPE=application
TARGET=ONLINE
STATE=ONLINE
```

Use `onsctl ping` to check that the ONS daemon is active.

Using a Shared Oracle Home

ONS requires that the `$ORACLE_HOME/opmn/log` directory is private for each ONS daemon. If using a cluster file system for `$ORACLE_HOME`, each node should define `$ORACLE_HOME/opmn/log` as a link to a node specific directory, for example, `$ORACLE_HOME/opmn/clusnode1/log`.

Events for Shadow Preconnect Services in Real Application Clusters

When using Transparent Application Failover (TAF) PRECONNECT, Real Application Clusters (RAC) high availability maintains a preconnect service to support TAF Preconnect and applications that are configured to manage work on secondary RAC instances. Secondary instances are RAC instances that are not supporting the primary service.

In this type of configuration, Oracle maintains the shadow service on all instances that do not support the primary service. You can use events to stop and start secondary work. The events are posted to callouts and to the Oracle Notification Service (ONS).

To use events, configure the payload with the following format:

```
SRV_PRECONNECT VERSION=1.0 service=db_unique_name.db_domain
database=database name instance=instance name host=host name
status=preconn_up reason=timestamp=27-Jan-2004 16:53:58
reported=Tue Jan 27 16:53:59 PST 2004
SRV_PRECONNECT VERSION=1.0 service=db_unique_name.db_domain
database=RACEY instance=instance name host=host name
status=preconn_down reason=timestamp=27-Jan-2004 16:58:01
reported=Tue Jan 27 16:58:02 PST 2004
```

Up Event Example:

```
@ SRV_PRECONNECT VERSION=1.0 service=MYSERV.us.oracle.com
database=RACEY instance=RACEY1 host=myhost-pc status=preconn_up
reason=timestamp=27-Jan-2004 16:53:58
reported=Tue Jan 27 16:53:59 PST 2004
```

Down Event Example:

```
@ SRV_PRECONNECT VERSION=1.0 service=MYSERV.us.oracle.com
database=RACEY instance=RACEY1 host=myhost-pc status=preconn_down
reason=timestamp=27-Jan-2004 16:58:01
reported=Tue Jan 27 16:58:02 PST 2004
```

High Availability Callouts and Oracle Notification Events

The notification interface is available as a server-side callout and as an Oracle Notification Services (ONS) event. The server-side callout is a script with the same payload as the ONS event that is run immediately on the server when the condition occurs. Use this method to start and stop server-side applications, to relocate low-priority services when high priority services arrive, and to post tickets for fault tracking. The following table describes the event payload.

Table A–3 Event payload parameters and descriptions

Parameter	Description
`Event type`	The event type for the component such as service, service_member, database, instance, or node
`Service name`	The service name; matches the configured service in `SERVICE$`
`Database name`	The database supporting the service; matches the initialization parameter value for `DB_UNIQUE_NAME`, which in turn defaults to the value of the initialization parameter `DB_NAME`
`Instance`	The name of the instance that supports the service; matches the instance name
`Node name`	The name of the node that supports the service or the node that has failed; matches the CSS node name
`Status`	The new status; values are `UP`, `DOWN`, and `NOT_RESTARTING`
`Cardinality`	Cardinality for the service on UP events
`Time stamp`	The local time zone to use when ordering notification events
`Incarnation`	Cluster incarnation for node down

When a session connects, mid-tiers can record the following values that match the high availability event payload.

```
sys_context('userenv', 'instance_name');
sys_context('userenv', 'server_host');
```

```
sys_context('userenv', 'service_name');
sys_context('userenv', 'db_unique_name');
```

Index

J

M

N

O

P

R

S

T

U

V

W