Oracle9*i* Real Application Clusters

Deployment and Performance

Release 2 (9.2)

March 2002

Part No. A96598-01

ORACLE

Oracle9*i* Real Application Clusters Deployment and Performance, Release 2 (9.2)

Part No. A96598-01

Primary Author: Mark Bauer.

Contributing Authors: David Austin, Kotaro Ono, Stefan Pommerenk, Joao Rimoli, and Michael Zoll.

Contributors: Wilson Chan, Sashikanth Chandrasekaran, Mitch Flatland, Rick Greenwald, Bill Kehoe, Merrill Holt, Raj Kumar, Neil MacNaughton, Vinay Srihari, and Tak Wang.

Graphic Designer: Valarie Moore.

Contents

List of Figures

List of Tables

Send Us Your Comments

Oracle9*i* Real Application Clusters Deployment and Performance, Release 2 (9.2)

Part No. A96598-01

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 Deployment and Performance explains the deployment considerations for implementing applications on an Oracle9*i* Real Application Clusters database. This manual also provides post-deployment information about monitoring the performance of Real Application Clusters databases.

Information in this manual applies to Real Application Clusters as it runs on all operating systems. Where necessary, this manual refers to platform-specific documentation.

> **See Also:** The *Oracle9i Real Application Clusters Documentation Online Roadmap* to help you use the online Oracle9*i* Real Application Clusters Documentation set

This preface contains these topics:

Audience

Oracle9i Real Application Clusters Deployment and Performance is written for database professionals working with Real Application Clusters. Using this document requires a conceptual understanding of Real Application Clusters processing and its software and hardware components as described in *Oracle9i Real Application Clusters Concepts*. You should have installed Real Application Clusters using the document *Oracle9i Real Application Clusters Setup and Configuration* and related platform-specific documentation.

All single-instance Oracle database deployment and performance methodologies apply to Real Application Clusters. Therefore, you should also be familiar with the information in *Oracle9i Database Performance Tuning Guide and Reference*, and *Oracle9i Data Warehousing Guide*.

Organization

The three parts of this book and their contents are:

Part I: Introduction to Real Application Clusters Deployment

Part I introduces the high-level aspects of deploying applications in Real Application Clusters by describing how to take advantage of Oracle9*i* features. It also describes the deployment of internet-based applications in e-commerce and data warehouse environments.

Chapter 1, "Introduction to Application Deployment for Real Application Clusters"

This chapter provides an overview of deployment considerations for Real Application Clusters environments with regard to Oracle9*i* features.

Chapter 2, "Application Deployment in Real Application Clusters"

This chapter describes the deployment of online e-commerce and data warehouse applications for Real Application Clusters.

Part II: Deployment and Performance for Oracle9*i* Real Application Clusters

Part II describes deployment and performance monitoring for Real Application Clusters.

Chapter 3, "Database Deployment Techniques in Real Application Clusters"

This chapter describes database deployment considerations for Real Application Clusters.

Chapter 4, "Monitoring Real Application Clusters Performance"

This chapter presents general performance monitoring recommendations for Real Application Clusters.

Chapter 5, "Monitoring Performance in Real Application Clusters with Performance Manager"

This chapter explains how to monitor Real Application Clusters databases with Oracle Enterprise Manager.

Part III: Oracle Real Application Clusters Reference

Part III: contains reference information for Real Application Clusters deployment and performance.

Appendix A, "Configuring Multi-Block Locks (Optional)"

This appendix explains how to override the Real Application Clusters default resource control mechanisms.

Appendix B, "Using Free Lists and Free List Groups in Real Application Clusters (Optional)"

This appendix explains how to use free lists and free list groups.

Glossary

This glossary defines important terms used in this book.

Related Documentation

For more information, see these Oracle resources:

- *Oracle9i Real Application Clusters Documentation Online Roadmap*
- *Oracle9i Real Application Clusters Concepts*
- *Oracle9i Real Application Clusters Setup and Configuration*
- *Oracle9i Real Application Clusters Administration*
- *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
- *Oracle9i Database Performance Tuning Guide and Reference*
- *Oracle9i Data Warehousing Guide*
- Your platform-specific Oracle Real Application Clusters Guard installation guide

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 9*i* Real Application Clusters Real Application Clusters Guard I platform-specific installation manuals
- *Oracle9i Real Application Clusters Guard II Concepts, Installation, and Administration* on the Real Application Clusters Guard II software CD

Operating System-Specific Administrative 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* for more information about administering Real Application Clusters Guard I

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

Oracle9*i* Real Application Clusters Management

- *Oracle9i Real Application Clusters Administration*
- *Oracle Enterprise Manager Administrator's Guide*
- *Getting Started with the Oracle Diagnostics Pack*

Generic Documentation

- *Oracle9i Database Concepts*
- *Oracle9i Net Services Administrator's Guide*
- *Oracle9i Database Reference*
- *Oracle9i Database New Features*

Many of the examples in this book use the sample schemas of the **seed database**, which is installed by default when you install Oracle. Refer to *Oracle9i Sample Schemas* for information about how these schemas were created and how to use them.

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 for Windows Operating Systems

Conventions in Text

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

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Database Concepts* Ensure that the recovery catalog and target database do *not* 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 `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run `U`*`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

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

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

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

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`

Convention	Meaning	Example
...	Horizontal ellipsis points indicate either: ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code	`CREATE TABLE ... AS subquery;` `SELECT col1, col2, ... , coln FROM employees;`

.
.
.
	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	`SQL> SELECT NAME FROM V$DATAFILE;`
		`NAME`
		`----------`
		/fsl/dbs/tbs_01.dbf
		/fs1/dbs/tbs_02.dbf
		.
		.
		.
		`/fsl/dbs/tbs_09.dbf`
		9 rows selected.

Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	`acctbal NUMBER(11,2);` `acct CONSTANT NUMBER(4) := 3;`
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/system_password` `DB_NAME = database_name`

UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Conventions for Windows Operating Systems

The following table describes conventions for Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Choose Start >	How to start a program.	To start the Database Configuration Assistant, choose Start > Programs > Oracle - *HOME_NAME* > Configuration and Migration Tools > Database Configuration Assistant.
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the file name begins with \\, then Windows assumes it uses the Universal Naming Convention.	`c:\winnt"\"system32` is the same as `C:\WINNT\SYSTEM32`
`C:\>`	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the *command prompt* in this manual.	`C:\oracle\oradata>`

Special characters	The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	`C:\>exp scott/tiger TABLES=emp QUERY=\"WHERE job='SALESMAN' and sal<1600\"`
		`C:\>imp SYSTEM/`*`password`*
		`FROMUSER=scott TABLES=(emp, dept)`

HOME_NAME	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	`C:\> net start Oracle`*`HOME_NAME`*`TNSListener`

Convention	Meaning	Example
ORACLE_HOME and *ORACLE_BASE*	In releases prior to Oracle8*i* release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory that by default used one of the following names: ■ `C:\orant` for Windows NT ■ `C:\orawin95` for Windows 95 ■ `C:\orawin98` for Windows 98 This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is `C:\oracle`. If you install the latest Oracle release on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is C:\oracle\orann, where nn is the latest release number. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle9i Database Getting Started for Windows* for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.	Go to the *ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\admin` directory.

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.

Part I

Introduction to Deployment in Real Application Clusters

Part I introduces application deployment for Real Application Clusters and it explains how to take advantage of Oracle's high availability and scalability features in Real Application Clusters environments. The chapters in Part I are:

1

Introduction to Application Deployment for Real Application Clusters

This chapter describes **Oracle Real Application Clusters** application deployment by explaining considerations for taking advantage of the scalability and high-performance features of Real Application Clusters. This chapter includes the following topics:

- Taking Full Advantage of Oracle9i Features
- Implementing Oracle9i Features with Real Application Clusters

Taking Full Advantage of Oracle9*i* Features

To optimally deploy applications with Oracle9*i* Real Application Clusters, consider the issues for the Oracle features described in this chapter. Proper implementation of these features minimizes deployment problems. It also ensures that your system takes full advantage of the breakthrough technology of **Cache Fusion** and the high-performance features of Real Application Clusters.

The feature descriptions in this chapter provide a starting point for Real Application Clusters application deployment that you can use *after* implementing single-instance deployment methods.

Implementing Oracle9*i* Features with Real Application Clusters

The considerations described in this section relate to features that are unique to Oracle and that enhance the performance of Real Application Clusters. The features discussed in this section are:

- Storage in Real Application Clusters
- High Availability and Failover in Real Application Clusters
- The Shared Server in Real Application Clusters
- Connection Load Balancing
- Transparent Application Failover in Real Application Clusters
- PL/SQL in Real Application Clusters
- Recovery Manager (RMAN) in Real Application Clusters

Storage in Real Application Clusters

The advanced storage features of Oracle9*i* provide exceptional performance for Real Application Clusters. These features include automatic segment-space management, automatic undo management, and the server parameter file. In addition, instead of using **raw devices**, on certain platforms you can store files for Real Application Clusters in cluster file system files.

Cluster File Systems in Real Application Clusters

Cluster file systems simplify Real Application Clusters installation and administration. Using cluster file systems eliminates the need to manage raw devices. Cluster file systems also offer scalable, low latency, highly resilient storage

that significantly reduces costs. Refer to your vendor documentation for details about implementing cluster file systems.

High Availability and Failover in Real Application Clusters

High availability configurations have redundant hardware and software that maintain operations despite failures and that avoid single points-of-failure. When failures occur, failover moves the processing performed by the failed component to a backup component. Oracle's failover process quickly re-masters resources, recovers partial or failed transactions, and rapidly restores the system.

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

Primary/Secondary Instance and Active/Active Instance Configurations

Primary / Secondary Instance configurations are the least complicated type of high availability configuration to configure. These are also the easiest type of configurations to administer. For example, the administrative overhead for a primary database in this configuration is the same as the overhead of a single-instance configuration.

In Primary / Secondary configurations, the second instance does not have to remain idle. For example, you can use the second instance for read-only operations. You do not have exceptional scalability with Primary / Secondary configurations, but you do have high availability.

Active / active instance configurations, on the other hand, have typically been more complex to configure. However, with the advent of Real Application Clusters Guard II, easy to manage *full* active configurations are available.

Real Application Clusters Guard II

Real Application Clusters Guard II extends the notion of a two-node active / active cluster to an n-node, fully active cluster where all instances can support the services in the cluster database.

> **See Also:** *Oracle9i Real Application Clusters Guard II Concepts, Installation, and Administration* for more information about Real Application Clusters Guard II

Oracle Real Application Clusters Guard I

Oracle Real Application Clusters Guard I, which is an enhanced configuration of Real Application Clusters tightly integrates Oracle's enhanced recovery features within the **cluster database** framework of your platform.

> **See Also:** *Oracle9i Real Application Clusters Real Application Clusters Guard I - Concepts and Administration* for more conceptual information about Real Application Clusters Guard I

Data Guard

Oracle9*i* Data Guard works with standby databases to protect your data against errors, failures, and corruptions that might otherwise destroy your database. Data Guard protects critical data by automating the creation, management, and monitoring aspects of standby database environments. Data Guard automates the otherwise manual process of maintaining a transactionally consistent copy of an Oracle database to recover from the loss or damage of the production database.

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

The Shared Server in Real Application Clusters

Real Application Clusters with the functionality of the **shared server** feature can process thousands of concurrently connected database users. Shared server efficiently manages the connection load for many users and it operates similarly to the way that a **transaction monitor** operates.

Real Application Clusters with shared server significantly enhances the performance of applications running on two or more smaller computers. You do not need to rewrite your applications to use shared server. In fact, some applications perform better with shared server than without.

With shared server configurations, user processes connect to a **dispatcher**. The dispatcher then directs multiple incoming network session requests to a common queue. When a server process becomes available, the dispatcher connects the incoming request to the idle dispatcher. When the connection is no longer needed, the server process is available for another request. Thus, a small set of server processes can serve a large number of clients.

Connection Load Balancing

The **connection load balancing** feature automatically distributes connections among active instances. Connection load balancing does this based on the workload of each **node** and instance in a cluster. You can use **connection load balancing**, in both shared server and **dedicated server** environments. Real Application Clusters and Cache Fusion combined with connection load balancing allow you to run all types of applications without application or data partitioning.

Note: You must install Oracle Net to use shared server and its **load balancing** features.

Transparent Application Failover in Real Application Clusters

The **transparent application failover (TAF)** feature automatically reconnects applications to the database if the connection fails. Because the reconnection happens automatically within the OCI library, you do not need to change the client application to use TAF.

Because most TAF functionality is implemented in client-side network libraries (OCI), the client must use the Oracle Net OCI libraries to take advantage of TAF functionality. Therefore, to implement TAF in Real Application Clusters, make sure you use JDBC OCI instead of PL/SQL packages.

Note: Although Real Application Clusters supports both thin JDBC and JDBC OCI, TAF is only supported with JDBC OCI.

You can also use TAF in Primary/Secondary Instance configurations. If you do this, then use the `INSTANCE_ROLE` parameter in the Connect Data portion of the connect descriptor to configure explicit secondary instance connections.

See Also: *Oracle9i Real Application Clusters Administration* for information about using the `INSTANCE_ROLE` parameter in Real Application Clusters and the *Oracle9i Net Services Administrator's Guide* for more detailed information and examples for this parameter

To use TAF, you must have a license for the **Oracle9i Enterprise Edition**. Because TAF was designed for Real Application Clusters, it is much easier to configure TAF for that environment. However, TAF is not restricted for use with Real Application

Clusters environments. You can also use TAF for single instance Oracle. In addition, you can use TAF for the following system types:

- Oracle Real Application Clusters Guard
- Replicated Systems
- Data Guard

See Also:

- *Oracle9i Real Application Clusters Concepts* for more conceptual information about TAF in Real Application Clusters
- *Oracle9i Net Services Administrator's Guide* and the *Oracle Call Interface Programmer's Guide* for more information about TAF
- *Oracle9i Replication* for more information about Oracle replication

PL/SQL in Real Application Clusters

PL/SQL is Oracle's procedural extension of SQL. PL/SQL is an advanced fourth-generation programming language that offers features such as data encapsulation, overloading, collection types, exception handling, and information hiding. PL/SQL also offers seamless SQL access, tight integration with the Oracle server, as well as tools, portability, and security.

See Also: The *PL/SQL User's Guide and Reference* for more information about PL/SQL

Recovery Manager (RMAN) in Real Application Clusters

Recovery Manager (RMAN) is an Oracle tool that you can use to backup, copy, restore, and recover each datafile, **control file**, and archived redo log. You can invoke RMAN as a command line utility or use it through **Oracle Enterprise Manager**.

RMAN automates many backup and recovery tasks. For example, RMAN automatically locates the appropriate backups for each datafile and copies them to the correct destinations. This eliminates the manual, error-prone effort of using operating system commands to accomplish the same task.

You must configure RMAN so that all instances can access all the archive logs throughout the cluster. When one instance fails, the surviving instance that performs recovery must access the archive logs of the failed instance.

See Also: *Oracle9i Real Application Clusters Administration* for details on configuring RMAN for use with Real Application Clusters and *Oracle9i Recovery Manager User's Guide and Reference* for detailed information about RMAN

The next chapter examines application development for online e-commerce and decision support systems. The remainder of this book examines the deployment and performance of applications that use Real Application Clusters databases.

2

Application Deployment in Real Application Clusters

This chapter discusses both the deployment of online e-commerce (OLTP) applications and the use of data warehouse applications in **Oracle Real Application Clusters** environments. This chapter also briefly describes application performance monitoring. The topics in this chapter are:

- Deployment Strategies for Real Application Clusters-Based Applications
- Deploying E-Commerce Applications with Real Application Clusters
- Deploying Data Warehouse Applications with Real Application Clusters
- Deploying Internet-Based Applications

Deployment Strategies for Real Application Clusters-Based Applications

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

Deploying E-Commerce Applications with Real Application Clusters

Cache Fusion makes Real Application Clusters databases the optimal deployment servers for online e-commerce applications. This is because these types of applications require:

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

The high availability features of Oracle and Real Application Clusters can re-distribute and load balance the workloads to surviving instances without interrupting processing. Real Application Clusters also provides excellent scalability so that if you add or replace a **node**, Oracle re-masters resources and re-distributes processing loads without reconfiguration or application re-partitioning.

The application workload management feature of Real Application Clusters is highly dynamic. Real Application Clusters can alter workloads in real-time based on changing business requirements. This occurs in a manageable environment with minimal administrative overhead. The dynamic resource allocation capabilities of the Cache Fusion architecture provide optimal performance for online applications with great deployment flexibility.

Flexible Implementation with Cache Fusion

E-commerce requirements, especially the requirements of online transaction processing systems, have frequently changing workloads. To accommodate this, Real Application Clusters remains flexible and dynamic despite changes in system load and system availability. Real Application Clusters addresses a wide range of service levels that, for example, fluctuate due to:

- Varying user demands

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

To accommodate these requirements, it is impractical and often too complex to partition packaged e-commerce applications. Once deployed, such applications can access hundreds or even thousands of tables. A common recommendation to meet these service levels is to purchase a larger server. Or you may be told to segment your application or application modules across distinct databases. Segmenting applications, however, can fragment data and constrain a global enterprise-wide view of your information.

Real Application Clusters eliminates the need to purchase excess hardware and it avoids application segmentation. Instead, Real Application Clusters meets these demands by dynamically migrating the mastering of database resources to adapt to changing business requirements and workloads.

Deploying Data Warehouse Applications with Real Application Clusters

This section discusses deploying data warehouse systems in Real Application Clusters environments by briefly describing the data warehouse features available in shared disk architectures. The topics in this section are:

- Speed-Up for Data Warehouse Applications on Real Application Clusters
- Dynamic Parallel-Aware Query Optimization

> **See Also:** *Oracle9i Data Warehousing Guide* for detailed information about implementing data warehouse applications in Real Application Clusters environments

Speed-Up for Data Warehouse Applications on Real Application Clusters

Real Application Clusters is ideal for data warehouse applications because it augments the single instance benefits of Oracle. Real Application Clusters does this by maximizing the processing available on all nodes of a **cluster database** to provide speed-up and scale-up for data warehouse systems.

For example, e-businesses strategically use data warehouse systems to acquire customers and expand markets. Companies can use product promotions to gather information using data warehouse systems. They can then customize client lists to best fit the profiles of the company's target demographics.

Flexible Parallelism within Real Application Clusters Environments

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

Real Application Clusters takes full advantage of parallel execution by distributing parallel processing to all the nodes in your cluster. The number of processes that can participate in parallel operations depends on the **degree of parallelism (DOP)** assigned to each table or index.

Function Shipping On loosely coupled systems, Oracle's parallel execution technology uses a function shipping strategy to perform work on remote nodes. Oracle's parallel architecture uses function shipping when the target data is located on the remote node. This delivers efficient parallel execution and eliminates unneeded internode data transfers over the **interconnect**.

Exploitation of Data Locality On some hardware systems, powerful data locality capabilities were more relevant when shared nothing hardware systems were popular. However, almost all current cluster systems use a shared disk architecture.

On shared nothing systems, each node has direct hardware connectivity to a subset of disk devices. On these systems it is more efficient to access local devices from the *owning* nodes. Real Application Clusters exploits this affinity of devices to nodes and delivers performance that is superior to shared nothing systems using cluster configurations and a shared disk architecture.

As with other elements of Cache Fusion, Oracle's strategy works transparently without data partitioning. Oracle dynamically detects the disk on which the target data resides and makes intelligent use of the data's location in the following two ways:

- Oracle spawns parallel execution server processes on nodes where the data is waiting to be processed
- Oracle assigns local data partitions to each sub-process to eliminate or minimize internode data movement

Dynamic Parallel-Aware Query Optimization

Oracle's cost-based optimizer considers parallel execution when determining the optimal execution plans. The optimizer dynamically computes intelligent heuristic defaults for parallelism based on the number of processors.

An evaluation of the costs of alternative access paths—table scans versus indexed access, for example—takes into account the degree of parallelism (DOP) available for the operation. This results in Oracle selecting execution plans that are optimized for parallel execution.

Oracle also makes intelligent decisions in Real Application Clusters environments with regard to intranode and internode parallelism. For intranode parallelism, for example, if a SQL statement requires six query sub-processes and six CPUs are idle on the local node, or the node to which the user is connected, then the SQL statement is processed using local resources. This eliminates query coordination overhead across multiple nodes.

Continuing with this example: if there are only two CPUs on the local node, then those two CPUs and four CPUs of another node are used to complete the SQL statement. In this manner, Oracle uses both internode and intranode parallelism to provide speed-up for query operations.

In all real world data warehouse applications, SQL statements are not perfectly partitioned across the different parallel execution servers. Therefore, some CPUs in the system complete their assigned work and become idle sooner than others. Oracle's parallel execution technology is able to dynamically detect idle CPUs and assign work to these idle CPUs from the execution queue of the CPUs with greater workloads. In this way, Oracle efficiently re-distributes the query workload across all of the CPUs in the system. Real Application Clusters extends these efficiencies to clusters by re-distributing the work across all the nodes of the cluster.

Load Balancing for Multiple Concurrent Parallel Operations

Load balancing distributes parallel execution server processes to spread CPU and memory use evenly among nodes. It also minimizes communication and remote I/O. Oracle does this by allocating parallel execution servers to the nodes that are running the fewest number of processes.

The load balancing algorithm maintains an even load across all nodes. For example, if a degree of parallelism of eight is requested on an eight-node system with one CPU for each node, the algorithm places two servers on each node. If the entire parallel execution server group fits on one node, then the load balancing algorithm places all the processes on a single node to avoid communications overhead. For

example, if you use a DOP of 8 on a two-node cluster with 16 CPUs for each node, then the algorithm places all 8 parallel execution server processes on one node.

Using Parallel Instance Groups

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

Establish instance group membership on an instance-by-instance basis by setting the INSTANCE_GROUPS initialization parameter to a name representing one or more instance groups. For example, on a 32-node system owned by both a Marketing and a Sales organization, you could assign half the nodes to one organization and the other half to the other organization using instance group names. To do this, assign nodes 1-16 to the Marketing organization using the following parameter syntax in your **initialization parameter file**:

```
sid|1-16|.INSTANCE_GROUPS=marketing
```

Then assign nodes 17-32 to Sales using the following syntax in the parameter file:

```
sid|17-32|.INSTANCE_GROUPS=sales
```

Activate the nodes owned by Sales to spawn a parallel execution server process by entering the following:

```
ALTER SESSION SET PARALLEL_INSTANCE_GROUP = 'sales';
```

In response, Oracle allocates parallel execution server processes to nodes 17-32. The default value for PARALLEL_INSTANCE_GROUP is all active instances.

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

Disk Affinity

Disk affinity refers to the relationship of an instance to the data that it accesses. The more often an instance accesses a particular set of data, the greater the affinity that instance has to the disk on which the data resides.

Disk affinity is used for parallel table scans, parallel temporary tablespace allocation, parallel DML, and parallel index scans. It is not used for parallel table

creation or parallel index creation. Access to temporary tablespaces preferentially uses local datafiles. It guarantees optimal space management extent allocation. Disks striped by the operating system are treated by disk affinity as a single unit.

Without disk affinity, Oracle attempts to balance the allocation of parallel execution servers evenly across instances. With disk affinity, Oracle allocates parallel execution servers for parallel table scans on the instances that most frequently access the requested data.

Assume that Oracle is performing a full table scan on table T and that instances 1, 2, and 3 have a affinity for that table and instance 4 does not:

- If a query requires two instances, then two instances from the set 1, 2, and 3 are used.
- If a query requires three instances, then instances 1, 2, and 3 are used.
- If a query requires four instances, then all four instances are used.
- If there are two concurrent operations against table T, each requiring three instances, and enough processes are available on the instances for both operations, then both operations use instances 1, 2, and 3. Instance 4 is not used. In contrast, without disk affinity, instance 4 is used.

> **See Also:** *Oracle9i* Real Application Clusters Concepts for more information about instance affinity

Deploying Internet-Based Applications

Testing your application's scalability, availability, and load balancing capabilities are some of the most challenging aspects of internet-based application deployment. In rapid prototyping environments, for example, you can internally test and benchmark your site with a limited number of Real Application Clusters instances on a test hardware platform.

All applications have limits on the number of users they support given the constraints of specific hardware and software architectures. To accommodate anticipated demand, you can estimate the traffic loads on your system and determine how many nodes you need. You should also consider peak workloads.

Your scalability tests must also simulate user access to your Web site. To do this, configure your traffic generator to issue pseudo `get` commands as used in the Hypertext Transfer Protocol (http). This tests your system's performance and load processing capabilities under fairly realistic conditions.

You can then conduct structured, Web-based testing using traffic generators to stress test your system. This type of persistent testing against your most aggressive performance goals should reveal any capacity limitations.

After making further enhancements as dictated by your testing results, prototype your site to early adopters. This enables you to obtain real-world benchmarks against which you can further refine your system's performance.

Note: These techniques are not Real Application Clusters-specific. You could also use them for deploying single-instance Web-based Oracle database applications.

Application Performance in Real Application Clusters

You use the same methodology to develop and tune applications in Real Application Clusters as you use for single-instance Oracle databases. If your application ran efficiently on a single-instance Oracle database, then it will run well on Real Application Clusters.

If necessary, refer to the deployment techniques outlined in Chapter 3, "Database Deployment Techniques in Real Application Clusters" in the unlikely event that your application experiences hot spots. You can also refer to the performance monitoring recommendations in Chapter 4, "Monitoring Real Application Clusters Performance".

Administrative Aspects of System Scalability for Real Application Clusters

You may need to add nodes before deployment or during production to accommodate growth requirements or to replace failed hardware. Adding a node to your Real Application Clusters environment involves two main steps:

- Adding a node at the clusterware layer
- Adding a node at the Oracle layer

To add a node, connect the hardware according to your vendor's installation instructions. Then install the operating system-dependent (OSD) clusterware. Complete the installation of the instance on the new node using the Oracle Universal Installer (OUI) and the Database Configuration Assistant (DBCA). On some platforms you can dynamically add nodes and instances.

> **See Also:** *Oracle9i Real Application Clusters Administration* for detailed procedures on adding nodes and instances

The next part of this book describes application deployment and performance monitoring considerations for Real Application Clusters databases.

Part II

Deployment and Performance in Real Application Clusters

Part II describes considerations for application deployment in Real Application Clusters. Part II also describes how to monitor statistics and, if needed, adjust parameters to improve Real Application Clusters performance. It also contains information about using Oracle Enterprise Manager to monitor Real Application Clusters database performance.

> **Note:** If your application performed well on a single-instance Oracle database, then you should not have to tune your application for deployment on Real Application Clusters.

Part II includes the following chapters:

- Chapter 3, "Database Deployment Techniques in Real Application Clusters"
- Chapter 4, "Monitoring Real Application Clusters Performance"
- Chapter 5, "Monitoring Performance in Real Application Clusters with Performance Manager"

3

Database Deployment Techniques in Real Application Clusters

This chapter describes database deployment techniques for **Oracle Real Application Clusters** environments. The topics in this chapter are:

- Principles of Database Deployment for Real Application Clusters
- Tablespace Use in Real Application Clusters
- Object Creation and Performance in Real Application Clusters
- Using Sequence Numbers in Real Application Clusters
- Application Tuning Recommendations for Real Application Clusters
- Advanced Queuing and Real Application Clusters
- Conclusions and a Summary of Guidelines

Principles of Database Deployment for Real Application Clusters

When deploying databases for Real Application Clusters, use the same methodologies that you would use for single-instance databases. If you have an effective single-instance design, then your application will run well on Real Application Clusters.

Tablespace Use in Real Application Clusters

Single-instance Oracle database tablespace usage methodologies also apply to tablespace use in Real Application Clusters databases. You control the objects that reside in specific tablespaces in Real Application Clusters using the same methods that you use to control objects in tablespaces in single-instance Oracle databases.

To simplify tablespace administration, Oracle Corporation strongly recommends that you use automatic segment-space management in Real Application Clusters environments. Automatic segment-space management greatly improves extent management and reduces the overhead associated with searching for free space and allocating it when inserting new data.

If you cannot use locally managed tablespaces which are required for automatic segment-space management, then refer to the discussion about using free list groups as described in Appendix B, "Using Free Lists and Free List Groups in Real Application Clusters (Optional)". You may, also want to configure sequence number generation if every node uses sequence numbers.

> **See Also:** *Oracle9i Database Performance Planning* for more information about tablespaces and performance methodologies

Object Creation and Performance in Real Application Clusters

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

Object Partitioning

Cache Fusion eliminates most of the costs associated with globally shared database partitions by efficiently synchronizing this data across the cluster. However, object partitioning, without changing your application, can improve performance for hot

blocks in tables and indexes. This is done by re-creating objects as hash or composite partitioned objects.

For example, consider a table that has a high insert rate which also uses a sequence number as the primary key of its index. All sessions on all nodes access the right-most index leaf block. Therefore, unavoidable index block splits can create a serialization point that results in a bottleneck. To resolve this, rebuild the table and its index, for example, as a 16-way hash partitioned object. This evenly distributes the load among 16 index leaf blocks.

Note: These techniques also apply to single-instance environments; they are not specific to Real Application Clusters.

Using Sequence Numbers in Real Application Clusters

When deploying applications for Real Application Clusters, cache the Oracle sequence numbers whenever possible. To optimize sequence number use, each instance's cache must be large enough to accommodate the sequences. The default cache size holds 20 sequence numbers. To increase this, for example to hold 200, use this syntax:

```
ALTER SEQUENCE sequence_name CACHE 200;
```

To suppress caching, however, use the ordering feature. It is normal to lose some numbers after instance failures or after executing the `SHUTDOWN` command. This is true even in single-instance configurations. If you cannot avoid ordering, then you may need to disable sequence caching. In this case, expect some performance overhead.

> **See Also:** *Oracle9i Database Concepts* for more information about sequences

Detecting Global Conflicts for Sequences

If sequences are insufficiently cached or not cached at all, then performance problems can result in an increase in wait times. In this case, examine the statistics in the `V$SYSTEM_EVENT` view to determine whether the problem is due to the use of sequences.

In such situations, the `DC_SEQUENCES` parameter's ratio of `DLM_CONFLICTS` to `DLM_REQUESTS` will be high. If this ratio exceeds 10 to 15%, and the row cache lock

wait time is a significant portion of the total wait time, then it is likely that the service time deterioration is due to insufficiently cached sequences.

Using Database Tables to Generate Sequence Numbers

If your application cannot afford to lose sequence numbers, then implement sequences by storing them in database tables. However, there can be some performance overhead associated with implementing this strategy. This is true even in single-instance environments. As a general recommendation, rows storing sequence numbers should be locked for only a brief period.

Real Application Clusters can experience a minor amount of additional overhead as a result of the cache coherence needed for storing sequence numbers. If a single data block stores several sequence numbers and if more than one instance needs those sequence numbers, then the data block can be frequently transferred among the instances.

To minimize the adverse effects of frequent block transfers, set PCTFREE to a high value so that Oracle stores only a single row of the table containing the sequence numbers in each data block. In this case, the cache transfers only occur when the instances concurrently request the same sequence number.

Application Tuning Recommendations for Real Application Clusters

This section explains how to identify and resolve performance issues in Real Application Clusters-based applications. It contains the following topics:

- Query Tuning Tips
- Transaction Processing Tips
- Advanced Queuing and Real Application Clusters

Query Tuning Tips

Query-intensive applications benefit from tuning techniques that maximize the amount of data for each I/O request. Begin monitoring performance before attempting to use these techniques and continue monitoring performance afterward to assess their effectiveness. The techniques are:

- Using Large Block Sizes
- Increasing the Value for DB_FILE_MULTIBLOCK_READ_COUNT

Using Large Block Sizes

Use a large block size to increase the number of rows that each operation retrieves. This also reduces the depth of your application's index trees. Your block size should be at least 8K if your database is used primarily for processing queries.

Increasing the Value for DB_FILE_MULTIBLOCK_READ_COUNT

Also set the value for `DB_FILE_MULTIBLOCK_READ_COUNT` to the largest possible value. Doing this improves the speed of full table scans by reducing the number of reads required to scan a table. Note that system I/O is limited by the block size multiplied by the number of blocks read.

If you use operating system **striping**, then set the stripe size to `DB_FILE_MULTIBLOCK_READ_COUNT` multiplied by the `DB_BLOCK_SIZE` multiplied by 2. If your system can differentiate index stripes from table data stripes, then use a stripe size of `DB_BLOCK_SIZE` multiplied by 2 for indexes.

Also consider:

- Using read-only tablespaces for datafiles and indexes
- Defining tablespaces that hold temporary segments as type `TEMPORARY`

> **See Also:** *Oracle9i Database Performance Planning* for more information about using parallelism for improving query performance in Data Warehouse environments

Transaction Processing Tips

Transaction-based applications generally write more data to disk than other application types. To improve the ability of the database writer processes (DBWR*n*) to write large amounts of data quickly, use asynchronous I/O. If the access is random, then consider using a smaller block size. Monitor your application's performance both before and after initiating this method to make sure your system's performance is acceptable.

> **Note:** You cannot use this technique on all systems types and not all platforms support asynchronous I/O.

Advanced Queuing and Real Application Clusters

Using advanced queuing in Real Application Clusters environments can introduce functionality and performance-related issues as described in this section under the following topics:

Queue Table Instance Affinity

Queue table instance affinity enables you to assign primary and secondary instance properties to queue tables. This enables automatic assignment of queue table ownership when instances shut down and restart. You can evaluate queue table instance affinity by querying the following views:

- DBA_QUEUE_TABLES
- USER_QUEUE_TABLES

Global Cache Service Resource Acquisition

Global Cache Service resource acquisition is more expensive than local resource or local enqueue acquisition. If you improperly deploy advanced queuing, then its resource control behavior can adversely affect performance in Real Application Clusters environments. To avoid this, consider doing the following:

- Disable the locks on queue tables
- Reduce the number of COMMITs

If you cannot use automatic segment-space management, then increase the number of blocks that are added to a free list when advancing the high water mark.

Advanced Queuing and Queue Table Cache Transfers

In general, cache transfers of queue table data blocks and queue table index blocks can occur under the following circumstances:

- Queues are accessed simultaneously from different instances
- Oracle incorrectly assigns queue table ownership so that a queue monitor schedules a queue from an instance that is different from the instance where the enqueue or dequeue operations are performed
- Oracle must perform space transactions on the queue table

The frequency of cache transfers for queue table blocks can be reduced by creating queue tables with affinity to a particular instance, and then accessing the queue or queue table from this instance.

See Also: *Oracle9i Application Developer's Guide - Advanced Queuing* for general information about using Advanced Queuing

Conclusions and a Summary of Guidelines

Real Application Clusters and Cache Fusion introduce an improved diskless algorithm that efficiently manages cache coherency and enables you to deploy less complicated database designs and still achieve optimal performance. Real Application Clusters with Cache Fusion eliminates most of the processing overhead that might normally exist when multiple instances contend for resources.

Generally speaking, 80% or more of any performance issues result from 20% or less of a given workload. If you first attempt to deal with the 20% by observing some simple guidelines, then you can produce tangible benefits with minimal effort. You can address these problems by implementing any or all of the following:

- Use automatic segment-space management and locally managed tablespaces to reduce extent management processing
- Use read-only tablespaces wherever data remains constant
- Use Oracle sequences to generate unique numbers and set the CACHE parameter to a high value

4

Monitoring Real Application Clusters Performance

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

- Overview of Monitoring Real Application Clusters Databases
- Configuration Recommendations for Optimal Performance
- Performance Views in Real Application Clusters
- Real Application Clusters Performance Statistics
- Using Statspack and Statistics to Monitor Real Application Clusters Performance

> **See Also:** Chapter 5, "Monitoring Performance in Real Application Clusters with Performance Manager" for information about monitoring Real Application Clusters performance with Oracle Enterprise Manager

Overview of Monitoring Real Application Clusters Databases

All single **instance** tuning practices apply to applications running on Real Application Clusters databases. Therefore, review and implement the single-instance tuning methodologies described in *Oracle9i Database Performance Planning*.

Real Application Clusters databases should not require any more tuning than single-instance Oracle databases. This is because Cache Fusion does not use disk writes for cache coherency. Therefore, your performance monitoring effort for Real Application Clusters databases should be limited to the steps outlined in this chapter.

Configuration Recommendations for Optimal Performance

This section describes the following Real Application Clusters-specific configuration recommendations:

- Using User-Mode IPC Protocols
- Sizing the Buffer Cache and Shared Pool
- Verifying the Interconnect Settings for Real Application Clusters

Using User-Mode IPC Protocols

Because Cache Fusion exploits high speed IPCs, Real Application Clusters benefits from the performance gains of the latest technologies for low latency communication links used in **cluster database** interconnects. You can expect even greater performance gains if you use more efficient protocols, such as **Virtual Interface Architecture (VIA)** and user-mode IPCs.

Cache Fusion reduces CPU use with user-mode IPCs, also known as **memory-mapped IPCs**, for UNIX, Windows NT, and Windows 2000 platforms. If the appropriate hardware support is available, then **operating system context switches** are minimized beyond the basic reductions achieved with Cache Fusion alone. This also eliminates costly data copying and system calls.

If your hardware efficiently implements them, then user-mode IPCs can reduce CPU use. This is because user processes in user-mode IPCs communicate without using operating system kernels. In other words, user processes do not have to switch from user execution mode to kernel execution mode.

Sizing the Buffer Cache and Shared Pool

The buffer cache and shared pool capacity requirements in Real Application Clusters are slightly greater than those in single-instance Oracle databases. To facilitate recovery, Real Application Clusters may need additional memory to maintain duplicates of data blocks, or Past Images (PI), for blocks cached in more than one instance. Therefore, increase the size of the buffer cache by about 10% and increase the size of the shared pool by about 15%.

As in single-instance Oracle environments, you may be able to improve the buffer cache hit ratio by more aggressively flushing buffers that are not used frequently. For example, you can increase database writer (DBWR*n*) activity by using incremental checkpoints.

The rate of incremental checkpointing depends on the settings for various parameters, such as `FAST_START_MTTR_TARGET`, `LOG_CHECKPOINT_TIMEOUT`, and `LOG_CHECKPOINT_INTERVAL`. Setting these parameters to nonzero values results in more frequent writing of dirty or cold buffers. Consequently, the duration of instance recovery decreases and buffer cache use is more efficient. This results in more space being available to cache data blocks.

> **See Also:** *Oracle9i Database Performance Planning* for more information about checkpointing and performance

Verifying the Interconnect Settings for Real Application Clusters

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

Influencing Interconnect Processing

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

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

Message traffic for Global Cache Service (GCS) and Global Enqueue Service (GES) processing must use the appropriate interconnect. If you are uncertain about the IP

address or the NIC that Real Application Clusters-related traffic uses, then execute the following platform-independent SQL*Plus statements:

```
SQL> oradebug setmypid
SQL> oradebug ipc
```

This command sequence causes Oracle to write information about the IP address that Oracle is using for interconnect traffic to a trace file in the `user_dump_dest` directory. Although you should rarely need to set this parameter, you can use the `CLUSTER_INTERCONNECTS` parameter to assign a private network IP address or NIC as in the following example:

```
CLUSTER_INTERCONNECTS=10.0.0.1
```

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

> **See Also:** *Oracle9i Real Application Clusters Administration* for information about setting the `CLUSTER_INTERCONNECTS` parameter as well as your platform-specific documentation for how Oracle interprets settings for this parameter

Performance Views in Real Application Clusters

Each instance has a set of instance-specific views. You can also query global dynamic performance views to retrieve performance information from all qualified instances. Global dynamic performance view names are prefixed with *GV$*.

A global view contains all columns from its respective instance-specific view as well as the `INST_ID` column. This column displays the **instance number** from which Oracle obtains the associated instance-specific information. You can use it as a filter to retrieve information from a subset of instances. For example, the following query retrieves information from instances 2 and 5:

```
SELECT INST_ID,NAME,VALUE FROM GV$SYSSTAT WHERE NAME LIKE 'global cache%' AND
(INST_ID =2 OR INST_ID=5);
```

Each global view contains a `GLOBAL` hint that creates a query executed in parallel to retrieve the contents of the local views on each instance. If the number of parallel execution processes in an instance reaches the limit of the value set for `PARALLEL_MAX_SERVERS` and you submit an additional query against a GV$ view, then Oracle spawns one additional **parallel execution** process for this purpose. The extra process is not available for parallel operations other than to perform GV$ queries.

Note: Oracle does not spawn additional parallel execution server processes to accommodate GV$ queries if you have set PARALLEL_MAX_SERVERS to zero for an instance.

If the number of parallel execution processes on an instance reaches the value set for PARALLEL_MAX_SERVERS and you issue multiple GV$ queries, then all queries except for the first query will fail. In most parallel queries, if a process could not be allocated, then it would result in either an error or in a sequential execution of the query by the query coordinator.

See Also:

- "Flexible Parallelism within Real Application Clusters Environments" on page 2-4
- *Oracle9i Database Reference* for restrictions on GV$ views and complete descriptions of related parameters and views

Creating Real Application Clusters Data Dictionary Views with CATCLUST.SQL

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

See Also: *Oracle9i Database Reference* for more information on dynamic performance views

Real Application Clusters Performance Statistics

This section provides an overview of statistics from V$ and GV$ views that you can use to evaluate block traffic in your cluster. Use these statistics to analyze interconnect block transfer rates as well as the overall performance of your Real Application Clusters database. This section includes the following topics:

- The Content of Real Application Clusters Statistics
- Recording Statistics

The Content of Real Application Clusters Statistics

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

Many statistics measure the work done by different components of the database kernel, such as the cache layer, the transaction layer, or the I/O layer. For example, timed statistics reveal the amount of time spent processing certain requests and the amount of time waited for specific events. Oracle records most statistics in each instance's System Global Area (SGA).

Recording Statistics

Oracle Corporation recommends that you record statistics about the rates at which certain events occur. Maintaining a history of system performance identifies trends as these statistics change. Performance trends also highlight problems that contribute to increased response times and reduced throughput. They also help identify processing requirements changes during peak capacity periods.

Oracle collects Cache Fusion-related performance statistics from the buffer cache and **Global Cache Service (GCS)** layers. The statistics that Oracle collects describe general Real Application Clusters performance statistics as well as statistics for block requests and block mode conversion waits.

Oracle records object level statistics by default, that is, Oracle sets the `STATISTICS_LEVEL` parameter to `typical`. To also record timed statistics, set the `TIMED_STATISTICS` parameter to true. Oracle records statistics in hundredths of seconds.

In addition to performance trends revealed by object level and timed statistics, also examine statistics that reveal information about specific transactions within your Real Application Clusters environment. Do this with utilities such as Oracle9*i* Statspack as described in the next section.

Note: Set `TIMED_STATISTICS` to `false` if you are not actively collecting statistics to avoid the performance overhead associated with statistics collection.

See Also: *Oracle9i Performance Methods* for more information about using the DBA_OBJECT_STATS, V$OBJ_STAT, and V$OBJ_STAT_NAME views

Using Statspack and Statistics to Monitor Real Application Clusters Performance

The remainder of this chapter explains how to use Statspack in Real Application Clusters and how to analyze its statistics to assess performance trends as described under the following headings:

- Using Statspack in Real Application Clusters
- Monitoring Performance by Analyzing GCS and GES Statistics
- Analyzing Wait Events
- Using "CACHE_TRANSFER" Views to Analyze Real Application Clusters Statistics

Using Statspack in Real Application Clusters

You can use Statspack for Real Application Clusters just as you would for single-instance Oracle databases. Statspack displays statistics that show performance trends over a period of time. It uses the following pages to display Real Application Clusters performance information:

- Cluster Statistics Page
- GES Statistics Page

Monitor the following statistics on these pages:

- Global Cache Service (GCS) statistics that reveal workload characteristics
- Global Enqueue Service (GES) statistics
- GCS and GES Messaging statistics

Most statistics such as counts, elapsed times, and wait times, are computed by Statspack. These statistics characterize the application workloads and profiles. Oracle computes these statistics in the following ways:

- Per transaction, for example, global cache current blocks received per transaction
- As a rate, for example, global cache current blocks received per second

- As a ratio, for example, cache hit ratio, parsed CPU per elapsed CPU

The first page of Statspack displays most of the load and profile data and the average Global Cache Service times per request, such as the time required to receive a current block. Statspack also displays the top wait events and their proportion to the total wait time. These statistics indicate the overhead associated with specific requests.

> **See Also:** The `spdoc.txt` documentation file in the `ORACLE_HOME/rdbms/admin` directory for more information about Statspack

Monitoring Performance by Analyzing GCS and GES Statistics

To analyze your cluster database's performance and to identify any contention in Real Application Clusters, examine block transfer rates and the wait events and statistics as described under the following headings:

- Analyzing Block Transfers in Real Application Clusters
- Analyzing Performance Using GCS and GES Statistics

Analyzing Block Transfers in Real Application Clusters

Analyze block transfers in Real Application Clusters to determine the cost of global processing and to quantify the resources required to maintain interinstance coherency. Do this by analyzing the statistics as described in the following sections. Use these procedures on an on-going basis to identify processing trends and optimize performance. The following statistics reveal Real Application Clusters-related performance characteristics:

- Buffer cache-related statistics, such as consistent gets and db block gets
- Global Cache Service statistics such as global cache current block receive time, global cache cr block receive time, and global cache open and convert requests
- Global Cache Service wait events, such as global cache s to x and global cache null to s

Response times for cache-to-cache transfers are not bounded by I/O-related factors other than log writes. This is because Oracle writes data block modifications to the redo logs before blocks are transmitted to a remote instance. In other words, Real Application Clusters does not cause overhead in terms of disk I/O other than to perform physical I/O for cache replacement, checkpoints, and the reading of newly requested blocks. This behavior is identical to single-instance Oracle environments.

Analyzing Performance Using GCS and GES Statistics

This section describes how to monitor Global Cache and Global Enqueue Service performance by identifying any contention as explained under the following topics:

- Global Cache Service Wait Events
- Global Cache Service Timings
- Global Enqueue Service Statistics
- Undesirable Global Cache Statistics

Global Cache Service Wait Events

Identify globally hot resources in each cache by examining the values for the following statistics:

- global cache busy
- buffer busy global cache
- buffer busy global CR

Use the V$SESSION_WAIT view to identify objects that have performance issues. Columns P1 and P2 identify the file and block number of the object as in the following example queries:

```
SELECT P1 FILE_NUMBER, P2 BLOCK_NUMBER FROM V$SESSION_WAIT WHERE EVENT = 'BUFFER
BUSY GLOBAL CR';
```

The output from this first query may look like this:

```
FILE_NUMBER BLOCK_NUMBER
----------- ------------
         12         3841
```

If you then issue the query:

```
SELECT OWNER,SEGMENT_NAME,SEGMENT_TYPE FROM DBA_EXTENTS WHERE FILE_ID = 12 AND
3841 BETWEEN BLOCK_ID AND BLOCK_ID+BLOCKS-1;
```

Then the output would be similar to:

```
OWNER      SEGMENT_NAME                   SEGMENT_TYPE
---------- ------------------------------ ---------------
SCOTT      W_ID_I                         INDEX PARTITION
```

If there is contention, then index leaf blocks are usually the most contended blocks. To reduce contention on individual blocks, you can use a smaller block size for the index tablespace, or hash partitioning. These modifications distribute the load among more distinct blocks.

Global Cache Service Timings

Examine Global Cache Service timings to determine whether the interconnect has latency problems. Do this by examining the statistic for:

- global cache cr / current block receive time

Long latencies can be caused by:

- A large number of processes in the run queue waiting for CPU or scheduling delays.
- Platform-specific operating system parameter settings that affect IPC buffering or process scheduling.
- Slow, busy, or faulty interconnects. In these cases, look for dropped packets, retransmits, or cyclic redundancy check (CRC) errors. Ensure that the network is private and that interinstance traffic is not routed through a public network.

> **Note:** Cyclic redundancy check errors usually indicate firmware or hardware problems. These errors most commonly appear in system log files.

The procedures you use to evaluate whether these types of performance issues are occurring in your Real Application Clusters database are platform-specific. On Solaris operating systems, for example, execute the following commands:

```
netstat -l
netstat -s
sar -c
sar -q
vmstat
```

Global Enqueue Service Statistics

If the wait event 'enqueue' is among the leading wait events in terms of wait time, then analyze the output from the V$ENQUEUE_STATS view to identify the enqueue with the highest wait time.

- SQ Enqueue—This indicates that there is contention for sequences. In almost all cases you can increase the cache size of sequences used by the application by executing an `ALTER SEQUENCE` statement. Ensure that the sequence uses the `NOORDER` keyword to avoid additional SQ enqueue contention.
- TX Enqueue—This is usually an application-related issue pertaining to row locking and it can also be a problem in single-instance Oracle databases. However, Real Application Clusters processing can magnify the effect of TX enqueue waits. Performance bottlenecks on leaf blocks of right-growing indexes may also appear as TX enqueue waits while index block splits are in progress.

  You can resolve TX enqueue performance issues by setting the value of the `INITRANS` parameter to be equal to the number of CPUs per node multiplied by the number of nodes in the cluster multiplied by 0.75. However, Oracle Corporation recommends that you avoid setting this parameter to be greater than 100. Therefore, you should also not set `MAXTRANS` to be greater than 100. If the value for either of these parameters is too high, then Oracle uses more space for the transaction layer block header and less is used for the data layer variable header. This can result in more I/O.

Undesirable Global Cache Statistics

The following are undesirable statistics, or statistics for which the values should always be zero or near-zero.

- global cache blocks lost—This statistic reveals any block losses during transfers and high values may indicate network problems. When using an unreliable IPC protocol such as UDP, the value for 'global cache blocks lost' may be non-zero. If this is the case, then the ratio of 'global cache blocks lost' divided by 'global cache current blocks served' plus 'global cache cr blocks served' should be as small as possible. A non-zero value for 'global cache blocks lost' does not necessarily indicate a problem, because Oracle retries the block transfer operation until it is successful. Unlike TCP/IP, UDP/IP is considered unreliable because UDP/IP provides very few error recovery services.
- global cache blocks corrupt—This statistic indicates whether any blocks were corrupted during transfers. High values for this statistic indicate an IPC, network, or hardware problem.

Analyzing Wait Events

Most events that show a high total time as reported in the dynamic performance views or in Statspack are actually normal. If response times increase and Statspack shows a high proportion of wait time for cluster accesses, then determine the cause of these waits. Statspack provides a breakdown of the wait events with the five highest values sorted by percentages. Specific statistics events that you should monitor are:

- global cache open s and global cache open x
- global cache null to s and global cache null to x
- global cache cr request
- Global Cache Service Utilization for Logical Reads

global cache open s and global cache open x

These events are associated with the initial access of particular data blocks by an instance. The duration of the wait is short and the completion of the wait is most likely followed by a read from disk. This is because the blocks have not been cached in any instances in the cluster database.

If these events are associated with high totals or high per-transaction wait times, then it is likely that data blocks are not cached in the local instance and that the blocks cannot be obtained from another instance which results in a disk read. At the same time, you may also observe sub-optimal buffer cache hit ratios.

global cache null to s and global cache null to x

These events are waited for when a block was used by an instance, transferred to another instance, and then requested again by the original instance. Processes waiting for these events are usually waiting for a block to be transferred from the instance that last modified it. If one instance requests cached data blocks from other instances, then it is normal that these events consume a greater proportion of the total wait time.

global cache cr request

This event is waited for when an instance has requested a data block for a consistent read and the transferred block has not yet arrived at the requesting instance.

General Remarks About Global Cache Service Times If global cache waits constitute a large proportion of the wait time as listed on the first page of your Statspack report, and if response times or throughput does not conform to your service level

requirements, then check the Global Cache Service workload characteristics on the Cluster Statistics page of the Statspack report.

If the average Global Cache Service time per request is high, then it could be the result of contention, system loads, or network issues. System logs and operating system statistics may also indicate that a network link is congested, that packets are routed through the public network instead of the private interconnect, or that the sizes of the run queues are increasing.

In cases where CPU use is close to the maximum and processes are queuing for the CPU, raising the priorities of the Global Cache Service processes (LMS*n*) to have priority over other processes can significantly lower Global Cache Service times. Also consider reducing the number of processes on the database server, adding CPUs to the server, or adding nodes to the cluster database.

> **See Also:** *Oracle9i Real Application Clusters Administration* for procedures to add nodes and instances to your Real Application Clusters database

Global Cache Service Utilization for Logical Reads

To estimate the use of the Global Cache Service relative to the number of buffer cache reads, or logical reads, divide the sum of Global Cache Service requests by the number of logical reads for a given statistics collection interval.

Oracle makes a Global Cache Service request whenever a user accesses a buffer cache to read or modify a data block and the block is not in the local cache. This results in a remote cache read, a disk read, or a change of access privileges. In other words, logical reads are a superset of Global Cache Service operations. The calculation is:

$$\frac{\text{global cache gets} + \text{global cache converts} + \text{global cache cr blocks received} + \text{global cache current blocks received}}{\text{consistent gets} + \text{db block gets}}$$

Some blocks may be very hot, or in other words, frequently requested by local and remote users. Sometimes a block transfer is delayed for a few milliseconds to permit

local users to complete their work. The following ratio provides a rough estimate of how probable this is:

$$\frac{\text{global cache defers}}{\text{global cache current blocks served}}$$

A ratio of more than 0.3 indicates a fairly hot data set. In such a case, closely analyze the blocks involved in busy waits. To do this, query columns such as NAME, TYPE, FORCED_READS, and FORCED_WRITES from the V$CACHE_TRANSFER view, or examine the CR_TRANSFERS and CURRENT_TRANSFERS columns in V$OBJ_STATS. Also examine the values shown for the "global cache busy", "buffer busy global cache", and "buffer busy global CR" statistics as described under the heading "Global Cache Service Wait Events" on page 4-9.

If you discover a problem, then identify the object causing it, the instance that is accessing the object, and how the object is being accessed. If necessary, alleviate the contention by reducing:

- Hot spots by spreading the accesses to index blocks or data blocks, possibly by using Oracle hash or range partitions wherever applicable—just as you would in single-instance Oracle databases.
- Concurrency on the object by implementing load balancing or resource management. For example, decrease the rate of modifications to that object (use fewer database processes).

As mentioned earlier, in single-instance Oracle databases *and* in Real Application Clusters, blocks are only written to disk for aging, cache replacement, or checkpoints. When a data block is replaced from the cache due to aging or when a checkpoint occurs and the block was previously changed in another instance but not written to disk, Oracle sends a message to notify the other instance that Oracle will write the data block to disk. This is called a **fusion write**, and the following ratio reveals the proportion of writes that Oracle manages in this way:

$$\frac{\text{DBWR fusion writes}}{\text{physical writes}}$$

The larger this ratio is, the higher the number of written blocks that have been copied with previous changes in other instances. A large ratio can result from

insufficiently sized caches or it can be because checkpoints have not occurred for some time. A large number can also indicate the proportion of the global working set that is composed of buffers written due to cache replacement or checkpointing. For example, 0.1 means that 10% of the buffers written to disk were globally dirty.

Note that a fusion write is not an additional write to disk. However, a fusion write requires messaging to coordinate the transfer with the other instance. Therefore, a fusion write is a subset of all physical writes incurred by an instance.

Using "CACHE_TRANSFER" Views to Analyze Real Application Clusters Statistics

Use the V$CACHE_TRANSFER and V$FILE_CACHE_TRANSFER views to examine Real Application Clusters Statistics. The V$CACHE_TRANSFER view shows the types and classes of blocks that Oracle transfers over the **interconnect** on a per-object basis. Use the FORCED_READS and FORCED_WRITES columns to determine which types of objects your Real Application Clusters instances share. Values in the FORCED_WRITES column provide counts of how often a certain block type experiences a transfer out of a local buffer cache because the current version was requested by another instance.

Use the V$FILE_CACHE_TRANSFER view to identify files that experience cache transfers. For example, V$CACHE_TRANSFER has a NAME column showing the name of an object. Therefore, use this view to assess block transfers per object.

Even though the shared disk architecture eliminates forced disk writes, the V$CACHE_TRANSFER and V$FILE_CACHE_TRANSFER views may still show the number of block mode conversions per block class or object. However, values in the FORCED_WRITES column will be zero.

5

Monitoring Performance in Real Application Clusters with Performance Manager

This chapter presents the **Oracle Real Application Clusters**-specific **Oracle Performance Manager** charts. You must install and configure Oracle Performance Manager to display the charts described in this chapter. Therefore, use this chapter as a supplement to information contained in the *Getting Started with the Oracle Diagnostics Pack*. This chapter contains the following topics:

- Oracle Performance Manager Overview
- Displaying the Oracle Performance Manager Charts for Real Application Clusters

See Also:

- *Oracle9i Real Application Clusters Setup and Configuration* for additional information about installing Oracle Performance Manager
- *Oracle9i Database Performance Guide and Reference* for detailed information about the statistics that the Performance Manager charts display and how to interpret the statistics
- *Oracle9i Database Reference* for more information about fields in the Performance Manager charts and the V$ views from which they are derived

Oracle Performance Manager Overview

Oracle Performance Manager, which is part of the Diagnostics Pack, displays key performance statistics that you can monitor to optimize the performance of your Real Application Clusters database. Oracle Performance Manager presents performance data that the Intelligent Agent captures and computes. The Oracle Performance Manager charts display information from instance-specific V$ views and from global GV$ views.

Using Performance Manager Charts for Previous Oracle Cluster Software Releases

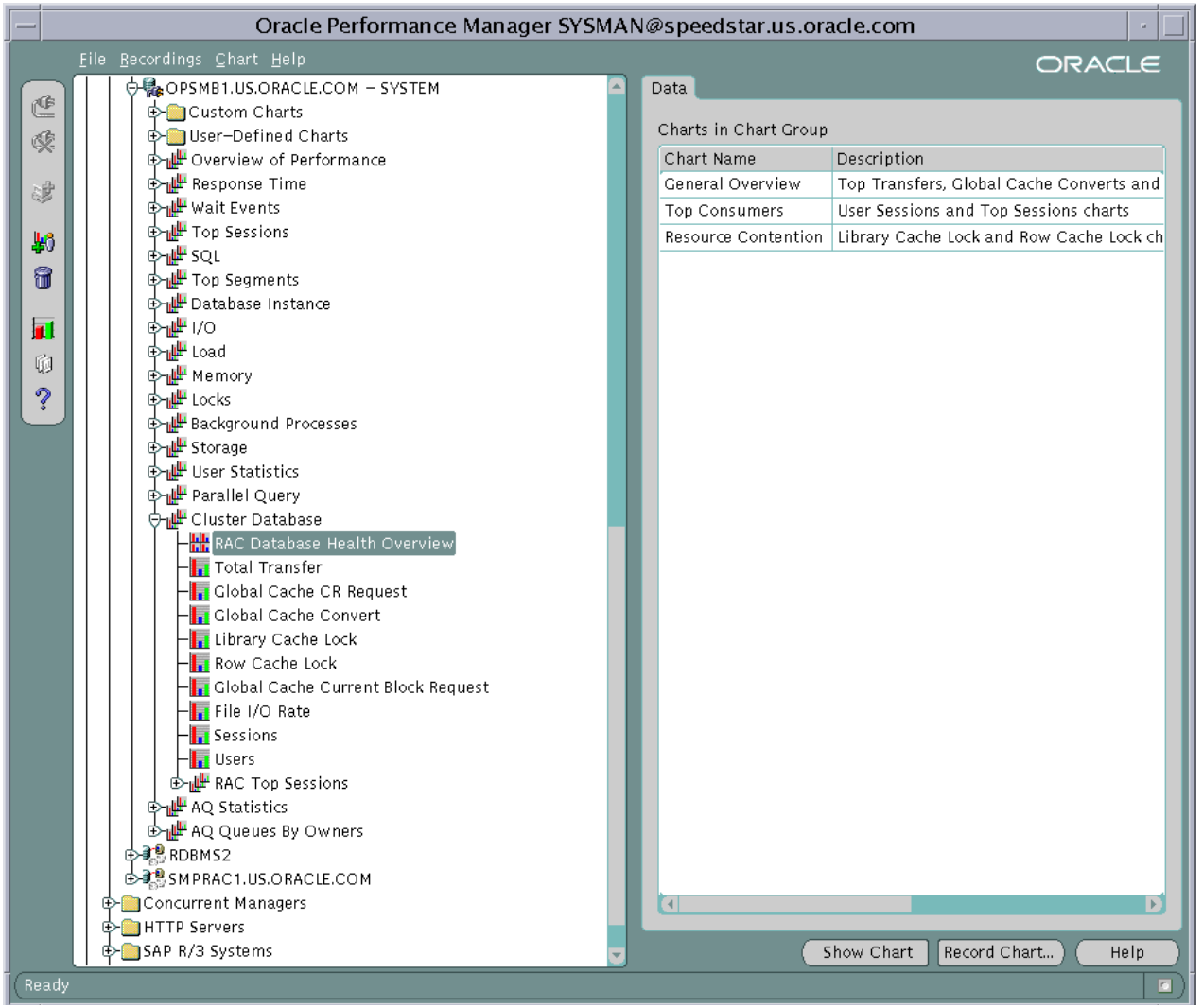
Use Oracle Performance Manager to examine the performance of current as well as previous versions of Oracle cluster databases. To use Performance Manager for pre-release 2 (9.2) versions, refer to the Oracle cluster software documentation for that version.

Note: Some charts that were documented in the Oracle Real Application Clusters release 1 (9.0.1) documentation are no longer available for that release.

Displaying the Oracle Performance Manager Charts for Real Application Clusters

Oracle Performance Manager has primary-level charts that you can access from icons that appear under the Cluster Database icon shown in Figure 5–1. Some of the primary charts also have lower-level charts that display performance information by instance, by file, and by object.

Figure 5–1 Oracle Performance Manager Real Application Clusters-Specific Charts



Real Application Clusters-Specific Performance Manager Chart Hierarchies

Table 5–1 shows the Performance Manager chart hierarchies. You can navigate from the primary charts in the left column to lower-level charts in the right column.

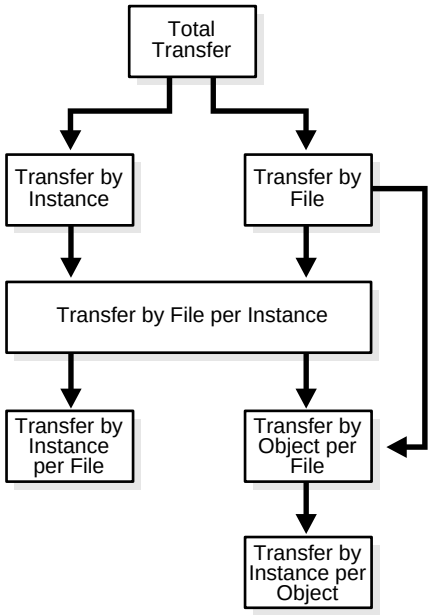
Table 5–1 Real Application Clusters-Specific Chart Hierarchies

Primary Chart	Lower-Level Charts
RAC Database Health Overview	-
Total Transfer Refer to Figure 5–2 for detail about the Total Transfer chart hierarchy.	Transfer By Instance Transfer By File From these two charts you can also access: Transfer By File Per Instance Transfer By Instance Per File Transfer By Object Per File Transfer By Instance Per Object
Global Cache CR Request	Global Cache CR Request By Instance
Global Cache Convert	Global Cache Convert By Instance
Library Cache Lock	Library Cache Lock By Instance
Row Cache Lock	Row Cache Lock By Instance
Global Cache Current Block Request	Global Cache Current Block Request By Instance
File I/O Rate	File I/O Rate By Object File I/O Rate By Instance
Sessions	
Users	Users By Instance
Real Application Clusters Top Sessions	-

See Also: Performance Manager online help text for more information about these charts

Figure 5–2 shows the chart hierarchy for the Total Transfer chart and its lower-level charts.

Figure 5–2 Real Application Clusters-Specific Enterprise Manager Chart Hierarchy



The primary- and lower-level charts are described under the following headings:

- Total Transfer Chart
- Global Cache CR Request Chart
- Global Cache Convert Chart
- Library Cache Lock Chart
- Row Cache Lock Chart
- Global Cache Current Block Request Chart
- File I/O Rate Chart
- Sessions Chart
- Users Chart

Real Application Clusters-Specific Charts in Performance Manager

The Performance Manager charts in this chapter typically display rates of activities as they occur during a time interval. This section describes the following Real Application Clusters-specific charts:

Total Transfer Chart

The Total Transfer chart displays the rates of the following activities for the entire cluster database:

- Logical Reads: number of times the consistent read and current blocks were requested (sum of values for 'consistent gets' and 'db block gets')
- Physical Reads: number of times the blocks were read from the disk, without bypassing the buffer cache (difference of values for 'physical reads' and 'physical reads direct')
- Cache Transfers: number of times consistent read and current blocks were read from the remote caches (sum of values for 'global cache cr blocks received' and 'global cache current blocks received')

To determine these rates for each instance, drill down to the Transfer By Instance chart.

You can use this chart to analyze performance in your Real Application Clusters database as follows:

- A low rate of physical reads combined with a low rate of cache-to-cache transfers indicates an ideal workload that results in scalable system if you need to add nodes.

- A high rate of physical reads combined with a high rate of cache-to-cache transfers indicates that a large working set of your application cannot be cached entirely in the cluster. Increasing the size of your buffer cache in each instance or adding another node can reduce this rate.
- If the rate of cache-to-cache transfers is a high percentage of your logical reads and the rate of physical reads is low, then drill down to the Transfer By File chart for further analysis.

Transfer By File Chart

The Transfer By File chart displays the rate of transfers for consistent read blocks as well as current blocks for each datafile of your cluster database. This statistic is derived from the sum of values for 'cr_transfers' and 'cur_transfers' from the GV$FILE_CACHE_TRANSFER view. Obtain the transfer rates on per-instance basis by drilling down to the Transfer By File Per Instance chart.

Use this chart to identify the file(s) with the highest transfer rates. After selecting a file you can drill down to the Transfer By Object Per File chart to identify the objects that are subject to maximum transfers. This chart also shows you how these objects are accessed from each node. This enables you to identify the hot objects such as tables and indexes. Then you can identify a corrective action such as the use of Oracle hash or range partitioning.

Global Cache CR Request Chart

The Global Cache CR Request chart displays the following statistic for the entire cluster database:

- The average global cache CR request time which is a ratio of the 'global cache cr block receive time' multiplied by 10, divided by the amount of 'global cache cr blocks received'

To determine the average CR request time for each instance, drill down to the Global Cache CR Request By Instance Chart.

If the chart displays a high value for CR request time, then investigate the possible causes such as a high system load, using a public interconnect instead of a private network, network errors, or poor CPU utilization by the LMS processes.

Global Cache Convert Chart

The Global Cache Convert chart displays the following statistics for the entire cluster database:

- The average global cache convert time which is a ratio of 'global cache convert time' multiplied by 10, divided by the 'global cache converts'
- The average global cache get time which is a ratio of the 'global cache get time' multiplied by 10 divided by the 'global cache gets'

To determine these statistics for each instance, drill down to the Global Cache Convert By Instance chart and consider the following points:

- A high average convert time can result from right-growing index trees. If this is true for your situation, then distribute access to the index leaf blocks by using hash or range partitioning on the index.
- High values for both convert and get times can be attributed to all the system, interconnect, network, LMS process, and CPU utilization-related problems discussed previously for the Global Cache CR Request By Instance Chart.

Library Cache Lock Chart

The Library Cache Lock chart displays the following statistics for the entire cluster database:

- The dlm_lock_requests which is the number of times a lock was requested for a database object
- The dlm_pin_requests which is the number of times a pin was requested for a database object
- The dlm_invalidations which is the number of invalidations received from other instances

To determine these statistics for each instance, drill down to the Library Cache Lock By Instance chart.

If you observe high number of lock and pin requests, then try to locate the sources for frequent PL/SQL unit execution, high amount of parsing, or the inefficient use of bind variables.

If you observe a high number of invalidations, then try to locate DDL statements or PL/SQL compile commands that result in schema objects being analyzed, dropped, or altered.

Row Cache Lock Chart

The Row Cache Lock chart displays the following statistics for the entire cluster database:

- The dlm_requests which is the number of times DLM lock requests are generated for consistent read blocks
- The dlm_conflicts which is the number of times when another instance already owned a lock on a consistent read block in a mode that conflicts with the mode requested by the instance
- The percentage of dlm_requests that result in dlm_conflicts

To determine these statistics for each instance, drill down to the Row Cache Lock By Instance chart. The statistics on this chart highlight any dictionary contention. If you observe a high percentage of DLM conflicts, then investigate the sources of dictionary contention, such as dictionary managed tablespaces, uncached sequences, or the frequent creation and dropping of database objects.

Global Cache Current Block Request Chart

The Global Cache Current Block Request chart displays the following statistics for the entire cluster database:

- The average global cache current block request time which is the ratio of 'global cache current block receive time' multiplied by 10, divided by 'global cache current blocks received'.
- The average global cache current block serve time which is the ratio of 'global cache current block serve time' multiplied by 10, divided by 'global cache current blocks served'.

To determine these statistics for each instance, drill down to the Global Cache Current Block Request By Instance chart. A high value for the serve time is generally attributed to poor performance of the LMS*n* processes. Just as for the average global cache CR request time, high values of average current block request and serve times can be caused by a high system load, the use of a public interconnect, network errors, the low priority of LMS*n* processes , or poor CPU utilization by the LMS*n* processes.

Real Application Clusters-Specific Versions of Oracle Charts

The remaining charts in this chapter are clusterized versions of single-instance Oracle database charts. The charts described in this section are:

- File I/O Rate Chart
- Sessions Chart
- Users Chart
- Real Application Clusters Top Sessions Chart

File I/O Rate Chart

The File I/O Rate chart displays the rate of physical reads and writes for all files in the cluster database. To determine the rate of reads and writes per datafile, drill down to the File I/O Rate By Object chart. To determine the rate of reads and writes per instance, drill down to the File I/O Rate By Instance chart.

Sessions Chart

The Sessions chart displays the sessions attached to the entire cluster database as well as related information, such as instance name, session ID, process ID, status, and user name.

Users Chart

The Users chart displays the following for the entire cluster database:

- The total number of user sessions logged on regardless of whether activity is generated
- The total number of active user sessions

To determine the number of logged-on and active users for each instance, drill down to the Users By Instance Chart.

Real Application Clusters Top Sessions Chart

The Real Application Clusters Top Sessions chart displays a set of sessions with the highest contribution to the cluster database instance activity based on selected sort statistics.

Real Application Clusters Database Health Overview Chart

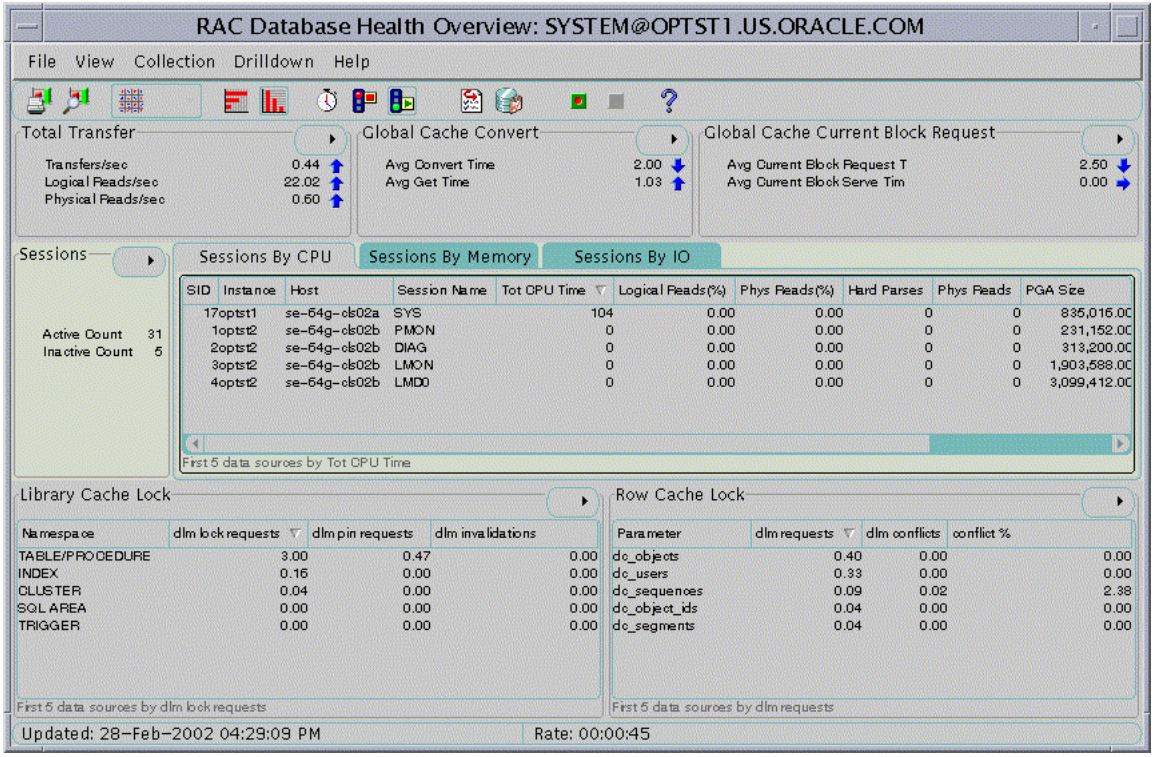
Figure 5–3 displays the Real Application Clusters Database Health Overview chart which is a group of standalone charts that shows key performance statistics for the entire cluster database. This chart consists of the following standalone charts:

- Total Transfer
- Global Cache Convert
- Global Cache Current Block Request
- Library Cache Lock
- Row Cache Lock

Drill downs from these charts are available in the same manner as the drill downs are available when these charts are used as standalone charts. In addition, the center of the overview chart contains a clusterized version of session information that is not available as standalone chart. You can customize the display of sessions information based on CPU, Memory, or I/O.

Note: You can also display this overview chart from the Enterprise Manager Console by selecting the menu item Performance View from the right-mouse menu of a cluster database.

Figure 5–3 Real Application Clusters Database Health Overview Chart



Part III

Real Application Clusters Reference

Part III contains **Oracle Real Application Clusters** reference information. The reference information in Part III includes:

- Appendix A, "Configuring Multi-Block Locks (Optional)"
- Appendix B, "Using Free Lists and Free List Groups in Real Application Clusters (Optional)"
- Glossary

A

Configuring Multi-Block Locks (Optional)

This appendix explains how to configure locks to manage access to 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)**. The topics in this appendix are:

- Before You Override the Global Cache and Global Enqueue Service Resource Control Mechanisms
- Deciding to Override Global Cache and Global Enqueue Service Processing
- Setting GC_FILES_TO_LOCKS
- Additional Considerations for Setting GC_FILES_TO_LOCKS
- Tuning Parallel Execution on Real Application Clusters
- Analyzing Real Application Clusters I/O Statistics
- Monitoring Multi-Block Lock Usage by Detecting False Forced Writes
- Lock Names and Lock Formats

Before You Override the Global Cache and Global Enqueue Service Resource Control Mechanisms

The default scheme provides exceptional performance for all system types in all Real Application Clusters environments. In addition, assigning locks requires additional administrative effort. Therefore, using the default scheme is preferable to performing the tasks required to override the default strategy as described in this appendix.

Note: Only use the information in this appendix for *exceptional* cases. An example of this is an application where the data access patterns are almost exclusively read-mostly.

Deciding to Override Global Cache and Global Enqueue Service Processing

Cache Fusion provides exceptional scalability and performance using cache-to-cache transfers of data that is not cached locally. In other words, before an **instance** reads a data block from disk, Oracle attempts to obtain the requested data from another instance's cache. If the requested block exists in another cache, then the data block is transferred across the **interconnect** from the holding instance to the requesting instance.

Real Application Clusters' resource control scheme guarantees the integrity of changes to data made by multiple instances. By default, each data block in an instance's buffer cache is protected by the Global Cache Service. The GCS tracks the access modes, roles, privileges, and states of these resources.

In rare situations, you may want to override the GCS, and the GES by configuring multi-block locks where one lock covers multiple data blocks in a file. If blocks are frequently accessed from the same instance, or if blocks are accessed from multiple **node**s but in compatible modes such as *shared* mode for concurrent reads, then a lock configuration may improve performance.

To do this, set the `GC_FILES_TO_LOCKS` parameter and specify the number of locks that Oracle uses for particular files. The syntax of the parameter also enables you to specify lock allocations for groups of files as well as the number of contiguous data blocks to be covered by each lock. If you indiscriminately use values for `GC_FILES_TO_LOCKS`, then adverse performance such as excessive **forced disk writes** can result. Therefore, only set `GC_FILES_TO_LOCKS` for:

- Read-only or read-mostly files and tablespaces
- Datafiles *except* those associated with the following: temporary tablespaces, tablespaces marked as `READ ONLY`, and tablespaces containing rollback segments.

> **Note:** You can set the `GC_FILES_TO_LOCKS` parameter for a particular file and thereby only disable Cache Fusion processing for that particular file. Therefore, you can use multi-block lock assignments and default GCS and GES (Cache Fusion) processing in the same Real Application Clusters environment.

When to Use Locks

Using multiple locks for each file can be useful for the types of data shown in Table A–1.

Table A–1 When to Use Locks

Situation	Reason
When the composition of the data is mostly read-only.	A few locks can cover many blocks without requiring frequent lock operations. These locks are released only when another instance needs to modify the data. Assigning locks can result in better performance on read-only data with **parallel execution** processing. If the data is strictly read-only, then consider designating the tablespace as read-only.
When a large amount of data is modified by a relatively small set of instances.	Lock assignments permit access to an un-cached database block to proceed without Parallel Cache Management activity. However, this is only possible if the block is already in the requesting instance's cache.

Using locking can cause additional cross-instance cache management activity because conflicts can occur between instances that modify different database blocks. Resolution of false forced disk writes or excessive forced disk writes can require writing several blocks from the cache of the instance that currently owns access to the blocks.

Setting GC_FILES_TO_LOCKS

Set the GC_FILES_TO_LOCKS initialization parameter to specify the number of locks covering data blocks in a datafile or set of datafiles. This section covers:

- GC_FILES_TO_LOCKS Syntax
- Lock Assignment Examples

GC_FILES_TO_LOCKS Syntax

The syntax for the GC_FILES_TO_LOCKS parameter enables you to specify the relationship between locks and files. The syntax for this parameter and the meanings of the variables as shown in Table A–2 are:

GC_FILES_TO_LOCKS="{*file_list*=*#locks*[!*blocks*] [EACH][:]} . . ."

Table A–2 GC_FILES_TO_LOCKS Variables and their Meanings

Variable	Meaning
file_list	*file_list* specifies a single file, range of files, or list of files and ranges as follows: *fileidA*[-*fileidC*][,*fileidE*[-*fileidG*]] ... Query the data dictionary view DBA_DATA_FILES to find the correspondence between file names and file ID numbers.
#locks	Sets the number of locks to assign to file_list.
!*blocks*	Specifies the number of contiguous data blocks to be covered by each lock; also called the *blocking factor*
EACH	Specifies *#locks* as the number of locks to be allocated to *each* file in file_list.

Note: All instance must have identical values for GC_FILE_TO_LOCKS. Also, do not use spaces within the quotation marks of the GC_FILES_TO_LOCKS parameter syntax.

The default value for *!blocks* is 1. When you specify *blocks*, contiguous data blocks are covered by each of the *#lock* locks. To specify a value for *blocks*, use the exclamation point (!) separator. You would primarily specify *blocks*, and not specify the EACH keyword to allocate sets of locks to cover multiple datafiles.

Always set the *!blocks* value to avoid interfering with data partitioning if you have also used the optional free list groups. Normally you do not need to preallocate extents. When a row is inserted into a table and Oracle allocates new extents, Oracle allocates contiguous blocks that are specified with *!blocks* in GC_FILES_TO_LOCKS to the free list group associated with an instance.

Lock Assignment Examples

For example, you can assign 300 locks to file 1 and 100 locks to file 2 by adding the following line to your **initialization parameter file**:

```
GC_FILES_TO_LOCKS = "1=300:2=100"
```

The following entry specifies a total of 1500 locks: 500 each for files 1, 2, and 3:

```
GC_FILES_TO_LOCKS = "1-3=500EACH"
```

By contrast, the following entry specifies a total of only 500 locks spread across the three files:

```
GC_FILES_TO_LOCKS = "1-3=500"
```

The following entry indicates that Oracle should use 1000 distinct locks to protect file 1. The data in the files is protected in groups of 25 contiguous locks.

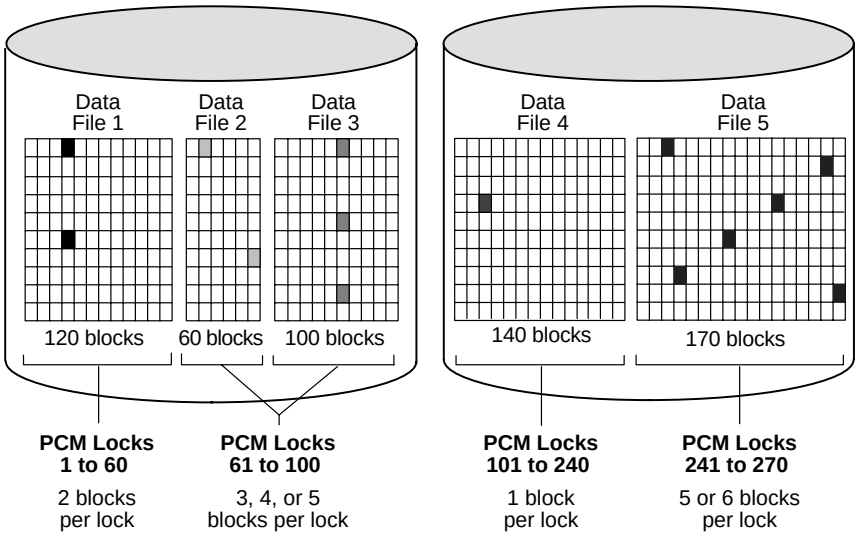
```
GC_FILES_TO_LOCKS = "1=1000!25"
```

If you define a datafile with the AUTOEXTEND clause or if you issue the ALTER DATABASE ... DATAFILE ... RESIZE statement, then you may also need to adjust the lock assignment.

When you add new datafiles, decide whether these new files should be subject to the default control of the GCS or whether you want to assign locks using the GC_FILES_TO_LOCKS initialization parameter.

The following examples show different methods of mapping blocks to locks and how the same locks are used on multiple datafiles.

Figure A–1 Mapping Locks to Data Blocks



Example 1 Figure A–1 shows an example of mapping blocks to locks for the parameter value `GC_FILES_TO_LOCKS = "1=60:2-3=40:4=140:5=30"`.

In datafile 1 shown in Figure A–1, 60 locks map to 120 blocks, which is a multiple of 60. Each lock covers two data blocks.

In datafiles 2 and 3, 40 locks map to a total of 160 blocks. A lock can cover either one or two data blocks in datafile 2, and two or three data blocks in datafile 3. Thus, one lock can cover three, four, or five data blocks across both datafiles.

In datafile 4, each lock maps exactly to a single data block, since there is the same number of locks as data blocks.

In datafile 5, 30 locks map to 170 blocks, which is not a multiple of 30. Each lock therefore covers five or six data blocks.

Each lock illustrated in Figure A–1 can be held in either shared read mode or read-exclusive mode.

Example 2 The following parameter setting allocates 500 locks to datafile 1; 400 locks each to files 2, 3, 4, 10, 11, and 12; 150 locks to file 5; 250 locks to file 6; and 300 locks collectively to files 7 through 9:

```
GC_FILES_TO_LOCKS = "1=500:2-4,10-12=400EACH:5=150:6=250:7-9=300"
```

This example assigns a total of (500 + (6*400) + 150 + 250 + 300) = 3600 locks. You can specify more than this number of locks if you add more datafiles.

Example 3 In Example 2, 300 locks are allocated to datafiles 7, 8, and 9 collectively with the clause "7-9=300". The keyword EACH is omitted. If each of these datafiles contains 900 data blocks, then for a total of 2700 data blocks, then each lock covers nine data blocks. Because the datafiles are multiples of 300, the nine locks cover three data blocks in each datafile.

Example 4 The following parameter value allocates 200 locks each to files 1 through 3; 50 locks to datafile 4; 100 locks collectively to datafiles 5, 6, 7, and 9; and 20 locks in contiguous 50-block groups to datafiles 8 and 10 combined:

```
GC_FILES_TO_LOCKS = "1-3=200EACH 4=50:5-7,9=100:8,10=20!50"
```

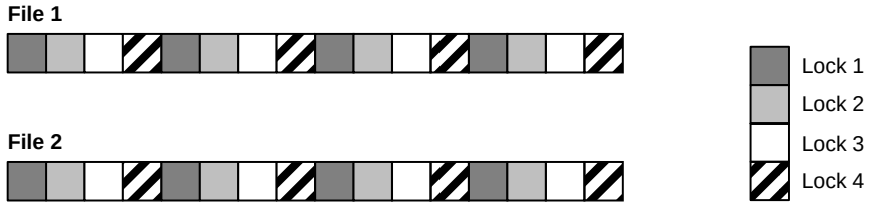
In this example, a lock assigned to the combined datafiles 5, 6, 7, and 9 covers one or more data blocks in each datafile, unless a datafile contains fewer than 100 data blocks. If datafiles 5 to 7 contain 500 data blocks each and datafile 9 contains 100 data blocks, then each lock covers 16 data blocks: one in datafile 9 and five each in the other datafiles. Alternatively, if datafile 9 contained 50 data blocks, half of the locks would cover 16 data blocks (one in datafile 9); the other half of the locks would only cover 15 data blocks (none in datafile 9).

The 20 locks assigned collectively to datafiles 8 and 10 cover contiguous groups of 50 data blocks. If the datafiles contain multiples of 50 data blocks and the total number of data blocks is not greater than 20 times 50, that is, 1000, then each lock covers data blocks in either datafile 8 or datafile 10, but not in both. This is because each of these locks covers 50 contiguous data blocks. If the size of datafile 8 is not a multiple of 50 data blocks, then one lock must cover data blocks in both files. If the sizes of datafiles 8 and 10 exceed 1000 data blocks, then some locks must cover more than one group of 50 data blocks, and the groups might be in different files.

Example 5 GC_FILES_TO_LOCKS="1-2=4"

In this example, four locks are specified for files 1 and 2. Therefore, the number of blocks covered by each lock is eight ((16+16) / 4). The blocks are not contiguous.

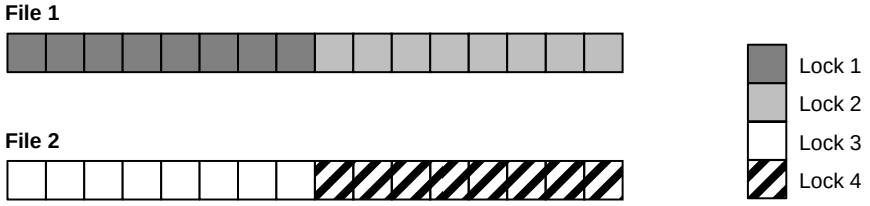
Figure A–2 GC_FILES_TO_LOCKS Example 5



Example 6 GC_FILES_TO_LOCKS="1-2=4!8"

In this example, four locks are specified for files 1 and 2. However, the locks must cover eight contiguous blocks.

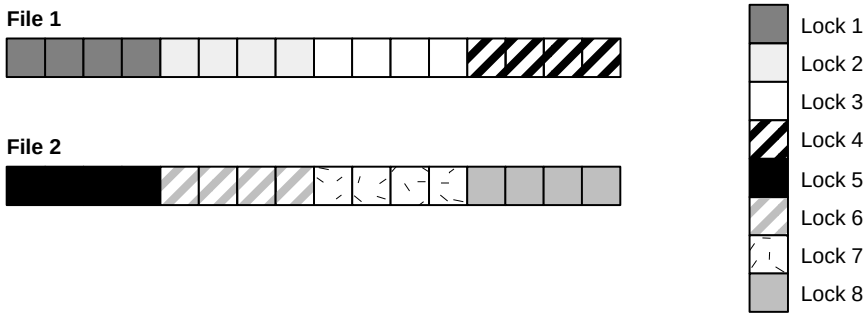
Figure A–3 GC_FILES_TO_LOCKS Example 6



Example 7 GC_FILES_TO_LOCKS="1-2=4!4EACH"

In this example, four locks are specified for file 1 and four for file 2. The locks must cover four contiguous blocks.

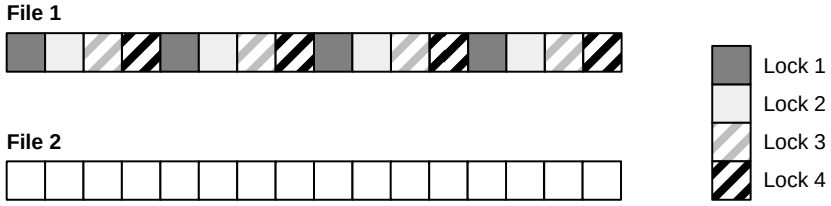
Figure A–4 GC_FILES_TO_LOCKS Example 7



Example 8 GC_FILES_TO_LOCKS="1=4:2=0"

In this example, file 1 has multi-block lock control with 4 locks. On file 2, locks are allocated.

Figure A–5 GC_FILES_TO_LOCKS Example 8



Additional Considerations for Setting GC_FILES_TO_LOCKS

Setting GC_FILES_TO_LOCKS in Real Application Clusters has further implications. For example, setting it can increase monitoring overhead and you may have to frequently adjust the parameter when the database grows or when you add files. Moreover, you cannot dynamically change the setting for GC_FILES_TO_LOCKS. To change the setting, you must stop the instances, alter the setting, and restart all the instances. In addition, consider the following topics in this section:

- Expanding or Adding Datafiles
- Files To Avoid Including in GC_FILES_TO_LOCKS Settings

Expanding or Adding Datafiles

Sites that run continuously cannot afford to shut down for parameter value adjustments. Therefore, when you use the GC_FILES_TO_LOCKS parameter, remember to provide room for growth or room for files to extend.

You must also carefully consider how you use locks on files that do not grow significantly, such as read-only or read-mostly files. It is possible that better performance would result from assigning fewer locks for multiple blocks. However, if the expected CPU and memory savings due to fewer locks do not outweigh the administrative overhead, use the resource control scheme of the Global Cache and Global Enqueue Services.

Files To Avoid Including in GC_FILES_TO_LOCKS Settings

Never include the following types of files in the GC_FILES_TO_LOCKS parameter list:

- Files that contain rollback segments.
- Files that are part of a TEMPORARY tablespace.
- Files with read-only data within a tablespace that is explicitly set to READ ONLY; the exception to this is a single lock that you can assign to ensure the tablespace does not have to contend for spare locks—but setting this lock is not mandatory—you can still leave this tablespace unassigned.

Tuning Parallel Execution on Real Application Clusters

To optimize parallel execution in Real Application Clusters environments when not using the default resource control scheme, you must accurately set the GC_FILES_TO_LOCKS parameter. Data block address locking in its default behavior assigns

one lock to each block. For example, during a full table scan, a lock must be acquired for each block read into the scan. To accelerate full table scans, you use one of the following three possibilities:

- For datafiles containing truly read-only data, set the tablespace to read only. Then lock operations do not occur.
- Alternatively, for data that is mostly read-only, assign very few hashed locks (for example, 2 shared locks) to each datafile. Then these are the only locks you have to acquire when you read the data.
- If you want data block address or fine-grain locking, group the blocks controlled by each lock, using the ! option. This has advantages over default data block address locking because with the default, you would need to acquire one million locks in order to read one million blocks. When you group the blocks, you reduce the number of locks allocated by the grouping factor. Thus a grouping of !10 would mean that you would only have to acquire one tenth as many locks as with the default. Performance improves due to the dramatically reduced amount of lock allocation. As a rule of thumb, performance with a grouping of !10 is comparable to the speed of hashed locking.

The following guidelines affect memory usage, and thus indirectly affect performance:

- Never allocate locks for datafiles of temporary tablespaces.
- Allocate specific locks for the SYSTEM tablespace. This practice ensures that data dictionary activity such as space management never interferes with the data tablespaces at a cache management level (error 1575).

Analyzing Real Application Clusters I/O Statistics

If you set GC_FILES_TO_LOCKS, then Cache Fusion is disabled. In this case, you can use three statistics in the V$SYSSTAT view to measure the I/O performance related to global cache synchronization:

- DBWR cross-instance writes
- Remote instance undo header writes
- Remote instance undo block writes

DBWR cross-instance writes occur when Oracle resolves inter-instance data block usage by writing the requested block to disk before the requesting node can use it.

Cache Fusion eliminates the disk I/O for current and consistent-read versions of blocks. This can lead to a substantial reduction in **physical writes** and reads performed by each instance.

Analyzing Real Application Clusters I/O Statistics Using V$SYSSTAT

You can obtain the following statistics to quantify the write I/Os required for global cache synchronization.

1. Use this syntax to query the V$SYSSTAT view:

```
SELECT NAME, VALUE FROM V$SYSSTAT
WHERE NAME IN ('DBWR cross-instance writes',
'remote instance undo block writes',
'remote instance undo header writes',
'physical writes');
```

Oracle responds with output similar to:

```
NAME                                                          VALUE
------------------------------------------------------------ ----------
physical writes                                              41802
DBWR cross-instance writes                                    5403
remote instance undo block writes                                0
remote instance undo header writes                               2
4 rows selected.
```

Where the statistic *physical writes* refers to all physical writes that occurred from a particular instance performed by DBWR, the value for **DBWR cross-instance writes** accounts for all writes caused by writing a dirty buffer containing a data block that is requested for modification by another instance. Because the DBWR

process also handles cross-instance writes, *DBWR cross-instance writes* are a subset of all *physical writes*.

2. Calculate the ratio of Real Application Clusters-related I/O to overall physical I/O using this equation:

$$\frac{\textit{DBWR cross-instance writes}}{\textit{physical writes}}$$

3. Use this equation to calculate how many writes to rollback segments occur when a remote instance needs to read from rollback segments that are in use by a local instance:

$$\frac{\textit{(remote instance undo header writes+ remote instance undo block writes)}}{\textit{DBWR cross-instance writes}}$$

The ratio shows how much disk I/O is related to writes to rollback segments.

4. To estimate the number or percentage of reads due to global cache synchronization, use the number of lock requests for conversions from NULL(N) to Shared mode (S) counted in V$LOCK_ACTIVITY and the **physical reads** statistics from V$SYSSTAT.

The following formula computes the percentage of reads that are only for local work where **lock buffers for read** represents the N-to-S block access mode conversions:

$$\frac{\textit{(physical reads - (lock buffers for read))} * 100}{\textit{physical reads}}$$

These so-called *forced reads* occur when a cached data block that was previously modified by the local instance had to be written to disk. This is due to a request from another instance, so the block is then re-acquired by the local instance for a read.

Monitoring Multi-Block Lock Usage by Detecting False Forced Writes

False forced writes occur when Oracle down-converts a lock that protects two or more blocks if the blocks are concurrently updated from different nodes. Assume that each node is updating a different block covered by the same lock. In this case, each node must write both blocks to disk even though the node is updating only one of them. This is necessary because the same lock covers both blocks.

Statistics are not available to show false forced write activity. To assess false forced write activity you can only consider circumstantial evidence as described in this section.

The following SQL statement shows the number of lock operations causing a write, and the number of blocks actually written:

```
SELECT VALUE/(A.COUNTER + B.COUNTER + C.COUNTER) "PING RATE"
  FROM V$SYSSTAT,
    V$LOCK_ACTIVITY A,
    V$LOCK_ACTIVITY B,
    V$LOCK_ACTIVITY C
WHERE A.FROM_VAL = 'X'
    AND A.TO_VAL = 'NULL'
    AND B.FROM_VAL = 'X'
    AND B.TO_VAL = 'S'
    AND C.FROM_VAL = 'X'
    AND C.TO_VAL = 'SSX'
    AND NAME = 'DBWR cross-instance writes';
```

Table A–3 shows how to interpret the forced disk write rate.

Table A–3 Interpreting the Forced Write Rate

Forced Disk Write Rate	Meaning
Less than 1	False forced writes may be occurring, but there are more lock operations than forced disk writes. DBWR is writing blocks fast enough, resulting in no writes for lock activity. This is also known as a *soft ping*, meaning I/O activity is not required for the forced disk write, only lock activity.
Equal to 1	Each lock activity involving a potential write causes the write to occur. False forced writes may be occurring.
Greater than 1	False forced writes are definitely occurring.

Use this formula to calculate the percentage of false forced writes:

$$\frac{(ping\_rate - 1)}{ping\_rate} * 100$$

Then check the total number of writes and calculate the number due to false forced writes:

```
SELECT Y.VALUE "ALL WRITES",
    Z.VALUE "PING WRITES",
    Z.VALUE * pingrate "FALSE PINGS",
FROM V$SYSSTAT Z,
    V$SYSSTAT Y,
WHERE Z.NAME = 'DBWR cross-instance writes'
AND Y.NAME = 'physical writes';
```

Here, *ping_rate* is given by the following SQL statement:

```
CREATE OR REPLACE VIEW PING_RATE AS
SELECT ((VALUE/(A.COUNTER+B.COUNTER+C.COUNTER))-1)/
    (VALUE/(A.COUNTER+B.COUNTER+C.COUNTER)) RATE
FROM V$SYSSTAT,
    V$LOCK_ACTIVITY A,
    V$LOCK_ACTIVITY B,
    V$LOCK_ACTIVITY C
WHERE A.FROM_VAL = 'X'
    AND A.TO_VAL   = 'NULL'
    AND B.FROM_VAL = 'X'
    AND B.TO_VAL   = 'S'
    AND C.FROM_VAL = 'X'
AND C.TO_VAL   = 'SSX'
AND NAME = 'DBWR cross-instance writes';
```

The goal is not only to reduce overall forced disk writes, but also to reduce false forced writes. To do this, look at the distribution of instance locks in `GC_FILES_TO_LOCKS` and check the data in the files.

Lock Names and Lock Formats

The following section describes the lock names and lock formats of locks. The topics in this section are:

- Lock Names and Lock Name Formats
- Lock Names
- Lock Types and Names

Lock Names and Lock Name Formats

Internally, Oracle global lock name formats use one of the following formats with parameter descriptions as shown in Table A–4:

- `type ID1 ID2`
- `type, ID1, ID2`
- `type (ID1, ID2)`

Table A–4 Lock Name Format Variable Descriptions

Variable	Meaning
`type`	A two-character type name for the lock type, for example, BL, TX, TM
`ID1`	The first lock identifier. The meaning and format of this identifier differs from one lock type to another.
`ID2`	The second lock identifier. The meaning and format of this identifier differs from one lock type to another.

For example, a space management lock might be named ST00. A lock might be named BL 1 900.

The clients of the lock manager define the lock type, for example *BL* for a lock, and two parameters, id1 and id2, and pass these parameters to the GCS API to open a lock. The lock manager does not distinguish between different types of locks. Each component of Oracle defines the type and the two parameters for its own needs, in other words, id1 and id2 have a meaning consistent with the requirements of each component.

Lock Names

All locks are Buffer Cache Management locks. Buffer Cache Management locks are of type *BL*. The syntax of lock names is `type ID1 ID2`, where:

- `type`—Is always BL because locks are buffer locks.
- `ID1`—The database address of the blocks.
- `ID2`—The block class.

Examples of lock names are:

- `BL (100, 1)`—This is a data block with lock element 100.
- `BL (1000, 4)`—This is a segment header block with lock element 1000.
- `BL (27, 1)`—This is an undo segment header with rollback segment Number 10. The formula for the rollback segment is 7 + (10 * 2).

Lock Types and Names

There are several different types and names of locks as shown in Table A–5:

Table A–5 Locks Types and Names

Type	Lock Name	Type	Lock Name
CF	Controlfile Transaction	PS	Parallel Execution Process Synchronization
CI	Cross-Instance Call Invocation	RT	Redo Thread
DF	Datafile	SC	System Change Number
DL	Direct Loader Index Creation	SM	SMON
DM	Database Mount	SN	Sequence Number
DX	Distributed Recovery	SQ	Sequence Number Enqueue
FS	File Set	SV	Sequence Number Value
KK	Redo Log *Kick*	ST	Space Management Transaction
IN	Instance Number	TA	Transaction Recovery
IR	Instance Recovery	TM	DML Enqueue
IS	Instance State	TS	Temporary Segment (also Table-Space)
MM	Mount Definition	TT	Temporary Table
MR	Media Recovery	TX	Transaction

Table A–5 (Cont.) Locks Types and Names

Type	Lock Name	Type	Lock Name
IV	Library Cache Invalidation	UL	User-Defined Locks
L[A-P]	Library Cache Lock	UN	User Name
N[A-Z]	Library Cache Pin	WL	Begin written Redo Log
Q[A-Z]	Row Cache	XA	Instance Registration Attribute Lock
PF	Password File	XI	Instance Registration Lock
PR	Process Startup		

B

Using Free Lists and Free List Groups in Real Application Clusters (Optional)

If you cannot use automatic segment-space management as Oracle Corporation recommends, then refer to the procedures in this appendix that describe how to use free lists and free list groups in **Oracle Real Application Clusters** environments. The sections in this appendix include:

- Using Free List Groups For Concurrent Inserts from Multiple Nodes
- Extent Management

Using Free List Groups For Concurrent Inserts from Multiple Nodes

If you cannot use locally managed tablespaces and automatic segment-space management, then consider managing free space manually by using free lists and free list groups. However, Oracle Corporation strongly recommends that you use automatic segment-space management.

The Purpose of Free Lists and Free List Groups

Without automatic segment-space management, when data is frequently inserted into a table from multiple nodes and the table is not partitioned, you can use free list groups to avoid performance issues. In such situations, performance issues can be due to concurrent access to data blocks, table segment headers, and other global resource demands.

Free list groups separate the data structures associated with the free space management of a table into disjoint sets that are available for individual instances. With free list groups, the performance issues among processes working on different instances is reduced because data blocks with sufficient free space for inserts are managed separately for each instance.

> **See Also:** *Oracle9i Real Application Clusters Concepts* for a conceptual overview of free list groups

Deciding Whether to Create Database Objects with Free List Groups

Free lists and free list groups are usually needed when random inserts to a table from multiple **instances** occur frequently. Processes looking for space in data blocks can contend for the same blocks and table headers. The degree of concurrency and the overhead of shipping data and header blocks from one instance to another can adversely affect performance. In these cases, use free list groups.

Identifying Critical Tables

You can identify tables that are subject to high insert rates by querying the V$SQL view and searching for INSERT commands as shown in the following example:

```
SELECT SUBSTR(SQL_TEXT,80), DECODE(COMMAND_TYPE,2,'INSERT'),EXECUTIONS
FROM V$SQL
WHERE COMMAND_TYPE = 2
ORDER BY EXECUTIONS;
```

Search for the table name in the string for the statements with the highest number of executions. These statements and the indexes that are built on them are candidates for free list groups.

Determining FREELIST GROUPS Reorganization Needs

You can monitor free list group performance by examining the rate of cache transfers and **forced disk writes** by using the V$CLASS_CACHE_TRANSFER view. V$CLASS_CACHE_TRANSFER view contains information about the number of cache transfers that occurred since instance startup for each class of block. If your output from the following select statement example shows a relatively high amount for segment header and free list forced disk writes, for example, more than 5% of the total, then consider changing the FREELIST GROUPS parameter for some tables to improve performance.

```
SELECT CLASS, (X_2_NULL_FORCED_STALE + X_2_S_FORCED_STALE) CACHE_TRANSFER
FROM V$CLASS_CACHE_TRANSFER;
```

Because the V$CLASS_CACHE_TRANSFER view does not identify cache transfers by object name, use other views to identify the objects that significantly contribute to the number of cache transfers. For example, the V$CACHE_TRANSFER view has information about each block in the buffer cache that is transferred. Block class 4 identifies segment headers and block class 6 identifies free list blocks. The output from the following select statement can show objects that could benefit from increased free list groups values:

```
SELECT NAME, CLASS#, SUM(XNC) CACHE_TRANSFER
FROM V$CACHE_TRANSFER
WHERE CLASS# IN (4,6)
GROUP BY NAME, CLASS#
ORDER BY CACHE_TRANSFER DESC;
```

Note: If you did not create your database with the Database Configuration Assistant, then certain Real Application Clusters-specific views such as V$CLASS_CACHE_TRANSFER are only available after you execute the CATCLUST.SQL script.

Creating Tables, Clusters, and Indexes with FREELISTS and FREELIST GROUPS

Create free lists and free list groups by specifying the FREELISTS and FREELIST GROUPS storage parameters in CREATE TABLE, CREATE CLUSTER or CREATE INDEX statements. The database can be opened in either exclusive or shared mode.

If you need to use free list groups, then the general rule is to create at least one free list group for each Real Application Clusters instance.

Note: You *cannot* change the value of FREELIST GROUPS with the ALTER TABLE, ALTER CLUSTER, or ALTER INDEX statements unless you export, drop, rebuild, and reload the table. However, you can dynamically change the setting for the FREELISTS parameter with the ALTER TABLE, ALTER INDEX, or ALTER CLUSTER statements.

FREELISTS Parameter

The FREELISTS parameter specifies the number of free lists in each free list group. The default and minimum value of FREELISTS is 1. The maximum value depends on the data block size. If you specify a value that is too large, then an error message informs you of the maximum value. The optimal value for FREELISTS depends on the expected number of concurrent inserts for each free list group for a particular table.

Note: Oracle ignores a setting for FREELISTS if the tablespace in which the object resides is in automatic segment-space management mode.

FREELIST GROUPS Parameter

Each free list group is associated with one or more instances at startup. The default value of FREELIST GROUPS is 1. This means that all existing free lists of a segment are available to all instances. As mentioned, you would typically set FREELIST GROUPS equal to the number of instances in your Real Application Clusters environment.

Free list group blocks with enough free space for inserts and updates are effectively disjoint once Oracle allocates them to a particular free list group. However, once data blocks that are allocated to one instance are freed by another instance, they are no longer available to the original instance. This might render some space unusable and possibly create a skew.

Note: With multiple free list groups, the free list structure is detached from the segment header and located in the free list block, which is a separate block. This reduces for the segment header performance issues and provides separate free block lists for instances.

Example The following statement creates a table named `department` that has seven free list groups, each of which contains four free lists:

```
CREATE TABLE department
        (deptno   NUMBER(2),
         dname    VARCHAR2(14),
         loc      VARCHAR2(13) )
        STORAGE ( INITIAL 100K        NEXT 50K
                  MAXEXTENTS 10       PCTINCREASE 5
                  FREELIST GROUPS 7   FREELISTS 4 );
```

Creating FREELISTS and FREELIST GROUPS for Clustered Tables

Use clustered tables to store records from different tables if the records are frequently accessed as a group by one or more `SELECT` statements. Using clustered tables can thus improve performance by reducing the overhead for processing reads. However, clustered tables may be less useful for DML statements.

You cannot specify `FREELISTS` and `FREELIST GROUPS` storage parameters in the `CREATE TABLE` statement for a clustered table. Instead, you must specify free list parameters for the *entire* cluster rather than for individual tables. This is because clustered tables use the storage parameters of the `CREATE CLUSTER` statement.

Without automatic segment-space management, Real Application Clusters enables instances to use multiple free lists and free list groups. Some hash clusters can also use multiple free lists and free list groups if you created them with a user-defined key for the hashing function and the key is partitioned by instance.

Note: Using the `TRUNCATE TABLE` *table_name* `REUSE STORAGE` syntax removes extent mappings for free list groups and resets the high water mark to the beginning of the first extent.

Creating FREELISTS for Indexes

You can also use the FREELISTS and FREELIST GROUPS parameters in the CREATE INDEX statement. However, you should be aware that inserting into an index differs from inserting into a table because the block Oracle uses is determined by the index key value.

For example, assume you have a table with multiple free list groups that also has an index with multiple free list groups. If two sessions connect to different instances and insert rows into that table, then Oracle uses different blocks to store the table data. This minimizes cache block transfers for the affected data segment. However, index segment cache block transfers can still occur if these sessions insert similar index key values. Therefore, you can only anticipate a slight reduction in cache transfers for the index segment header because Oracle must use more index blocks to store the index free lists.

> **See Also:** *Oracle9i SQL Reference* for more information on the SQL mentioned in this section

Associating Instances and User Sessions with Free List Groups

When Oracle creates an object with multiple free list groups, the number of a free list group block becomes part of the object's data dictionary definition. This is important because instances and users need to be associated with a free list group block. You can establish this association statically by assigning a fixed **instance number** to an instance using an initialization parameter, or by specifying the instance number in DDL statements.

Associating Instances with Free List Groups

You can associate instances with free list groups as follows:

- INSTANCE_NUMBER parameter—You can use various SQL clauses with the INSTANCE_NUMBER initialization parameter to associate extents of data blocks with instances.
- SET INSTANCE clause—You can use the SET INSTANCE clause of the ALTER SESSION statement to ensure a session uses the free list group associated with a particular instance regardless of the instance to which the session is connected. For example:

  ```
  ALTER SESSION SET INSTANCE = inst_no
  ```

The SET INSTANCE clause is useful when an instance fails and users re-connect to other instances. For example, consider a database where space is preallocated to the

free list groups in a table. If an instance fails and all the users are failed over to other instances, then their session can be set to use the free list group associated with the failed instance.

If you omit the SET INSTANCE clause, then the failed over sessions would begin inserting data into blocks and extents would be allocated to the instance that they failed over to. Later, when the failed instance is restored and the users connect to it again, the data they inserted would be part of a set of blocks associated with the other instance's free list group. Thus, interinstance communication could increase.

Extent Management

This section discuses the following topics:

- Preallocating Extents
- Extent Management and Locally Managed Tablespaces

Preallocating Extents

Before Oracle inserts rows into a table, the table only has an initial extent with a number of free blocks allocated to it. Otherwise the table is empty. Therefore, you might consider preallocating space for the table in a free list group. This guarantees an optimal allocation of extents containing free blocks to the free list groups, and therefore to the instances. Preallocation also avoids extent allocation overhead.

The advantage of doing this is that the physical storage layout can be determined in advance. Moreover, the technique of allocating extents enables you to select the physical file or volume from which the new extents are allocated. However, you should consider whether and how to implement the ALLOCATE EXTENT clause and how to use a few Oracle initialization parameters when you preallocate as described in the following paragraphs:

- Preallocating Extents with The ALLOCATE EXTENT Clause
- Extent Preallocation Using MAXEXTENTS, MINEXTENTS, and INITIAL

Preallocating Extents with The ALLOCATE EXTENT Clause

The ALLOCATE EXTENT clause of the ALTER TABLE or ALTER CLUSTER statement enables you to preallocate an extent to a table, index, or cluster with parameters to specify the extent size, datafile, and a group of free lists with which to associate the object.

Exclusive and Shared Modes and the ALLOCATE EXTENT Clause You can use the ALTER TABLE (or CLUSTER) ALLOCATE EXTENT statement while the database is running in exclusive mode, as well as in shared mode. When an instance runs in exclusive mode, the instance still follows the same rules for locating space. A transaction can use the master free list or the specific free list group for that instance.

The SIZE Parameter and the ALLOCATE EXTENT CLAUSE The SIZE parameter of the ALLOCATE EXTENT clause is the extent size in bytes, rounded up to a multiple of the block size. If you do not specify SIZE, then Oracle calculates the extent size according to the values of the NEXT and PCTINCREASE storage parameters.

Oracle does not use the value of SIZE as a basis for calculating subsequent extent allocations, which are determined by the values set for the NEXT and PCTINCREASE parameters.

The DATAFILE Parameter and the ALLOCATE EXTENT Clause This parameter specifies the datafile from which to take space for an extent. If you omit this parameter, then Oracle allocates space from any accessible datafile in the tablespace containing the table.

The filename must exactly match the string stored in the **control file** and the filename is case-sensitive. You can query the FILE_NAME column of the DBA_DATA_FILES data dictionary view for this string.

The INSTANCE Parameter and the ALLOCATE EXTENT Clause This parameter assigns the new space to the free list group associated with the **instance number** *integer*. At startup, each instance acquires a unique instance number that maps the instance to a group of free lists. The lowest instance number is 1, not 0; the maximum value is operating system-specific. The syntax is:

```
ALTER TABLE tablename ALLOCATE EXTENT (... INSTANCE n )
```

where *n* maps to the free list group with the same number. If the instance number is greater than the number of free list groups, then it is hashed as follows to determine the free list group to which it is assigned:

$$modulo(n,\#\_freelistgroups) + 1$$

If you do not specify the INSTANCE parameter, then the new space is assigned to the table but not allocated to any group of free lists. Such space is included in the master free list of free blocks as needed when no other space is available.

Note: Use a value for INSTANCE that corresponds to the number of the free list group you wish to use rather than the actual instance number.

See Also: *Oracle9i Real Application Clusters Administration* for more information about the INSTANCE parameter

Extent Preallocation Using MAXEXTENTS, MINEXTENTS, and INITIAL

You can prevent automatic extent allocations by preallocating extents to free list groups associated with particular instances and by setting MAXEXTENTS to the current number of extents (preallocated extents plus MINEXTENTS). You can minimize the initial allocation when you create the table or cluster by setting MINEXTENTS to 1, which is the default, and by setting INITIAL to its minimum value which is two data blocks, or 10K for a block size of 2048 bytes. To also minimize performance issues among instances for data blocks, create multiple datafiles for each table and associate each instance with a different file.

If you expect to increase the number of nodes in your system, then enable for additional instances by creating tables or clusters with more free list groups than the current number of instances. You do not have to allocate space to those free list groups until it is needed. Only the master free list of free blocks has space allocated to it automatically.

To associate a data block with a free list group, either lower the data block's usage to be less than the value set for PCTUSED by a process running on an instance using that free list group, or specifically allocate the block to that free list group. Therefore, a free list group that is never used does not leave unused free data blocks.

Extent Management and Locally Managed Tablespaces

Allocating and deallocating extents are expensive operations that you should minimize. Most of these operations in Real Application Clusters require interinstance coordination. In addition, a high rate of extent management operations can more adversely affect performance in Real Application Clusters environments than in single instance environments. This is especially true for dictionary managed tablespaces.

Identifying Extent Management Issues

If the "row cache lock" event is a significant contributor to the non-idle wait time in V$SYSTEM_EVENT, then there is a performance issue in the data dictionary cache. Extent allocation and deallocation operations could cause this.

V$ROWCACHE provides data dictionary cache information for DC_USED_EXTENTS and DC_FREE_EXTENTS. This is particularly true when the values for DLM_CONFLICTS for those parameters increase significantly over time. This means that excessive extent management activity is occurring.

Minimizing Extent Management Operations

Proper storage parameter configuration for tables, indexes, temporary segments, and rollback segments decreases extent allocation and deallocation frequency. Do this using the INITIAL, NEXT, PCTINCREASE, MINEXTENTS, and OPTIMAL parameters.

Using Locally Managed Tablespaces

You can greatly reduce extent allocation and deallocation overhead if you use locally managed tablespaces. For optimal performance and space use, segments in locally managed tablespaces should ideally have similar space allocation characteristics. This enables you to create the tablespace with the proper uniform extent size that corresponds to the ideal extent size increment calculated for the segments.

For example, you could put tables with relatively high insert rates in a tablespace with a 10MB uniform extent size. On the other hand, you can place small tables with limited DML activity in a tablespace with a 100K uniform extent size. For an existing system where tablespaces are not organized by segment size, this type of configuration can require significant reorganization efforts with limited benefits. For that reason, the compromise is to create most of your tablespaces as locally managed with AUTOALLOCATE instead of UNIFORM extent allocation.

> **See Also:** *Oracle9i SQL Reference* for more information about the AUTOALLOCATE and UNIFORM clauses of the CREATE TABLESPACE statement

Glossary

buffer busy global cache

A wait event that is signaled when a process has to wait for a block to become available because another process is obtaining a resource for this block.

buffer busy waits

A wait event that is signaled when a process cannot get a buffer because another process is using the buffer at that moment.

cache convert waits

A statistic showing the total number of waits for all up-convert operations, such as **global cache null to S**, **global cache null to X**, and **global cache S to X**.

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.

cache open waits

A statistic showing the total number of waits for **global cache open S** and **global cache open X**.

cluster

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

cluster database

The generic term for an **Oracle 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 membership across the cluster. The CM also monitors process health, specifically the health of the database instance. The **Global Enqueue Service Monitor (LMON)**, a background process that monitors the health of the **Global Cache Service (GCS)**, registers and de-registers from the CM.

connection load balancing

A feature that balances the number of active connections among various instances and **shared server** dispatchers for the same service.

connect-time failover

A client connect request is forwarded to another listener if the first listener is not responding. Connect-time failover is enabled by **service registration**, because the listener knows whether an instance is up prior to attempting a connection.

consistent gets

Consistent gets is a statistic showing the number of buffers that are obtained in consistent read (CR) mode.

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 any transactions on those instances have committed.

Console

The **Oracle Enterprise Manager** Console gives you a central point of control for the Oracle environment through an intuitive graphical user interface (GUI) that provides powerful and robust system management.

control file

A file that records the physical structure of a database and contains the database name, the names and locations of associated databases and online redo log files, the timestamp of the database creation, the current log sequence number, checkpoint information and various other records about the database's structure and health.

CR blocks received per transaction

The number of CR blocks shipped from the instance that has a block in exclusive access mode to the instance requesting a CR version of this block.

cr request retry

A statistic that quantifies a wait that is incurred whenever Oracle re-submits a consistent read request when Oracle detects that the holding instance is no longer available.

db block changes

A statistic that shows the number of current buffers obtained in exclusive mode for DML.

db block gets

A statistic that shows the number of current buffers obtained for a read.

DBWR cross-instance writes

Also known as *forced writes*, a statistic showing the number of writes that an instance has to perform to disk to make a previously exclusively held block available for another instance to read into its buffer cache. DBWR cross-instance writes are practically eliminated with Cache Fusion, unless you specify a value greater than 0 (zero) for the `GC_FILES_TO_LOCKS` parameter.

decision support system (DSS)

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

dedicated server

A server that requires a dedicated server process for each user process. There is one server process for each client. **Oracle Net** sends the address of an existing server process back to the client. The client then resends its connect request to the server address provided. Contrast this with the **shared server.**

degree of parallelism (DOP)

Specifies the number of processes, or threads, used in parallel execution. Each parallel process or thread can use one or two parallel execution processes depending on the SQL statement's complexity.

DFS Lock Handles

Pointers to global resources. To perform operations on global enqueue service resources, the process first needs to acquire a DFS handle.

dispatcher

A process that enables many clients to connect to the same server without the need for a dedicated server process for each client. A dispatcher handles and directs multiple incoming network session requests to shared server processes. See also **shared server.**

flow control messages sent

The number of flow-control (nullreq and nullack) messages that are sent by the LMS process.

flow control messages received

The number of flow-control (nullreq and nullack) messages received by the LMD process.

forced disk writes

The forced writing of a data block to disk by one instance when the data block is requested by another instance for a DML operation. Forced Writes are practically eliminated in Oracle9*i* with Cache Fusion, but they remain relevant if you specify 1:1 or 1:n releasable or fixed resources with the `GC_FILES_TO_LOCKS` parameter. In this case, Cache Fusion is disabled for that tablespace.

global cache bg acks

A wait event that only can occur during startup or shutdown of an instance when the LMS process finalizes its operations.

global cache busy

A wait event that occurs whenever a session has to wait for an ongoing operation on the resource to complete.

global cache cr cancel wait

A wait event that occurs whenever a session waits for the AST to complete for a canceled block access request. Cancelling the request is part of the Cache Fusion Write Protocol.

global cache converts

A statistic showing resource converts of buffer cache blocks. This statistic is incremented whenever GCS resources are converted from Null to Exclusive, Shared to Exclusive, or Null to Shared.

global cache convert time

The accumulated time that all sessions require to perform global conversions on GCS resources.

global cache convert timeouts

A statistic that is incremented whenever a resource operation times out.

global cache cr block flush time

The time waited for a log flush when a CR request is served. Once LGWR has completed flushing the changes to a buffer that is on the log flush queue, LMS can send it. It is part of the serve time.

global cache cr blocks received

When a process requests a consistent read for a data block that is not in its local cache, it sends a request to another instance. Once the request is complete, in other words, the buffer has been received, Oracle increments the statistic.

global cache cr block receive time

A statistic that records the total time required for consistent read requests to complete. In other words, it records the accumulated round-trip time for all requests for consistent read blocks.

global cache cr blocks served

A statistic showing the number of requests for a consistent read block served by LMS. Oracle increments this statistic when the block is sent.

global cache cr block build time

A statistic showing the time that the LMS process requires to create a consistent read block on the holding instance

global cache cr block send time

A statistic showing the time required by LMS to initiate a send of a consistent read block. For each request, timing begins when the block is sent and stops when the send has completed. This statistic only measures the time it takes to initiate the send; it does not measure the time elapsed before the block arrives at the requesting instance.

global cache cr cancel wait

Await event that occurs when a session waits for the acquisition interrupt to complete for a canceled CR request. Cancelling the CR request is part of the Cache Fusion write protocol.

global cache cr request

A wait event that occurs whenever a process has to wait for a pending CR request to complete. The process waited for either shared access to a block to be granted before reading the block from disk into the cache, or it waited for the LMS of the holding instance to send the block.

global cache current block flush time

A statistic showing the time it takes to flush the changes to a block to disk, otherwise known as a *forced log flush*, before the block is shipped to the requesting instance

global cache current block pin time

A statistic showing the time it takes to the pin the current block before shipping it to the requesting instance. Pinning a block is necessary to disallow further changes to the block while it is prepared to be shipped to another instance.

global cache current blocks received

A statistic showing the number of current blocks received from the holding instance over the **interconnect**.

global cache current block receive time

A statistic showing the accumulated round-trip time for all requests for current blocks

global cache current block send time

A statistic showing the time it takes to send the current block to the requesting instance over the **interconnect**.

global cache current blocks served

A statistic showing the number of current blocks shipped to the requesting instance over the **interconnect**

global cache freelist wait

A statistic showing when a wait event that occurs if Oracle must wait after it detects that the local element free list is empty.

global cache freelist waits

A statistic showing the number of times Oracle found the resource element free list empty.

global cache gets

A statistic showing the number of buffer gets that result in opening a new resource with the GCS.

global cache get time

A statistic showing the accumulated time of all sessions needed to open a GCS resource for a local buffer.

global cache initialization parameters

Initialization parameters that determine the size of the collection of global that protect the database buffers on all instances.

global cache null to S

A wait event that occurs whenever a session has to wait for a resource conversion to complete.

global cache null to X

A wait event that occurs whenever a session has to wait for this resource conversion to complete.

global cache open S

A wait event that occurs when a session has to wait for receiving permission for shared access to the requested resource.

global cache open X

A wait event that occurs when a session has to wait for receiving a exclusive access to the requested resource.

global cache S to X

A wait event that occurs whenever a session has to wait for this resource conversion to complete.

global cache pending ast

A wait event that can occur when a process waits for an acquisition interrupt before Oracle closes a resource element.

global cache pred cancel wait

A wait event that occurs when a session must wait for the acquisition interrupt to complete for a canceled predecessor read request. Cancelling a predecessor read request is part of the Cache Fusion write protocol.

global cache retry prepare

A wait event that occurs whenever Oracle fails to prepare a buffer for a consistent read or Cache Fusion request, and when Oracle cannot ignore or skip this failure.

Global Cache Service (GCS)

The process that implements Cache Fusion. It maintains block modes for blocks in the global role and is responsible for block transfers among instances. The Global Cache Service accomplishes these tasks using background processes such as the Global Cache Service process (LMS) and the Global Enqueue Service process (GES).

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 LMS*n* varies depending on the amount of messaging traffic among nodes in the cluster. The LMS*n* handle the acquisition interrupt and blocking interrupt requests from a remote instance for Global Cache Service resources. For cross-instance consistent read requests, LMS*n* creates a consistent read version of the block and sends it to the requesting instance. LMS*n* also controls the flow of messages to and from remote instances.

global database name

The full name of the database that uniquely identifies it from another database. The global database name is of the form `database_name.database_domain`, for example, sales.us.acme.com.

Global Enqueue Service (GES)

This service coordinates enqueues that are shared globally.

Global Enqueue Service Daemon (LMD)

The resource agent process that manages Global Enqueue Service resource requests. The LMD process also handles deadlock detection Global Enqueue Service requests. Remote resource requests are requests originating from another instance.

Global Enqueue Service Monitor (LMON)

The background process that monitors the entire cluster to manage global resources. LMON manages instance and process expirations and the associated recovery for the Global Cache and Global Enqueue Services. In particular, LMON handles the part of recovery associated with global resources. LMON-provided services are also known as cluster group services (CGS).

global lock async converts

A statistic showing the number of resources that Oracle converted from an incompatible mode.

global lock sync gets

A statistic showing the number of GCS resources that Oracle must open synchronously. Sync gets are mostly for GES resources (for example, library cache resources).

global lock async gets

A statistic showing the number of GES resources that Oracle must open asynchronously. Async gets are only used for GES resources and include the number of global cache gets.

global lock get time

A statistic showing the accumulated time for all GES resources that Oracle needed to open.

global lock sync converts

The number of GES resources that Oracle converted from an incompatible mode. Sync converts occur mostly for GES resources.

global lock convert time

A statistic showing the accumulated time for all global lock sync converts and global lock async converts.

hybrid database

A database that has both OLTP and Data Warehouse processing characteristics.

initialization parameter file

A file with parameter settings that initialize the database (`init`*`db_name`*`.ora`). In the case of Real Application Clusters, it initializes the instances within a cluster (`initsid.ora`). The default single initialization parameter file is known as `SPFILE.ORA`.

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 **Oracle System Identifier (SID)**, **instance name**, **instance number**, **rollback segments**, and **thread ID**.

instance groups

Instance groups limit the number of instances that participate in a parallel operations. You can create any number of instance groups, each consisting of one or more instances. You can then specify which instance group is to be used for any or all parallel operations. Parallel execution servers will only be used on instances that are members of the specified instance group.

instance name

Represents the name of the instance and is used to uniquely identify a specific instance when several instances share common service names. The instance name is identified by the `INSTANCE_NAME` parameter in the initialization parameter file. The instance name is identical to **Oracle System Identifier (SID)**.

instance number

A number that associates extents of data blocks with particular instances. The instance number enables you to start up an instance and ensure that it uses the

extents allocated to it for inserts and updates. This ensures that it does not use space allocated for other instances. The instance cannot use data blocks in another free list unless the instance is restarted with that instance number.

You can use various SQL options with the `INSTANCE_NUMBER` initialization parameter to associate extents of data blocks with instances.

The instance number is depicted by the `INSTANCE_NUMBER` parameter in the instance initialization file, `init`*`sid`*`.ora`.

interconnect

The communication link between the nodes.

Inter-Process 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**.

listener

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

load balancing

Load balancing is the even distribution of active database connections among instances. In the context of parallel execution, load balancing refers to the distribution of parallel execution server processes to spread work among the CPUs and memory resources.

lock buffers for read

A statistic showing the number of up-converts from Null to Shared.

lock gets per transaction

A statistic showing the number of global lock sync gets and global lock async gets per transaction.

lock converts per transaction

A statistic showing the number of global local sync converts and global lock async converts per transaction.

messages flow controlled

The number of messages intended to be sent directly but that are instead queued and delivered later by LMD / LMS.

messages received

The number of messages received by the LMD process.

messages sent directly

The number of messages sent directly by Oracle processes.

messages sent indirectly

The number of messages explicitly queued by Oracle processes.

node

A node is machine on which one or more instances reside.

operating system context switches

Operating system context switches occur when a thread's time allotment has elapsed, when a thread with a higher priority has become ready to run, or when a running thread needs to wait, for example, for I/O to complete.

operating system-dependent (OSD) clusterware

Software that consists of several software components developed by Oracle or other vendors. The OSD layer maps the key operating system/cluster-ware services required for proper operation of **Real Application Clusters**.

Oracle Data Gatherer

The Oracle Data Gatherer collects performance statistics for the **Oracle Performance Manager**. You must install the Oracle Data Gatherer on a **node** on your network.

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 Enterprise Manager Console

A suite of GUI tools that make up the Oracle Enterprise Manager product.

Oracle Intelligent Agent

A process that runs on each of the **node** that functions as the executor of jobs and events sent by the console by way of the Management Server. The Oracle Intelligent

Agent ensures high availability since the agent can function regardless of the status of the **Console** or network connections.

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. The main function of Oracle Net is to establish network sessions and transfer data between a client machine and a server or between two servers. Once a network session is established, Oracle Net acts as a data courier for the client and the server.

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.

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 System Identifier (SID)

A name that identifies a specific instance of a running pre-release 8.1 Oracle database. 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 `init`*`db_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.

Oracle9*i* Enterprise Edition

Oracle9*i* Enterprise Edition is an object-relational database management system (ORDBMS). It provides the applications and files to manage a database. All other Real Application Clusters components are layered on top of the Oracle9*i* Enterprise Edition.

parallel automatic tuning

A feature that automatically controls values for all parameters related to parallel execution. These parameters affect several aspects of server processing, namely, the

degree of parallelism (DOP), the adaptive multiuser feature, and memory sizing. Initialize and automatically tune parallel execution by setting the initialization parameter `PARALLEL_AUTOMATIC_TUNING` to `true`.

parallel execution

Multiple processes operating together to complete a single database transaction. Parallel execution works on both single instance and cluster database Oracle installations. Parallel execution is also referred to *parallel query.*

physical reads

A statistic showing the number of disk reads that had to be performed when a request for a data block could not be satisfied from a local cache.

physical writes

The number of write I/Os performed by the DBWN*n* processes. This number includes the number of DBWR cross instance writes (forced writes) in Oracle9*i* when `GC_FILES_TO_LOCKS` is set. Setting `GC_FILES_TO_LOCKS` for a particular datafile will enable the use of the old ping protocol, and will not leverage the Cache Fusion architecture.

raw devices

Disks or partitions on disk drives that do not have a file system set up on them. Raw devices are used for Real Application Clusters since they enable the sharing of disks.

raw volumes

See **raw devices**.

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. You can invoke RMAN as a command line utility from the operating system (O/S) prompt or use the GUI-based Enterprise Manager Backup Manager.

remote instance undo block writes

A statistic showing the number of undo blocks written to disk by DBWn as part of a forced write.

remote instance undo header writes

A statistic showing the number of rollback segment header blocks written to disk by DBWn as part of a forced write.

repository database

A database, such as that used by **Oracle Enterprise Manager,** that is a set of tables in an Oracle database, to store data to manage Real Application Clusters environments. This database is separate from any shared Real Application Clusters database on the **node**s.

seed database

A preconfigured, ready-to-use database that requires minimal user input to create.

server clustering

See **Oracle Real Application Clusters**.

Server Management

A comprehensive, integrated system management solution for managing Real Application Clusters environments. Server Management enables you to manage cluster databases in heterogeneous environments. Server Management is part of the open client/server architecture of **Oracle Enterprise Manager**. In addition to managing cluster databases, Server Management enables you to schedule jobs, perform event management, monitor performance, and obtain statistics to monitor Real Application Clusters performance.

service name

A logical representation of a database, which 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 is the **global database name**, a name comprised of the database name (`DB_NAME`) and domain name (`DB_DOMAIN`), entered during installation or database creation.

service registration

A feature by which the PMON process (or shared server Dispatcher processes when using shared server) automatically registers information with a listener. Because

this information is registered with the listener, you do not need to configure the `listener.ora` file with this static information.

shared server

A server configured to enable many user processes to share very few server processes. This means increases the number of users that can be supported. With shared server, many user processes connect to a **dispatcher**. Contrast this with **dedicated server**.

star schemas

Query-centric schemas that when represented in a diagram have a *fact* table at the center. The fact table usually contains the data element that is central to queries operating against the schema. A fact table is often quite large and is surrounded by several *dimension* tables that contain data that are attributes of the data in the fact table. Star schemas simplify query development because it is intuitive as to how to join attributes in the dimension tables with the fact table data. Star schemas are best suited for data warehouse environments and are thus less useful for OTLP environments.

striping

Refers to the interleaving of a related block of data across disks. If you properly implement striping, then it reduces I/O and improves performance. Because striping software is operating system-dependent, rely on your vendor documentation to ensure proper installation and configuration.

Transmission Control Protocol/Interconnect Protocol (TCP/IP)

A set of protocols that enable cooperating computers to share resources across a network.

thread ID

The number of a redo thread for an instance. Any available redo thread number can be used, but an instance cannot use the same thread number as another instance.

transaction monitor

A class of software products that provide a transaction execution layer on top of the operating system. Transaction monitors combine database updates and submit them to a database. In doing this, the transaction monitor manages some of the consistency and correctness of the database. The monitor ensures that the rules of transaction atomicity are adhered to; updates take place completely or not at all. The advantages of using transaction monitors include increased throughput.

transparent application failover (TAF)

A runtime failover mechanism for high-availability environments, such as Real Application Clusters and Oracle Real Application Clusters Guard, that refers to the failover and re-establishment 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.

User Datagram Protocol (UDP)

A protocol similar to TCP/IP. However, it is simpler to administer. It is considered less reliable than TCP/IP because, for example, it does not guarantee message ordering.

User-mode IPC

An IPC-based (Inter-process Communication) protocol that directly accesses network hardware. As opposed to kernel-mode IPC, with user-mode IPC the protocol avoids the overhead of copying data into kernel space, making system calls, and incurring context switches.

Virtual Interface Architecture (VIA)

An implementation of user mode IPC.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W