Oracle9*i*

Database Performance Methods

Release 1 (9.0.1)

June 2001

Part No. A87504-02

ORACLE®

Oracle9i Database Performance Methods, Release 1 (9.0.1)

Part No. A87504-02

Primary Author: Michele Cyran

Contributing Author: Andrew Holdsworth

Contributors: Jorn Bartels, Maria Colgan, Bjorn Engsig, Cecilia Gervasio, Connie Dialeris Green, Mattias Jankowitz, Peter Kilpatrick, Anjo Kolk, JP Polk, Virag Saksena, Sabrina Whitehouse, Graham Wood

Graphic Designer: Valarie Moore

Contents

Send Us Your Comments

Oracle9i Database Performance Methods, Release 1 (9.0.1)

Part No. A87504-02

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

This book describes ways to improve Oracle performance by starting with good application design and using statistics to monitor application performance. It explains the Oracle Performance Improvement Method and the Emergency Performance Method, and it provides techniques for dealing with performance problems.

This preface contains these topics:

Audience

Oracle9i Database Performance Methods is a high-level aid for people responsible for the operation, maintenance, and performance of Oracle. To use this book, you could be a database administrator, application designer, programmer, or manager. You should be familiar with Oracle9*i*, the operating system, and application design before reading this manual.

Organization

This document contains:

Chapter 1, "Designing and Developing for Performance"

This chapter describes performance issues to consider when designing Oracle applications.

Chapter 2, "Monitoring and Improving Application Performance"

This chapter describes the Oracle Performance Improvement Method and the importance of statistics for application performance improvements.

Chapter 3, "Emergency Performance Techniques"

This chapter describes techniques for dealing with performance emergencies.

Related Documentation

Before reading this manual, you should have already read *Oracle9i Database Concepts*, the *Oracle9i Application Developer's Guide - Fundamentals*, and the *Oracle9i Database Administrator's Guide*.

For more information about Oracle Enterprise Manager and its optional applications, see *Oracle Enterprise Manager Concepts Guide*, *Oracle Enterprise Manager Administrator's Guide*, and *Oracle Enterprise Manager Performance Monitoring and Planning Guide*.

For more information about tuning the Oracle Application Server, see the *Oracle Application Server Performance and Tuning Guide*.

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 on how these schemas were created and how you can use them yourself.

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://technet.oracle.com/membership/index.htm`

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://technet.oracle.com/docs/index.htm`

Conventions

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

- Conventions in Text
- Conventions in Code Examples

Conventions in Text

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

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles, emphasis, syntax clauses, or placeholders.	*Oracle9i Database Concepts* You can specify the *parallel_clause*. Run `U`*`old_release`*`.SQL` where *old_release* refers to the release you installed prior to upgrading.
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, user names, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Specify the `ROLLBACK_SEGMENTS` parameter. Use the `DBMS_STATS.GENERATE_STATS` procedure.
lowercase monospace (fixed-width font)	Lowercase monospace typeface indicates executables 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, user names and roles, program units, and parameter values.	Enter `sqlplus` to open SQL*Plus. 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.

Conventions in Code Examples

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

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

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

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (digits [ , precision ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`
. . .	Horizontal ellipsis points indicate either: ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code	`CREATE TABLE ... AS subquery;` `SELECT col1, col2, ... , col`*`n`* `FROM employees;`
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as it is shown.	`acctbal NUMBER(11,2);` `acct CONSTANT NUMBER(4) := 3;`
Italics	Italicized text indicates variables for which you must supply particular values.	`CONNECT SYSTEM/`*`system_password`*

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.	`SELECT last_name, employee_id FROM employees;` `sqlplus hr/hr`

Documentation Accessibility

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

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

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.

1

Designing and Developing for Performance

Good system performance begins with design and continues throughout the life of your system. Carefully consider performance issues during the initial design phase, and it will be easier to tune your system during production.

This chapter contains the following subjects:

Oracle's New Methodology

System performance has become increasingly important as computer systems get larger and more complex and as the Internet plays a bigger role in business applications. In order to accommodate this, Oracle Corporation has designed a new performance methodology. It is based on years of Oracle designing and performance experience, and it explains clear and simple activities that can dramatically improve system performance.

Performance strategies vary in their effectiveness, and systems with different purposes, such as operational systems and decision support systems, require different performance skills. This book examines the considerations that any database designer, administrator, or performance expert should focus their efforts on.

System performance is designed and built into a system. It does not just happen. Performance problems are usually the result of contention for, or exhaustion of, some system resource. When a system resource is exhausted, the system is unable to scale to higher levels of performance. This new performance methodology is based on careful planning and design of the database, to prevent system resources from becoming exhausted and causing down-time. By eliminating resource conflicts, systems can be made scalable to the levels required by the business.

See Also: *Oracle9i Database Performance Guide and Reference*

Understanding Investment Options

With the availability of relatively inexpensive, high-powered processors, memory, and disk drives, there is a temptation to buy more system resources to improve performance. In many situations, new CPUs, memory, or more disk drives can indeed provide an immediate performance improvement. However, any performance increases achieved by adding hardware should be considered a short-term relief to an immediate problem. If the demand and load rates on the application continue to grow, then the chance that you will face the same problem in the near future is very likely.

In other situations, additional hardware does not improve the system's performance at all. Poorly designed systems perform poorly no matter how much extra hardware is allocated. Before purchasing additional hardware, make sure that there is no serialization or single threading going on within the application. Long-term, it is generally more valuable to increase the efficiency of your application in terms of the number of physical resources used per business transaction.

Understanding Scalability

The word *scalability* is used in many contexts in development environments. The following section provides an explanation of scalability that is aimed at application designers and performance specialists.

What is Scalability?

Scalability is a system's ability to process more workload, with a proportional increase in system resource usage. In other words, in a scalable system, if you double the workload, then the system would use twice as many system resources. This sounds obvious, but due to conflicts within the system, the resource usage might exceed twice the original workload.

Examples of bad scalability due to resource conflicts include the following:

- Applications requiring significant concurrency management as user populations increase
- Increased locking activities
- Increased data consistency workload
- Increased operating system workload
- Transactions requiring increases in data access as data volumes increase
- Poor SQL and index design resulting in a higher number of logical I/Os for the same number of rows returned
- Reduced availability, because database objects take longer to maintain

An application is said to be unscalable if it exhausts a system resource to the point where no more throughput is possible when it's workload is increased. Such applications result in fixed throughputs and poor response times.

Examples of resource exhaustion include the following:

- Hardware exhaustion
- Table scans in high-volume transactions causing inevitable disk I/O shortages
- Excessive network requests, resulting in network and scheduling bottlenecks
- Memory allocation causing paging and swapping
- Excessive process and thread allocation causing operating system thrashing

This means that application designers must create a design that uses the same resources, regardless of user populations and data volumes, and does not put loads on the system resources beyond their limits.

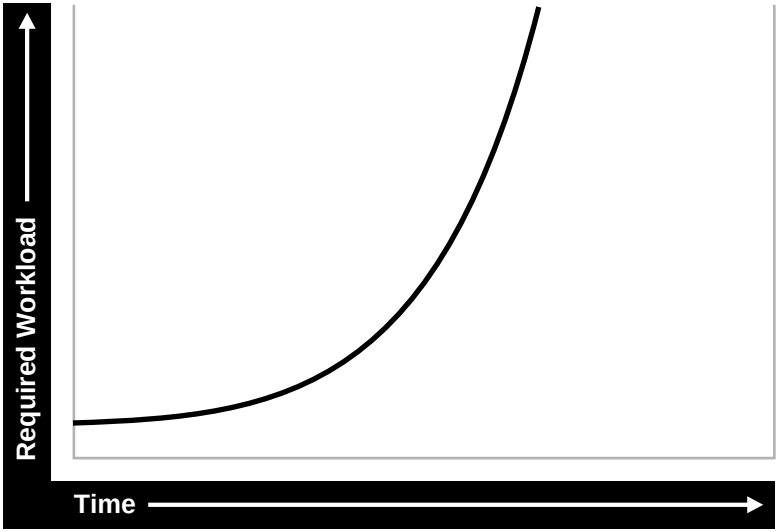
Internet Scalability

Applications that are accessible through the Internet have more complex performance and availability requirements. Some applications are designed and written only for Internet use, but even typical back-office applications, such as a general ledger application, might require some or all data to be available online.

Characteristics of Internet age applications include the following:

- Availability 24 hours a day, 365 days a year
- Unpredictable and imprecise number of concurrent users
- Difficulty in capacity planning
- Availability for any type of query
- Multi-tier architectures
- Stateless middleware
- Rapid development timescale
- Minimal time for testing

Figure 1–1 Internet Growth Curve



The above diagram illustrates the classic Internet / e-business growth curve, with demand growing at an increasing rate. Applications must scale with the increase of workload and also when additional hardware is added to support increasing demand. Design errors can cause the implementation to reach its maximum, regardless of additional hardware resources or re-design efforts.

Internet applications are challenged by very short development timeframes with limited time for testing and evaluation. However, bad design generally means that at some point in the future, the system will need to be re-architected or re-implemented. If an application with known architectural and implementation limitations is deployed on the Internet, and if the workload exceeds the anticipated demand, then there is real chance of failure in the future. From a business perspective, poor performance can mean a loss of customers. If Web users do not get a response in seven seconds, then the user's attention could be lost forever.

In many cases, the cost of re-designing a system with the associated downtime costs in migrating to new implementations exceeds the costs of properly building the original system. The moral of the story is simple: design and implement with scalability in mind from the start.

Factors Preventing Scalability

When building applications, designers and architects should aim for as close to perfect scalability as possible. This is sometimes called *linear* scalability, where system throughput is directly proportional to the number of CPUs.

In real life, linear scalability is impossible for reasons beyond a designer's control. However, making the application design and implementation as scalable as possible should ensure that current and future performance objectives can be achieved through expansion of hardware components and the evolution of CPU technology.

Factors Preventing Linear Scalability

1. Poor Application Design, Implementation, and Configuration

   The application has the biggest impact on scalability. For example:

   - Poor schema design can cause expensive SQL that does not scale.
   - Poor transaction design can cause locking and serialization problems.
   - Poor connection management can cause poor response times and unreliable systems.

   However, the design is not the only problem. The physical implementation of the application can be the weak link. For example:

   - Systems can move to production environments with bad I/O strategies.
   - The production environment could use different execution plans than those generated in testing.
   - Memory-intensive applications that allocate a large amount of memory without much thought for freeing the memory at runtime can cause excessive memory usage.
   - Inefficient memory usage and memory leaks put a high stress on the operating virtual memory subsystem. This impacts performance and availability.

2. Incorrect Sizing of Hardware Components

   Bad capacity planning of all hardware components is becoming less of a problem as relative hardware prices decrease. However, too much capacity can mask scalability problems as the workload is increased on a system.

3. Limitations of Software Components

   All software components have scalability and resource usage limitations. This applies to application servers, database servers, and operating systems. Application design should not place demands on the software beyond what it can handle.

4. Limitations of Hardware Components

   Hardware is not perfectly scalable. Most multiprocessor machines can get close to linear scaling with a finite number of CPUs, but after a certain point each additional CPU can increase performance overall, but not proportionately. There might come a time when an additional CPU offers no increase in performance, or even degrades performance. This behavior is very closely linked to the workload and the operating system setup.

Note: These factors are based on Oracle Corporation's Server Performance group's experience of tuning unscalable systems.

System Architecture

There are two main parts to a system's architecture:

- Hardware and Software Components
- Configuring the Right System Architecture for Your Requirements

Hardware and Software Components

Hardware Components

Today's designers and architects are responsible for sizing and capacity planning of hardware at each tier in a multi-tier environment. It is the architect's responsibility to achieve a balanced design. This is analogous to a bridge designer who

must consider all the various payload and structural requirements for the bridge. A bridge is only as strong as its weakest component. As a result, a bridge is designed in balance, such that all components reach their design limits simultaneously.

The main hardware components are the following:

- CPU
- Memory
- I/O Subsystem
- Network

CPU There can be one or more CPUs, and they can vary in processing power from simple CPUs found in hand-held devices to high-powered server CPUs. Sizing of other hardware components is usually a multiple of the CPUs on the system.

Memory Database and application servers require considerable amounts of memory to cache data and avoid time-consuming disk access.

I/O Subsystem The I/O subsystem can vary between the hard disk on a client PC and high performance disk arrays. Disk arrays can perform thousands of I/Os per second and provide availability through redundancy in terms of multiple I/O paths and hot pluggable mirrored disks.

Network All computers in a system are connected to a network, from a modem line to a high speed internal LAN. The primary concerns with network specifications are bandwidth (volume) and latency (speed).

> **See Also:** *Oracle9i Database Performance Guide and Reference* for more information on tuning these resources

Software Components

The same way computers have common hardware components, applications have common functional components. By dividing software development into functional components, it is possible to comprehend the application design and architecture better. Some components of the system are performed by existing software bought to accelerate application implementation or to avoid re-development of common components.

The difference between software components and hardware components is that while hardware components only perform one task, a piece of software can perform the roles of various software components. For example, a disk drive only stores and retrieves data, but a client program can manage the user interface and perform business logic.