

Oracle9i Real Application Clusters

Concepts

Release 2 (9.2)

March 2002

Part No. A96597-01

Oracle9i Real Application Clusters Concepts, Release 2 (9.2)

Part No. A96597-01

Copyright © 1998, 2002 Oracle Corporation. All rights reserved.

Primary Author: Mark Bauer.

Contributors: David Austin, Wilson Chan, Sashikanth Chandraserkaran, Jonathan Creighton, Lisa Eldridge, Mitch Flatland, Rick Greenwald, Eugene Ho, Bill Kehoe, Raj Kumar, Kotaro Ono, Stefan Pommerenk, Rebecca Reitmeyer, Joao Rimoli, Daniel Semler, Vinay Srihari, Alok Srivastava, Tak Wang, Shari Yamaguchi, and Tolga Yurek.

Graphic Designer: Valarie Moore.

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent and other intellectual and industrial property laws. Reverse engineering, disassembly or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark, and Oracle9i, Oracle Names, SQL*Plus, Oracle Store, and PL/SQL are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

Send Us Your Comments	xiii
Preface.....	xv
What's New in Real Application Clusters?.....	xxv
Part I Cluster Database Processing Fundamentals	
1 Introduction to Real Application Clusters	
What Is Real Application Clusters?	1-2
The Benefits of Real Application Clusters.....	1-3
Lower Overall Cost of Ownership.....	1-3
Expanded Scalability.....	1-3
High Availability	1-4
Transparency.....	1-4
Buffer Cache Management.....	1-4
Row Locking.....	1-5
Multiversion Read Consistency.....	1-5
Recovery Manager, Online Backups, and Archiving.....	1-5
2 Real Application Clusters Systems Architecture	
Overview of Cluster Database System Components.....	2-2
Nodes and Their Components.....	2-3
Cluster Interconnect and Interprocess Communication (Node-to-Node)	2-3

Memory, Interconnect, and Storage	2-4
The High-Speed IPC Interconnect	2-4
Shared Disk Storage and the Cluster File System Advantage	2-4

3 Real Application Clusters Software Architecture

The Operating System-Dependent Clusterware	3-2
The Cluster Manager.....	3-2
The Node Monitor	3-2
The Interconnect.....	3-3
Real Application Clusters Shared Disk Components	3-3
Real Application Clusters-Specific Daemon and Instance Processes	3-3
The Global Services Daemon	3-4
Instance Processes Specific to Real Application Clusters	3-4
The Global Cache and Global Enqueue Service	3-6
Application Transparency	3-6
Global Resource Directory with Distributed Architecture	3-6
Resource Mastering and Affinity	3-7
GCS and GES Interaction with the Cluster Manager	3-7

4 Scalability in Real Application Clusters

Scalability Features of Real Application Clusters	4-2
All System Types Benefit from Real Application Clusters	4-2
Transaction Systems and Real Application Clusters.....	4-2
Data Warehouse Systems and Real Application Clusters	4-3
Levels of Scalability	4-3
Network Scalability	4-4
Network Scalability and Client/Server Connectivity	4-4
Operating System Scalability	4-6

Part II Resource Coordination in Real Application Clusters

5 Real Application Clusters Resource Coordination

Overview of Real Application Clusters Resource Coordination	5-2
The Contents of the Global Resource Directory	5-2
Real Application Clusters Synchronization Processes	5-3
Enqueues	5-3

Past Images	5-3
Resource Modes and Roles	5-4
Resource Modes	5-4
Resource Roles	5-4
Global Cache Service Operations	5-5
System Change Number Processing	5-6
Lamport SCN Generation.....	5-7

6 Cache Fusion and the Global Cache Service

Overview of Cache Fusion Processing	6-2
Concurrent Reads on Multiple Nodes.....	6-2
Concurrent Reads and Writes on Different Nodes.....	6-2
Concurrent Writes on Different Nodes	6-2
Write Protocol and Past Image Tracking	6-3
Resource Control, Cache-to-Cache Transfer, and Cache Coherency	6-3
Block Access Modes and Buffer States	6-4
Cache Fusion Scenarios	6-4
Requesting a Changed Block for a Modification Operation	6-5
Writing Blocks to Disk	6-6
Real Application Clusters Recovery and Cache Fusion	6-7

7 Resource Coordination by the Global Enqueue Service

Global Enqueue Service Processing	7-2
Global Enqueue Concurrency Control	7-2
Resources Managed by the Global Enqueue Service	7-3
Dictionary Cache Locks	7-3
Library Cache Locks.....	7-3

Part III Implementing Real Application Clusters

8 Real Application Clusters Storage Considerations

Overview of Storage in Real Application Clusters	8-2
Datafiles in Real Application Clusters	8-2
Datafile Verification in Real Application Clusters	8-2
Adding Datafiles in Real Application Clusters.....	8-2

Parameter File Storage in Real Application Clusters	8-3
Location of the Server Parameter File.....	8-4
Redo Log File Storage in Real Application Clusters	8-4
Automatic Segment-Space Management	8-5
Managing Undo Space in Real Application Clusters	8-5
Private and Public Rollback Segments	8-5

9 Manageability Tools for Real Application Clusters Environments

Overview of Manageability in Real Application Clusters	9-2
Manageability for Real Application Clusters Installation, Setup, and Configuration	9-2
Manageability for Real Application Clusters Administration	9-3
Oracle Enterprise Manager.....	9-3
The Database Configuration Assistant	9-4
The Server Control (SRVCTL) Utility	9-5
Global Services Daemon Administration Commands	9-5
Manageability for Real Application Clusters Performance Monitoring	9-5
Monitoring Performance with Oracle Enterprise Manager.....	9-6
Monitoring Performance with Statspack.....	9-6
Manageability for Real Application Clusters Backup and Recovery	9-6

Part IV High Availability and Real Application Clusters

10 High Availability Concepts and Best Practices in Real Application Clusters

Understanding High Availability	10-2
Configuring Real Application Clusters for High Availability	10-2
Cluster Components and High Availability	10-3
Disaster Planning	10-4
Failure Protection Validation	10-4
Failover and Real Application Clusters	10-5
Failover Basics	10-5
Client Failover	10-6
Uses of Transparent Application Failover	10-7
Server Failover	10-11
Failover Processing in Real Application Clusters	10-12

Detecting Failure.....	10-13
Reorganizing Cluster Membership.....	10-13
Performing Database Recovery	10-14
High Availability Configurations	10-16
Default N-Node Configurations.....	10-17
Basic High Availability Configurations	10-17
Shared High Availability Node Configurations	10-24
Full Active Configurations with Real Application Clusters Guard II	10-25
Deploying High Availability	10-25

Part V Reference

A Restrictions

Compatibility.....	A-2
Restricted SQL Statements	A-2
Maximum Number of Datafiles	A-2

B Using Multi-Block Lock Assignments (Optional)

When to Use Locks	B-2
How to Use Locks	B-2
Lock Granularity	B-2
Understanding Lock Management.....	B-2

Glossary

Index

List of Figures

2-1	Cluster Components for Cluster Database Processing	2-2
3-1	Real Application Clusters-Specific Instance Processes	3-5
6-1	Requesting a Changed Block for a Modification Operation	6-5
6-2	Writing Blocks to Disk	6-6
10-1	Primary/Secondary Configurations in Dedicated Server Environments.....	10-19
10-2	Primary/Secondary Configurations and Node Failure with Dedicated Server	10-20
10-3	Primary/Secondary Configurations in Shared Server Environments	10-22
10-4	Primary/Secondary Configurations and Node Failure with Shared Server	10-23
B-1	Lock-to-Block Assignment Examples for GC_FILES_TO_LOCKS Settings	B-3

List of Tables

5-1	Global Cache Service Resource Modes	5-4
-----	---	-----

Send Us Your Comments

Oracle9i Real Application Clusters Concepts, Release 2 (9.2)

Part No. A96597-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. 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?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, 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
500 Oracle Parkway, Mailstop 4op11
Redwood Shores, CA 94065
USA

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

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

Preface

Oracle9i Real Application Clusters Concepts describes the concepts of Real Application Clusters for all operating systems. Read this manual before reading *Oracle9i Real Application Clusters Setup and Configuration*, *Oracle9i Real Application Clusters Administration*, and *Oracle9i Real Application Clusters Deployment and Performance*.

Before reading about Real Application Clusters concepts, however, you should be familiar with single-instance Oracle database processing as described in *Oracle9i Database Concepts*.

See Also: The *Oracle9i Real Application Clusters Documentation Online Roadmap* to use the online Real Application Clusters Documentation set

This preface contains the following topics:

- [Audience](#)
- [Organization](#)
- [Related Documentation](#)
- [Conventions](#)
- [Documentation Accessibility](#)

Audience

Real Application Clusters Concepts is intended for database administrators and application developers who work with Real Application Clusters.

Organization

This book presents Real Application Clusters concepts in five parts. It begins by describing cluster database processing fundamentals for Real Application Clusters. The book then describes resource processing among instances. It then explains the fundamentals of implementing Real Application Clusters. Then it describes high availability and Real Application Clusters. The book ends with reference material that includes an appendix describing the implementation restrictions for Real Application Clusters, an appendix on the concepts of lock setting, and a glossary of terms.

Part I "Cluster Database Processing Fundamentals"

Part I introduces Real Application Clusters, describes its architecture and its software components, and explains its scalability features.

Chapter 1, "Introduction to Real Application Clusters"

This chapter introduces cluster database processing with Real Application Clusters.

Chapter 2, "Real Application Clusters Systems Architecture"

This chapter describes the systems components and high-level architectural models that typify cluster environments.

Chapter 3, "Real Application Clusters Software Architecture"

This chapter describes the Real Application Clusters software components.

Chapter 4, "Scalability in Real Application Clusters"

This chapter describes the scalability features of Real Application Clusters.

Part II "Resource Coordination in Real Application Clusters"

Part II describes interinstance resource coordination and explains the processing that the Global Cache Service (GCS) and Global Enqueue Service (GES) perform.

Chapter 5, "Real Application Clusters Resource Coordination"

This chapter describes the interinstance coordination that occur in Real Application Clusters environments.

Chapter 6, "Cache Fusion and the Global Cache Service"

This chapter provides a detailed discussion of Cache Fusion and the interinstance coordination activities managed by the GCS.

Chapter 7, "Resource Coordination by the Global Enqueue Service"

This chapter provides details on and the interinstance coordination activities managed by the GES.

Part III "Implementing Real Application Clusters"

Part III describes the implementation considerations and manageability components for Real Application Clusters.

Chapter 8, "Real Application Clusters Storage Considerations"

This chapter describes the storage considerations for Real Application Clusters applications.

Chapter 9, "Manageability Tools for Real Application Clusters Environments"

This chapter describes the manageability components for Real Application Clusters environments.

Part IV "High Availability and Real Application Clusters"

Part IV describes how you can implement high availability with Real Application Clusters.

Chapter 10, "High Availability Concepts and Best Practices in Real Application Clusters"

This chapter describes the concepts and best practices for using Real Application Clusters to implement high availability.

Part V "Reference"

Part V contains reference information about Real Application Clusters.

Appendix A, "Restrictions"

This appendix lists the deployment restrictions for Real Application Clusters.

Appendix B, "Using Multi-Block Lock Assignments (Optional)"

This appendix describes how to use optional lock setting to override the interinstance coordination provided by Cache Fusion.

Glossary

This glossary defines important terms used in this book.

Related Documentation

- After reading this manual, Oracle Corporation recommends that you read *Oracle9i Real Application Clusters Setup and Configuration*, *Oracle9i Real Application Clusters Administration*, and *Oracle9i Real Application Clusters Deployment and Performance*.
- You can also read *Oracle9i Real Application Clusters Real Application Clusters Guard I - Concepts and Administration* for information about Oracle Real Application Clusters Guard I and *Oracle9i Real Application Clusters Guard II Concepts, Installation, and Administration* on the Real Application Clusters Guard II software CD for information about Real Application Clusters Guard II.

See Also: The *Oracle9i Real Application Clusters Documentation Online Roadmap* to use the online Real Application Clusters Documentation set

For more information, see these Oracle resources:

Installation Guides

- *Oracle9i Installation Guide Release 2 (9.2.0.1) for UNIX Systems: AIX-Based Systems, Compaq Tru64, HP 9000 Series HP-UX, Linux Intel, and Sun Solaris*
- *Oracle9i Database Installation Guide for Windows*
- *Oracle Diagnostics Pack Installation*

- Oracle Real Application Clusters Guard I installation manuals; there are several platform-specific titles available
- *Oracle9i Real Application Clusters Guard II Concepts, Installation, and Administration* on the Real Application Clusters Guard II software CD

Operating System-Specific Administrator's Guides

- *Oracle9i Administrator's Reference Release 2 (9.2.0.1) for UNIX Systems: AIX-Based Systems, Compaq Tru64, HP 9000 Series HP-UX, Linux Intel, and Sun Solaris*
- *Oracle9i Database Administrator's Guide for Windows*
- *Oracle9i Real Application Clusters Real Application Clusters Guard I - Concepts and Administration*
- *Oracle9i Real Application Clusters Guard II Concepts, Installation, and Administration* on the Real Application Clusters Guard II software CD

Cluster Database Management

- *Oracle9i Database Concepts*
- *Oracle9i Database Administrator's Guide*
- *Oracle Enterprise Manager Administrator's Guide*
- *Getting Started with the Oracle Diagnostics Pack*

In North America, printed documentation is available for sale in the Oracle Store at

<http://oraclestore.oracle.com/>

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

<http://www.oraclebookshop.com/>

Other customers can contact their Oracle representative to purchase printed documentation.

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/admin/account/membership.html>

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/docs/index.htm>

To access the database documentation search engine directly, please visit

<http://tahiti.oracle.com>

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.
<i>Italics</i>	Italic typeface indicates book titles or emphasis.	<i>Oracle9i Database Concepts</i> Ensure that the recovery catalog and target database do <i>not</i> reside on the same disk.
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.

Convention	Meaning	Example
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 <code>sqlplus</code> to open SQL*Plus. The password is specified in the <code>orapwd</code> file. Back up the datafiles and control files in the <code>/disk1/oracle/dbs</code> directory. The <code>department_id</code> , <code>department_name</code> , and <code>location_id</code> columns are in the <code>hr.departments</code> table. Set the <code>QUERY_REWRITE_ENABLED</code> initialization parameter to <code>true</code> . Connect as <code>oe</code> user. The <code>JRepUtil</code> class implements these methods.
lowercase italic monospace (fixed-width) font	Lowercase italic monospace font represents placeholders or variables.	You can specify the <code>parallel_clause</code> . Run <code>Uold_release.SQL</code> where <code>old_release</code> 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 (<i>digits</i> [, <i>precision</i>])
{ }	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]

Convention	Meaning	Example
...	Horizontal ellipsis points indicate either: <ul style="list-style-type: none"> 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 	<pre>CREATE TABLE ... AS subquery; SELECT col1, col2, ... , coln FROM employees;</pre>
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	<pre>SQL> SELECT NAME FROM V\$DATAFILE; NAME ----- /fs1/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fs1/dbs/tbs_09.dbf 9 rows selected.</pre>
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	<pre>acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;</pre>
<i>Italics</i>	Italicized text indicates placeholders or variables for which you must supply particular values.	<pre>CONNECT SYSTEM/system_password DB_NAME = database_name</pre>
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.	<pre>SELECT last_name, employee_id FROM employees; SELECT * FROM USER_TABLES; DROP TABLE hr.employees;</pre>
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.	<pre>SELECT last_name, employee_id FROM employees; sqlplus hr/hr CREATE USER mjones IDENTIFIED BY ty3MU9;</pre>

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 Corporation 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 Corporation does not own or control. Oracle Corporation neither evaluates nor makes any representations regarding the accessibility of these Web sites.

What's New in Real Application Clusters?

This section describes the new features for Oracle9i release 2 (9.2) of Real Application Clusters and provides cross-references where you can find additional information. The following section describes the new features in Oracle9i release 2 (9.2) for Real Application Clusters:

- [Release 2 New Features in Real Application Clusters](#)

Release 2 New Features in Real Application Clusters

- **Oracle Universal Installer and the Database Configuration Assistant**

There are many clusterization enhancements for the Oracle Universal Installer and the Database Configuration Assistant. In addition, two extra tablespaces are required to accommodate the XML DB and Oracle Data Mining features.

See Also: *Oracle9i Real Application Clusters Installation and Configuration* for more information about enhancements to the Database Configuration Assistant

- **Database Upgrade Assistant (DBUA)**

The Database Migration Assistant is now called the Database Upgrade Assistant (DBUA) and it supports Real Application Clusters.

See Also: The *Oracle9i Database Migration* guide for information about using the DBUA

- **Cluster File System Support**

You can now use a cluster file system for Real Application Clusters on certain platforms.

See Also: *Oracle9i Real Application Clusters Installation and Configuration* and the Release 2 (9.2) README file for additional details about raw partition tablespace size requirements and your platform-specific documentation for more information about using cluster file systems

- **Dynamic Local and Remote Listener Parameters**

You can use the `ALTER SYSTEM SET SQL` statement to dynamically update the `LOCAL_LISTENER` and `REMOTE_LISTENER` parameters. In addition, when you add or remove nodes from a Real Application Clusters database, Oracle dynamically updates these parameters. During reconfiguration, the process monitor (PMON) process records reconfiguration information with all the listeners in the cluster for cross-instance listener registration.

- **Recovery Manager Enhancements for Real Application Clusters**

The Recovery Manager (RMAN) autolocation feature for Real Application Clusters automatically discovers which nodes of a Real Application Clusters configuration can access the files that you want to back up or restore by autolocating the following file types:

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

See Also: *Oracle9i Recovery Manager User's Guide and Reference* for more information about Recovery Manager and *Oracle9i Real Application Clusters Administration* for more information about using Recovery Manager in Real Application Clusters environments

- **Real Application Clusters Guard II**

Real Application Clusters Guard II supports comprehensive workload management to maintain high availability for Real Application Clusters databases and their applications. Real Application Clusters Guard II transfers application loads based on the concept of service names. Therefore, Real Application Clusters Guard II supports workload management based on service levels as well as applications using database services.

Service names have been adopted for high availability because you do not have to make application changes to implement them. In addition, service names provide location transparency to the database instances that offer the service. Service names enable a single-system image that simplifies the configuration, operation, and recovery of workloads.

See Also: *Oracle9i Real Application Clusters Guard II Concepts, Installation, and Administration* on the Real Application Clusters Guard II software CD

Part I

Cluster Database Processing Fundamentals

Part I describes the fundamentals of Real Application Clusters processing. The chapters in Part I are:

- [Chapter 1, "Introduction to Real Application Clusters"](#)
- [Chapter 2, "Real Application Clusters Systems Architecture"](#)
- [Chapter 3, "Real Application Clusters Software Architecture"](#)
- [Chapter 4, "Scalability in Real Application Clusters"](#)

Introduction to Real Application Clusters

This chapter introduces cluster database processing and the features of Real Application Clusters. This chapter includes the following topics:

- [What Is Real Application Clusters?](#)
- [The Benefits of Real Application Clusters](#)

What Is Real Application Clusters?

Real Application Clusters harnesses the processing power of multiple interconnected computers. Real Application Clusters software and a collection of hardware known as a **cluster** unite the processing power of each component to create a robust computing environment.

Real Application Clusters is a breakthrough cluster software architecture with scalability and high availability features that exceed the capabilities of previous Oracle cluster-enabled software products. Real Application Clusters offers significant advantages for all system types. With the increased functionality of Real Application Clusters, all systems and applications can efficiently exploit clustered environments.

You can use Real Application Clusters to deliver high performance, increased throughput, and **high availability**. Before deploying Real Application Clusters, however, you should understand Real Application Clusters processing.

In Real Application Clusters environments, all active instances can concurrently execute transactions against a shared database. Real Application Clusters coordinates each instance's access to the shared data to provide data consistency and data integrity.

Harnessing the power of clusters offers obvious advantages. A large task divided into subtasks and distributed among multiple nodes is completed sooner and more efficiently than if you processed the entire task on one node. Cluster processing also provides increased performance for larger workloads and for accommodating rapidly growing user populations.

With Real Application Clusters, you can scale applications to meet increasing data processing demands without changing the application code. As you add resources such as nodes or storage, Real Application Clusters extends the processing powers of these resources beyond the limits of the individual components.

Data warehouse applications that access read-only data are prime candidates for Real Application Clusters. In addition, Real Application Clusters successfully manages **Online Transaction Processing (OLTP)** systems and hybrid systems which combine the characteristics of both read-write and read-only applications. Real Application Clusters also serves as an important component of robust high availability solutions, tolerating failures with little or no downtime.

The Benefits of Real Application Clusters

This section describes the following benefits of Real Application Clusters:

- Lower Overall Cost of Ownership
- Expanded Scalability
- High Availability
- Transparency
- Buffer Cache Management
- Row Locking
- Recovery Manager, Online Backups, and Archiving

Lower Overall Cost of Ownership

Real Application Clusters lowers the overall cost of ownership more effectively than other cluster database products. This is due in great part to the single-system image afforded by the Real Application Clusters architecture.

The degree of system consolidation that you can achieve with Real Application Clusters enables you to use significantly *less* software and computing equipment than other cluster database products. Moreover, you can deploy Real Application Clusters for *any* configuration requiring high availability and scalability. In addition, the comprehensive, easy-to-use manageability tools that interact with Real Application Clusters *keep* your overall costs low by greatly reducing the administrative overhead throughout the life of your project.

Expanded Scalability

A scalable environment enables you to improve performance and add capacity by adding nodes. On some platforms you can add nodes dynamically while your cluster is running.

The number of nodes that Real Application Clusters can actually support is significantly more nodes than any known implementation. Small systems configured primarily for high availability might only have two nodes. Large configurations, however, might have 32 to 64 nodes.

Note: Whether you can dynamically add nodes to your cluster depends on the capabilities of your clusterware. If you use Oracle clusterware, then you can dynamically add nodes and instances on most platforms.

See Also: [Chapter 4, "Scalability in Real Application Clusters"](#) for more information about scalability

High Availability

High availability refers to systems with redundant components that provide consistent, uninterrupted service, even during failures. In most high availability configurations, nodes are isolated from each other so that a failure on one node does not affect the entire system.

Transparency

The concept of **transparency** implies that Real Application Clusters environments are functionally equivalent to single-instance Oracle database configurations. In other words, you do not need to make code changes to deploy applications on Real Application Clusters if your applications ran efficiently on single-instance Oracle configurations.

The Cache Fusion feature, which implies a fusing of the multiple buffer caches in a cluster, simplifies the administration of Real Application Clusters environments. With Real Application Clusters and Cache Fusion, you also do not need to perform capacity planning.

Transparency is also realized by efficient resource use at both the application and system levels. For example, you do not need to perform time-consuming resource configurations by examining data access patterns because Real Application Clusters does this automatically.

Buffer Cache Management

Oracle stores resources, such as data block information, in a buffer cache that resides in memory. Storing this information locally reduces database operations and disk I/O. Because each instance has its own memory, Real Application Clusters coordinates the buffer caches of multiple nodes while minimizing disk I/O. This optimizes performance and expands the effective memory to be nearly equal to the sum of all memory in your cluster database.

To do this, Real Application Clusters uses the **Global Cache Service (GCS)** to coordinate operations among the multiple buffer caches and to optimize Oracle's high performance features. The **Global Enqueue Service (GES)** also assists in synchronization by managing intranode communications. The GCS and GES are described in more detail later in Part II of this book.

See Also: *Oracle9i Database Concepts* for detailed information about the buffer cache

Row Locking

Oracle row locking and multi-version read consistency provide a high degree of concurrency and throughput. Row locking ensures that transactions that modify different rows in the same data block do not need to wait for each other to commit.

See Also: *Oracle9i Database Concepts* for more information about row locking

Multiversion Read Consistency

Multiversion read consistency ensures that read operations do not block write operations and that write operations do not block read operations. Multiversion read consistency creates snapshots or read consistent versions of blocks that have been modified by a transaction that has not committed. This approach has two key benefits:

- Read operations do not have to wait for resources because users modifying data do not prevent readers from reading
- A snapshot view of the data is available from a specific point in time

Recovery Manager, Online Backups, and Archiving

Real Application Clusters supports the full functionality of **Recovery Manager (RMAN)** and **Oracle Enterprise Manager**. Real Application Clusters also supports all Oracle backup and archiving features that are available in single-instance Oracle databases. This includes both online and offline backups of either an entire database or individual tablespaces.

If you operate Oracle in ARCHIVELOG mode, then when a log file is full Oracle converts it to an archive log file before overwriting the log file with information. In Real Application Clusters, each instance automatically archives its own redo log

files, or one or more instances can archive the redo log files for some or all of the instances in the cluster database.

If you operate your database in `NOARCHIVELOG` mode, then you can only make offline backups. If you cannot afford data loss, then Oracle strongly recommends that you use `ARCHIVELOG` mode.

See Also: *Oracle9i Real Application Clusters Administration* for more information about RMAN in Real Application Clusters

The remaining chapters in Part I describe the Real Application Clusters architecture and its scalability features.

Real Application Clusters Systems Architecture

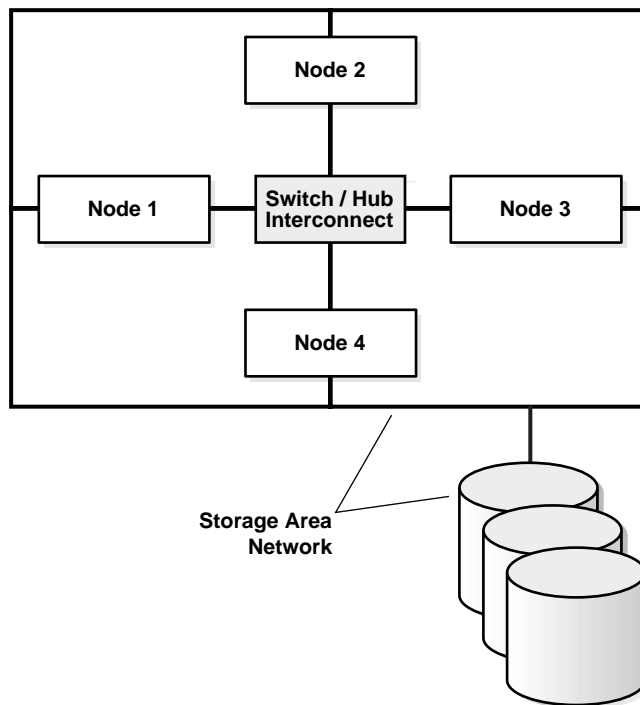
This chapter describes the system components and architectural models that typify most cluster database environments. It describes the hardware for nodes along with the hardware and software that unites the nodes into a cluster database. Topics in this chapter include:

- [Overview of Cluster Database System Components](#)
- [Memory, Interconnect, and Storage](#)
- [The High-Speed IPC Interconnect](#)
- [Shared Disk Storage and the Cluster File System Advantage](#)

Overview of Cluster Database System Components

A cluster database comprises two or more nodes that are linked by an **interconnect**. The interconnect serves as the communication path between each **node** in the cluster database. Each Oracle **instance** uses the interconnect for the messaging that synchronizes each instance's use of shared resources. Oracle also uses the interconnect to transmit data blocks that the multiple instances share. The primary type of shared resource is the datafiles that all the nodes access. **Figure 2-1** is a high-level view of how the interconnect links the nodes in a cluster database and how the cluster accesses the shared datafiles that are on storage devices.

Figure 2-1 Cluster Components for Cluster Database Processing



The cluster and its interconnect are linked to the storage devices, or **shared disk subsystem**, by a storage area network. The following sections describe the nodes and the interconnect in more detail:

- [Nodes and Their Components](#)

- [Cluster Interconnect and Interprocess Communication \(Node-to-Node\)](#)

Nodes and Their Components

A node has the following main components:

- **CPU**—The main processing component of a computer which reads from and writes to the computer's main memory.
- **Memory**—The component used for programmatic execution and the buffering of data.
- **Interconnect**—The communication link between the nodes.
- **Storage**—A device that stores data. This is usually persistent storage that must be accessed by read-write transactions to alter its contents.

You can purchase these components in many configurations. Their arrangement determines how the nodes access memory and storage.

Note: Oracle Corporation recommends that you deploy Real Application Clusters with configurations that have been certified for use with Real Application Clusters databases.

Cluster Interconnect and Interprocess Communication (Node-to-Node)

Real Application Clusters uses a high-speed [interprocess communication \(IPC\)](#) component for internode communications. The IPC defines the protocols and interfaces required for Real Application Clusters environments to transfer messages between instances. Messages are the fundamental units of communication in this interface. The core IPC functionality is built on an asynchronous, queued messaging model.

Memory, Interconnect, and Storage

All cluster databases use CPUs in generally the same way. However, you can deploy different configurations of memory, storage, and the interconnect for different purposes. The architecture on which you deploy Real Application Clusters depends on your processing goals.

Each node in a cluster database has one or more CPUs. Nodes with multiple CPUs are typically configured to share main memory. This enables you to deploy a scalable system.

The High-Speed IPC Interconnect

The high-speed **interprocess communication (IPC)** interconnect is a high-bandwidth, low latency communication facility that links the nodes in the cluster. The interconnect routes messages and other cluster communications traffic to coordinate each node's access to resources.

You can use Ethernet, a **Fiber Distributed Data Interface (FDDI)**, or other proprietary hardware for your interconnect. Also consider installing a backup interconnect in case your primary interconnect fails. The backup interconnect enhances high availability and reduces the likelihood of the interconnect becoming a single point-of-failure.

Real Application Clusters supports user-mode and memory-mapped IPCs. These types of IPCs substantially reduce CPU consumption and IPC latency.

Shared Disk Storage and the Cluster File System Advantage

Real Application Clusters requires that all nodes have simultaneous access to the shared disks to give the instances concurrent access to the database. The implementation of the shared disk subsystem is based on your operating system: you can use either a cluster file system or place the files on raw devices. Cluster file systems greatly simplify the installation and administration of Real Application Clusters.

Memory access configurations for Real Application Clusters are typically uniform. This means that the overhead for each node in the cluster to access memory is the same. However, typical storage access configurations are both uniform and non-uniform. The storage access configuration that you use is independent of your memory configuration.

As for memory configurations, most systems also use uniform disk access for Real Application Clusters databases. Uniform disk access configurations in a cluster database simplify disk access administration.

Real Application Clusters Software Architecture

This chapter describes the **Oracle Real Application Clusters**-specific architectural components. Some of these components are supplied with the Oracle database software while others are vendor-specific. Topics in this chapter include descriptions of the following Real Application Clusters components:

- **The Operating System-Dependent Clusterware**
- **Real Application Clusters Shared Disk Components**
- **Real Application Clusters-Specific Daemon and Instance Processes**
- **The Global Cache and Global Enqueue Service**

The Operating System-Dependent Clusterware

Real Application Clusters processing uses **operating system-dependent (OSD) clusterware** to access the operating system and for cluster-related service processing such as communicating information about instance startup and shutdown. Vendors provide the OSD clusterware for UNIX operating systems, and Oracle provides the OSD clusterware for Windows NT and Windows 2000 operating systems. The OSD has the following subcomponents:

- **The Cluster Manager**
- **The Node Monitor**
- **The Interconnect**

The Cluster Manager

The **cluster manager (CM)** oversees internode messaging that travels over the **interconnect** to coordinate internode operations. The cluster manager also provides a global view of the cluster and the nodes and instances that are members of it. The cluster manager also controls cluster membership.

The Node Monitor

The cluster manager includes a subset of functionality known as a node monitor. The node monitor polls the status of each **resource** in the cluster including the nodes, the interconnect hardware and software, and the shared disks. In the Oracle-supplied Cluster Manager for Windows, the node monitor also polls the Oracle instances.

The cluster manager informs clients and the Oracle server when the status of cluster resources change. This is because Real Application Clusters manages cluster membership by reconfiguring the cluster database when a joining instance registers with the cluster manager or when an existing instance disconnects from it.

The node monitor also serves the cluster manager by:

- Providing node management interface modules
- Discovering and tracking the membership states of the nodes by providing a common view of node membership across the cluster
- Detecting and diagnosing changes in the states of active nodes and communicating information about those change events

The Interconnect

The interprocess communication (IPC) software, or interconnect, as described in [Chapter 2](#), is another key OSD component. The IPC controls messaging among the nodes. Real Application Clusters also uses the IPC to transfer data blocks between instances.

Real Application Clusters Shared Disk Components

Real Application Clusters databases have the same components as single-instance Oracle databases. These include one or more control files, a set of online redo log files, the optional archive log files, datafiles, and so on. Therefore, you must provide shared disk access for each control file and datafile and for each online redo log member of every redo log group. You must also configure shared disks for the undo tablespace datafiles to use the recommended automatic undo management feature.

For Windows NT and Windows 2000 only, you must provide shared access to a **voting** or **quorum disk** on which Oracle stores cluster configuration information. You can place this disk on a cluster file system (CFS) or on a raw device. The node monitor uses the quorum disk configuration information to manage the cluster configuration.

The Oracle configuration and administrative tools also require access to cluster configuration data stored on shared disks. You must configure a shared disk resource to use the Database Configuration Assistant (DBCA), Oracle Enterprise Manager (EM), and the Server Control (SRVCTL) command-line administrative utility. On Windows NT and Windows 2000, the voting disk and configuration data share the same disk resource.

Parameter file management is simplified in Real Application Clusters if you use the server parameter file. Store this file on a shared disk to administer global and instance-specific parameter settings in one file.

See Also: *Oracle9i Real Application Clusters Setup and Configuration* for information about tablespace requirements and *Oracle9i Real Application Clusters Administration* for more information about using the server parameter file in Real Application Clusters

Real Application Clusters-Specific Daemon and Instance Processes

This section describes the Real Application Clusters-specific daemon and instance processes under the following headings:

- [The Global Services Daemon](#)
- [Instance Processes Specific to Real Application Clusters](#)

The Global Services Daemon

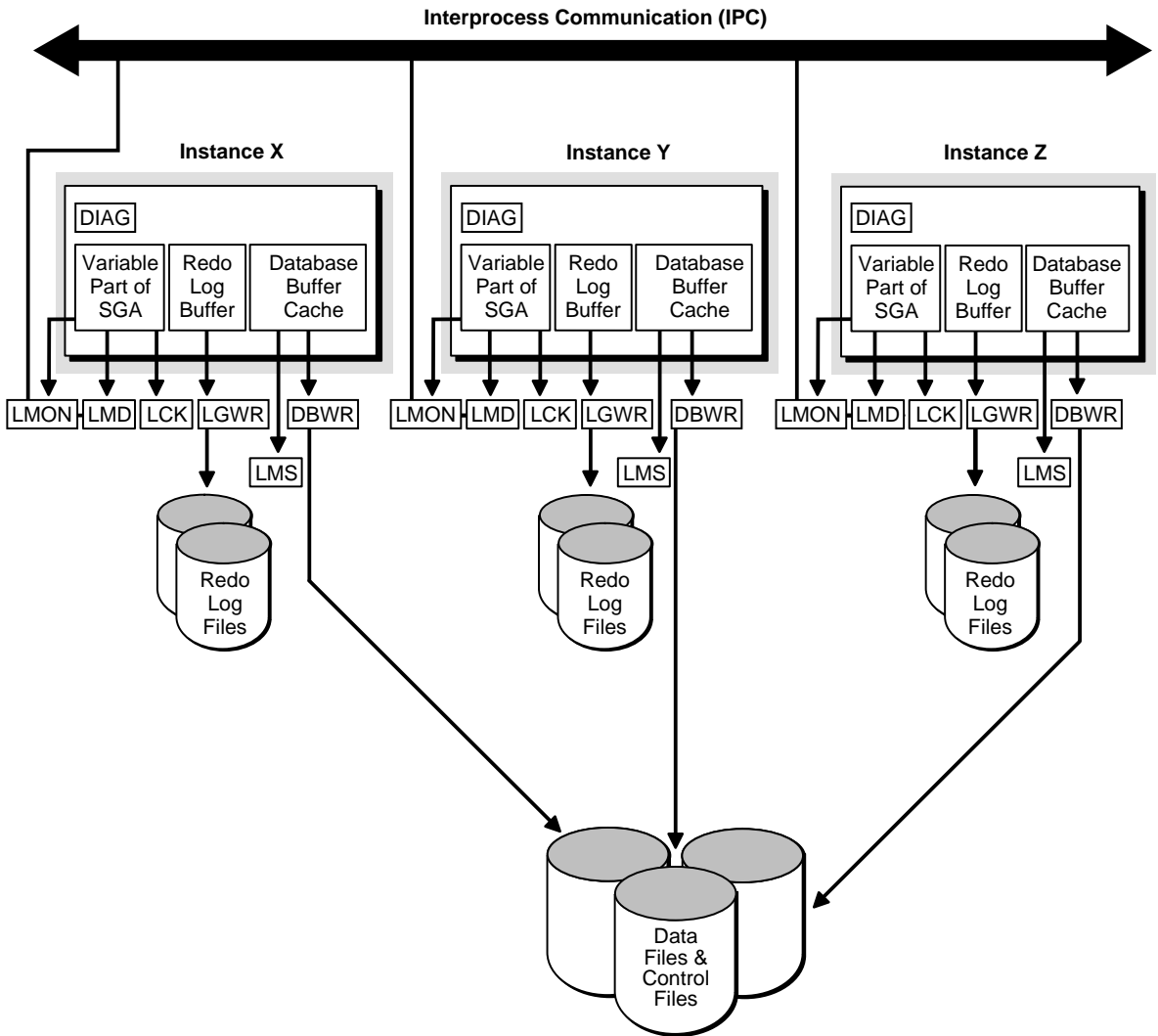
The Global Services Daemon (GSD) runs on each node with one GSD process per node. The GSD coordinates with the cluster manager to receive requests from clients such as the DBCA, EM, and the SRVCTL utility to execute administrative job tasks such as instance startup or shutdown. The GSD is not an Oracle instance background process and is therefore not started with the Oracle instance.

Instance Processes Specific to Real Application Clusters

A Real Application Clusters database has the same processes as single-instance Oracle databases such as process monitor (PMON), database writer (DBWR n), log writer (LGWR), and so on. There are also additional Real Application Clusters-specific processes as shown in [Figure 3-1](#). The exact names of these processes and the trace files that they create are platform-dependent.

- Global Cache Service Processes (LMS n), where n ranges from 0 to 9 depending on the amount of messaging traffic, control the flow of messages to remote instances and manage global data block access. LMS n processes also transmit block images between the buffer caches of different instances. This processing is part of the Cache Fusion feature.
- The Global Enqueue Service Monitor (LMON) monitors global enqueues and resources across the cluster and performs global enqueue recovery operations. Enqueues are shared memory structures that serialize row updates.
- The Global Enqueue Service Daemon (LMD) manages global enqueue and global resource access. Within each instance, the LMD process manages incoming remote resource requests.
- The Lock Process (LCK) manages non-Cache Fusion resource requests such as library and row cache requests.
- The Diagnosability Daemon (DIAG) captures diagnostic data about process failures within instances. The operation of this daemon is automated and it updates an alert log file to record the activity that it performs.

Figure 3-1 Real Application Clusters-Specific Instance Processes



The Global Cache and Global Enqueue Service

The **Global Cache Service (GCS)** and **Global Enqueue Service (GES)** are integrated components of Real Application Clusters that coordinate simultaneous access to the shared database and to shared resources within the database. These services maintain consistency and data integrity. The GCS and GES on each instance, as well as the Cluster Manager, use the IPC to communicate between instances and within the cluster. This section describes the following features of the GCS and GES:

- [Application Transparency](#)
- [Global Resource Directory with Distributed Architecture](#)
- [Resource Mastering and Affinity](#)
- [GCS and GES Interaction with the Cluster Manager](#)

Application Transparency

The coordination of access to resources that is performed by the GCS and GES is transparent to applications. Applications in Real Application Clusters use the same concurrency mechanisms as in single-instance Oracle databases.

Global Resource Directory with Distributed Architecture

The GCS and GES maintain a **Global Resource Directory** to record information about resources. The Global Resource Directory resides in memory, is distributed throughout the cluster, and is available to all active instances. In this distributed architecture, each node participates in the management of information in the directory. This distributed scheme provides **fault tolerance** and enhanced runtime performance.

The GCS and GES ensure the integrity of the Global Resource Directory even if multiple nodes fail. The shared database is always accessible if at least one instance is active after recovery is completed. The fault tolerance of the resource directory also enables Real Application Clusters instances to start and stop at any time, in any order.

Resource Mastering and Affinity

The Global Cache and Global Enqueue Services maintain information about each resource within a cluster. The GCS and GES nominate one instance to manage *all* information about a particular resource. This instance is called the **resource master**. The GCS evaluates resource mastering periodically and changes the resource master based on data access patterns. This reduces network traffic as well as resource acquisition time.

GCS and GES Interaction with the Cluster Manager

The GCS and GES operate independently of the cluster manager. However, these services rely on the cluster manager for timely and correct information about the statuses of the instances in the cluster. If these services cannot obtain the information they need from a particular instance, then Oracle shuts down the unresponsive instance. This ensures the integrity of Real Application Clusters databases because each instance must be aware of all other active instances to coordinate shared disk access.

Scalability in Real Application Clusters

This chapter provides detail about the scalability features of Real Application Clusters and includes the following topics:

- [Scalability Features of Real Application Clusters](#)
- [All System Types Benefit from Real Application Clusters](#)
- [Levels of Scalability](#)

Scalability Features of Real Application Clusters

You can implement Real Application Clusters using several features that enhance application performance and maintain high availability levels. These features:

- Balance workloads among the nodes by controlling multiple server connections during periods of heavy use
- Provide persistent, fault tolerant connections between clients and your Real Application Clusters database

If you install Real Application Clusters with one of the preconfigured databases, then the **Database Configuration Assistant (DBCA)** automatically configures most of these features.

See Also: *Oracle9i Net Services Administrator's Guide* for information about manually configuring **scalability** features

All System Types Benefit from Real Application Clusters

This section describes why all system types benefit from Real Application Clusters. It includes the following topics:

- [Transaction Systems and Real Application Clusters](#)
- [Data Warehouse Systems and Real Application Clusters](#)

Note: If your application was well designed and ran efficiently on a single-instance Oracle database, then it will scale well on Real Application Clusters.

Transaction Systems and Real Application Clusters

Transaction systems are characterized by relatively brief database update transactions. Applications for **transaction systems**, which include e-business and traditional online transaction processing (OLTP) systems, perform well on Real Application Clusters. This is because Real Application Clusters provides excellent scale up. Cache Fusion enables updates to occur from multiple instances without overhead.

Because transaction systems manage more transactions as additional users connect to the system, you can increase capacity by dynamically adding nodes to the cluster. However, poorly designed systems do not scale well on either single-instance Oracle databases or on Real Application Clusters databases—poorly designed applications will likely experience performance degradation as demand increases.

Data Warehouse Systems and Real Application Clusters

Data warehouse systems perform well on Real Application Clusters because applications with low update activity can access the database through different instances without additional overhead. If the data blocks are not modified, then multiple instances can read the same blocks into their buffer caches and perform queries on the blocks without additional I/O. Data warehouse systems usually benefit from scale up and may experience speed up as well.

Oracle Parallel Execution in Real Application Clusters

Real Application Clusters provides the framework for parallel execution. Parallel execution performs efficiently in Real Application Clusters because it can distribute portions of a large SQL statement across multiple instances. The transaction is completed more quickly because it executes on multiple CPUs.

In Real Application Clusters, Oracle determines at runtime whether it will run parallel execution server processes on only one instance, or whether it will run processes on multiple instances. In general, Oracle tries to use only one instance when sufficient resources are available. This reduces cross-instance message traffic.

Levels of Scalability

Implementing cluster databases requires optimal scalability on the following levels as described in this section:

- [Network Scalability](#)
- [Network Scalability and Client/Server Connectivity](#)
- [Operating System Scalability](#)

Note: Inappropriately designed applications might not fully use the potential scalability of a system. Likewise, regardless of how well your applications scale, you will not obtain optimal performance if you attempt to deploy your applications on hardware that does not scale.

Network Scalability

Interconnects are a key factor in cluster scalability. That is, every system must have a connection to the CPUs, whether the connection is a high-speed bus or a low-speed Ethernet connection. Bandwidth and latency of the interconnect then determine the scalability of the hardware.

Network Scalability as a Function of Bandwidth and Latency

Hardware scalability requires very low latency and most interconnects have sufficient bandwidth to accommodate Real Application Clusters internode processing. Other internode operations, such as parallel execution, also rely on high bandwidth.

Network Scalability and Client/Server Connectivity

In Real Application Clusters, client/server connections are established using several subcomponents. A client submits a connection request to a listener process residing on the target node. The listener grants a connection to a client so the client can access a particular node based on several factors. How the listener grants the connection depends on how you configure the connection options as described under the following headings:

- [Connection Load Balancing Feature](#)
- [Enhanced Network Scalability and the Shared Server Feature](#)

Connection Load Balancing Feature

The connection load balancing feature provides exceptional benefits in Real Application Clusters environments where there are multiple instances and dispatchers. Connection load balancing improves performance by balancing the number of connections among various instances and shared server dispatchers for the same service. Note that if your application is partitioned, you may not want to use this feature.

Because of service registration's ability to cross-register with remote listeners, a listener is always aware of all instances. Thus a listener can send an incoming client request for a specific service to the least-loaded instance and least-loaded dispatcher. Service registration also facilitates connect-time failover and client load balancing that you can use with or without a shared server. In addition, Oracle dynamically updates the initialization parameters that control connection load balancing when you add or remove nodes from your Real Application Clusters database.

Connect-Time Failover for Multiple Listeners Service registration informs listeners whether an instance is running or stopped. This enables connect-time failover when multiple listeners support a service. To use this feature, configure a client to fail over client requests to a different listener if the initial listener connection fails. The reconnection attempts continue until the client successfully connects to a listener. If an instance fails, then a listener returns a network error.

Client Load Balancing and Connection Load Balancing for Multiple Listeners Oracle Net provides client load balancing and connection load balancing. When more than one listener supports a service, a client can randomly send connection requests to the various listeners. The random nature of the connection requests distributes the load to avoid overburdening a single listener.

In addition to load balancing, clients can also specify that their connection request automatically fails over to a specific listener if a connection cannot be made with the listener chosen at random. A connection could fail either because the specified listener is not up or because the instance is not up and the listener therefore cannot accept the connection.

Configuring Client Load and Connection Load Balancing To enable client load balancing, configure a service in your initialization parameter file and in the `tnsnames.ora` file. To do this, set the `LOAD_BALANCE` parameter to `ON`, `YES`, or `TRUE` and provide the same service name in the connect descriptor.

Note: Oracle Net uses the PMON process to automatically register service information. Therefore, you do not need to modify a `listener.ora` file with static instance information.

For connection load balancing, connections to the dispatchers on each node are based on the number of active connections. Oracle Net assigns new connections to the node that has the lowest processing load and fewest connections. If you have configured shared servers, then Oracle Net connects to the least loaded dispatcher.

Note: Clients must be connected to server machines in a scalable manner. This means that your network must also be scalable!

See Also: *Oracle9i Net Services Administrator's Guide* for additional conceptual and configuration information about the features described in this section

Enhanced Network Scalability and the Shared Server Feature

The shared server enables enhanced scalability in that for a given amount of memory you can support more users than when using dedicated servers. With shared server, the incremental memory cost as you add a user is less than when you use a dedicated server.

An increasing number of Oracle9i features require the shared server such as:

- Oracle9i JServer
- Connection Pooling
- Connection Concentration Feature
- Advanced Queueing

Service Registration and Shared Server Functionality An important feature of shared server is **service registration**. This feature provides the listener with at least one **service name** and **instance name** as well as the network addresses of the database. It also provides information about the current state and load of all instances and shared server dispatchers. With this information, the listener forwards client connection requests to the appropriate dispatchers and dedicated servers.

Because this information is registered with the listener, you do not need to configure the **listener.ora** file with static database service information. Service registration also extends benefits to connection load balancing which is a feature of the shared server.

Note: Service registration is also available in dedicated servers. However, dedicated servers do not provide the same scalability as shared servers.

Operating System Scalability

The scalability of your Real Application Clusters database also depends on your operating system's scalability. Software scalability can be an important issue if one node is a shared memory system—that is, a system where multiple CPUs connect to a symmetric multiprocessor with a single memory location. Methods of synchronization in the operating system can determine the scalability of the system. In asymmetrical multiprocessing, for example, only a single CPU can handle I/O interrupts.

The next part of this book, Part II, provides detail about how the Global Cache and Global Enqueue Services coordinate resources within Real Application Clusters.

Part II

Resource Coordination in Real Application Clusters

Part II describes the Real Application Clusters resource coordination that synchronizes data access and ensures data integrity. The chapters in Part II are:

- [Chapter 5, "Real Application Clusters Resource Coordination"](#)
- [Chapter 6, "Cache Fusion and the Global Cache Service"](#)
- [Chapter 7, "Resource Coordination by the Global Enqueue Service"](#)

Real Application Clusters Resource Coordination

This chapter provides an overview of **Real Application Clusters** resource coordination and resource management. The topics in this chapter include:

- [Overview of Real Application Clusters Resource Coordination](#)
- [Resource Modes and Roles](#)
- [System Change Number Processing](#)

Overview of Real Application Clusters Resource Coordination

Cluster operations require synchronization among all instances to control shared access to resources. Real Application Clusters uses the **Global Resource Directory** to record information about how resources are used within a cluster database. The **Global Cache Service (GCS)** and **Global Enqueue Service (GES)** manage the information in this directory. Each instance maintains part of the global resource directory in the System Global Area (SGA).

Resource coordination within Real Application Clusters occurs at both an instance level and at a cluster database level. Instance level resource coordination within Real Application Clusters is referred to as **local** resource coordination. Cluster level coordination is referred to as **global** resource coordination.

The processes that manage local resource coordination in a cluster database are identical to the local resource coordination processes in single instance Oracle. This means that row and block level access, space management, system change number (SCN) creation, and data dictionary cache and library cache management are the same in Real Application Clusters as in single instance Oracle. Global resource coordination, however, is somewhat more complex as described later in this chapter.

The Contents of the Global Resource Directory

Each instance's portion of the Global Resource Directory contains information about the current statuses of all shared resources. Oracle also uses information in the Global Resource Directory during instance failure recovery or cluster reconfigurations.

The types of shared resource information in the Global Resource Directory include:

- Data block identifiers, such as a data block addresses
- Locations of the most current versions of data blocks if they have been read into buffer caches on multiple nodes in the cluster
- The **modes** in which data blocks are being held by instances: (N) Null, (S) Shared, or (X) Exclusive
- The **roles** in which data blocks are being held by each instance: local or global

Real Application Clusters Synchronization Processes

When an instance requests a resource, such as a data block, Real Application Clusters locally manages the instance's acquisition of the block. If this block is later modified by one or more instances, then Oracle performs further synchronization on a global level to permit shared access to this block across the cluster. Synchronization in this case requires intranode messaging as well as the preparation of consistent read versions of the block and the transmission of copies of the block between memory caches within the cluster database.

The **Global Cache Service Processes (LMSn)** locate, prepare, and transmit the most current block images when one or more instances request a data block that is being updated by another instance. To coordinate GCS processing, the GES (GES) transmits intranode messages over the interconnect.

Note: Oracle uses shared memory for cluster database processing within a node. Therefore, messaging is not necessary.

Therefore, in Real Application Clusters some elements of local coordination become global when remote instances require locally held resources. This is especially true for enqueues and past images of data blocks which are described in the following sub-sections.

Enqueues

An **enqueue** is a memory structure that serializes updates to a particular row. Coordination of enqueues is not needed until another instance requires the resource to which an enqueue is assigned.

You do not have to configure global enqueues. Their number is calculated automatically at startup and Oracle records the calculated values in the `alert.log` file.

Past Images

Past images are copies of dirty blocks that Oracle maintains until writes affecting the versions of information in the dirty blocks are complete. The GCS manages past images and the GCS also uses them in failure recovery.

Resource Modes and Roles

As a result of interinstance block transfers, the same data block can exist in multiple caches. The block can also be held by multiple instances in different modes depending on whether a holding instance modifies the data or merely reads the contents of the block. The modes and roles that the GCS uses for resource coordination within a cluster, such as the sharing of data blocks, are described under the following headings:

- [Resource Modes](#)
- [Resource Roles](#)

Resource Modes

A **resource mode** determines whether the holder of the resource can modify it. [Table 5–1](#) compares the three modes, **null (N) mode**, **shared (S) mode**, and **exclusive (X) mode**.

Table 5–1 Global Cache Service Resource Modes

Resource Mode	Identifier	Description
Null	N	Holding a resource at this level does not convey access rights.
Shared	S	A protected read. When a resource is held at this level, an instance cannot modify it. Multiple instances can read the resource.
Exclusive	X	When a resource is held at this level, it grants the holding instance exclusive access. Other instances cannot write to the resource. Consistent reads of older resources may still be available.

Resource Roles

Oracle assigns a **resource role** to the resource on the holding instance. The roles are either local or global and are therefore mutually exclusive. The evolution of local-to-global roles is as follows:

- When a block is first read into an instance's cache and other instances have not read the block, the block is **locally managed**. The GCS therefore assigns a *local* role to the block.

- If the block has been modified by the local instance and transmitted to another instance, then it is considered **globally managed** and the GCS assigns a *global* role to the block.

All resources have the local role if they only exist in only one cache. If a data block was changed in one instance and subsequently transferred to another instance, then the buffer containing the data block is considered globally **dirty** and the resource therefore has a global role.

Global Cache Service Operations

The GCS tracks the locations, modes, and roles of data blocks. The GCS therefore also manages the access privileges of various instances in relation to resources. Oracle uses the GCS for cache coherency when the current version of a data block is in one instance's buffer cache and another instance requests that block for modification.

If an instance reads a block in exclusive mode, then in subsequent operations multiple transactions *within* the instance can share access to a set of data blocks without using the GCS. This is only true, however, if the block is not transferred out of the local cache. If the block is transferred out of the local cache, then the GCS updates the **Global Resource Directory** that the resource has a global role; whether the resource's mode converts from exclusive to another mode depends on how other instances use the resource.

Global Cache Service Processing Example

Assume that an instance in a cluster database needs to update a data block. At the same time, another instance needs to update the same block. Without the cache coherency provided by the GCS, both instances could simultaneously update the same block. However, due to the synchronization controls of the GCS, only one instance can update the block; the other instance must wait.

By ensuring that only one instance can update a block at any time, the GCS maintains cache coherency. The blocks do not have to be held by an instance in exclusive mode until the instance's transaction completes. For example, one instance can modify a row in a block while a transaction from another instance modifying a different row in the same block has not committed its changes.

Cache Coherency and the Global Cache Service

The GCS ensures cache coherency by requiring instances to acquire a resource at a cluster-wide level before modifying a database block. Thus, the GCS synchronizes global cache access, allowing only one instance at a time to modify a block.

Oracle's multi-versioning architecture distinguishes between current data blocks and one or more **consistent read (CR)** versions of a block. The current block contains changes for all committed and yet-to-be-committed transactions.

A **consistent read (CR)** version of a block represents a consistent snapshot of the data at a point in time. The LMS_n processes produce consistent read versions by applying rollback segment information to past images. Both current and consistent read blocks are managed by the GCS.

If one instance holds a block that it has modified and another instance requests it, then the holding instance maintains a past image (PI) of the block. In the event of failure, Oracle can reconstruct current and consistent read versions of the block by reading PIs.

System Change Number Processing

To synchronize resources within a cluster, especially data blocks, Oracle must track successive generations of data block changes. That is, Oracle records each change made to a data block by assigning a numeric identifier to each version of a block. This enables Oracle to generate redo logs in an orderly manner and to accommodate subsequent recovery processing.

Oracle uses a logical timestamp known as a system change number (SCN) to order data block change events within each instance and across all instances. The main purpose of SCNs is to mark redo log entries to synchronize recovery processing.

Oracle assigns an SCN to each transaction. Conceptually, there is a global serial point that generates SCNs. In practice, however, SCNs can be read and generated in parallel. One of the SCN generation schemes is called **Lamport SCN Generation**.

In single instance Oracle, the System Global Area (SGA) maintains and increments SCNs from an instance that has mounted the database in exclusive mode. In Real Application Clusters, the SCN must be maintained globally and its implementation can vary by platform. The SCN can be managed by the GCS, by the Lamport SCN generation scheme, or by using a hardware clock or dedicated SCN server.

Lamport SCN Generation

The Lamport SCN generation scheme is efficient and scalable because it generates SCNs in parallel on all instances. In this scheme, all messages between instances carry SCNs. Multiple instances can generate SCNs in parallel without additional communication among these instances.

On most platforms, Oracle uses the Lamport SCN generation scheme when the value for `MAX_COMMIT_PROPAGATION_DELAY` is larger than a platform-specific threshold. This is generally the default and this value is typically set to 7 seconds. Note that by changing this value you change the SCN generator used by your cluster database.

Examine the `alert.log` after instance startup to determine whether the Lamport SCN generation scheme is in use. If the value for `MAX_COMMIT_PROPAGATION_DELAY` is smaller than the threshold value, then Oracle uses the hardware clock SCN generation scheme.

[Chapter 6](#) describes Cache Fusion processing in more detail.

Cache Fusion and the Global Cache Service

This chapter describes **Cache Fusion** processing and explains **Global Cache Service (GCS)** operations. It also explains the resource control mechanisms in Real Application Clusters, illustrates common Cache Fusion scenarios, and describes recovery processing in Real Application Clusters. The topics in this chapter include:

- [Overview of Cache Fusion Processing](#)
- [Resource Control, Cache-to-Cache Transfer, and Cache Coherency](#)
- [Block Access Modes and Buffer States](#)
- [Cache Fusion Scenarios](#)
- [Real Application Clusters Recovery and Cache Fusion](#)

Overview of Cache Fusion Processing

By default, a resource is allocated for each data block that resides in the cache of an instance. Due to **Cache Fusion** and the elimination of disk writes that occur when other instances request blocks for modifications, the performance overhead to manage shared data between instances is greatly diminished. Not only do Cache Fusion's concurrency controls greatly improve performance, but they also reduce the administrative effort for Real Application Clusters environments.

Cache Fusion addresses several types of concurrency as described under the following headings:

- [Concurrent Reads on Multiple Nodes](#)
- [Concurrent Reads and Writes on Different Nodes](#)
- [Concurrent Writes on Different Nodes](#)

Concurrent Reads on Multiple Nodes

Concurrent reads on multiple nodes occur when two instances need to read the same data block. Real Application Clusters resolves this situation without synchronization because multiple instances can share data blocks for read access without cache coherency conflicts.

Concurrent Reads and Writes on Different Nodes

A read request from an instance for a block that was modified by another instance and not yet written to disk can be a request for either the current version of the block or for a read-consistent version. In either case, the **Global Cache Service Processes (LMSn)** transfer the block from the holding instance's cache to the requesting instance's cache over the interconnect.

Concurrent Writes on Different Nodes

Concurrent writes on different nodes occur when the same data block is modified frequently by different instances. In such cases, the holding instance completes its work on the data block after receiving a request for the block. The GCS then converts the resources on the block to be globally managed and the LMSn processes transfer a copy of the block to the cache of the requesting instance. The main features of this processing are:

- The **Global Cache Service (GCS)** tracks a each version of a data block, and each version is referred to as a **past image (PI)**. In the event of a failure, Oracle can reconstruct the current version of a block by using the information in a PI.
- The cache-to-cache data transfer is done through the high speed IPC interconnect, thus eliminating disk I/O.
- Cache Fusion limits the number of **context switches** because of the reduced sequence of round trip messages. Reducing the number of context switches enables greater cache coherency protocol efficiency. The database writer (DBWn) processes are not involved in Cache Fusion block transfers.

Write Protocol and Past Image Tracking

When an instance requests a block for modification, the **Global Cache Service Processes (LMSn)** send the block from the instance that last modified it to the requesting instance. In addition, the LMSn process retains a PI of the block in the instance that originally held it.

Writes to disks are only triggered by cache replacements and during checkpoints. For example, consider a situation where an instance initiates a write of a data block and the block's resource has a global role. However, the instance only has the PI of the block and not the most current buffer. Under these circumstances, the instance informs the GCS and the GCS forwards the write request to the instance where the most recent version of the block is held. The holder then sends a completion message to the GCS. Finally, all other instances with PIs of the block delete them.

Resource Control, Cache-to-Cache Transfer, and Cache Coherency

The GCS assigns and opens resources for each data block read into an instance's buffer cache. Oracle closes resources when the resources do not manage any more buffers or when buffered blocks are written to disk due to cache replacement. When Oracle closes a resource, it returns the resource to a list from which Oracle can assign new resources.

Block Access Modes and Buffer States

An additional concurrency control concept is the **buffer state** which is the state of a buffer in the local cache of an instance. The buffer state of a block relates to the access mode of the block. For example, if a buffer state is **exclusive current (XCUR)**, an instance owns the resource in exclusive mode.

To see a buffer's state, query the `STATUS` column of the `V$BH` dynamic performance view. The `V$BH` view provides information about the block access mode and their buffer state names as follows:

- With a block access mode of `NULL` the buffer state name is `CR`—An instance can perform a consistent read of the block. That is, if the instance holds an older version of the data.
- With a block access mode of `S` the buffer state name is `SCUR`—An instance has shared access to the block and can only perform reads.
- With a block access mode of `X` the buffer state name is `XCUR`—An instance has exclusive access to the block and can modify it.
- With a block access mode of `NULL` the buffer state name is `PI`—An instance has made changes to the block but retains copies of it as past images to record its state before changes.

Only the `SCUR` and `PI` buffer states are Real Application Clusters-specific. There can be only one copy of any one block buffered in the `XCUR` state in the cluster database at any time. To perform modifications on a block, a process must assign an `XCUR` buffer state to the buffer containing the data block.

For example, if another instance requests read access to the most current version of the same block, then Oracle changes the access mode from exclusive to shared, sends a current read version of the block to the requesting instance, and keeps a `PI` buffer if the buffer contained a **dirty block**.

At this point, the first instance has the current block and the requesting instance also has the current block in shared mode. Therefore, the role of the resource becomes global. There can be multiple **shared current (SCUR)** versions of this block cached throughout the cluster database at any time.

Cache Fusion Scenarios

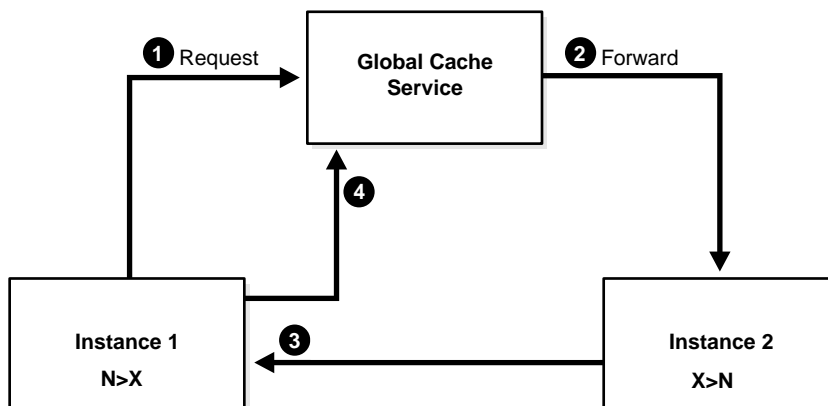
The following scenarios illustrate the most important points of Cache Fusion processing. These scenarios do not address all possible configurations. For example, this section does not describe read operations. The scenarios in this section are:

- Requesting a Changed Block for a Modification Operation
- Writing Blocks to Disk

Requesting a Changed Block for a Modification Operation

The scenario in [Figure 6–1](#) assumes that the data block has been changed, or **dirty**, by only one instance and held in exclusive mode (X). Furthermore, this scenario assumes that the block has only been accessed by the instance that changed it. That is, only one copy of it exists cluster-wide. In other words, the block has a local role (L).

Figure 6–1 Requesting a Changed Block for a Modification Operation



1. The instance attempting to modify the block, instance 1, submits a request to the GCS.
2. The GCS transmits the request to the holder, instance 2.
3. Instance 2 receives the message and the LMS process sends the block to instance 1. Before sending the block, the resource is downgraded in instance 2 from exclusive to null mode (N) and instance 2 retains the dirty buffer as a PI. The role changes to global (G) because the block may become dirty in more than one instance. Along with the block, instance 2 communicates to the requesting instance that instance 2 retained a PI in null (N) mode. In the same message, instance 2 also specifies that the requestor must retain the block in exclusive (X) mode and with a global (G) role.

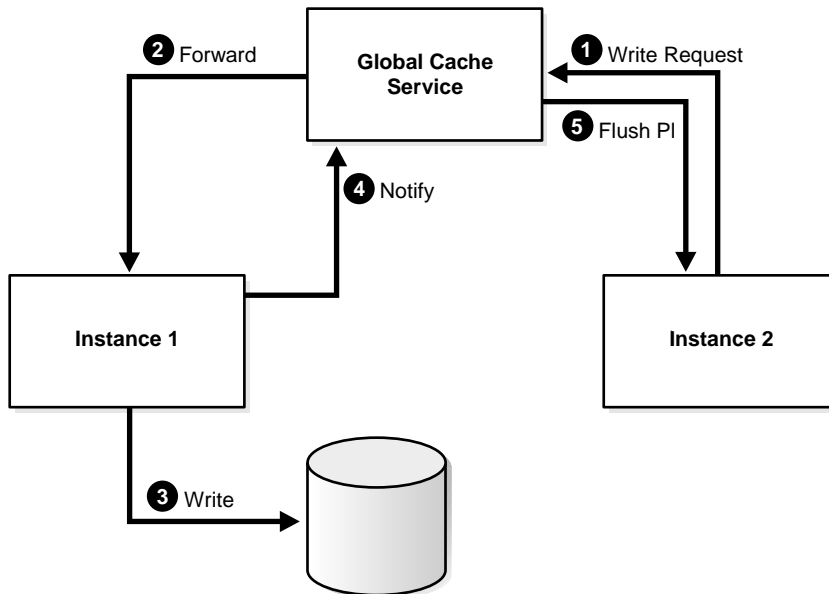
- On receipt of the block, instance 1 informs the GCS that it holds the block in exclusive mode and with a global role. Note that the data block is not written to disk before the resource is granted to instance 1.

Writing Blocks to Disk

The scenario in [Figure 6–2](#) illustrates how an instance can perform a checkpoint at any time or replace buffers in the cache due to free buffer requests. Because multiple versions of the data block with changes could exist in the caches of instances in the cluster, a write protocol managed by the GCS ensures that only the most current version of the data is written to disk. It must also ensure that all previous versions are purged from the other caches. A write request for a data block can originate in any instance that has the current or past image of the block.

In this scenario, assume that the instance holding a PI buffer in null mode requests that Oracle write the buffer to disk.

Figure 6–2 *Writing Blocks to Disk*



- Instance 2 sends a write request to the GCS.
- The GCS forwards the request to instance 1, the current block holder.

3. Instance 1 receives the write request and writes the block to disk.
4. Instance 1 records the completion of the write operation with the GCS and informs the GCS that the resource role can become local because instance 1 performed the write of the current block.
5. After receipt of the notification, the GCS orders all PI holders to discard, or *flush*, their PIs; the PIs are no longer needed for recovery. The buffer is free and the resource previously held in null mode is closed.

The next section explains Real Application Clusters recovery.

Real Application Clusters Recovery and Cache Fusion

In Real Application Clusters recovery, the amount of recovery processing required after node failures is proportional to the number of failed nodes. In general, data blocks become available immediately after they are recovered.

When an instance fails and the failure is detected by another Oracle instance, Oracle performs the following recovery steps:

1. During the first phase of recovery, which is the GES reconfiguration, Oracle first reconfigures the GES enqueues. Then Oracle reconfigures the GCS resources. During this time, all GCS resource requests and write requests are temporarily suspended. However, processes and transactions can continue to modify data blocks as long as these processes and transactions have already acquired the necessary enqueues.
2. After the reconfiguration of enqueues that the GES controlled, a log read and the remastering of GCS resources occur in parallel. At the end of this step the block resources that need to be recovered have been identified.
3. Buffer space for recovery is allocated and the resources that were identified in the previous reading of the log are claimed as recovery resources. Then, assuming that there are PIs of blocks to be recovered in other caches in the cluster database, resource buffers are requested from other instances. The resource buffers are the starting point of recovery for a particular block.
4. All resources and enqueues required for subsequent processing have been acquired and the Global Resource Directory is now **unfrozen**. Any data blocks that are not in recovery can now be accessed. Note that the system is already partially available.

5. The cache layer recovers and writes each block identified in step 2, releasing the recovery resources immediately after block recovery so that more blocks become available as cache recovery proceeds.
6. After all blocks have been recovered and the recovery resources have been released, the system is again fully available. Recovered blocks are available after recovery completes.

In summary, the recovered database or recovered portions of the database become available earlier, and before the completion of the entire recovery sequence. This makes the system available sooner and it makes recovery more scalable.

If neither the PI buffers nor the current buffer for a data block are in any of the surviving instances' caches, then Oracle performs a log merge of the failed instances. As mentioned for recovery in general, the performance overhead of a log merge is proportional to the number of failed instances and to the size of the redo logs for each instance. You can, however, control the size of the log with Oracle's checkpoint features. With its advanced design, Real Application Clusters recovery can manage multiple simultaneous failures and sequential failures. The shared server feature is also resilient to instance failures during recovery.

See Also: *Oracle9i Real Application Clusters Administration* for more information about recovery in Real Application Clusters

[Chapter 7](#) describes the resource coordination performed by the Global Enqueue Service.

Resource Coordination by the Global Enqueue Service

This chapter describes the interinstance resource coordination messaging that the **Global Enqueue Service (GES)** performs in Real Application Clusters. The GES coordinates interinstance coordination using **resource**, data, and interinstance data requests. Topics in this chapter include:

- [Global Enqueue Service Processing](#)
- [Global Enqueue Concurrency Control](#)
- [Resources Managed by the Global Enqueue Service](#)

See Also: [Chapter 6, "Cache Fusion and the Global Cache Service"](#) for information about the [Global Cache Service \(GCS\)](#)

Global Enqueue Service Processing

The GES manages all non-Cache Fusion intrainstance resource operations and tracks the status of all Oracle enqueueing mechanisms. The GES does this for resources that are accessed by more than one instance. The primary resources that the GES controls are dictionary cache locks and library cache locks. The GES manages the communication regarding these resources that occurs between instances. However, these resources do not protect datafile blocks. The GES also performs deadlock detection to all deadlock sensitive enqueues and resources.

Global Enqueue Concurrency Control

Other parts of an Oracle database also globally protect their data structures. These other parts are the:

- Data Dictionary Access Layer
- Library Cache Layer
- Transaction Management Code Layer

Each layer has specific protocols that other layers use to access their respective data structures. All layers of Oracle use the services of the GES API to acquire, convert, and release resources that reside in these layers. The GES uses parameters that affect global enqueue resources to automatically calculate the exact memory requirements for enqueue resource management operations when an instance starts up.

At the transaction layer, global enqueue resources are sometimes held only for a very short time and quickly released. Normally, TX locks, for example, are acquired when a transaction starts and they are released when the transaction commits. In some cases they are used to signal an event, such as in the case of library cache locks where DDL statements may invalidate library cache objects. In general, transaction and table lock processing operate the same way in Real Application Clusters as they do in single-instance Oracle databases. The next section more closely examines the resources managed by the GES.

Resources Managed by the Global Enqueue Service

This section explains how Oracle uses GES enqueues to manage concurrency for resources that operate *on* transactions, tables, and other entities within an Oracle Real Application Clusters environment. The following resources are local in single-instance Oracle databases, but they are global when they are under the control of the GES:

- [Dictionary Cache Locks](#)
- [Library Cache Locks](#)

Dictionary Cache Locks

Each Real Application Clusters database instance has a dictionary cache, or **row cache**, containing data dictionary information. The data dictionary structure is the same for Oracle instances in a cluster database as it is for instances in single instance Oracle. However, in Real Application Clusters, Oracle synchronizes all the dictionary caches throughout the cluster. Real Application Clusters uses latches to do this, just as in single-instance Oracle databases.

Consider a five-node Real Application Clusters environment in which a user drops a table on one node. Because each dictionary cache, of which there are five, may have a copy of the definition of the dropped table, the node dropping the table from its cache must inform the other four dictionary caches to drop their copies of the definition. Real Application Clusters handles this automatically with messages managed by the GES.

Library Cache Locks

When a database object, such as a table, view, procedure, package, or index, is referenced during the parsing of a SQL, DML, DDL, PL/SQL, or Java statement, the process that parses the statement acquires a library cache lock. In Oracle9i, the lock is held only until the parse or compilation completes, or, for the duration of the parse call.

Part III of this book describes Real Application Clusters implementation topics.

Part III

Implementing Real Application Clusters

Part III describes several topics specific to Real Application Clusters implementation. The chapters in Part III are:

- [Chapter 8, "Real Application Clusters Storage Considerations"](#)
- [Chapter 9, "Manageability Tools for Real Application Clusters Environments"](#)

Real Application Clusters Storage Considerations

This chapter describes Real Application Clusters storage considerations. Topics in this chapter include:

- [Overview of Storage in Real Application Clusters](#)
- [Datafiles in Real Application Clusters](#)
- [Parameter File Storage in Real Application Clusters](#)
- [Redo Log File Storage in Real Application Clusters](#)
- [Managing Undo Space in Real Application Clusters](#)

Overview of Storage in Real Application Clusters

The primary difference between Real Application Clusters storage and storage for single-instance Oracle databases is that all datafiles and configuration files in Real Application Clusters must reside on either a cluster file system or on shared raw devices. You must also create at least two additional redo logs for each instance. You must also create one device for each instance for its own tablespace for automatic undo management, or one rollback segment tablespace for the database if you do not use automatic undo management.

If your platform does not support a cluster file system or if you do not want to use it to store datafiles for Real Application Clusters, then you must create additional raw devices as described in *Oracle9i Real Application Clusters Setup and Configuration*. The rest of this chapter provides additional information about Real Application Clusters-specific storage topics.

Datafiles in Real Application Clusters

All Real Application Clusters instances must be able to access the same datafiles. The composition of database files is the same for Oracle in both Real Application Clusters and in single instance Oracle. Therefore, you do not have to alter datafiles to use them in Real Application Clusters. The physical placement of the datafiles, however, changes. This is because, as mentioned, all instances must access the files.

Datafile Verification in Real Application Clusters

The first Real Application Clusters instance to start verifies access to all online files so it can determine whether to perform media recovery. Additional instances can operate without access to all of the online datafiles, but attempts to use an unverified file will fail and Oracle will generate an error message.

Adding Datafiles in Real Application Clusters

When you add a datafile to a tablespace or bring a datafile online, all instances verify access to the file. If you add a new datafile onto a disk that other instances cannot access, then verification fails but the instances can continue running. Verification can also fail if instances access different copies of the same datafile.

If verification fails for any instance, then diagnose and fix the problem. Use the `ALTER SYSTEM CHECK DATAFILES` statement to verify access. To perform recovery, you do not need to recover entire files. Instead, you can recover individual blocks.

See Also:

- *Oracle9i Recovery Manager User's Guide* for a description of Block Media Recovery (BMR)
- *Oracle9i SQL Reference* for details about using the ALTER SYSTEM CHECK DATAFILES statement
- *Oracle9i Database Administrator's Guide* for information about the Oracle Managed File (OMF) feature and related file naming conventions

Parameter File Storage in Real Application Clusters

You can use two types of files for parameter administration: the server-side parameter file or one or more client-side parameter files. Oracle Corporation recommends that you administer parameters using server parameter files to simplify parameter administration and to take advantage of Oracle's advanced self-tuning capabilities. When Oracle self-tunes, it automatically modifies parameter settings in the server parameter file.

The **Database Configuration Assistant (DBCA)** creates the server parameter file as a binary file. By default, Oracle creates the server parameter file based on one **parameter file (PFILE)**. The server parameter file uses an asterisk (*) to identify global parameter settings and the Oracle **system identifier (SID)** designator to indicate instance-specific settings.

You can only change parameter settings in the server parameter file using either **Oracle Enterprise Manager** or ALTER SYSTEM SET SQL statements. If you use the traditional client-side parameter files, then parameter changes that Oracle makes as a result of self-tuning are not preserved after shutdown.

Note: Avoid modifying the values for self-tuning parameters in the server parameter file; overriding these settings can adversely affect performance.

Location of the Server Parameter File

If you use the DBCA to create your Real Application Clusters database, then the single-instance Oracle database default location that the DBCA provides for the server parameter file is inappropriate for Real Application Clusters. This is because all instances must use the same server parameter file. Therefore, you must override the default location provided by the DBCA.

The value you use to override the default location, however, depends upon your file system configuration. If your platform supports a cluster file system or if you are using a single-node Real Application Clusters database, then you can place the server parameter file on the file system. Otherwise you must enter a raw device name for the location of the server parameter file.

See Also: *Oracle9i Real Application Clusters Setup and Configuration* and *Oracle9i Real Application Clusters Administration* for more information about configuring and using the server parameter file

Redo Log File Storage in Real Application Clusters

In Real Application Clusters, each instance writes to its own set of online redo log files. The redo written by an instance is called a **thread** of redo. Each online redo log file is associated with a particular thread number. When an online redo log is archived, Oracle records its thread number to identify it in case Oracle performs recovery.

A **private thread** is a redo log created by using the `ALTER DATABASE ADD LOGFILE` statement with the `THREAD` clause. A **public thread** is a redo log created using the `ALTER DATABASE ADD LOGFILE` statement but without specifying a `THREAD` clause.

If the `THREAD` initialization parameter is specified, then the instance starting up acquires the thread identified by that value as a private thread. If the `THREAD` initialization parameter has the default value 0, then the instance acquires a public thread. Once acquired, the acquiring instance uses the redo thread exclusively.

Note: Online redo log files can be multiplexed or *mirrored*.

See Also: *Oracle9i Database Concepts* and the *Oracle9i Database Administrator's Guide* for descriptions of multiplexed redo log files

Automatic Segment-Space Management

The ability of transactions from different instances to concurrently insert data into the same table occurs in Real Application Clusters without contention to locate free space for new records. To accomplish this, Oracle closely manages the blocks that have space for transactions that could cause rows to exceed their available space. Oracle does this for each database object, such as a table, cluster, or index.

To control space management, Oracle Corporation strongly recommends that you use automatic segment-space management. This feature uses bitmaps to manage free space within segments. If you prefer to use free lists, then create as many free lists groups as you have instances in your Real Application Clusters database.

See Also: *Oracle9i Real Application Clusters Deployment and Performance* for more information about free lists and free list groups and the *Oracle9i Database Administrator's Guide* for more information about automatic segment-space management

Managing Undo Space in Real Application Clusters

Undo blocks are records of previous data values that were later changed by transactions, even if the transactions are not committed. Oracle requires undo blocks to maintain read consistency, to undo changes made by transactions that roll back or terminate, and to recover the database.

Oracle Corporation strongly recommends that you use automatic undo management to control undo space in Real Application Clusters. This feature is not only easier to administer than rollback segment undo, but the performance of space management with automatic undo management is superior to space management performance in rollback undo mode.

Note: If you use automatic undo management, then each instance requires its own undo tablespace.

Private and Public Rollback Segments

If you use rollback segment undo mode, then each Real Application Clusters instance shares use of the SYSTEM rollback segment and requires at least one dedicated rollback segment. Instances can acquire either private or public rollback

segments at startup and use them exclusively. The exception to this is the `SYSTEM` rollback segment. However, other instances can read rollback segments to create read-consistent snapshots or to perform instance recovery.

Oracle Corporation recommends that you use private rollback segments for each instance in Real Application Clusters, and that you use the same rollback segment identifiers each time you startup a particular instance. For private rollback segments, you must always specify the `SID` when using the `ROLLBACK_SEGMENTS` parameter. Instances use rollback segments until the rollback segments are taken offline or when the acquiring instance is shut down as specified in the rollback segment parameter.

A public rollback segment is offline and not used by an instance until an instance that needs a rollback segment starts up, acquires it, and brings it online. Once online, the acquiring instance uses the public rollback segment exclusively.

See Also: *Oracle9i Database Administrator's Guide* for information about rollback segment performance and for information about the implications of adding rollback segments. That manual also contains details about automatic undo management, undo tablespaces, and managing rollback segments.

Manageability Tools for Real Application Clusters Environments

Oracle has many manageability tools that simplify the configuration and administration of Real Application Clusters databases. The topics in this chapter include:

- [Overview of Manageability in Real Application Clusters](#)
- [Manageability for Real Application Clusters Installation, Setup, and Configuration](#)
- [Manageability for Real Application Clusters Administration](#)
- [Manageability for Real Application Clusters Performance Monitoring](#)
- [Manageability for Real Application Clusters Backup and Recovery](#)

See Also: *Oracle9i Real Application Clusters Real Application Clusters Guard I - Concepts and Administration* and *Oracle9i Real Application Clusters Guard II Concepts, Installation, and Administration* on the Real Application Clusters Guard II software CD for more information about the manageability tools for these features

Overview of Manageability in Real Application Clusters

Oracle provides manageability tools for all phases of deploying Real Application Clusters databases. The Oracle manageability framework provides out-of-the-box operability and scalability by providing user-friendly tools that simplify the installation, configuration, administration, and deployment of Real Application Clusters databases.

You can install Real Application Clusters using either a completely automated process or by using the customized installation process. Once your system is configured, your Real Application Clusters database is ready for data loading and production operations.

On-going administration of your Real Application Clusters database is simplified with features from Oracle Enterprise Manager (EM). Enterprise Manager is a graphical user interface-based tool that provides a cluster-wide view of all activity within Real Application Clusters database environments. You can also quickly execute administrative operations with the Server Control (SRVCTL) command-line utility and the Global Services Daemon (GSD) commands.

Real Application Clusters also takes full advantage of Oracle9i self-tuning features. In addition, you can use the full set of Oracle Recovery Manager (RMAN) features to automate backup and recovery operations. The rest of this chapter describes these features in more detail.

Manageability for Real Application Clusters Installation, Setup, and Configuration

The Oracle Universal Installer works with the Database Configuration Assistant and the Network Configuration Assistant to install the Oracle9i and Real Application Clusters software and to configure Real Application Clusters databases. The Installer offers several preconfigured installations that require little or no user input. You can select from configuration types that include Transaction Processing, Data Warehouse, and a General Purpose configuration. These configuration types install preconfigured options that are suitable for each type of processing environment.

You can also run the Installer and configuration tools separately to manually install the Oracle Real Application Clusters software and configure your cluster database environment. In addition, for some platforms, you can store the datafiles for Real Application Clusters on a cluster file system. This greatly simplifies your installation and administration procedures. Otherwise, before installing Real Application Clusters you must configure raw devices.

See Also: *Oracle9i Real Application Clusters Setup and Configuration* for more information about installing Oracle9i software and your platform-specific documentation for information about cluster file system support

Manageability for Real Application Clusters Administration

Oracle provides several tools to simplify the administration of Real Application Clusters databases as described under the following headings:

- [Oracle Enterprise Manager](#)
- [The Database Configuration Assistant](#)
- [The Server Control \(SRVCTL\) Utility](#)
- [Global Services Daemon Administration Commands](#)

Oracle Enterprise Manager

Oracle Enterprise Manager provides control over both instance- and database-level operations. You can use EM to view the entire cluster as a single entity or to obtain detailed information about individual instances. The information that EM makes available for cluster databases is the same as for single-instance Oracle databases. That is, you can use EM to administer cluster databases and their related elements by navigating through the same master and detail views and menus that are available in a single-instance Oracle database.

Oracle Enterprise Manager Requirements

To use Enterprise Manager, the The Oracle Intelligent Agent must reside on each node that is part of your Real Application Clusters database. The Oracle Intelligent Agent is installed when the Oracle Universal Installer installs the Oracle database software.

Administrative Tasks You Can Accomplish with Enterprise Manager

You can use Enterprise Manager to start and stop instances, listeners, and cluster databases that the Oracle Intelligent Agent discovers. In addition, you can start and stop specific groups of instances and their services. Oracle Enterprise Manager also displays the statuses of all instances in your Real Application Clusters database. You can also use Enterprise Manager to perform the following administrative tasks:

- Display and set initialization parameter values in the server parameter file

- Control automatic undo management to further simplify the management of undo space
- Examine sessions and session details on all instances and terminate sessions on any instance
- Control resource management by assigning different resource plans to each instance to increase operating efficiency

You can also use EM to view the tablespaces of both cluster file system files or datafiles on raw devices. Use EM to either bring these objects online or to take them offline as needed. You can also use EM to assign redo log groups to specific threads of cluster database instances and to manage schema objects and users.

Enterprise Manager also enables you to automate repetitive tasks by creating and managing jobs that you execute within your Real Application Clusters environment. You can schedule jobs on any node or group of nodes on which the Intelligent Agent is running. You can also schedule jobs against a cluster database or against cluster database instances. As with single-instance Oracle databases, Enterprise Manager can also process reports for multiple cluster database instances or for the entire cluster database environment and publish these reports to a Web location.

See Also: *Oracle Enterprise Manager Concepts Guide* for more information about Oracle Enterprise Manager

The Database Configuration Assistant

The Database Configuration Assistant has several features with which you can administer Real Application Clusters environments. You can operate the DBCA from any node in your cluster regardless of whether the instance or instances on that node are started.

Use the Instance Management feature of the DBCA to add or remove instances, nodes, and databases from your cluster. You can also use Instance Management to develop and manage database creation scripts. This feature also enables you to reverse engineer a schema by creating scripts for preexisting databases. Instance Management enables you to make instance-specific object assignments. That is, you can assign rollback segments and redo log groups to specific instances.

The DBCA also supports Oracle-Managed Files (OMF) for operating systems that support a cluster file system. OMF greatly simplifies your file management overhead. Whenever Oracle requires additional files or no longer needs files, OMF automatically adds or deletes them.

The Server Control (SRVCTL) Utility

You can use the command-line Server Control (SRVCTL) utility to perform many database-level administrative tasks that you can perform with Oracle Enterprise Manager. For example, you can use SRVCTL to accomplish cluster database tasks and database configuration tasks.

At the cluster database level you can use SRVCTL to start and stop cluster databases, to start and stop cluster database instances, and to obtain the status of a cluster database and its cluster database instances. You can also use SRVCTL to perform the following cluster database configuration tasks:

- Add and delete cluster database configuration information
- Add an instance to and delete an instance from a cluster database configuration
- Move instances in a cluster database configuration from one node to another
- Set and unset the environment for an instance in a cluster database configuration
- Set and unset the environment for an entire cluster database in a cluster database configuration

You can also use SRVCTL to start and stop a group of programs that includes the Global Services Daemon, Virtual IP Addressing, listeners, and EM Agents.

Global Services Daemon Administration Commands

You can use `gsdctl` commands to start, stop, and obtain the status of the Global Services Daemon (GSD). These `gsdctl` commands operate on all platforms to control GSD operations on all the nodes in your Real Application Clusters database.

See Also: *Oracle9i Real Application Clusters Administration* for more information about the manageability components for administering Oracle Real Application Clusters

Manageability for Real Application Clusters Performance Monitoring

You can use two Oracle manageability tools to monitor Real Application Clusters performance as described under the following headings:

- [Monitoring Performance with Oracle Enterprise Manager](#)
- [Monitoring Performance with Statspack](#)

Monitoring Performance with Oracle Enterprise Manager

Oracle Enterprise Manager has performance charts for monitoring applications that run on Real Application Clusters databases. The performance charts collect information from all active instances and aggregate them into a single, comprehensive view. Use these charts to monitor both the cluster database and the cluster database instances.

Monitoring Performance with Statspack

Statspack displays Real Application Clusters statistics to show performance trends over time. Statspack displays Global Cache Service (GCS) statistics that reveal performance characteristics as well as Global Enqueue Service (GES) statistics. Statspack also displays GCS and GES messaging statistics.

See Also: *Oracle9i Real Application Clusters Deployment and Performance* for more information about Statspack and monitoring Real Application Clusters performance

Manageability for Real Application Clusters Backup and Recovery

Real Application Clusters supports the complete range of functionality of Oracle Recovery Manager (RMAN). After you configure your archive log scheme as described in *Oracle9i Real Application Clusters Administration* and all the archive logs are readable by all instances in your cluster database, you can automate your backup and recovery operations using RMAN.

See Also: *Oracle9i Recovery Manager User's Guide* for more information about using RMAN

Part IV

High Availability and Real Application Clusters

Part IV includes the information on Oracle's products and features that provide high availability. The chapter in Part IV is:

- [Chapter 10, "High Availability Concepts and Best Practices in Real Application Clusters"](#)

High Availability Concepts and Best Practices in Real Application Clusters

This chapter describes the concepts and some of the best practices for implementing high availability in Real Application Clusters. The topics in this chapter are:

- [Understanding High Availability](#)
- [Configuring Real Application Clusters for High Availability](#)
- [Disaster Planning](#)
- [Failure Protection Validation](#)
- [Failover and Real Application Clusters](#)
- [Failover Processing in Real Application Clusters](#)
- [High Availability Configurations](#)
- [Deploying High Availability](#)

Understanding High Availability

Computing environments configured to provide nearly full-time availability are known as **high availability** systems. Such systems typically have redundant hardware and software that makes the system available despite failures. Well-designed high availability systems do not have single points-of-failure. Any hardware or software component that can fail has a redundant component of the same type.

When failures occur, the **failover** process moves processing performed by the failed component to the backup component. This process remasters system-wide resources, recovers partial or failed transactions, and restores the system to normal, preferably within a matter of microseconds. The more transparent that failover is to users, the higher the availability of the system.

Oracle offers several products and features that provide high availability. These include Real Application Clusters, **Oracle Real Application Clusters Guard I**, **Oracle Real Application Clusters Guard II**, Oracle Replication, and **Oracle9i Data Guard**. You can use these products in various combinations to meet your specific high availability needs. Real Application Clusters systems are inherently high availability environments that can provide continuous service for both planned and unplanned outages. Real Application Clusters Guard II provides *continuous* service despite unplanned failures *and* for online maintenance operations.

See Also: *Oracle9i Real Application Clusters Real Application Clusters Guard I - Concepts and Administration* and *Oracle9i Real Application Clusters Guard II Concepts, Installation, and Administration* for more information about these features

Configuring Real Application Clusters for High Availability

Real Application Clusters builds higher levels of availability on top of the standard Oracle features. All single instance high availability features such as Fast-Start Recovery and online reorganizations also apply to Real Application Clusters. Fast-Start Recovery can greatly reduce the mean time to recovery (MTTR) with minimal effects on online application performance. Online reorganizations reduce the duration of planned downtime and you can perform many operations online while users update the underlying objects.

In addition to these features, Real Application Clusters exploits the redundancy provided by clustering to deliver availability with $n-1$ node failures in an n -node cluster. In other words, all users have access to all data as long as there is one available node in the cluster. To configure Real Application Clusters for high

availability, consider the hardware and software components of your cluster as described the following section.

Cluster Components and High Availability

This section describes high availability and cluster components in the following sections:

- [Cluster Nodes](#)
- [Cluster Interconnects](#)
- [Database Software](#)

See Also: [Chapter 2](#) for more information about these components

Cluster Nodes

Real Application Clusters environments are fully redundant because all nodes access all the database. The failure of one node does not affect another node's ability to process transactions. As long as the cluster has one surviving node, all database clients can process all transactions, although the clients may be subject to increased response times due to capacity constraints on the surviving node.

Cluster Interconnects

Interconnect redundancy is often overlooked in clusters. This is because the mean time to failure (MTTF) is generally several years. Therefore, cluster interconnect redundancy might not be a high priority. Also, depending on the system and sophistication level, a redundant cluster interconnect could be cost prohibitive.

However, a redundant cluster interconnect is an important aspect of a fully redundant cluster. Without this, a system is not truly free of single points-of-failure. Cluster interconnects can fail for a variety of reasons and you cannot prevent all of them.

Database Software

In Real Application Clusters, Oracle executables are installed on either the cluster file system or on the local disks of each node; and at least one instance runs on each node of a cluster. Note that if your platform supports a cluster file system (CFS) and you use it, then only one copy of the Oracle Real Application Clusters software will be installed. All instances have equal access to all data and can process any

transactions. In this way, Real Application Clusters ensure full database software redundancy.

Disaster Planning

Real Application Clusters is primarily a single site, high availability solution. This means the nodes in the cluster generally exist within the same building, if not the same room. Therefore, disaster planning can be critical. Depending on how mission critical your system is, and the potential exposure of your system's location for such disasters, disaster planning can be an important high availability component.

Oracle offers other solutions such as Oracle9i Data Guard to facilitate more comprehensive disaster recovery planning. You can use these solutions with Real Application Clusters where one cluster hosts the primary database and another remote system or cluster hosts the disaster recovery database. However, Real Application Clusters are not required on either site for disaster recovery.

See Also : *Oracle9i Data Guard Concepts and Administration* for more information about Data Guard

Failure Protection Validation

Once you have carefully considered your system level issues, validate that your Real Application Clusters environment optimally protects against failures. Use the following list of failure points to plan and troubleshoot your failure protection system:

- Cluster component
- CPU
- Memory
- Interconnect software
- Operating System
- Cluster Manager
- Oracle database instance media
- Corrupt or lost control file, log file, or datafile
- Dropped or deleted database object
- Human error

Real Application Clusters environments protect against cluster component failures and software failures. However, media failures and human error could still cause system downtime. Real Application Clusters, as with single-instance Oracle databases, operates on one set of files. For this reason, you should adopt best practices to avoid the adverse effects of media failures.

RAID-based redundancy practices avoid file loss but might not prevent rare cases of file corruptions. Also, if you mistakenly drop a database object in an Real Application Clusters environment, then you can recover that object the same way you would in a single instance database. These are the primary limitations in an otherwise very robust and highly available Real Application Clusters system.

Once you deploy your system, the key issue is the transparency of failover and its duration as described in the following section.

Failover and Real Application Clusters

This section describes the principles of failover and the features Real Application Clusters offers to implement failover in high availability systems. Topics in this section include:

- [Failover Basics](#)
- [Client Failover](#)
- [Uses of Transparent Application Failover](#)
- [Server Failover](#)

Failover Basics

Failover requires that highly available systems have accurate instance monitoring or **heartbeat** mechanisms. In addition to having this functionality for normal operations, the system must be able to quickly and accurately synchronize resources during failover.

The process of synchronizing, or **remastering**, requires the graceful shutdown of the failing system as well as an accurate assumption of control of the resources that were mastered on that system. In Real Application Clusters, your system records resource information to remote nodes as well as local. This makes the information needed for failover and recovery available to the recovering instances.

See Also:

- *Oracle9i Real Application Clusters Real Application Clusters Guard I - Concepts and Administration* for information about how to set up Oracle Real Application Clusters Guard I on your system
- *Oracle9i Real Application Clusters Guard II Concepts, Installation, and Administration* for information about using Oracle Real Application Clusters Guard II
- *Oracle9i Real Application Clusters Administration* for details about recovery in Real Application Clusters
- *Oracle9i Recovery Manager User's Guide* for details on recovery

Duration of Failover

The duration of failover includes the time a system requires to remaster system-wide resources and the time to recover from failures. The duration of the failover process can be a relatively short interval on certified platforms.

- For existing users, failover entails both server and client failover actions
- For new users, failover only entails the duration of server failover processing

Client Failover

It is important to hide system failures from database client connections. Such connections can include application users in client server environments or middle-tier database clients in multitiered application environments. Properly configured failover mechanisms transparently reroute client sessions to an available node in the cluster. This capability in the Oracle database is referred to as Transparent Application Failover.

Transparent Application Failover

Transparent Application Failover (TAF) enables an application user to automatically reconnect to a database if the connection fails. Active transactions roll back, but the new database connection, which is achieved using a different node, is identical to the original. This is true regardless of how the connection fails.

Elements Affected by Transparent Application Failover

There are several elements associated with active database connections. These include:

- Client/Server database connections
- Users' database sessions executing commands
- Open cursors used for fetching
- Active transactions
- Server-side program variables

Transparent Application Failover automatically restores some of these elements. For example, during normal client/server database operations, a client maintains a connection to the database so the client and server can communicate. If the server fails, then so does the connection. The next time the client tries to use the connection the client issues an error. At this point, the user must log in to the database again.

With Transparent Application Failover, however, Oracle automatically obtains a new connection to the database. This enables users to continue working as if the original connection had never failed. Therefore, with Transparent Application Failover, a client notices no connection loss as long as one instance remains active to serve the application.

See Also: *Oracle9i Net Services Administrator's Guide* for background and configuration information about Transparent Application Failover

Uses of Transparent Application Failover

While the ability to fail over client sessions is an important benefit of Transparent Application Failover, there are other useful scenarios where Transparent Application Failover improves system availability. These topics are discussed in the following subsections:

- [Transactional Shutdowns](#)
- [Quiescing the Database](#)
- [Load Balancing](#)
- [Database Client Processing During Failover](#)
- [Transparent Application Failover Restrictions](#)

Transactional Shutdowns

It is sometimes necessary to take nodes out of service for maintenance or repair. For example, if you want to apply patch releases without interrupting service to

application clients. Transactional shutdowns facilitate shutting down selected nodes rather than an entire database. Two transactional shutdown options are available:

- Use the `TRANSACTIONAL` clause of the `SHUTDOWN` statement to remove a node from service so that the shutdown event is deferred until all existing transactions are completed. In this way, client sessions can be migrated to another node of the cluster at transaction boundaries.
- Use the `TRANSACTIONAL LOCAL` clause of the `SHUTDOWN` statement to perform transactional shutdown on a specified local instance. You can use this statement to prevent new transactions from starting locally, and to perform an immediate shutdown after all local transactions have completed. With this option, you can gracefully move all sessions from one instance to another by shutting down selected instances transactionally.

After performing a transactional shutdown, Oracle routes newly submitted transactions to an alternate node. An immediate shutdown is performed on the node when all existing transactions complete.

See Also: ["Transparent Application Failover Processing During Shutdowns"](#) on page 10-10

Quiescing the Database

You may need to perform administrative tasks that require isolation from concurrent user transactions or queries. To do this, you can use the quiesce database feature. This prevents you, for example, from having to shut down the database and re-open it in restricted mode to perform such tasks.

To do this, you can use the `ALTER SYSTEM` statement with the `QUIESCE RESTRICTED` clause. The `QUIESCE RESTRICTED` clause enables you to perform administrative tasks in isolation from concurrent user transactions or queries.

Note: You cannot open the database on one instance if the database is *being quiesced* on another node. In other words, if you issued the `ALTER SYSTEM QUIESCE RESTRICTED` statement but it is not finished processing, you cannot open the database. Nor can you open the database if it is already in a quiesced state.

See Also: *Oracle9i Real Application Clusters Administration* and the *Oracle9i Database Administrator's Guide* for more detailed information about the quiesce database feature and *Oracle9i SQL Reference* for more information about the `ALTER SYSTEM QUIESCE RESTRICTED` syntax

Load Balancing

A database is available when it processes transactions in a timely manner. When the load exceeds a node's capacity, client transaction response times are adversely affected and the database availability is compromised. It then becomes important to manually migrate client sessions to a less heavily loaded node to maintain response times and application availability.

In Real Application Clusters, the Transport Network Services (TNS) listener files provide automated load balancing across nodes in both shared server and dedicated server configurations. Because the parameters that control cross-instance registration are also dynamic, Real Application Clusters' load balancing feature automatically adjusts for cluster configuration changes. For example, if you add a node to your cluster database, then Oracle updates all the listener files in the cluster with the new node's listener information.

Database Client Processing During Failover

Failover processing for query clients is different than the failover processing for Database Manipulation Language clients. The important issue during failover operations in either case is that the failure is masked from existing client connections as much as possible. The following subsections describe both types of failover processing.

Query Clients At failover, in-progress queries are reissued and processed from the beginning. This might extend the duration of the next query if the original query required longer to complete. With Transparent Application Failover (TAF), the failure is masked for query clients with an increased response time being the only issue affecting the client. If the client query can be satisfied with data in the buffer cache of the surviving node to which the client reconnected, then the increased response time is minimal. Using TAF's `PRECONNECT` method eliminates the need to reconnect to a surviving instance and thus further minimizes response time. However, `PRECONNECT` allocates resources awaiting the failover event.

After failover, server-side recovery must complete before access to the datafiles is allowed. The client transaction experiences a system pause until server-side recovery completes, if server-side recovery has not already completed.

You can also use a callback function through an OCI call to notify clients of the failover so that the clients do not misinterpret the delay for a failure. This prevents the clients from manually attempting to reestablish connections.

Database Manipulation Language Clients Database Manipulation Language (DML) database clients perform `INSERT`, `UPDATE`, and `DELETE` operations. Oracle handles certain errors and performs a reconnect when those errors occur.

Without this application code, `INSERT`, `UPDATE`, and `DELETE` operations on the failed instance return an un-handled Oracle error code. Upon re-submission, Oracle routes the client connections to a surviving instance. The client transaction then stops only momentarily until server-side recovery completes.

Transparent Application Failover Processing During Shutdowns

Queries that cross the network *after* shutdown processing completes will failover. However, Oracle returns an error for queries that are in progress *during* shutdowns. Therefore, TAF only operates when the operating system returns a network error and the instance is completely down.

Applications that use TAF for transactional shutdown must be written to process the error ORA-01033 "ORACLE initialization or shutdown in progress". In the event of a failure, an instance will return error ORA-01033 once shutdown processing begins. Such applications need to periodically retry the failed operation, even when Oracle reports multiple ORA-01033 errors. When shutdown processing completes, TAF recognizes the failure of the network connection to instance and restores the connection to an available instance.

Connection load balancing improves connection performance by balancing the number of active connections among multiple dispatchers. In single-instance Oracle environments, the listener selects the least loaded dispatcher to manage incoming client requests. In Real Application Clusters environments, connection load balancing also has the capability of balancing the number of active connections among multiple instances.

Due to dynamic service registration, a listener is always aware of all of the instances and dispatchers regardless of their locations. Depending on the load information, a listener determines to which instance and to which dispatcher to send incoming client requests if you are using the shared server configuration.

In shared server configurations, listeners select dispatchers using the following criteria in the order shown:

1. Least loaded node

2. Least loaded instance
3. Least loaded dispatcher for that instance

In dedicated server configurations, listeners select instances in the following order:

1. Least loaded node
2. Least loaded instance

If a database service has multiple instances on multiple nodes, then the listener chooses the least loaded instance on the least loaded node. If you have configured the shared server, then the least loaded dispatcher of the selected instance is chosen.

See Also: *Oracle9i Net Services Administrator's Guide* for more information about load balancing

Transparent Application Failover Restrictions

When a connection fails, you might experience the following:

- All PL/SQL package states on the server are lost at failover
- `ALTER SESSION` statements are lost
- If failover occurs when a transaction is in progress, then each subsequent call causes an error message until the user issues an `OCITransRollback` call. Then Oracle issues an Oracle Call Interface (OCI) success message. Be sure to check this message to see if you must perform additional operations.
- Oracle fails over the database connection and if `TYPE=SELECT` in the `FAILOVER_MODE` section of the service name description, Oracle also attempts to fail over the query
- Continuing work on failed-over cursors can result in an error message

If the first command after failover is not a `SQL SELECT` or `OCIStmtFetch` statement, then an error message results. Failover only takes effect if the application is programmed with OCI release 8.0 or greater.

Server Failover

Server-side failover processing in Real Application Clusters is different from host-based failover solutions that are available on many server platforms. The following subsections describe both types of failover processing.

Real Application Clusters Failover

Real Application Clusters provides rapid server-side failover. This is accomplished by the concurrent, active-active architecture in Real Application Clusters. In other words, multiple Oracle instances are concurrently active on multiple nodes and these instances synchronize access to the same database.

All nodes also have concurrent ownership and access to all disks. When one node fails, all other nodes in the cluster maintain access to all the disks; there is no disk ownership to transfer, and database application binaries are already loaded into memory.

Depending on the size of the database, the duration of failover can vary. The larger the database, or the greater the size of its datafiles, the greater the failover benefit of using Real Application Clusters.

Host-Based Failover

Many operating system vendors and other cluster software vendors offer high availability application failover products. These failover solutions monitor application services on a given primary cluster node. They then fail over services to a secondary cluster node as needed. Host-based failover solutions generally have one active instance performing useful work for a given database application. The secondary node monitors the application service on the primary node and initiates failover when the primary node service is unavailable.

Failover in host-based systems usually includes the following steps.

1. Detecting failure by monitoring the heartbeat
2. Reorganizing cluster membership in the Cluster Manager
3. Transferring disk ownership from the primary node to a secondary node
4. Restarting application and database binaries (Oracle executables)
5. Performing application and database recovery
6. Reestablishing client connections to the failover node

Failover Processing in Real Application Clusters

The following subsections describe server failover recovery processing in Real Application Clusters:

- [Detecting Failure](#)
- [Reorganizing Cluster Membership](#)

- [Performing Database Recovery](#)

Detecting Failure

Real Application Clusters relies on the Cluster Manager software for failure detection because the Cluster Manager maintains the heartbeat functions. The time it takes for the Cluster Manager to detect that a node is no longer in operation is a function of a configurable heartbeat timeout parameter.

The use of this parameter varies, depending on your platform. Defaults can vary significantly, depending on the clusterware you use, such as Sun Cluster or the Hewlett-Packard Service Guard OPS Edition. The parameter value is inversely related to the number of false failure detections because the cluster might incorrectly determine that a node is failing due to transient failures if the timeout interval is set too low. When a failure is detected, cluster reorganization occurs.

Reorganizing Cluster Membership

When a node fails, Oracle must alter the node's cluster membership status. This is known as a **cluster reorganization** and it usually happens quickly. The duration of cluster reorganization is proportional to the number of surviving nodes in the cluster.

The [Global Cache Service \(GCS\)](#) and [Global Enqueue Service \(GES\)](#) provide the Cluster Manager interfaces to the software and expose the cluster membership map to the Oracle instances when nodes are added or deleted from the cluster. The LMON process on each cluster node communicates with the Cluster Manager on the respective node and exposes that information to the respective instances. LMON also provides two more useful functions by:

- Continually sending messages from the node on which it runs
- Often writing to the shared disk

If a node fails to perform these two functions, then other nodes consider that node to no longer be a member of the cluster. Such a failure causes a change in a node's membership status within the cluster. Then LMON initiates recovery actions that include remastering of the Global Cache Service (GCS) and Global Enqueue Service (GES) resources and instance recovery.

At this stage, the Real Application Clusters environment is in a state of system pause, and client transactions that do not have the needed resources to complete will suspend until Oracle completes recovery processing. Other in-progress transactions, however, continue processing.

Instance Membership Recovery

The process of **instance membership recovery (IMR)** guarantees that all members of a cluster are functional by:

- Ensuring that communications are viable between all members, and that all members are capable of responding.
- Providing a mechanism for removing members that are not active. IMR arbitrates the membership and removes members that it decides no longer belong to the cluster.
- Voting on membership using the control file. Each member writes a bitmap to the control file. This is part of the checkpoint progress record.
- Removing members based on a communication failure. IMR will perceive members as having failed if they do not transmit periodic heartbeat messages to the control file, or if they do not respond to a query about their status.
- Settling the membership votes and locking the **control file voting results record (CFVRR)** with an arbiter.

All instances read the CFVRR. If a member is not in the membership map, then IMR assumes a node has expired. Appropriate diagnostic information is provided. As IMR is currently configured, all members wait indefinitely for notification of node expiration. There is no forced removal of instances. Part of the fault tolerance of Real Application Clusters is a provision for the possibility that the IMR arbiter itself could fail.

Performing Database Recovery

When an instance fails, Oracle must remaster the GCS resources from the failed instance onto the surviving cluster nodes and perform instance recovery as discussed in the following sections:

- [Remastering Global Cache Service Resources of the Failed Instance](#)
- [Instance Recovery](#)

Remastering Global Cache Service Resources of the Failed Instance

The time required for remastering resources is proportional to the number of GCS resources in the failed instance. This number in turn depends upon the size of the buffer caches.

During this phase, all resources previously mastered at the failed instance are redistributed across the remaining instances. These resources are reconstructed at

their new master instance. All other resources previously mastered at surviving instances are not affected. For any resource request, there is a $1/n$ chance that the request will be satisfied locally and a $(n-1)/n$ chance that the request involves remote operations.

In the case of a cluster database having only one surviving instance, all resource operations are satisfied locally. Once the remastering of a failed instance's GCS resource completes, Oracle recovers the in-progress transactions of the failed instance. This is known as **instance recovery**.

Instance Recovery

Instance recovery includes cache recovery and transaction recovery. Instance recovery requires that an active Real Application Clusters instance detects failure and performs recovery processing for the failed instance. The first Real Application Clusters instance that detects the failure, using its LMON process, controls the recovery of the failed instance by taking over its redo log files and performing instance recovery. This is why the redo log files must be on either a cluster file system file or on a shared raw device.

Instance recovery is complete when Oracle has replayed the online redo log files of the failed instance. Because Oracle can perform transaction recovery in a deferred fashion, any suspended client transactions can begin processing when cache recovery is complete.

See Also: *Oracle9i Recovery Manager User's Guide and Reference* for a description of Block Media Recovery (BMR)

Cache Recovery

For cache recovery, Oracle replays the online redo logs of the failed instance. You can also make Oracle perform cache recovery using parallel execution so that parallel processes, or threads, replay the redo logs of the failed Oracle instance. It could also be important that you keep the time interval for redo log replay to a predictable duration. The Fast-Start Recovery feature in Oracle9i enables you to control this.

Oracle also provides nonblocking rollback capabilities. This means that full database access can begin as soon as Oracle has replayed the online log files. After cache recovery completes, Oracle begins transaction recovery.

See Also: *Oracle9i Database Performance Guide and Reference* for more information on how to use Fast-Start Recovery

Transaction Recovery

Transaction recovery comprises rolling back all uncommitted transactions of the failed instance. Uncommitted transactions are **in-progress** transactions that did not commit.

The Oracle9i Fast-Start Rollback feature performs this as deferred processing that runs in the background. Oracle uses a multiversion read consistency technology to provide on-demand rollback of only those rows blocked by expired transactions. This enables new transactions to progress with minimal delay. New transactions do not have to wait for long-running expired transactions to be rolled back. Therefore, large transactions generally do not affect database recovery time.

Just as with cache recovery, Oracle9i Fast-Start Rollback rolls back expired transactions in parallel. However, single-instance Oracle databases roll back expired transactions using the CPU of one node.

Real Application Clusters provides cluster-aware Fast-Start Rollback capabilities that use all the CPU nodes of a cluster to perform parallel rollback operations. Each cluster node spawns a recovery coordinator and recovery processes to assist with parallel rollback operations. The Fast-Start Rollback feature is thus *cluster aware* because the database is aware of and uses all cluster resources for parallel rollback operations.

While the default behavior is to defer transaction recovery, you could choose to configure your system so that transaction recovery completes before allowing client transactions to progress. In this scenario, the ability of Real Application Clusters to parallelize transaction recovery across multiple nodes is a more visible user benefit.

High Availability Configurations

This section discusses the following Real Application Clusters high availability configurations:

- [Default N-Node Configurations](#)
- [Basic High Availability Configurations](#)
- [Shared High Availability Node Configurations](#)
- [Full Active Configurations with Real Application Clusters Guard II](#)

Default N-Node Configurations

The Real Application Clusters n -node configuration is the default environment. All nodes of the cluster participate in client transaction processing and client sessions can be load balanced at connect time. Response time is optimized for available cluster resources, such as CPU and memory, by distributing the load across cluster nodes to create a highly available environment.

Benefits of N-Node Configurations

In the event of node failures, an instance on another node performs the necessary recovery actions. The database clients on the failed instance can be load balanced across the surviving ($n-1$) instances of the cluster. The increased load on each of the surviving instances can be minimized and availability increased by keeping response times within acceptable bounds. In this configuration, the database application workload can be distributed across all nodes and therefore provide optimal use of cluster machine resources.

Basic High Availability Configurations

You can easily configure a basic high availability system for Real Application Clusters in two-node environments. The primary instance on one node accepts user connections while the **secondary instance** on the other node accepts connections when the primary node fails, or when specifically selected through the `INSTANCE_ROLE` parameter. You can configure this manually by controlling the routing of transactions to specific instances. However, Real Application Clusters provides the **Primary/Secondary Instance Configuration** feature to accomplish this automatically.

Primary/Secondary Instance Configurations

Configure the Primary/Secondary Instance feature by setting the `initsid.ora` parameter `ACTIVE_INSTANCE_COUNT` to 1. In a two-node environment, the instance that first mounts the database assumes the **primary instance role**. The other instance assumes the role of **secondary instance**. If the **primary instance** fails, then the secondary instance assumes the primary role. When the failed instance returns to active status, it assumes the secondary instance role.

Remote Clients and the Primary/Secondary Configuration The secondary instance becomes the primary instance only after the Cluster Manager informs it about the failure of the primary instance. This occurs before GCS and GES reconfiguration and cache and transaction recovery processes begin. The redirection to the

surviving instance happens transparently; application programming is not required. You only need to make minor configuration changes to the client connect strings.

In Primary/Secondary Instance configurations, both instances run concurrently, as in any n -node Real Application Clusters environment. However, database application users only connect to the designated primary instance. The primary node masters all of the GCS and GES resources. This minimizes communication between the nodes and provides performance levels that are nearly comparable to traditional single instance databases.

The secondary instance can be used by specially configured clients, known as administrative clients, for batch query reporting operations or database administration tasks. This enables some level of utilization of the second node. It also helps off-load CPU capacity from the primary instance and justify the investment in redundant nodes.

The Primary/Secondary Instance configuration works in both dedicated server and shared server environments. However, it functions differently in each as described in the following sections:

- [Primary/Secondary Instance Configurations in Dedicated Server Environments](#)
- [Primary/Secondary Instance Configurations and the Shared Server](#)

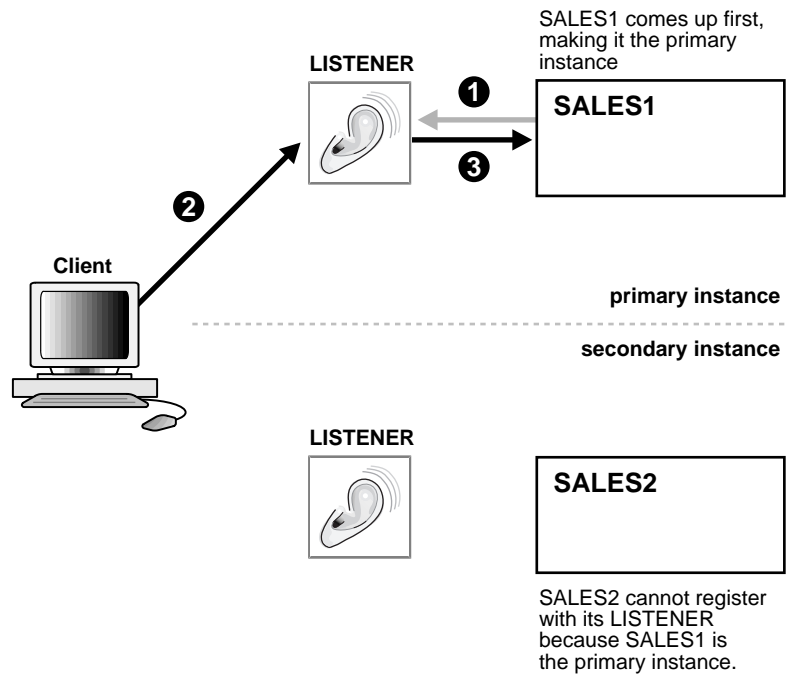
Primary/Secondary Instance Configurations in Dedicated Server Environments

In current high availability configurations, dedicated server environments do not use cross-instance listener registration. Connection requests made to a specific instance's listener can only be connected to that instance's service. This behavior is similar to the default n -node configuration in dedicated server environments.

Figure 10–1 shows a cluster configuration before a node failure.

1. SALES1 is in contact with a listener.
2. A client is in contact with a listener.
3. SALES1 becomes the primary instance.

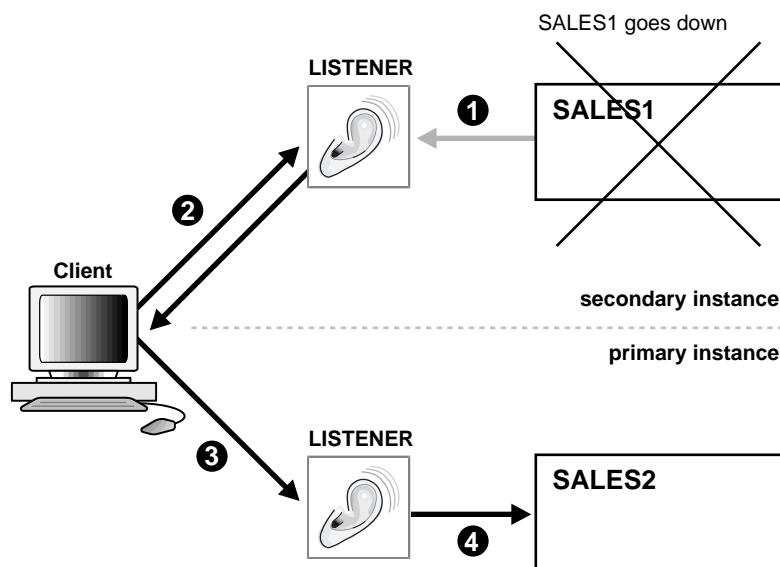
Figure 10-1 Primary/Secondary Configurations in Dedicated Server Environments



When the primary instance fails, as shown in [Figure 10–2](#), the following steps occur:

1. The failure of SALES1 is communicated throughout the cluster.
2. A reconnection request from the client is rejected by the failed instance's listener.
3. The secondary instance performs recovery and becomes the primary instance.
4. Upon resubmitting the client request, the client reestablishes the connection through the new primary instance's listener that connects the client to the new primary instance. Note that the connection is reestablished automatically when you use address lists or if your client is configured to use connection failover.

Figure 10–2 Primary/Secondary Configurations and Node Failure with Dedicated Server



Primary/Secondary Instance Configurations and the Shared Server

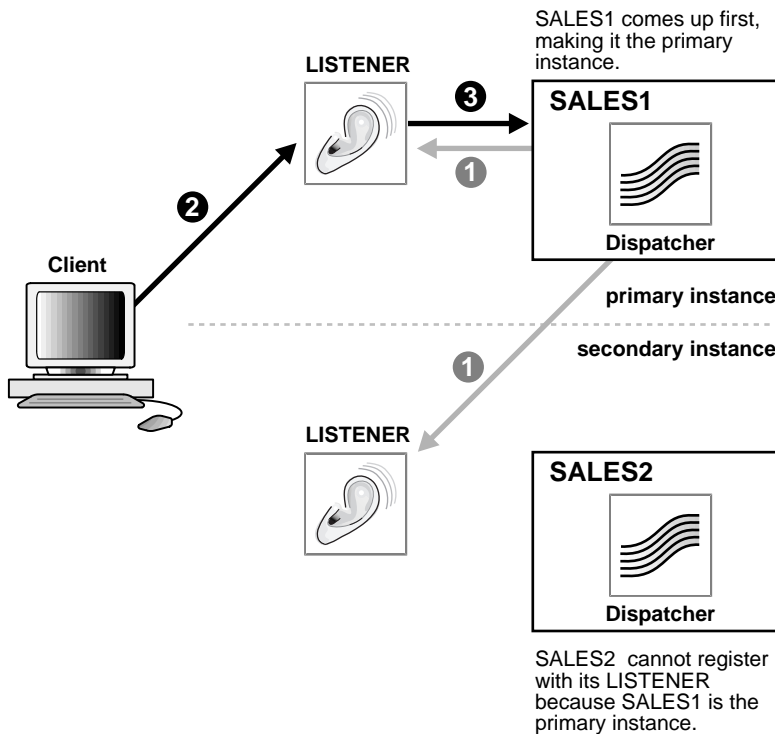
Real Application Clusters provides reconnection performance benefits when running in shared server mode. This is accomplished by the cross-registration of all the dispatchers and listeners in the cluster.

In the Primary/Secondary configurations, the primary instance's dispatcher registers as the primary instance with both listeners, as shown in [Figure 10-3](#):

- A client could connect to either listener. (Only the connection to the primary node listener is illustrated.)
- The client contacts the listener.
- The listener then connects the client to the dispatcher. (Only the listener/dispatcher connection on the primary node is illustrated.)

See Also: *Oracle9i Real Application Clusters Setup and Configuration* for information about configuring client connect strings

Figure 10–3 Primary/Secondary Configurations in Shared Server Environments



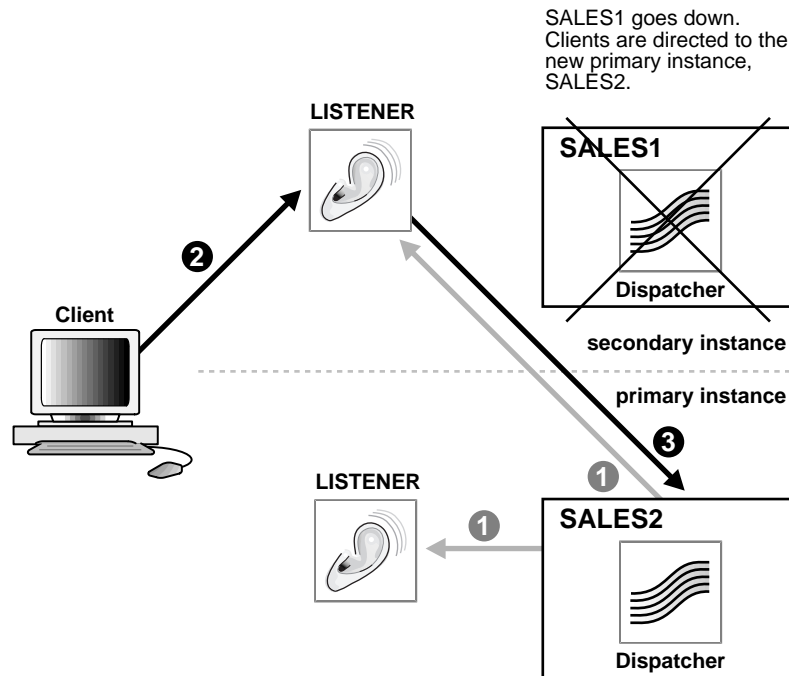
Specially configured clients can use the secondary instance for batch operations. For example, batch reporting tasks or index creation operations can be performed on the secondary instance.

See Also: *Oracle9i Real Application Clusters Administration* for instructions about how to connect to secondary instances

Figure 10–4 shows how a failed primary instance is replaced by a *new* primary instance.

1. If the primary node fails, then the dispatcher in the secondary instance registers as the *new* primary instance with the listeners.
2. The client requests a reconnection to the database through either listener.
3. The listener directs the request to the new primary instance’s dispatcher.

Figure 10–4 Primary/Secondary Configurations and Node Failure with Shared Server



Warming the Library Cache on the Secondary Instance

Maintaining information about frequently executed SQL and PL/SQL statements in the library cache improves the performance of the Oracle database server. In Real Application Clusters primary and secondary instance configurations, the library cache associated with the primary instance contains up-to-date information. If failover occurs, then the benefit of that information is lost unless the library cache on the secondary instance is populated beforehand.

Use the `DBMS_LIBCACHE` package to transfer information in the library cache of the primary instance to the library cache of the secondary instance. This process is called **warming the library cache**. It improves performance immediately after failover because the new primary library cache does not need to be populated with parsed SQL statements and compiled PL/SQL units.

See Also: *Oracle9i Real Application Clusters Real Application Clusters Guard I - Concepts and Administration* for more information about installing and configuring the library cache warming feature and *Oracle9i Supplied PL/SQL Packages and Types Reference* for more information about using `DBMS_LIBCACHE`

Benefits of Basic High Availability Configurations

There are several reasons for using the Primary/Secondary Instance feature for this scenario instead of a default two-node configuration. The Primary/Secondary Instance feature provides:

- A viable transition path for upgrading to an n -node configuration
- A highly available solution for applications that do not need to scale beyond one node
- Performance that is comparable to single instance databases
- A gradual way to migrate from a single instance application environment to a Real Application Clusters environment

Shared High Availability Node Configurations

Operating Real Application Clusters in an n -node configuration optimally utilizes cluster resources. However, as discussed previously, this is not always possible or advisable. On the other hand, the financial investment required to have an idle node for failover might be prohibitive. These situations might instead be best suited for a shared high availability node configuration.

This type of configuration typically has several nodes each running a separate application module or service where all application services share one Real Application Clusters database. In addition, you can configure a separate designated node as a failover node. While an instance is running on that node, no users are connected to it during normal operations. In the event that one of the nodes fails, Oracle can redirect the workload to the failover node.

While this configuration is useful for applications that need to run on separate nodes, it works best if a middle-tier application or transaction processing monitor directs the appropriate application users to the appropriate nodes. Unlike the Primary/Secondary Instance Configuration, there is no database setup that automates the workload transition to the failover node. Instead, the application or middle-tier software directs users from the failed application node to the failover node. The application also must control failing back the users once the failed node

is operational. Failing back frees the failover node for processing user work from subsequent node failures.

Benefits of Shared High Availability Node Configurations

In this configuration, application performance is maintained in the event of a failover. In the n -node configuration, application performance could degrade by $1/n$ due to the same workload being redistributed over a smaller set of cluster nodes.

Full Active Configurations with Real Application Clusters Guard II

High availability as well as improved manageability is available with Real Application Clusters Guard II which is a full instance environment that enables you to control all the instances on which services run as well as their failover properties. On failure, Real Application Clusters Guard II transfers application service loads to other available nodes without service interruptions.

See Also: *Oracle9i Real Application Clusters Guard II Concepts, Installation, and Administration* on the Real Application Clusters Guard II software CD for more information about Real Application Clusters Guard II

Deploying High Availability

Real Application Clusters provides a fully redundant fault resilient environment. All cluster nodes have an active instance that has equal access to all data and resources. If a node fails, then users can access the data using a surviving instance on another node. In-progress transactions on the failed node are recovered by the first node that detects the failure. In this way, there is minimal interruption to end-user application availability with Real Application Clusters.

Part V

Reference

Part V includes the following appendixes and reference information:

- [Appendix A, "Restrictions"](#)
- [Appendix B, "Using Multi-Block Lock Assignments \(Optional\)"](#)
- [Glossary](#)

A

Restrictions

This appendix documents Real Application Clusters compatibility issues and restrictions. Topics in this appendix include:

- [Compatibility](#)
- [Restricted SQL Statements](#)
- [Maximum Number of Datafiles](#)

Compatibility

Real Application Clusters runs with any Oracle database created in **exclusive (X) mode**. Oracle in exclusive mode can access a database created or modified by the Real Application Clusters software. In addition, each instance must have its own set of redo logs.

Restricted SQL Statements

In cluster database mode, the following operations are not supported:

- Creating a database (`CREATE DATABASE`)
- Creating a control file (`CREATE CONTROLFILE`)
- Switching the database's archiving mode (the `ARCHIVELOG` and `NOARCHIVELOG` options of `ALTER DATABASE`)

To perform these operations, shut down all instances and start up one instance in exclusive mode.

Maximum Number of Datafiles

The number of datafiles supported by Oracle is operating system-specific. Within this limit, the maximum number allowed depends on the values used in the `CREATE DATABASE` command. This in turn is limited by the physical size of the control file. This limit is the same in cluster database mode as in exclusive mode, but the additional instances of Real Application Clusters restrict the maximum number of files more than a single instance system.

See Also: *Oracle9i SQL Reference* and your Oracle operating system-specific documentation

Using Multi-Block Lock Assignments (Optional)

This appendix describes configuring locks to cover multiple blocks. Refer to this appendix only for a *limited set of rare* circumstances to override Oracle Real Application Clusters' default resource control scheme as performed by the Global Cache Service (GCS) and the Global Enqueue Service (GES). Overriding Cache Fusion to use locks is only desirable in *rare* cases. Topics in this appendix include:

- [When to Use Locks](#)
- [How to Use Locks](#)
- [Lock Granularity](#)
- [Understanding Lock Management](#)

Note: You never have to set `GC_FILES_TO_LOCKS`. Real Application Clusters and its resource control mechanism from Cache Fusion provide optimal performance without using locks.

When to Use Locks

The methodology described in this appendix has been superseded by **Cache Fusion** in Real Application Clusters. Overriding Cache Fusion to perform lock setting is *only* desirable in Real Application Clusters installations that have a large number of read-mostly blocks.

How to Use Locks

Locks can manage one or more blocks of any class: data blocks, undo blocks, segment headers, and so on. The amount of cross-instance activity, such as read/write or write/write operations, and the corresponding performance of Real Application Clusters, is evaluated in terms of *forced writes*. If you override Cache Fusion, then a forced write can occur when one instance requests a block that is held by another instance: Oracle writes the block to disk so the requesting instance can read it in its most current state.

See Also: *Oracle9i Real Application Clusters Deployment and Performance* for detailed information about allocating locks

Lock Granularity

There are two levels of lock granularity: 1:1 locks, which is the default, and 1:*n* locks. 1:*n* locks implies that a lock manages two or more data blocks as defined by the value for *n*. With 1:*n* locks, a few locks can manage many blocks and thus reduce lock operations. For read-only data, 1:*n* locks can perform faster than 1:1 locks during certain operations such as parallel execution. The `GC_FILES_TO_LOCKS` parameter is only useful to get 1:*n* lock granularity.

Understanding Lock Management

The number of locks assigned to datafiles and the number of data blocks in those datafiles determines the number of data blocks managed by a single lock.

When you set the `GC_FILES_TO_LOCKS` parameter for a file, then the number of blocks for each lock can be expressed as shown in [Figure B-1](#) for each file level. This example assumes values of `GC_FILES_TO_LOCKS=1:300,2:200,3-5:100`.

Figure B–1 Lock-to-Block Assignment Examples for `GC_FILES_TO_LOCKS` Settings

$$\text{File 1: } \frac{\text{file1 blocks}}{300 \text{ locks}}$$

$$\text{File 2: } \frac{\text{file2 blocks}}{200 \text{ locks}}$$

$$\text{File 3: } \frac{\text{sum (file3, file4, file5 blocks)}}{100 \text{ locks}}$$

If the size of each file, in blocks, is a multiple of the number of locks assigned to it, then each 1:*n* lock manages exactly the number of data blocks given by the equation.

If the file size is *not* a multiple of the number of locks, then the number of data blocks for each 1:*n* lock can vary by one for that datafile. For example, if you assign 400 locks to a datafile that contains 2,500 data blocks, then 100 locks manage 7 data blocks each and 300 locks manage 6 blocks. Any datafiles not specified in the `GC_FILES_TO_LOCKS` initialization parameter use the remaining locks.

If *n* files share the same 1:*n* locks, then the number of blocks for each lock can vary by as much as *n*. If you assign locks to individual files, either with separate clauses of `GC_FILES_TO_LOCKS` or by using the keyword `EACH`, then the number of blocks for each lock does not vary by more than one.

If you assign 1:*n* locks to a set of datafiles collectively, then each lock usually manages one or more blocks in each file. Exceptions can occur when you specify contiguous blocks (by using the `!blocks` option) or when a file contains fewer blocks than the number of locks assigned to the set of files.

See Also: *Oracle9i Real Application Clusters Deployment and Performance* for details on how to use the `GC_FILES_TO_LOCKS` parameter

Glossary

acquisition interrupt

A software synchronization feature used for notification. Real Application Clusters also uses a **blocking interrupt**.

agent

In a client/server model, the part of the system that performs information preparation and exchange on behalf of a client or server. In the phrase, *intelligent agent*, it implies an automatic process that can communicate with other agents to perform some collective tasks on behalf of one or more users.

blocking interrupt

A software synchronization feature that notifies the process holding the access right to a resource that another process needs to access the same resource in an incompatible mode. (The **shared (S) mode** and **exclusive (X) mode**, for example, are incompatible). See also **acquisition interrupt**.

buffer state

The state of a buffer in the local cache of an instance.

cache coherency

The synchronization of data in multiple caches so that reading a memory location through any cache will return the most recent data written to that location through any other cache. Sometimes called *cache consistency*.

Cache Fusion

A diskless cache coherency mechanism in **Real Application Clusters** that provides copies of blocks directly from a holding instance's memory cache to a requesting instance's memory cache.

cluster

A set of instances that cooperates to perform the same task.

cluster database

The generic term for a **Real Application Clusters** database.

clustered database

See **cluster database**.

clustering

See **cluster database**.

cluster manager (CM)

An operating system-dependent component that discovers and tracks the membership state of each **node** by providing a common view of cluster membership across the **cluster**. The CM also monitors process health, specifically the health of the database instance. The **Global Enqueue Service Monitor (LMON)** process, a background process that monitors the health of the **Global Cache Service (GCS)**, registers and de-registers from the CM.

connect-time failover

See **failover**.

consistent read (CR)

The **Global Cache Service (GCS)** ensures that a consistent read block (also known as the master copy data block) is maintained. The consistent read block is the master block version that records information about all changes to a block. It is held in at least one System Global Area (SGA) in the cluster if the block is to be changed. If an **instance** needs to read the block, then the current version of the block may reside in many buffer caches as a shared resource. Thus, the most recent copy of the block in all System Global Areas contains all changes made to that block by all instances, regardless of whether transactions on those instances have committed.

context switches

An operating system running a program runs in either user mode or operating system mode. Switching between user and operating system mode is a context switch. For example, a program that makes a system call while in user mode makes a context switch. Context switches can hinder performance because, in this example, the context of the user program must be stored while also transferring the context of the operating system kernel into memory. Performance can be even more adversely affected when multiple system calls compete for operating system resources.

control file voting results record (CFVRR)

Part of the recovery process. The CFVRR is a file record created by the [instance membership recovery \(IMR\)](#) arbiter after it settles the membership *votes* of nodes in a [cluster](#).

cooked partition

A portion of a physical disk where an extended partition is created, logical partitions are assigned, and formatting has been completed. In contrast, an unformatted partition is called a [raw partition](#).

CR

See [consistent read \(CR\)](#).

daemon

Abbreviation for disk and execution monitor. A program that is not invoked explicitly, but lies dormant waiting for specific conditions to occur.

Database Configuration Assistant (DBCA)

An Oracle tool for creating and deleting databases and for managing database templates.

Data Guard

See [Oracle9i Data Guard](#).

DBCA

See [Database Configuration Assistant \(DBCA\)](#).

Decision Support System (DSS)

Database and application environments that help with decision support or data warehouse systems.

diagnosability daemon

A Real Application Clusters background process that captures diagnostic data on [instance](#) process failures. No user control is required for this daemon.

dirty block

A data block that has been changed by an instance.

DSS

See [Decision Support System \(DSS\)](#).

EMCA

See [Enterprise Manager Configuration Assistant \(EMCA\)](#).

enqueue

Shared memory structures that serialize access to database resources. Enqueues are local to one [instance](#) if Real Application Clusters is not enabled. When you enable Real Application Clusters, enqueues can be local or global to a database. (See also: [latch](#), [lock](#), and [resource](#).)

Enterprise Manager

See [Oracle Enterprise Manager](#).

Enterprise Manager Configuration Assistant (EMCA)

A tool for creating, deleting, and modifying Enterprise Manager configurations and settings.

exclusive current (XCUR)

The buffer state name for an exclusive resource.

exclusive (X) mode

A write-only global resource mode. In this mode no other access is allowed. See also: [null \(N\) mode](#) and [shared \(S\) mode](#).

failover

The means of failure recognition and recovery used by Real Application Clusters.

fault tolerance

The ability of a system or component to continue normal operation despite the presence of hardware or software faults. This normally involves some degree of redundancy.

Fiber Distributed Data Interface (FDDI)

A standard of the American National Standards Institute (ANSI) for a 100 Mb per second local area network architecture. The underlying medium is often optical fiber and the topology is a dual-attached, counter-rotating token ring.

Fibre Channel

The generic term for a high speed serial data transfer architecture recently standardized by the American National Standards Institute (ANSI). The Fibre Channel architecture was developed by the Fibre Channel Industry Association (FCIA), a consortium of computer and mass storage manufacturers. The best-known Fibre Channel standard is the Fibre Channel Arbitrated Loop (FC-AL). (See also: [Storage Area Network \(SAN\)](#)).

five nines availability

A colloquial term for 99.999% system availability.

forced disk write

In Real Application Clusters, a particular data block can only be modified by one [instance](#) at a time. If one instance modifies a data block that another instance needs, then whether a forced disk write is required depends on the type of request submitted for the block.

free list group

A set of free lists available for use by one or more instances.

global cache element

An Oracle-specific data structure representing a [Cache Fusion](#) resource. There is a 1:1 corresponding relationship between a global cache element and a Cache Fusion resource in the [Global Cache Service \(GCS\)](#).

Global Cache Service (GCS)

The process that implements [Cache Fusion](#). It maintains the block mode for blocks in the global role. It is responsible for block transfers between instances. The Global Cache Service employs various background processes such as the [Global Cache Service Processes \(LMSn\)](#) and [Global Enqueue Service Daemon \(LMD\)](#).

Global Cache Service Processes (LMSn)

The processes that handle remote **Global Cache Service (GCS)** messages. Real Application Clusters provides for up to 10 Global Cache Service Processes. The number of LMSn varies depending on the amount of messaging traffic amount the nodes in the **cluster**. The LMSn handle the **acquisition interrupt** and **blocking interrupt** requests from a remote **instance** for Global Cache Service resources. For cross-instance consistent read requests, LMSn creates a consistent read version of the block and sends it to the requesting instance. LMSn also controls the flow of messages to and from remote instances.

Global Enqueue Service (GES)

This service coordinates local and global enqueues.

Global Enqueue Service Daemon (LMD)

The resource **agent** process that manages **Global Enqueue Service (GES)** resource requests. The LMD process also handles deadlock detection **Global Enqueue Service (GES)** requests. Remote resource requests are requests originating from another **instance**. (See also: **daemon**.)

Global Enqueue Service Monitor (LMON)

The process that monitors the entire **cluster** to manage global resources. LMON manages **instance** and process expirations and the associated recovery for the **Global Cache Service (GCS)** and **Global Enqueue Service (GES)**. In particular, LMON handles the part of recovery associated with global resources.

Global Resource Directory

The data structures associated with global resources. It is distributed across all instances in a **cluster**.

Global Services Daemon (GSD)

A component that receives requests from **SRVCTL** to execute administrative job tasks, such as startup or shutdown. The command is executed locally on each node, and the results are sent back to SRVCTL. The **daemon** by default. It should not be deleted.

GSD

See **Global Services Daemon (GSD)**.

HA

See **high availability**.

hardware failover

A type of **failover** performed by the platform-specific **cluster manager (CM)**. If a node or the **instance** running on it fails, the cluster manager restarts the instance on another node in the **cluster**. Restarting the Oracle instance requires moving the IP addresses, volumes, and file systems containing the Oracle datafiles. It also requires starting the Oracle server and opening the datafiles on the new node.

heartbeat

A periodic message that shows that an **instance** is active.

high availability

A system type with redundant components that provides consistent and uninterrupted service, even in the event of hardware or software failures.

IMR

See **instance membership recovery (IMR)**.

initdbname.ora

An initialization parameter file that contains global parameters that apply to an entire cluster.

initsid.ora

An initialization parameter file that contains parameters unique for an **instance** and that points to **initdbname.ora** for database parameters.

instance

The combination of the System Global Area (SGA) and each process for the Oracle database. The memory and processes of an instance manage the associated database's data and serve the database users. Each instance has unique **system identifier (SID)**, instance name, rollback segments, and **thread** ID.

instance membership recovery (IMR)

The method used by Real Application Clusters guaranteeing that all **cluster** members are functional. IMR polls and arbitrates the membership. Any members that do not show a heartbeat in the control file record or who do not respond to periodic status queries are presumed to have expired. The Instance Membership Recovery arbiter settles the membership *votes* of nodes in a cluster, and creates a **control file voting results record (CFVRR)**.

instance name

Represents the name of the **instance** and is used to uniquely identify a specific instance when clusters share common services names. The instance name is identified by the INSTANCE_NAME parameter in the instance initialization file, initsid.ora. The instance name is the same as the Oracle system identifier (sid). The instance name is the same as the **system identifier (SID)**.

interconnect

The communication link between the nodes.

interprocess communication (IPC)

A high-speed operating system-dependent transport component. The IPC transfers messages between instances on different nodes. Also referred to as the **interconnect**.

interrupt

See **acquisition interrupt** and **blocking interrupt**.

IPC

See **interprocess communication (IPC)**.

kernel

In the context of databases, the set of foreground and background processes that implement a database.

latch

A simple, low-level serialization mechanism that protect in-memory data structures in the System Global Area (SGA). Latches do not protect datafiles, are automatic, and are held for a very short time in exclusive mode. Because latches are synchronized within a node, they do not facilitate internode synchronization. (See also: **enqueue**, lock, and **resource**.)

LCK process

A process that manages **instance** global enqueue requests and cross-instance call operations. Workload is automatically shared and balanced when there are multiple **Global Cache Service Processes (LMSn)**.

library cache invalidation

Objects in the library cache can become invalid due to object dependencies. The object is recompiled on its next use.

listener

A process that resides on the server to listen for incoming client connection requests and manage the traffic to the server. When a client requests a network session with a server a listener receives the request. If the client information matches the listener information, then the listener grants a connection to the server.

listener.ora

A listener configuration file that identifies the protocol addresses on which the listener is accepting connection requests and the services the listener listens for.

LMD

See [Global Enqueue Service Daemon \(LMD\)](#).

LMON

See [Global Enqueue Service Monitor \(LMON\)](#).

LMS_n

See [Global Cache Service Processes \(LMS_n\)](#).

local enqueues

Synchronization mechanisms that coordinate concurrent access to shared data structures in [cluster](#) databases.

master free list

A list of blocks containing available space from any extent in a table.

mastering

See [resource mastering](#).

mean time between failures (MTBF)

The average time (usually expressed in hours) that a component works without failure. It is calculated by dividing the total number of failures into the total number of operating hours observed. The term can also mean the length of time a user can reasonably expect a device or system to work before a failure occurs.

N

See [null \(N\) mode](#).

node

A node is a machine on which an **instance** resides.

null (N) mode

A null (N) mode indicates that the holding process has an interest in a resource. However, access rights are only conferred when in **exclusive (X) mode** or **shared (S) mode**.

Online Transaction Processing (OLTP)

The processing of transactions by computers in real time. (See also: **transaction systems**.)

operating system-dependent (OSD) clusterware

Clusterware tailored for various operating systems. OSD clusterware provides communication links between the operating system and the **Real Application Clusters** software.

Oracle9i Data Guard

An Oracle **high availability** product that provides a backup database on ready standby status. This product was formerly called **Standby Database**.

Oracle Database Configuration Assistant (DBCA)

See **Database Configuration Assistant (DBCA)**.

Oracle Enterprise Manager

A system management tool that provides an integrated solution for centrally managing your heterogeneous environment. Oracle Enterprise Manager combines a graphical console, management server, Oracle Intelligent Agent, repository database, and tools to provide an integrated, comprehensive systems management platform for managing Oracle products.

Oracle Net

A software component that enables connectivity. It includes a core communication layer called the Oracle Net foundation layer and network protocol support. Oracle Net enables services and their applications to reside on different computers and communicate as peer applications.

Oracle Net Configuration Assistant

A post-installation tool that configures basic network components after installation.

Oracle Net Services

The term that encompasses all of the Oracle networking components, including: **Oracle Net**, the **listener**, Oracle Connection Manager, Oracle Names, **Oracle Net Configuration Assistant**, and Oracle Net Manager.

Oracle Performance Manager

An add-on application for **Oracle Enterprise Manager** that offers a variety of tabular and graphic performance statistics for Real Application Clusters. The statistics represent the aggregate performance for all instances running on Real Application Clusters.

Oracle Real Application Clusters

A breakthrough architecture that enables clusters to access a shared database. Real Application Clusters includes the software component that provides the necessary Real Application Clusters scripts, initialization files, and datafiles to make the Oracle9i Enterprise Edition an Oracle9i Real Application Clusters database.

Oracle Real Application Clusters Guard I

A **failover** protection feature. Oracle Real Application Clusters Guard is an integral component of Real Application Clusters. Oracle Real Application Clusters Guard provides the following functions:

- Automated, fast recovery and bounded recovery time from problems that cause the Oracle **instance** to fail
- Automatic capture of diagnostic data when certain types of failures occur
- Enforced **Primary/Secondary Instance Configuration**. Clients connecting through Oracle Net Services are properly routed to the primary node even if connected to another node in the cluster
- Elimination of delays that clients experience when reestablishing connections after a failure

Oracle Real Application Clusters Guard II

Real Application Clusters Guard II enables you to monitor clusters that use Real Application Clusters to maintain availability. On failure, Real Application Clusters Guard II optimally transfers application service loads to other active nodes. In Real Application Clusters Guard II, all instances are active and are able to support services. Instances are brought in automatically to support a service while they are also available to support other services.

Oracle System Identifier

See [system identifier \(SID\)](#).

OSD

See [operating system-dependent \(OSD\) clusterware](#).

parallel execution

Divides the work of processing certain types of SQL statements among multiple parallel execution server processes. Commonly used in [Decision Support System \(DSS\)](#) applications.

parameter file (PFILE)

A file used by an Oracle server that provides specific values and configuration settings that are used at database startup. The keyword `PFILE` is used in the startup command.

past image (PI)

A copy of a [dirty block](#) that is used by the [Global Cache Service \(GCS\)](#). Past images of blocks are maintained until writes covering those versions are recorded. Past images are used in failure recovery.

Performance Manager

See [Oracle Performance Manager](#).

PFILE

See [parameter file \(PFILE\)](#).

PI

See [past image \(PI\)](#).

ping

A synonymous term for [forced disk write](#).

planned downtime

Includes routine operations, maintenance, and upgrades that cause the system to be unavailable to users. (See also: [unplanned downtime](#).)

primary instance

In primary/secondary configurations, the **instance** through which all clients access the database. (See also: **secondary instance**.)

primary instance role

In primary/secondary configurations, the **instance** that mounts the database first assumes the primary role. It performs the work requested by application sessions. If the primary instance fails or is shut down, then **failover** occurs, and another instance assumes the primary instance role. (See also: **Primary/Secondary Instance Configuration** and **secondary instance role**.)

Primary/Secondary Instance Configuration

A configuration in which the primary **instance** is the instance where all clients access the database. The secondary instance provides backup services to the primary instance in case the primary instance fails. (See also: **primary instance**, **primary instance role**, **secondary instance**, and **secondary instance role**.)

private rollback segment

A rollback segment that is acquired exclusively by an **instance** when the instance opens a database.

public rollback segment

A rollback segment that is available to any **instance** that requires a rollback segment.

RAID

See **Redundant Array of Independent Disks (RAID)**.

raw device

A disk drive that does not yet have a file system set up. Raw devices are used for Real Application Clusters since they enable the sharing of disks. See also **raw partition**.

raw partition

A portion of a physical disk that is accessed at the lowest possible level. A raw partition is created when an extended partition is created and logical partitions are assigned to it without any formatting. Once formatting is complete, it is called a **cooked partition**. See also **raw device**.

raw volumes

See [raw device](#).

Real Application Clusters

See [Oracle Real Application Clusters](#).

Recovery Manager (RMAN)

An Oracle tool that enables you to back up, copy, restore, and recover datafiles, control files, and archived redo logs. It is included with the Oracle server and does not require separate installation.

Redundant Array of Independent Disks (RAID)

A hardware architecture that combines multiple hard disk drives to allow rapid access to a large volume of stored data.

repository database

A set of tables in an Oracle database that stores data required by [Oracle Enterprise Manager](#). This database is separate from the database on the nodes.

resource

In the context of the [Global Cache Service \(GCS\)](#), a concurrency control on data blocks. (See also: [global cache element](#), [enqueue](#), [lock](#), [resource](#), and [resource mastering](#).)

resource mastering

The method by which the [Global Cache Service \(GCS\)](#) and [Global Enqueue Service \(GES\)](#) control [resources](#). (See also: [Global Resource Directory](#).)

resource mode

A concurrency control that defines global access rights for instances in a cluster.

resource role

A concurrency control that defines whether a data block is cached in only one instance (local) or if it cached in multiple instances (global).

RMAN

See [Recovery Manager \(RMAN\)](#).

row cache

The memory that stores recently accessed data for an individual record (or row). Row caches are used so that subsequent requests for the data held in the same row can be processed more quickly.

S

See **shared (S) mode**.

scalability

The ability to add additional nodes to **Real Application Clusters** environments and maintain and often achieve markedly improved performance.

scale up

The factor that expresses how much more work can be done in the same time period by a larger system. the factor that expresses how much more work can be done in the same time period by a larger system. (See also: **speed up**.)

SCUR

See **shared current (SCUR)**.

secondary instance

In a **Primary/Secondary Instance Configuration**, the **instance** that provides backup services to the **primary instance** in case the primary instance fails.

secondary instance role

In a **Primary/Secondary Instance Configuration**, the second **instance** to mount the database assumes the secondary role. The instance with the primary role performs the work that is requested by application sessions, but selected tasks such as reporting and planned operations can be performed by the instance with the secondary instance role. (See also: **primary instance**, **primary instance role**, and **secondary instance**.)

Sequence Number Value Enqueue (SV)

An **enqueue** utilized by a session when it needs a sequence value.

server clustering

See **Real Application Clusters**.

Server Control utility

See [SRVCTL](#).

service name

A logical representation of a database. This is the way a database is presented to clients. A database can be presented as multiple services and a service can be implemented as multiple database instances. The service name is a string that includes:

- The global database name
- A name comprised of the database name (`DB_NAME`)
- Domain name (`DB_DOMAIN`)

The service name is entered during installation or database creation.

If you are not sure what the global database name is, you can obtain it from the combined values of the `SERVICE_NAMES` parameter in the common database initialization file, [initdbname.ora](#).

The service name is included in the `CONNECT_DATA` part of the connect descriptor.

service registration

A feature whereby PMON, the process monitor, automatically registers information with a [listener](#). Because this information is registered with the listener, the `listener.ora` file does not need to be configured with this static information.

shared cache

The aggregation of the buffer caches of all instances in a cluster database.

shared current (SCUR)

The buffer state name for a shared resource.

shared (S) mode

A protected read resource mode. No writes are allowed in shared mode. In shared mode, any number of users can have simultaneous read access to a resource. (See also: [exclusive \(X\) mode](#) and [null \(N\) mode](#).)

shared server

A server that is configured to allow many user processes to share very few server processes, so that the number of users that can be supported is increased. With shared server configuration, many user processes connect to a dispatcher.

SID

See [system identifier \(SID\)](#).

SMP

See [Symmetric Multi-Processor \(SMP\)](#).

speed up

The concept that more hardware can perform the same task in less time than the original system. With added hardware, speed up holds the task constant and measures time savings. (See also: [scale up](#).)

SRVCTL

The Service Control (SRVCTL) utility that administrators can use to manage instances and [Real Application Clusters](#) databases. SRVCTL is installed on each node. The SRVCTL utility gathers information about all the instances for [Oracle Enterprise Manager](#). SRVCTL acts as a single point of control between the Oracle Intelligent Agent and the nodes. Only one node's Oracle Intelligent Agent is used to communicate to SRVCTL. SRVCTL on that node then communicates to the other nodes through Java Remote Method Invocation (RMI).

Standby Database

See [Oracle9i Data Guard](#).

Storage Area Network (SAN)

A high speed network of shared storage devices. Typically, SAN architecture enables access to all storage devices by all servers on a local area network (LAN) or wide area network (WAN). Most SANs conform to [Fibre Channel](#) standards.

SV

See [Sequence Number Value Enqueue \(SV\)](#).

Symmetric Multi-Processor (SMP)

Two or more similar processors connected through a high-bandwidth link and managed by one operating system, where each processor has equal access to I/O devices.

system identifier (SID)

The Oracle system identifier (SID) identifies a specific [instance](#) of the running Oracle software. For a [Real Application Clusters](#) database, each [node](#) within the [cluster](#) has an instance referencing the database.

The database name, specified by the `DB_NAME` parameter in the `INITDB_NAME.ORA` file, and unique **thread** ID make up each node's SID. The thread ID starts at 1 for the first instance in the cluster, and is incremented by 1 for the next instance, and so on.

TAF

See **Transparent Application Failover (TAF)**.

thread

Each Oracle **instance** has its own set of online redo log groups. These groups are called a thread of online redo. In non-Real Application Clusters environments, each database has only one thread that belongs to the instance accessing it. In Real Application Clusters environments, each instance has a separate thread, that is, each instance has its own online redo log. Each thread has its own current log member.

three nines availability

A colloquial term for 99.9% system availability.

tnsnames.ora

A file that contains net service names. This file is needed on clients, nodes, the Console, and the Oracle Performance Manager machine.

transaction free list

A list of blocks freed by uncommitted transactions.

transaction systems

Application systems characterized by updates to the database. Examples of transaction systems include e-business systems and ERP applications. More specialized examples include telephone call and billing systems, credit card transactions, and airline reservation systems. Transaction systems are also known as **Online Transaction Processing (OLTP)** systems.

transparency

An action is transparent if it takes place without any effect that is visible to users.

Transparent Application Failover (TAF)

A runtime **failover** for **high availability** environments, such as **Real Application Clusters** and **Oracle Real Application Clusters Guard I**. TAF refers to the failover and reestablishment of application-to-service connections. It enables client applications to automatically reconnect to the database if the connection fails, and

optionally resume a `SELECT` statement that was in progress. This reconnect happens automatically from within the Oracle Call Interface (OCI) library.

unplanned downtime

System downtime that includes system faults, data and media errors, and site outages that cause the system to be unavailable to users. (See also: [planned downtime](#).)

volume manager

Software that provides storage management for a [Storage Area Network \(SAN\)](#). Usually host-based [Redundant Array of Independent Disks \(RAID\)](#) software. Most volume managers use a graphical user interface (GUI) console to show the partitioning and status of storage volumes. Volume managers can create volumes from various disks, and those volumes can encompass multiple disks. (See also: [raw partition](#).)

warming the library cache

The process of transferring information about parsed SQL statements and compiled PL/SQL units from the library cache on the primary [instance](#) to the library cache on the secondary instance. Warming the cache improves performance after [failover](#) because the library cache is already populated.

X

See [exclusive \(X\) mode](#).

XCUR

See [exclusive current \(XCUR\)](#).

Index

Numerics

1 to 1 locks, setting, B-2

A

access, 2-4
ACTIVE_INSTANCE_COUNT initialization
 parameter, 10-17
active/active configurations
 Real Application Clusters Guard II, 10-25
ADD LOGFILE clause
 THREAD clause, 8-4
administration
 manageability components in Real Application
 Clusters, 9-3
ALERT file, 8-2
allocation
 locks, B-3
ALTER DATABASE statement
 ADD LOGFILE, 8-4
 setting the log mode, A-2
ALTER SYSTEM SET
 and server parameter file administration, 8-3
applications
 scalability, 4-3, 4-6
architectural overview
 Real Application Clusters, 3-2
architecture
 components, 3-1
 for cluster database processing, 2-1
ARCHIVELOG mode
 automatic archiving, 1-6
 changing mode, A-2
 online and offline backups, 1-6
archiving
 backups, 1-5
archiving redo log files

 online archiving, 1-5, 1-6
asymmetrical multiprocessing, 4-6
automatic segment-space management
 recommended in Real Application Clusters, 8-5
automatic undo management, 8-5
 recommended in Real Application Clusters, 8-5
availability
 and the interconnect, 2-4
 benefit of cluster databases, 1-4

B

 backup
 manageability components in Real Application
 Clusters, 9-6
backups
 offline, 1-5
 online, 1-5
bandwidth, 2-4
 network, 4-4
best practices, 10-1
bitmaps
 automatic segment-space management, 8-5
Block Media Recovery, 10-15
blocks
 images, 5-3
 modifications to multiple versions, 5-3
 writing to disk, 6-6
buffer cache management, 1-4
buffer state, 6-4

C

cache
 flushing dictionary, 7-3
cache coherency
 and the Global Cache Service (GCS), 5-5
Cache Fusion
 definition, 6-2
 processing, 6-1
 scenarios, 6-4
cache recovery, 10-15
checkpoint, 6-6
client

- failover, 10-6
- load balancing, 4-5
- randomization, 4-5
- cluster
 - components, 2-2
 - definition, 1-2
 - storage access, 2-4
- cluster database
 - availability, 1-4
- cluster database processing
 - definition, 1-2
 - hardware for, 2-1
 - when advantageous, 4-2
- cluster file system, 3-3
 - benefits, 2-4
 - storage in Real Application Clusters, 8-2
- Cluster Hardware Architecture, 2-1
- Cluster Manager, 3-2
 - failure detection, 10-13
- Cluster Manager (CM)
 - interaction with Global Cache Service (GCS), 3-7
 - node monitoring, 3-2
 - purpose, 3-2
- cluster reorganization, 10-13
- compatibility
 - shared and exclusive modes, 8-2
- components
 - for high availability, 10-3
- concurrent reads and writes on different nodes, 6-2
- concurrent reads on multiple nodes, 6-2
- concurrent writes on different nodes, 6-2
- configurations
 - for high availability, 10-16
- connect-time failover, 4-5
- consistency
 - multiversion read, 1-5
- consistent read (CR), 5-6
- consistent read versions, 5-3
- Control File, 10-14
- Control File Voting Results Record, 10-14
- CPUs, 2-3
- CREATE CONTROLFILE statement
 - exclusive mode, A-2
- CREATE DATABASE statement

- exclusive mode, A-2
- cross-registration, 10-21

D

- data blocks, 1-4
- data warehouse, 4-3
- database
 - backups, 1-5
- Database Configuration Assistant (DBCA), 4-2, 8-3
 - and the Global Services Daemon (GSD), 3-4
 - manageability in Real Application Clusters, 9-4
 - server parameter file and default location, 8-4
- database instance registration
 - client load balancing, 4-5
 - connect-time failover, 4-5
- database writer (DBWR), 6-3
- datafiles
 - mapping locks to blocks, B-2
 - maximum number, A-2
 - shared, 8-2
 - unspecified for locks, B-3
 - verification, 8-2
- DBMS_LIBCACHE
 - warming the library cache, 10-23
- deadlock detection, 7-2
- Decision Support Systems (DSS), 4-3
- dedicated server
 - and Primary/Secondary Instance, 10-18
- DIAG
 - Diagnosability Daemon, 3-4
- Diagnosability Daemon (DIAG), 3-4
- dictionary cache, 7-3
 - locks, 7-3
- dirty block, 6-4
- disaster planning
 - high availability, 10-4
- disk subsystems, 2-4

E

- enqueue
 - definition, 5-3
- enqueues
 - and recovery processing, 6-7

Ethernet
as used in Real Application Clusters, 2-4
exclusive (X) mode, 5-4
exclusive current (XCUR), 6-4
exclusive mode
compatibility, A-2

F

failover
basics of, 10-5
connect-time, 4-5
definition, 10-2
duration of, 10-6
host-based, 10-12
recovery processing in Real Application Clusters, 10-12
server-side, 10-11
transparent application failover, 4-5
failure
ALERT file, 8-2
detection by the Cluster Manager, 10-13
instance and recovery, 6-7
protection validation, 10-4
Fast-start Recovery, 10-2
Fast-start Rollback, 10-16
fault tolerance, 3-6
FDDI
as used in Real Application Clusters, 2-4
features, new, xxvi
file
ALERT, 8-2
archiving redo log, 1-6
maximum number, A-2
naming conventions, 8-3
redo log, 1-6
size, B-3
files
redo log, 8-4
free space management
automatic segment-space management, 8-5

G

Global Cache Service, 6-1

described, 3-6
distributed architecture, 3-6
LMON, use of, 10-13
Global Cache Service (GCS), 3-6, 5-2
data block tracking operations, 5-5
fault tolerant, 3-6
features, 3-6
interacting with CM, 3-7
processing example, 5-5
Global Cache Service Processes (LMSn), 3-4
processing for synchronization, 5-3
Global Enqueue Service (GCS) reconfiguration
phase of recovery processing, 6-7
Global Enqueue Service (GES), 3-6, 5-2
resource coordination, 7-1
Global Enqueue Service Daemon (LMD), 3-4
Global Enqueue Service Monitor (LMON), 3-4
global enqueues
automatically calculated, 5-3
Global Resource Directory, 3-6, 5-2
contents of, 5-2
Global Services Daemon (GSD)
manageability in Real Application Clusters, 9-5
Real Application Clusters-specific background
process, 3-4

H

hardware
for cluster database processing, 2-1
scalability, 4-4
Hardware and Network Scalability, 4-4
high availability, 1-2
benefit of cluster databases, 1-4
cluster interconnects, 10-3
configurations for, 10-16
configurations, benefits of, 10-24
definition, 1-4, 10-2
high availability node configurations
benefits of, 10-25
host-based failover, 10-12

I

initialization parameter files, 8-3

- initsid.ora parameter, 10-17
- Input/Output, 1-4
- installation
 - manageability components in Real Application Clusters, 9-2
- Instance, 10-14
- instance, xv
- instance failure
 - recovery, 6-7
- instance membership recovery (IMR), 10-14
- instance name, 4-6
- instance recovery
 - rollback segments, 8-6
- instances
 - recovery, 10-15
 - thread number, 8-4
- interconnect, 2-3, 3-2
 - and scalability, 4-4
 - as a cluster component, 2-2
 - definition, 2-4
 - redundancy of, 10-3
- interprocess communication (IPC), 2-3, 2-4
- intranode messaging
 - for synchronization processing, 5-3
- introduction
 - Real Application Clusters, 1-1
- I/O
 - interrupts, 4-6

L

- Lamport SCN generation, 5-6
- latency, 2-4
 - network, 4-4
- LCK
 - Lock Process, 3-4
- Levels of Scalability, 4-3
- library cache
 - warming, 10-23
- library cache locks, 7-3
 - parsing of statements, 7-3
- listener
 - connect-time failover, 4-5
 - transparent application failover, 4-5
- listener.ora file, 4-6

- LMD
 - Global Enqueue Service Daemon, 3-4
- LMON
 - and cluster reorganization, 10-13
 - Global Enqueue Service Monitor, 3-4
- LMSn
 - Global Cache Service Processes, 3-4
- load balancing
 - client load balancing, 4-5
- local coordination
 - becoming global, 5-3

- local resource coordination, 5-2
- Lock Process (LCK), 3-4
- locks, B-2
 - enqueue, 5-3
 - library cache, 7-3
 - mapping blocks to, B-2
 - set of files, B-3

M

- manageability
 - administration, 9-3
 - backup and recovery, 9-6
 - components in Real Application Clusters, 9-1
 - installation, 9-2
 - overview in Real Application Clusters, 9-2
 - performance monitoring, 9-6
- manageability
 - setup and configuration, 9-2
- MAX_COMMIT_PROPAGATION_DELAY
 - parameter
 - and Lamport SCN generation, 5-7
- mean time to failure (MTTF), 10-3
- media failure
 - access to files, 8-2
- memory, 1-4, 2-3, 2-4
- memory cache
 - transfers for synchronization, 5-3
- memory-mapped IPCs
 - how used, 2-4
- messages
 - access to files, 8-2

- ALERT file, 8-2
- mode
 - archiving, 1-6
- modes
 - resource, 5-4
- MTTR, 10-2
- multitiered application environments, 10-6
- multiversion read consistency, 1-5

N

- network
 - bandwidth, 4-4
 - latency, 4-4
 - scalability, 4-4
- networks
 - performance, improving by randomizing client requests, 4-5
- new features, xxvi
- N-node
 - Real Application Clusters configurations, 10-17
- NOARCHIVELOG mode
 - changing mode, A-2
 - offline backups, 1-6
- node
 - definition, 2-3
 - failure, 1-4
- node monitoring, 3-2
- nodes
 - hardware for, 2-1
 - high availability, 10-3
- null (N) mode, 5-4

O

- offline backups, 1-5
- OMF feature, 8-3
- online archiving, 1-5
- online backups, 1-5
- online redo log files
 - thread of redo, 8-4
- Online Transaction Processing (OLTP), 1-2
- operating system scalability, 4-6
- operating system-dependent (OSD) layer, 3-2
- Oracle

- backing up, 1-5
 - datafile compatibility, 8-2
- Oracle Call Interface (OCI), 10-11
- Oracle Enterprise Manager
 - and server parameter file administration, 8-3
 - manageability in Real Application Clusters, 9-3
 - performance in Real Application Clusters, 9-6

- Oracle Enterprise Manager (EM)
 - and the Global Services Daemon (GSD), 3-4
- Oracle Managed File feature, 8-3
- Oracle Parallel Execution, 4-3
 - described, 4-3
- Oracle Real Application Clusters Guard I, 10-2
- Oracle Real Application Clusters Guard II, 10-2
- Oracle9i Data Guard, 10-2
- Overview, 2-2

P

- parallel mode
 - file operation restrictions, A-2
- parameters
 - file (PFILE), 8-3
 - file, server, 8-3
 - settings, 8-3
- past image (PI)
 - definition, 6-3
- performance
 - manageability components in Real Application Clusters, 9-6
- PIs, discarded, 6-7
- PL/SQL, 7-3
- Preface, xv, xxv
- primary instance, 10-17
- primary instance role, 10-17
- Primary/Secondary Instance configuration, 10-17
 - dedicated server environments, 10-18
 - definition, 10-17
 - shared server environments, 10-21
 - warming the library cache, 10-23
- private rollback segments
 - acquisition, 8-5

Q

quorum disk, 3-3

R

RAID

in disaster protection, 10-5

randomizing requests among listeners, 4-5

read consistency

multiversion, 1-5

read-only access, 1-5

Real Application Clusters

architectural overview, 3-2

cost of ownership, 1-3

definition, 1-2

dictionary cache locks, 7-3

instance processes, 3-4

Oracle Parallel Execution, 4-3

recovery, 6-7

resource coordination overview, 5-2

shared disk components, 3-3

synchronization processes, 5-3

Real Application Clusters

system types that benefit from, 4-2

Real Application Clusters connect-time

failover, 4-5

Real Application Clusters Guard II, 10-25

recovery

access to files, 8-2

cache, 10-15

manageability components in Real Application Clusters, 9-6

media failure, 8-2

of instances, 10-15

Real Application Clusters and Cache

Fusion, 6-7

rolling back, 8-6

Recovery in Real Applications Clusters, 6-7

Recovery Manager (RMAN)

manageability components in Real Application Clusters, 9-6

recovery processing

in Real Application Clusters, 10-12

redo log file

archiving, 1-6

overwriting, 1-6

thread of redo, 8-4

remastering

during failover, 10-5

resources, 10-14

resources, 10-14

and recovery processing, 6-7

and system change numbers (SCN), 5-6

coordination, global, 5-2

coordination, local, 5-2

coordination, overview of, 5-2

mode, 5-4

mode, and buffer state, 6-4

remastering, 10-14

role, 5-4

resources

information in Global Resource Directory, 5-2

restrictions

file operations, A-2

roles

resource, 5-4

rollback segment undo mode

in Real Application Clusters, 8-5

rollback segment undo mode

private rollback segments, 8-5

row level locking

resource sharing system, 1-5

row locking, 1-5

rows

updates and enqueues, 5-3

S

scalability, xvi, 1-3

applications, 4-3, 4-6

four levels of, 4-3

hardware, 4-4

levels of, 4-3

network, 4-5

shared memory system, 4-6

- scale
 - applications, 1-2
- SCN, 5-6
- secondary instance, 10-17
- Server Control (SRVCTL)
 - manageability in Real Application Clusters, 9-5
 - Server Control (SRVCTL) Utility
 - and the Global Services Daemon (GSD), 3-4
 - server parameter file, 8-3
 - location in Real Application Clusters, 8-4
 - server-side failover, 10-11
 - service name, 4-6
 - service registration, 4-6
 - setting, B-2
 - setup and configuration
 - manageability components in Real Application Clusters, 9-2
- shared (S) mode, 5-4
- shared current (SCUR), 6-4
- shared high availability node configurations, 10-24
- shared memory system
 - scalability, 4-6
- shared mode
 - datafiles, 8-2
- Shared Server
 - and Primary/Secondary Instance, 10-18
- SHUTDOWN TRANSACTIONAL command, 10-8
- SHUTDOWN TRANSACTIONAL LOCAL command, 10-8
- shutdowns
 - transactional, 10-7
- SID, 8-3
 - and server parameter file, 8-3
- single node cluster databases, 7-3
- space management
 - automatic segment-space management, 8-5
- SQL statement
 - restrictions, A-2
- startup, 3-4
 - verifying access to files, 8-2
- Statspack
 - manageability and performance in Real Application Clusters, 9-6

- statistics available for Real Application Clusters, 9-6
- storage, 2-2, 2-3
 - storage
 - cluster file system, 8-2
 - structured query language (SQL), 7-3
 - switch archiving mode, A-2
 - symmetric multiprocessor, 4-6
 - synchronization processes
 - in Real Application Clusters, 5-3
 - System Change Number (SCN)
 - incrementation, 5-6
 - Lamport, 5-6
 - systems
 - types that benefit from Real Application Clusters, 4-2
 - system-specific Oracle documentation
 - datafiles, maximum number, A-2

T

- tablespace
 - backups, 1-5
- thread, 8-4
- THREAD option
 - private thread creation, 8-4
 - public thread creation, 8-4
- THREAD parameter
 - instance acquiring thread, 8-4
- transaction recovery, 10-16
- transaction systems, 4-2
- transactions
 - committed data, 1-5
 - concurrent, 1-5, 5-5
 - row locking, 1-5
 - updates, 1-5
- transparency
 - definition, 1-4
- Transparent Application Failover
 - definition, 10-6
 - uses of, 10-7

U

undo space management, 8-5

updates

 concurrent, 1-5

user-mode interprocess communication

 how used in Real Application Clusters, 2-4

V

V\$BH view, 6-4

voting disk, 3-3

W

warming the library cache

 DBMS_LIBCACHE, 10-23

Windows NT/Windows 2000

 quorum disk, 3-3

 voting disk, 3-3

write protocol and past image tracking, 6-3