Oracle Database 10*g*: Performance Tuning

Student Guide

D19165GC11

Edition 1.1

February 2007

D49414

ORACLE®

Authors

James Spiller

Jean-Francois Verrier

Technical Contributors and Reviewers

Mirza Ahmad
Christian Bauwens
Tom Best
Maria Billings
Harald Van Breederode
Mary Bryksa
Peter Caldwell
Immanuel Chan
Edward Choi
Patrick Cullen
Andy Fortunak
Joel Goodman
Bert Van Gorkom
Scott Gossett
Connie Green
Laurie Holdren
Sanjeev Joglekar
Pete Jones
Donna Keesling
Teria Kidd
Vishal Kohli
Subodh Kumar
Pierre Labrousse
Herve Lejuene
Stefan Lindblad
Roderick Manalac
Louise Morin
Howard Ostrow
Jagadeesh Papajah
Gopi Ramamoorthy
Joerg Reinschmidt
Rick Shavwer
Ira Singer
Janet Stern
Bransislav Valny
Christian Weinfurtner
Lester Wells
James Womack
Anthony Woodell
Fang Wu

Editors

Atanu Raychaudhuri

Nita Pavitran

Richard Wallis

Graphic Designer

James Hans

Publisher

Jobi Varghese

Contents

8 Tuning the Shared Pool

9 Tuning the Buffer Cache

15 Performance Tuning: Summary

Preface

Profile

Prerequisites:

- Oracle Database 10*g*: Administration Workshop I (D17090GC30)
- Oracle Database 10*g*: Administration Workshop II (D17092GC30)

or

- Oracle Database 10*g*: New Features Administrators (D17079GC21)

or

- Equivalent experience

Typographic Conventions

Typographic Conventions in Text

Convention	Element	Example
Bold	Emphasized words and phrases in Web content only	To navigate within this application, do **not** click the Forward and Back buttons.
Bold italic	Glossary term (if there is a glossary)	The ***algorithm*** inserts the new key.
Brackets	Key names	Press [Enter].
Caps and lowercase	Buttons, check boxes, application triggers, windows	Click the Executable button. Select the Can't Delete Card check box. Assign a When-Validate-Item trigger to the ORD block. Open the Master Schedule window.
Angle brackets	Menu paths	Select File > Save.
Commas	Key sequences	Press and release the following keys one at a time: [Alt], [F], [D]
Courier new, case sensitive (default is lowercase)	Code output, directory names, file names, passwords, path names, user input, usernames	Code output: `debug.set ('I', 300);` Directory: `bin` (DOS), `$FMHOME` (UNIX) File name: Locate the `init.ora` file. Password: Use `tiger` as your password. Path name: Open `c:\my_docs\projects`. User input: Enter `300`. Username: Log in as `HR`.
Initial cap	Graphics labels (unless the term is a proper noun)	Customer address (*but* Oracle Payables)
Italic	Emphasized words and phrases, titles of books and courses, variables	Do *not* save changes to the database. For further information, see *Oracle Database SQL Reference 10g Release 1(10.1).* Enter *`user_id`*`@us.oracle.com`, where *`user_id`* is the name of the user.

Typographic Conventions (continued)

Typographic Conventions in Text (continued)

Convention	Element	Example
Quotation marks	Interface elements with long names that have only initial caps; lesson and chapter titles in cross-references	Select “Include a reusable module component” and then click Finish. This subject is covered in the lesson titled “Working with Objects.”
Uppercase	SQL column names, commands, functions, schemas, table names, database trigger names	Use the `SELECT` command to view information stored in the `LAST_NAME` column of the `EMPLOYEES` table.

Typographic Conventions in Code

Convention	Element	Example

Lowercase	Column names, table names, database trigger names	`SELECT last_name`
		`FROM employees;`
		
		`CREATE OR REPLACE TRIGGER secure_employees`

	Passwords	`CREATE USER scott` `IDENTIFIED BY tiger;`
	PL/SQL objects	`items.DELETE(3);`
Lowercase italic	Syntax variables	`CREATE ROLE` *`role`*
Uppercase	SQL commands and functions	`SELECT first_name` `FROM employees;`

Typographic Conventions (continued)

Typographic Conventions in Navigation Paths

This course uses simplified navigation paths, such as the following example, to direct you through Oracle applications.

Example:

Invoice Batch Summary

(N) Invoice > Entry > Invoice Batches Summary (M) Query > Find (B) Approve

This simplified path translates to the following:

1. (N) In the Navigator window, select Invoice > Entry > Invoice Batches Summary.
2. (M) In the menu, select Query > Find.
3. (B) Click the Approve button.

Notation:

(N) = navigator	(I) = icon
(M) = menu	(H) = hyperlink
(T) = tab	(B) = button

1

Introduction

Course Objectives

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

- Use the Oracle database tuning methodology that is appropriate to the available tool
- Use database advisors to proactively tune an Oracle database
- Use the tools based on Automatic Workload Repository to tune the database
- Use Statspack reports to tune the database
- Diagnose and tune common database performance problems
- Use Enterprise Manager performance-related pages to monitor an Oracle database

ORACLE

Agenda

Day	Lesson	Topic
1	1	Introduction
1	2	Performance Tuning: Overview
1	3	Statistics and Wait Events
1	4	Metrics, Alerts, and Baselines
1	5	Using Statspack
2	6	Using Automatic Workload Repository
2	7	Reactive Tuning
2	8	Tuning the Shared Pool
3	9	Tuning the Buffer Cache
3	10	Automatic Shared Memory Management

Agenda

Day	Lesson	Topic
3	11	Checkpoint and Redo Tuning
4	12	Tuning I/O
4	13	Tuning PGA and Temporary Space
4	14	Tuning Block Space Usage
4	15	Performance Tuning: Summary

ORACLE

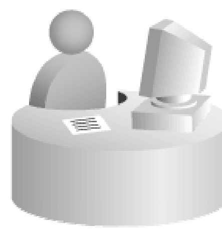
Tuning Questions

- Who tunes?
- What is tuned?
- Which tuning methodology is used?

Who Tunes?

The following individuals are involved with tuning:

- **Database administrators**
- **Application architects**
- **Application designers**
- **Application developers**
- **System administrators**



Who Tunes?

Everyone involved with the Oracle database software—including system architects, designers, developers, database administrators, and system administrators—should think about performance.

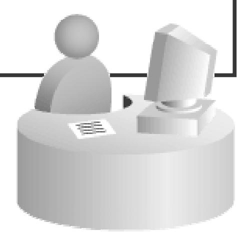
If problems develop, the database administrator (DBA) usually makes the first attempt at resolving them. Therefore, the DBA should have an accurate overview of all the applications in the database and their interactions with each other. DBAs often enlist the aid of either developers for application tuning or system administrators for tuning OS and I/O issues. This course is intended for the DBA who is responsible for ongoing tuning and monitoring of an Oracle database. However, anyone involved in the design, development, and deployment of an Oracle database can also benefit.

What Is Tuned?

Performance tuning areas:

- **Application:**
  - **Poorly written SQL**
  - **Serialized resources**
  - **Poor session management**
- **Instance tuning:**
  - **Memory**
  - **Database structure**
  - **Instance configuration**
- **Operating system:**
  - **I/O**
  - **Swap**
  - **Parameters**

Focus of this course



ORACLE

What Is Tuned?

To have a well-tuned system, you must carefully design systems and applications. Common performance problems can be grouped as follows:

- Application issues: Poorly written SQL, serialized resources, and poor session management
- Instance issues: Memory, I/O, and instance configuration
- Operating system issues: Swap, I/O, parameter settings, and network issues

You achieve the largest return on investment of time and effort by tuning the application. Tuning the SQL statements, the access paths, and the storage structures are all important parts of application tuning.

Instance tuning and application tuning use different sets of skills and tools. Application tuning is dependent on the type of application. Data warehouse applications and online transaction processing applications use different access methods and data structures for performance. Operating system tuning is dependent on the operating system being used.

Separate Oracle University courses address the specific skills and tools used in application tuning: *Oracle Database 10g: SQL Tuning Workshop* for OLTP tuning and statement tuning, and *Oracle Database 10g: Implement and Administer a Data Warehouse* for data warehouse issues.

What Is Tuned? (continued)

Some database issues require assistance from the system administrator. This course addresses some of the generic issues. Separate courses are available for Linux- and Windows-based systems that deal with OS-specific issues. The Linux course covers many issues that are generic to UNIX and UNIX-like operating systems.

What to Tune in the Instance

Instance tuning areas:

- Memory:
  - Insufficient memory
  - Poorly allocated memory
- I/O:
  - Insufficient bandwidth
  - Poorly allocated disk space
  - Poor database configuration
- Instance configuration:
  - Inappropriate instance parameters
  - Poor recovery and availability configuration

What to Tune in the Instance

This course focuses on instance tuning: memory, I/O, and instance configuration. This is the typical realm of the DBA. The workshop examples use well-tuned SQL statements so that you can focus on instance issues. Outside of the practice environment, problems will not be so clearly defined. Application issues can appear to be instance issues. Application tuning and instance tuning overlap. Sometimes an instance can be tuned to compensate for application problems.

Instance tuning areas can be broken down further:

- Memory issues: Insufficient memory or poorly allocated memory
- I/O issues: Insufficient bandwidth, poorly allocated disk space, or poor database configuration
- Instance configuration: Inappropriate instance parameters, and poor recovery and availability configuration

Which Tuning Methodology Is Used?

The procedures used to tune depend on the tool:

- **Basic tools**
  - **Dynamic performance views**
  - **Statistics**
  - **Metrics**
  - **Enterprise Manager pages**
- **AWR or Statspack**
- **Automatic Database Diagnostic Monitor (ADDM)**
- **DBA scripts**

ORACLE

Which Tuning Methodology Is Used?

The methodology that you use varies with the tools that are available.

If you have Oracle Database 10*g*: Enterprise Edition with the optional tuning packs, Automatic Database Diagnostic Monitor (ADDM) is available along with other Automatic Workload Repository (AWR)–based tools.

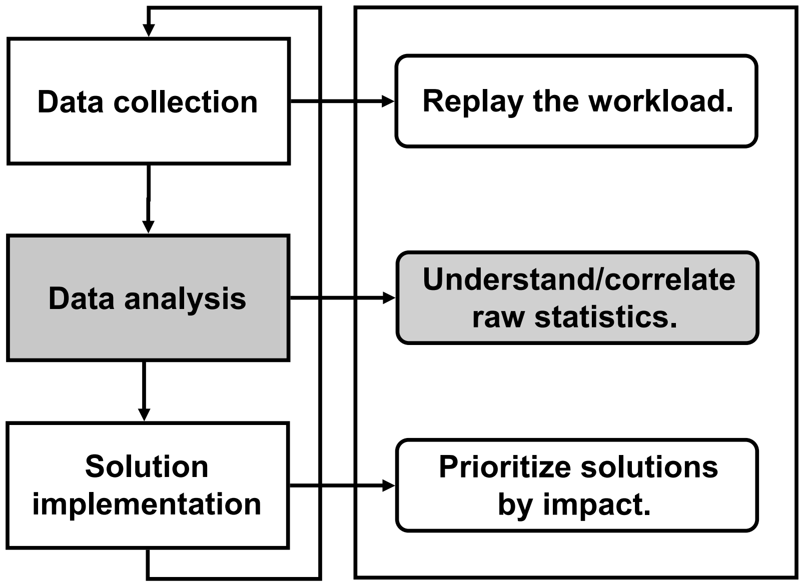
If you have Oracle Database 10*g*: Standard Edition, the tool to use is Statspack.

The steps for using both tools (ADDM and Statspack) are covered in this course.

In addition, many DBAs have developed their own tools and scripts for tuning.

All tuning tools depend on the basic tools of the dynamic performance views, statistics, and metrics that are collected by the instance.

Traditional Performance Tuning Methodology: Challenges

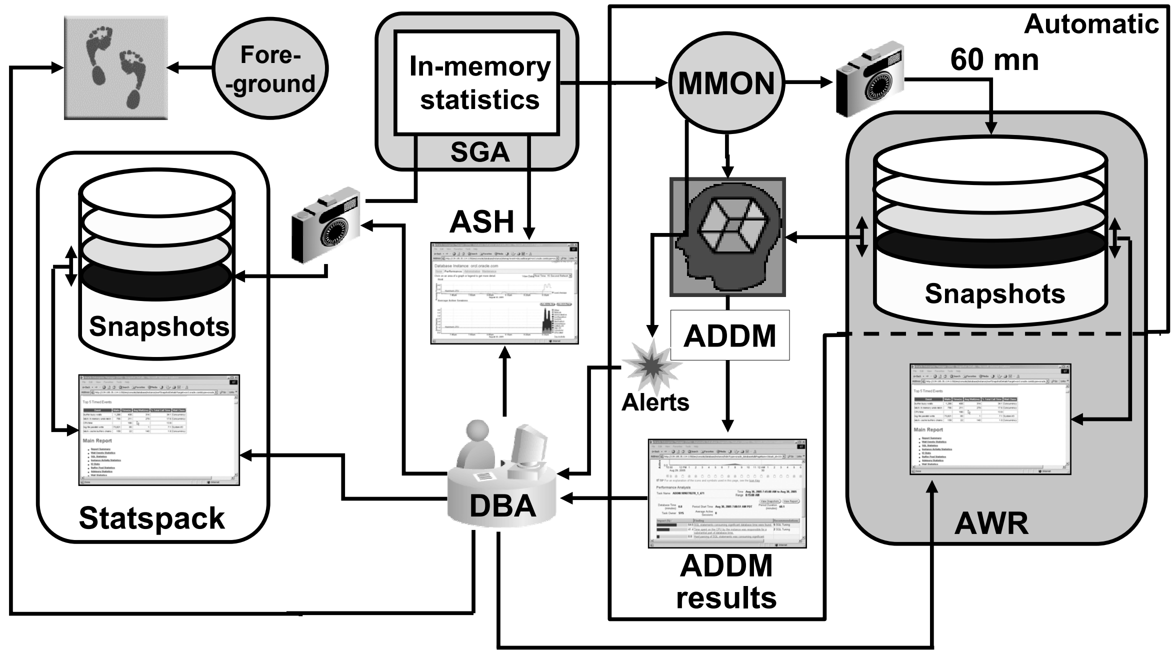


Traditional Performance Tuning Methodology: Challenges

There are three main phases in any performance tuning methodology.

- Data collection: In this phase, you need to identify the information relevant to diagnosing performance problems and set up an infrastructure to collect that information on a regular basis. However, your biggest challenge is to be able to replay the workload that causes the problems you encounter so that you can evaluate your solution.
- Data analysis: This phase is probably the most difficult because it needs an expert to understand and correlate all the relevant statistics together.
- Solution implementation: In this phase, you are often faced with multiple solutions for different problems identified during the previous phase. However, you need to use your own judgment to prioritize and quantify the solutions by impact.

Performance Monitoring Solutions



Performance Monitoring Solutions

In addition to the reactive tuning capabilities of previous releases (such as Statspack, SQL trace files, and performance views), Oracle Database 10*g* includes new methodologies to monitor your database.

- Proactive monitoring with Automatic Database Diagnostic Monitor (ADDM): This component is the ultimate solution for Oracle database tuning. ADDM automatically identifies bottlenecks within the Oracle database. Additionally, working with other manageability components, ADDM makes recommendations about the options available for fixing these bottlenecks.
- Reactive monitoring:
  - Server-generated alerts: The Oracle database server can automatically detect problematic situations. Reacting to a detected problem, the Oracle database server sends you an alert message with possible remedial action.
  - The Oracle database server has powerful new data sources and performance-reporting capabilities. Database Control provides an integrated performance management console that uses all relevant data sources. Using a drill-down method, you can manually identify bottlenecks with just a few mouse clicks.

Performance Monitoring Solutions (continued)

Data sources are included to capture important information about your database's health—for example, memory statistics (for current diagnostics) as well as statistics history stored in Automatic Workload Repository (AWR).

AWR simplifies performance data collection, and it has been designed with a high degree of manageability, automation, and data collection efficiency, and after careful consideration of the data volume collected. AWR is a part of the Database Diagnostics pack along with other features such as Automatic Database Diagnostic Monitor (ADDM).

If you use the Database Diagnostics pack, the first place to go for performance diagnostic help is ADDM. ADDM simplifies performance diagnosis by making it automatic, using the data collected by AWR. If you are licensed to use the Diagnostics Pack, you should use ADDM to perform the diagnostic work for you. For up-to-date license information specific to your release, refer to the *Oracle Database Licensing Information* manual. If you use the Database Diagnostics pack, you should use only AWR to capture performance data.

Using Features of the Packs

Monitoring and tuning with packs

Database Diagnostics Pack

- Automatic Workload Repository
- ADDM (Automated Database Diagnostic Monitor)
- Performance Monitoring (Database and Host)
- Event Notifications: Notification Methods, Rules and Schedules
- Event history/metric history (Database and Host)
- Blackouts
- Dynamic metric baselines
- Memory performance monitoring

Database Tuning Pack

- SQL Access Advisor
- SQL Tuning Advisor
- SQL Tuning Sets
- Reorganize Objects

Configuration Management Pack

- Database and Host Configuration
- Deployments
- Patch Database and View Patch Cache
- Patch staging
- Clone Database
- Clone Oracle Home
- Search configuration
- Compare configuration
- Policies

Monitoring and tuning without packs

- SQL traces
- Statspack
- System statistics
- Wait model
- Time model
- OS statistics
- Metrics
- Service statistics
- Histograms
- Optimizer statistics
- SQL statistics

ORACLE

Using Features of the Packs

The management packs names and features are listed in the left section of the slide. They all require a separate license that can be purchased only with Oracle Database Enterprise Edition. The features in these packs are accessible through Oracle Enterprise Manager Database Control, Oracle Enterprise Manager Grid Control, and APIs provided with the Oracle database software.

- Oracle Database Diagnostic Pack provides automatic performance diagnostic and advanced system-monitoring functionality. The following are part of this pack:
  - The `DBMS_WORKLOAD_REPOSITORY` package
  - The `DBMS_ADVISOR` package, if you specify `ADDM` as the value for the `ADVISOR_NAME` parameter, or if you specify any value starting with the `ADDM` prefix for the value of the `TASK_NAME` parameter
  - The `V$ACTIVE_SESSION_HISTORY` dynamic performance view
  - All data dictionary views beginning with the `DBA_HIST_` prefix, along with their underlying tables
  - All data dictionary views with the `DBA_ADVISOR_` prefix if queries to these views return rows with the value `ADDM` in the `ADVISOR_NAME` column or a value of `ADDM*` in the `TASK_NAME` column or the corresponding `TASK_ID`

Using Features of the Packs (continued)

  - The following reports found in the /rdbms/admin/ directory of the ORACLE_HOME directory are part of this pack: awrrpt.sql, awrrpti.sql, addmrtp.sql, addmrpti.sql, awrrpt.sql, awrrpti.sql, addmrpt.sql, addmrpti.sql, ashrpt.sql, ashrpti.sql, awrddrpt.sql, and awrddrpi.sql.

- Oracle Tuning Pack provides expert performance management for the Oracle database environment, including SQL tuning and storage optimization. Oracle Diagnostic Pack is a prerequisite product to Oracle Tuning Pack. Therefore, to use Tuning Pack, you must also have a Diagnostic Pack. The following are part of this pack:
  - The DBMS_SQLTUNE package
  - The DBMS_ADVISOR package, when the value of the ADVISOR_NAME parameter is either SQL Tuning Advisor or SQL Access Advisor
  - The sqltrpt.sql report found in the /rdbms/admin/ directory of the ORACLE_HOME directory
- Oracle Configuration Management Pack automates the time-consuming and often error-prone process of software configuration, software and hardware inventory tracking, patching, cloning, and policy management.

If you cannot purchase the packs mentioned above, especially the Database Diagnostics and Database Tuning packs, you can still use the traditional approach to performance tuning and monitoring by using Statspack reports, SQL traces, and most of the base statistics shown in the previous slide.

Tuning Scenario Alternatives

Without ADDM	With ADDM
Examine system utilization.	Review ADDM recommendations.
Look at wait events.	ADDM recommends use of `CURSOR_SHARING`.
Observe latch contention.	
See wait on shared pool and library cache latch.	
Review `V$SYSSTAT`.	
See parse time elapsed > parse time CPU and #hard parses greater than normal.	
Identify SQL by reviewing `V$SQL` for many statements with same hash plan.	
Examine objects accessed and review SQL.	
Identify hard parse issue by observing the SQL contains literals.	
Enable cursor sharing.	

Tuning Scenario Alternatives

The slide compares the tuning steps with and without the use of ADDM for a hard-parse scenario.

Here are the steps involved when you do not use ADDM:

1. You receive a call from a user complaining that the system is slow.
2. You examine the server machine and see that there are plenty of resources available, so obviously the slowdown is not due to the OS problems on the machine.
3. You look at the database instance and see that many of the sessions are waiting on latch-free waits.
4. Drilling down into the latches, you see that most of the latch-free waits are on library cache and shared pool latches.
5. From experience and referring to a number of books on the subject, you know that these latches are often associated with hard-parsing issues. For a double check, you look at the rate at which the statistics parse time elapsed and parse time CPU are increasing. You also observe that the elapsed time is accumulating much faster than CPU time. Your suspicion is confirmed.

Tuning Scenario Alternatives (continued)

6. At this stage, you have several ways to proceed, all of which try to identify skewed data distribution. One way is to look at the statistics for parse count (hard) for all sessions to see whether there are one or more sessions responsible for the majority of the hard parses. An alternative is to examine the shared pool to determine whether there are many statements with the same SQL plan but with different SQL text.
   In your example, you do the latter and find that there are a small number of plans associated with many different SQL texts.
7. When you review a few of these SQL statements, you determine that the SQL statements contain literal strings in `WHERE` clauses and so each of the statements must be separately parsed.
8. Having seen cases like this before, you can now say that the root cause of the problem is hard parsing caused by not using bind variables. You can thus move on to fixing the problem.
   The following are the steps involved when using ADDM for the same scenario:
   a. You receive a call from a user complaining that the system is slow.
   b. You examine the latest ADDM report and the first recommendation reads:

```
FINDING 3: 31% impact (7798 seconds)
------------------------------------
SQL statements were not shared due to the usage of literals.
This resulted in additional hard parses which were consuming
significant database time.
RECOMMENDATION 1: Application Analysis, 31% benefit (7798
seconds)
ACTION: Investigate application logic for possible use of bind
variables instead of literals. Alternatively, you may set the
parameter "cursor_sharing" to "force".
RATIONALE: SQL statements with PLAN_HASH_VALUE 3106087033 were
found to be using literals. Look in V$SQL for examples of such
SQL statements.
```

From this information, you immediately know that over 30% of the time is being spent parsing, and a recommendation to resolve the situation is provided. Note that the finding also includes a suspect plan hash value to enable you to quickly examine a few sample statements.

Tuning Methodology

Tuning steps:

1. **Tune from the top down:**
   – **The design before tuning the application code**
   – **The code before tuning the instance**
2. **Tune the area with the greatest potential benefit:**
   – **Identify the longest waits**
   – **Identify the largest service times**
3. **Stop tuning when the goal is met.**

Tuning Methodology

Oracle has developed a tuning methodology based on years of experience. The methodology presented in this course is also presented in the *Oracle Database Performance Tuning Guide*. This methodology is applied independent of the tools that you use. The ADDM tool follows this methodology automatically. The basic steps are the following:

- To be sure that the problem is in the database instance, check the OS statistics and general machine health before tuning the instance.
- Tune from the top down. Start with the design, then the application, and then the instance. As an example, try to eliminate the full table scans causing the I/O contention before tuning the tablespace layout on disk. The design should use appropriate data structures for the application and load characteristics. For example, reverse key indexes may work well in a RAC environment to reduce hot blocks due to a sequential primary key, but they may also lead to a large number of blocks being shipped across the interconnects if every instance is inserting into the same table. The application should avoid processes that require serialization through a single resource. A simple example is a single check number generator used by multiple processes. Tuning at the instance level is often limited by design and application choices.

Tuning Methodology (continued)

- Tune the area with greatest potential benefit. The tuning methodology presented in this course is simple. Identify the biggest bottleneck and tune it. Then repeat this process. All the various tuning tools have some way to identify the SQL statements, resource contention, or services that are taking the most time. Oracle Database 10*g* provides a time model and metrics to automate the process of identifying bottlenecks.
- Stop tuning when you meet your goal. This step implies that you already set tuning goals.

This is a general approach to tuning the database instance and may require multiple passes.

From a practical perspective, tuning during the design and development phases of a project tends to be more top-down. The tuning efforts during testing and production phases are often reactive and bottom-up. In all phases, tuning depends on actual test cases because theoretical tuning does not know all the variables that can be present. After a problem area is suspected or discovered, a test case is created and the area is tuned as in all the examples given in this course. Tune the area that has the greatest potential benefit. Reduce the longest waits and the longest service times.

Summary

In this lesson, you should have learned how to:

- Identify tuning tools
- Use a tuning methodology

2

Performance Tuning: Overview

Objectives

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

- Write appropriate tuning goals
- Apply the tuning methodology
- Balance performance and safety trade-offs
- Identify common tuning problems
- Log a performance service request with Oracle Support

General Tuning Session

Tuning sessions have the same procedure:

1. **Define the problem and state the goal.**
2. **Collect current statistics.**
3. **Consider some common performance errors.**
4. **Build a trial solution.**
5. **Implement and measure the change.**
6. **Decide: "Did the solution meet the goal?"**
   - **If answer is "No," go to step 3 and repeat.**
   - **If answer is "Yes," create a new baseline.**

ORACLE

General Tuning Session

When tuning, you focus on specific areas that offer the greatest return for your tuning effort. The steps are generic and apply to any performance-monitoring tool. Automatic Database Diagnostics Monitor (ADDM) automates many of the steps in the tuning methodology.

Often a knowledgeable DBA can resolve problems before they become apparent to users. For example, a company knows that it is going to increase the number of users who access the database. The DBA can start planning the modifications that must take place to prevent a systemwide slowdown due to limited resources. This type of proactive tuning requires that the DBA be familiar with all aspects of the database, the user base, and the applications.

The recommended tuning methodology is as follows:

1. Define the problem and state the goal: This is the analysis step. Whether the information source is users, database statistics, or metrics, you should collect accurate, factual data that corresponds to the problem. State the problem in terms that are measurable and directly related to the database operations. As an example, if the run time on the XYZ report is two times the baseline, then the goal becomes: Make the run time on the XYZ report equal to or less than the baseline.

General Tuning Session (continued)

2. Collect current statistics: Examine the host system and database statistics. Collect a full set of operating system and database statistics and compare these to your baseline statistics. The baseline statistics are a set of statistics that were taken when the instance was running acceptably. Examine the differences to determine what has changed on the system. Did the XYZ report change? Did the data change? Is the session that is producing the report waiting on something?
3. Consider common performance errors: From your list of differences in the collected statistics, make a comparison with common performance errors. Determine whether one of these errors has occurred on your system.
4. Build a trial solution: Include a conceptual model in your solution. The purpose of this model is to assist you with the overall picture of the database. You are looking for answers to the following questions:
   - Why is the performance degraded?
   - How can you resolve the problem to meet your goal?
5. Implement and measure the change: After you have developed the trial solution, make the appropriate change. Make only one change at a time. If you make multiple changes at the same time, you will not know which change is effective. If the changes do not solve the problem, you do not know if some changes helped and others hindered. Collect statistics to measure the change.
6. "Did the solution meet the goal?" Compare the current and the baseline sets.
   - "No": If you determine that more tuning is required, return to step 3 and repeat the process.
   - "Yes": If your solution meets the goal, make the current set of statistics the new baseline set. You met your goal. Stop tuning.

Defining the Problem

To discover and define the problem:

- **Listen to user feedback.**
- **Check the alert log and trace files for errors.**
- **Check the parameter file for any diagnostic or inappropriate parameter settings.**
- **Check memory, I/O, and CPU usage. Identify processes with resource usage anomalies.**
- **Identify and tune SQL statements that are heavy consumers of CPU or I/O.**
- **Collect instance and operating system (OS) statistics.**

ORACLE

Defining the Problem

Problems can arise at any time. A proactive DBA watches for problems and corrects them before they are noticed by users. The discovery and definition step in the past has been tedious and sometimes dependent on listening to user feedback. User feedback is important, but is often subjective and not reproducible. In Oracle Database 10*g*, many of the following information sources can be viewed from the Enterprise Manager interface.

- Check the logs for error messages that might give a quick clue to the nature of the problem. Do not overlook system- and application-specific logs.
- Ensure that the initialization parameter settings make sense for the system. Suggest different settings for different times of the day, if it helps.
- Check for CPU and disk queuing, disk utilization, and memory swapping. These are the signs of an overloaded system. If the application is reasonably well tuned, sometimes the best recommendation is to add more hardware.
- Use the available tools, such as Statspack or ADDM, to identify SQL statements in the applications that are consuming the most resources.
- Collect instance and OS statistics. Statspack reports point to components where the greatest waits and the greatest use of resources occur. ADDM goes further by focusing on those components with the greatest potential benefit.

Setting the Priority

Choose the problem that has the greatest impact:

- **Analyze system performance in terms of work done (CPU or service time) versus time spent waiting for work (wait time).**
- **Determine which component consumes the greatest amount of time.**
- **Drill down to tune that component (if appropriate).**

Setting the Priority

Determine which problem to tune first. In the performance reports, you see many statistics; even a well-tuned database shows a set of top wait events. The Oracle server provides a set of wait-event statistics for processes that are idle or waiting. The Oracle server also records CPU utilization for processes that are running. To determine the impact of a particular event, it must be compared with the overall time spent.

Each request to the database server has a response time consisting of a wait time and a service time. The service time is the time spent actively working on the request (CPU time). The wait time is by definition the time waiting for whatever reason. Both service time and wait time may be tuned. To tune the service time, the processing, the SQL, the access path, or the data storage structure has to change. Wait times can be tuned by reducing contention for the resource where the wait is occurring.

Each server process is typically in one of three states:

- Idle: Waiting for something to do (sleeping)
- Running code: Using the CPU or in a run queue
- Waiting (blocked):
  - For some resource to become available
  - For a requested activity to complete

Tuning Methodology: Setting the Priority (Example)

```
Time Model System Stats  DB/Inst: ORCL Snaps: 1-11
-> Ordered by % of DB time desc, Statistic name

Statistic                              Time (s) % of DB time
------------------------------ --------- ---------
sql execute elapsed time             467.0      77.1
DB CPU                               414.2      68.4
parse time elapsed                   200.5      33.1
hard parse elapsed time              109.0      18.0
…
DB time                              605.8
```

ORACLE

Tuning Methodology: Setting the Priority (Example)

You can determine the top-priority tuning tasks by comparing the time spent in various waits and tasks with the overall wait time and service time. Both major tools report this comparison to guide your tuning efforts with the Time Model System Stats. For example, the Statspack report in the slide shows that the Database CPU time is 414.2 seconds. The time spent in user calls is 68.4% of the total DB time. The `sql execute elapsed time` shows 467 seconds; this time includes wait times. Just from this limited view, the wait times for the SQL execution are significant, and would lead you to examine the wait statistics related to the SQL execution.

Further investigation reveals that the waits are inherent in the design of the application and cannot be changed. Repeat the process with the next area, `parse time elapsed`.

The `% of DB time` values indicate a relative impact tuning this area could have. If the `hard parse elapsed time` could be eliminated, then the maximum possible improvement is 109 seconds or 18%. The actual improvement may be much less, depending on the amount of improvement you can get from that area.

Common Tuning Problems

The most common tuning problems:

- SQL statements (application tuning)
- Session management
- Shared pool sizing and contention
- Buffer cache sizing and contention
- Data block contention
- Redo log and redo buffer tuning
- Undo tuning
- I/O issues
- Locking issues

ORACLE

Common Tuning Problems

The most common tuning problem in any Oracle database is SQL statements or application tuning. The SQL problems can originate in the application design—for example, over-normalization and serialized resources.

Note: The following Oracle University course is dedicated to SQL tuning: *Oracle Database 10g: SQL Tuning*.

Session management is also an application-level problem, usually appearing as a large number of connects and disconnects, but it could also include the number of open cursors and cursor caching issues.

Memory issues are high on the list of instance-tuning problems. These issues include sizing of the various parts of the System Global Area (SGA) and contention for memory resources. Data blocks and header blocks can be accessed by only one process at a time. Several processes attempting to access the same index blocks or table header blocks can create contention.

In OLTP applications, the amount of redo and undo generated can create bottlenecks in memory or I/O. In any database, I/O issues, such as database file layout on disk or RAID devices, can be a source of performance problems.

Locking issues are not usually a problem, but when you have locking issues they become very important.

ADDM Tuning Session

An ADDM tuning session has the same procedure as the general tuning session, but it combines steps.

1. **View the ADDM report.**
   a. **Collect current statistics; compare with previous set.**
   b. **Compare to performance issues knowledge base.**
   c. **Define the problem and make recommendations.**
2. **Review recommendations.**
   d. **Build a trial solution.**
3. **Implement the recommendation.**
   e. **Implement and measure the change.**
4. **Review the next ADDM report.**
   f. **Decide: “Did the solution meet the goal?”**

ORACLE

ADDM Tuning Session

Automatic Database Diagnostic Monitor (ADDM) internally follows the steps of the general tuning session. The steps that you follow while using ADDM are shown in the slide. The general steps are shown as substeps to the ADDM steps.

Effective Tuning Goals

Effective tuning goals are:

- **Specific**
- **Measurable**
- **Achievable**

Effective Tuning Goals

The elimination of a defined problem becomes the tuning goal. Goals are also derived from related Service Level Agreements (SLAs). An SLA is often a contractual or a business requirement that must be met. The goal may be based on an SLA or a problem. The SLA says that user response time to a particular request must be no more than 30 seconds. The problem is that the average response time is 25 seconds and increasing. A tuning goal is that user response time to a particular request should be 20 seconds.

Both tuning goals and SLAs must have three characteristics to be effective. They must be:

- Specific
- Measurable
- Achievable

"Make the instance run as fast as possible" is not specific. A specific goal would be the following: "The month-end report suite must complete in less than four hours."

Effective Tuning Goals (continued)

A measurable goal has objective quantities that can be measured. There is no doubt whether the goal is being met when it is measurable. A goal that is specific is easily made measurable as well. The goal of “User response time to a request is 10 seconds” is easily stated, but is this for all user requests? Is it the average response time? How do you measure average response time? Having specific definitions for the words of your goal is essential. By restating the goal as “User response time to a particular request is 20 seconds or less,” you can determine objectively when the goal has been met.

Achievable goals are possible and within the control of the individuals who are responsible for tuning.

The following are examples of unachievable goals within constraints of the typical DBA:

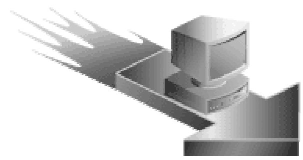
- When the goal is to tune the instance to create a high-performance application, but you are not allowed to change the SQL or the data structures, there is a limited amount of tuning that is possible.
- When the goal is to have a response time of one second, but the network latency between the server and the client is two seconds. Without a change to the network, a response time of one second is impossible.

Even these situations are not impossible to change in an absolute sense, but the DBA always has business constraints that limit the amount of money and resources that can be applied to the solution.

You should always establish measurable tuning goals. Without a tuning goal, it is difficult to determine when you have performed enough tuning.

Tuning Objectives

The objectives of tuning are:
- Minimizing response time
- Increasing throughput
- Increasing load capabilities
- Decreasing recovery time



ORACLE

Tuning Objectives

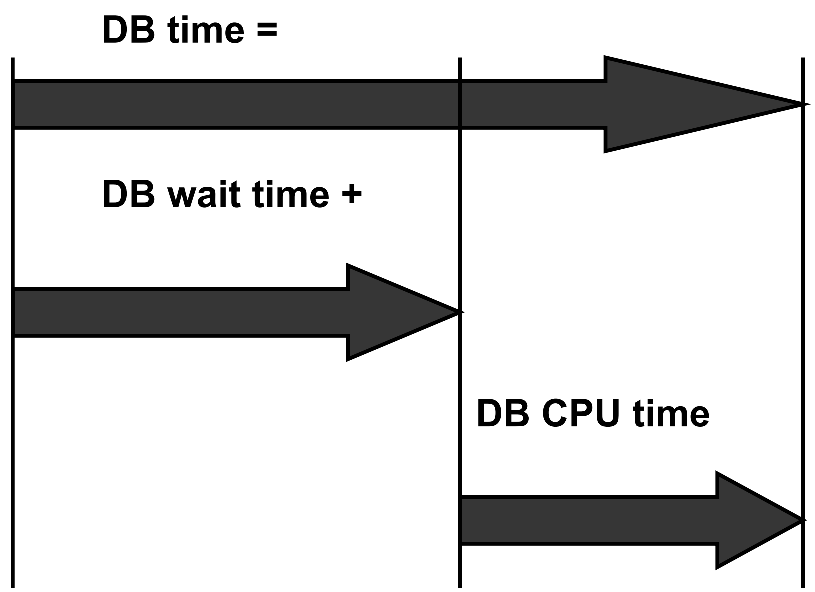
The objectives of tuning goals can be summarized as “Get more done in less time.” Depending on your environment, this translates into:
- Minimizing response time or reducing user wait time
- Increasing throughput (which means decreasing the time to perform a job or set of jobs)
- Increasing load capabilities (which means allowing more tasks or releasing capacity for other tasks)

In some environments, you make trade-offs. In high-volume online transaction processing (OLTP) environments, you may allow longer user response time to get more total transactions from many users. Studies have shown that in a Web-based environment, user response time must be less than seven seconds or the user goes somewhere else. In this case, everything else is subordinate to response time.

Business requirements affect tuning goals. Performance may be limited by safety concerns, as in the goal “Decrease recovery time.”

In a business environment where down time may be measured in hundreds or thousands of dollars per minute, the overhead of protecting the instance from failure and reducing recovery time is more important than the user response time.

DB Time



DB Time

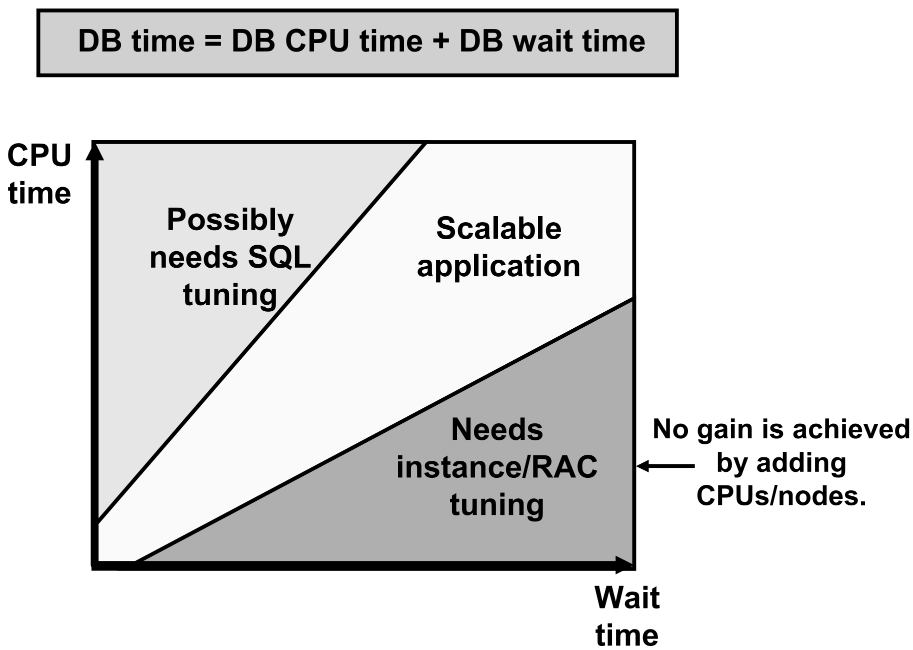
Tuning is not just about reducing waits. Its goal is to improve end-user response time and/or minimize the average resources used by each individual request. Sometimes these go together, and in other cases there is a trade-off (for example, parallel query). In general, we can say that tuning is the avoidance of consuming or holding resources in a wasteful manner.

Any request to the database is composed of two distinct segments: a wait time (DB wait time) and a service time (DB CPU time). The wait time is the sum of all the waits for various resources. The CPU time is the sum of the time that is spent actually working on the request or waiting on the OS run queue. These times are not necessarily composed of one wait and one block of CPU time.

Tuning consists of reducing or eliminating the wait time and reducing the CPU time. This definition applies to any application type: online transaction processing (OLTP) or data warehouse (DW).

Note: A very busy system shows longer DB CPU time due to waits on the run queue. An overloaded system causes processes to wait on the run queue, which inflates the DB CPU time for all processes.

CPU and Wait Time Tuning Dimensions



CPU and Wait Time Tuning Dimensions

When tuning your system, it is important that you compare the CPU time with the wait time of your system. By comparing CPU time with wait time, you can determine how much of the response time is spent on useful work and how much on waiting for resources potentially held by other processes. As a general rule, systems in which CPU time is dominant usually need less tuning than the ones in which wait time is dominant. Alternatively, heavy CPU usage can be caused by badly written SQL statements.

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

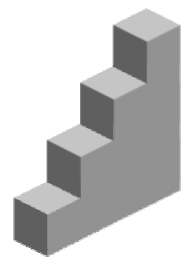
When contention is evidenced by increased wait time, adding more CPUs to a node, or nodes to a cluster, would provide very limited benefit. Conversely, a system in which the proportion of CPU time does not decrease significantly as load increases can scale better and would most likely benefit from adding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

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

Tuning Life Cycle Phases

An application life cycle can be divided into different phases:

- Application design and development
- Testing: Database configuration
- Deployment: Adding a new application to an existing database
- Production: Troubleshooting and tuning



Tuning Life Cycle Phases

Development: Application Design and Programming

Whenever possible, you should start tuning at this level. With a good design, many tuning problems do not occur. For example, although it is typically a good practice to fully normalize tables to reduce redundancy, this can result in a high number of table joins. By denormalizing tables, the performance of the application can improve dramatically.

Testing: Database Configuration

The testing phase is a continuation of development, with more realistic tests that should include production hardware and operating system.

Deployment: Adding a New Application to an Existing Database

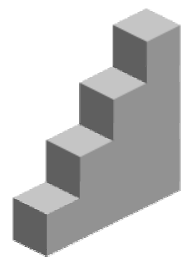
When adding a new application to an existing system, the workload changes. You should accompany any major change in the workload with performance monitoring.

Production: Troubleshooting and Tuning

Use tools to identify bottlenecks. Examine the reports to form a hypothesis about the cause of the bottleneck. From the hypothesis, you can develop and implement a solution. Run a test load against the database to determine whether the solution eliminated the bottleneck.

Tuning Steps During Life Cycle

1. Tune the design.
2. Tune the application.
3. Tune memory.
4. Tune I/O.
5. Tune contention.
6. Tune the operating system.



Tuning Steps During Life Cycle

The tuning methodology that you follow during the development of a new system is the same methodology that you follow for a production system. For a new system, there can be many unknown factors; therefore, you should carefully follow the sequence of steps in the slide. A contention problem may have its roots in the design where multiple processes are serializing on a single resource such as a sequence number generator. Fixing the design is the best way to fix the problem.

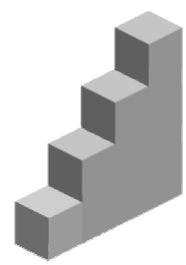
The rationale for this structure is that improvements early in the sequence can save you from having to deal with problems later. For example, if your application uses many full table scans, this may show up as excessive I/O. However, there is no point in resizing the buffer cache or redistributing disk files if you can rewrite the queries so that they access only four blocks instead of 4,000 blocks.

The first two steps are typically the responsibility of the system architects and application developers; however, the DBA is often involved in application tuning. The differences in tuning the later phases of the life cycle are primarily in what is allowed. Many DBAs in a production environment are not allowed to change the SQL or data structures. However, a design change to improve performance may warrant a change request to the application vendor or development team.

Application Design and Development

The application can be tuned even in the design and development phases by building and tuning test cases.

- **Check normalization against major functions.**
- **Check data structures against access times.**
- **Look at points where processes are serialized.**
- **Tune the major reports.**
- **Tune the high-volume processes.**



Application Design and Development

The tuning methodology that you follow during the design and development phases focuses on the common bottlenecks of any system: normalization, data structures, serialization points, major reports, and high-volume requests. Very early in the design, the major functions of the application are known. The level of normalization of the data has serious performance consequences. Over-normalization can lead to large multiway joins that can use all the available host resources. Under-normalization brings another set of problems: inconsistent data, complex data checking, and difficulty migrating data to other schemas or databases. The solution requires a fully normalized design and careful denormalization with built-in consistency checks.

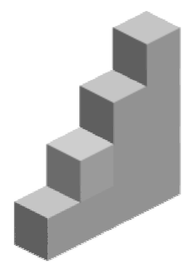
Choosing the proper data structures can provide large performance benefits such as partitioned tables for large data sets. Avoiding resource contention in the design increases its scalability. The major reports required for the application should be prototyped and tuned for expected run times. High-volume functions—in terms of either data or executions—should also be prototyped.

Each of these areas has test cases. These test cases are tuned with the same methodology that is used in the production database: collect statistics, tune the bottleneck, test, and repeat until the goal is met.

Testing: Database Configuration

The testing phase allows tuning at a deeper level:

- **Check physical layout.**
- **Monitor for resource contention.**
  - **Memory utilization**
  - **Locks**
  - **Disk hot spots**
- **Test for resource exhaustion.**



Testing: Database Configuration

The test phase enables you to test at a deeper level. The test case should exercise the application functions, expected loads, and stress tests of improbable loads. These kinds of tests can give valuable insight into the best physical layouts and the best OS and hardware configurations. It is important to monitor hot spots, even on fast disks. You should plan the data configuration to enable a shorter recovery time and faster data access. Incorporate the business requirements for recovery time and availability as much as possible to allow for the overhead of these requirements.

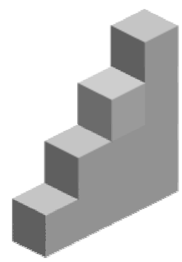
Test with loads that exhaust the machine resources. These tests identify the most used resources. These are the resources that limit the scalability of the system.

The DBA uses the time model at each phase to identify the bottlenecks and tuning sessions to fix the bottlenecks at each level.

Deployment

Deployment of:

- **New application and database**
  - **Take baseline.**
  - **Monitor growth and performance.**
- **New application in existing database**
  - **Take baseline before deployment.**
  - **Take baseline after deployment.**
  - **Compare baselines.**



ORACLE

Deployment

When a new application is initially deployed, the performance expectations often differ from reality. There are two variations to consider here: the first for a new application on a new database, and the second for an application added to an existing database.

The new application on a new database has no baseline; tuning is based on current performance. Take regular performance baseline reports. As the application grows in data set size or number of users, compare additional performance reports to the baseline. This enables you to tune before the performance degrades to an unacceptable level.

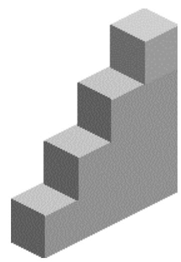
When a new application is added to an existing database, compare baseline performance reports from before and after the application is deployed. These reports show the resources that the new application is using and possible contention for resources with the existing applications.

Production

Tuning is reactive:

What has changed?

Where is the baseline?



Production

Tuning the other phases in the application life cycle is generally proactive. You are looking for possible problems before they are apparent to users.

Tuning a production database is very often reactive tuning. Something has gone wrong: a report that ran in minutes is now taking hours, response time has increased, users are complaining, and backups are not finishing in the allotted time.

Something has changed. Are there more users? Is there a new report or application running? Is there something in the OS that has changed?

Tuning a production system that previously ran acceptably and now has a problem depends on knowing or deducing what has changed. Compare the baseline statistics report taken when the database was running acceptably with a new report taken while the problem is visible. The differences should be visible.

Tuning a production database when there are no baseline statistics is more difficult but possible. The same methodology is used, and the prioritized components are tuned. The time model is used to identify the problem. Possible solutions are considered, starting with design in the top-down steps. A tuning session is then used to test the solutions.

Collecting a Baseline Set of Statistics

A baseline set of statistics is used to:

- **Provide a set of statistics that are collected when the system is operating within the bounds set**
- **Compare the baseline statistics with current statistics**
- **Create a hypothesis about what has changed on the system**

Collecting a Baseline Set of Statistics

Every production database should have at least one baseline set of statistics stored for reference, in case the database performance degrades. Using this set of statistics, you can determine whether degradation has in fact occurred or whether it is the users' perception that has changed.

If the degradation is a fact, then you can determine which area of the database has changed and focus the tuning effort there. You can create a hypothesis about what is the cause of the problem and how to resolve it.

After you have developed the hypothesis, you test it against the system and collect a new set of statistics. You compare the new set of statistics to the baseline and determine whether an improvement has occurred.

If the new set of statistics fulfills the performance goals that were set for the system, you use them as the new baseline. You should keep previous baseline sets to provide a long-term view of changes on the system.

The way you collect statistics and create baselines is different depending on the tools you use. In addition to collecting statistics, Automatic Database Diagnostic Monitor (ADDM) analyzes the differences and recommends solutions.

Performance Versus Safety Trade-Offs

Factors that affect performance:

- **Multiple control files**
- **Multiple redo log members in a group**
- **Frequent checkpointing**
- **Backing up data files**
- **Performing archiving**
- **Block check sums**
- **Number of concurrent users and transactions**

Performance Versus Safety Trade-Offs

There is always a trade-off between performance and safety. When you increase the database performance, there can be an adverse effect on safety. Conversely, the safer you make the database, the slower it runs.

Oracle recommends that there are at least two control files, and one is required. Many DBAs use three or four. More control files require more writes and thus more overhead. Multiple redo log members reduce the chance of loss of data due to a disk failure but increase the overhead of writing redo. Frequent checkpoints reduce the mean time to recovery but can increase the number of physical writes. Each item listed in the slide has a performance cost associated with it.

You must decide how safe to make your database and what level of performance you require, and then configure and tune your database accordingly. The safety requirements of the database are often determined by business needs. The uptime requirement, the mean time to recovery, and the amount of data that could be lost in a disk or system crash are all safety issues. In general, you should address the safety issues and then tune the database for performance with the safety requirements in place.

Filing a Performance Service Request

File a performance service request:

- Is the problem instancewide or query specific?
- Identify the root cause.
- Provide Statspack or AWR reports, plus OS statistics.
- Provide Remote Diagnostics Agent (RDA) reports.
- Provide `SQL_TRACE` reports.

Filing a Performance Service Request

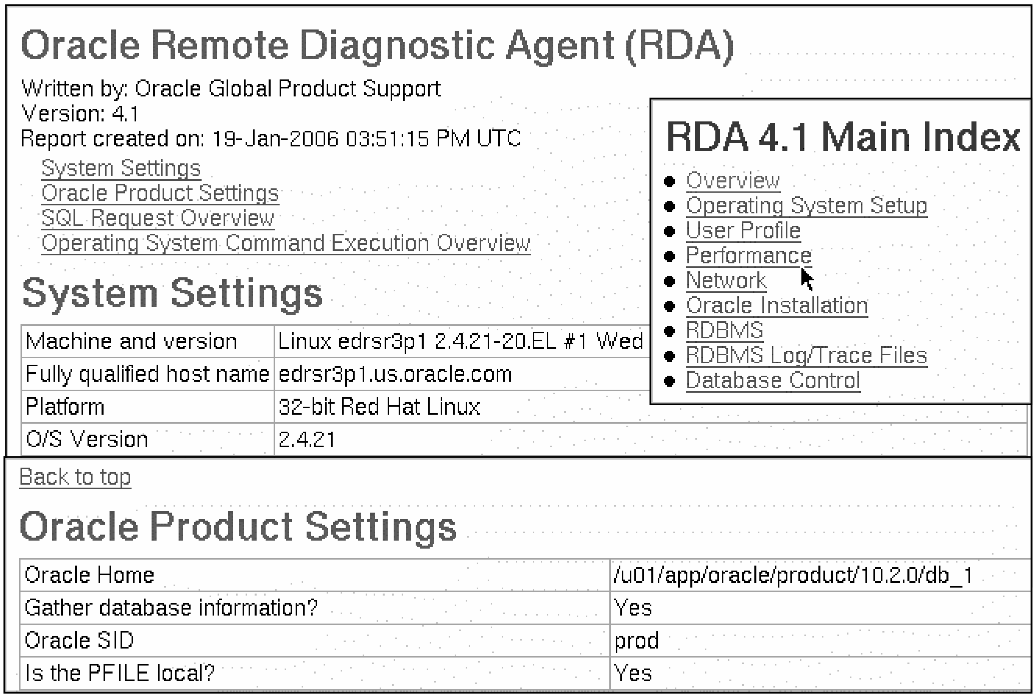
Oracle Support Services (OSS) provides a document, *Note: 210014.1 How to Log a Good Performance Service Request*, to guide you in logging a performance service request.

The performance problems that you encounter are often unique to your application and database configuration. The problem is occasionally something that you cannot tune.

OSS needs certain information:

- Is the problem instancewide or query specific? When does the problem occur? What is working and what is not? Give examples of both acceptable performing SQL and poorly performing SQL.
- What is the root cause? Create Statspack or AWR reports when the performance is “good” and when it is “bad,” and then compare. Check the OS, network, and database log files for clues. Remove recent changes one at a time and record the results; even things that do not change the problem help narrow the search for a root cause. You may not be able to determine the root cause.
- Provide Statspack or AWR reports and OS statistics during the time when the database is exhibiting the problem, and a baseline when the problem does not appear, if possible. Take short-period snapshots when the problem is evident.

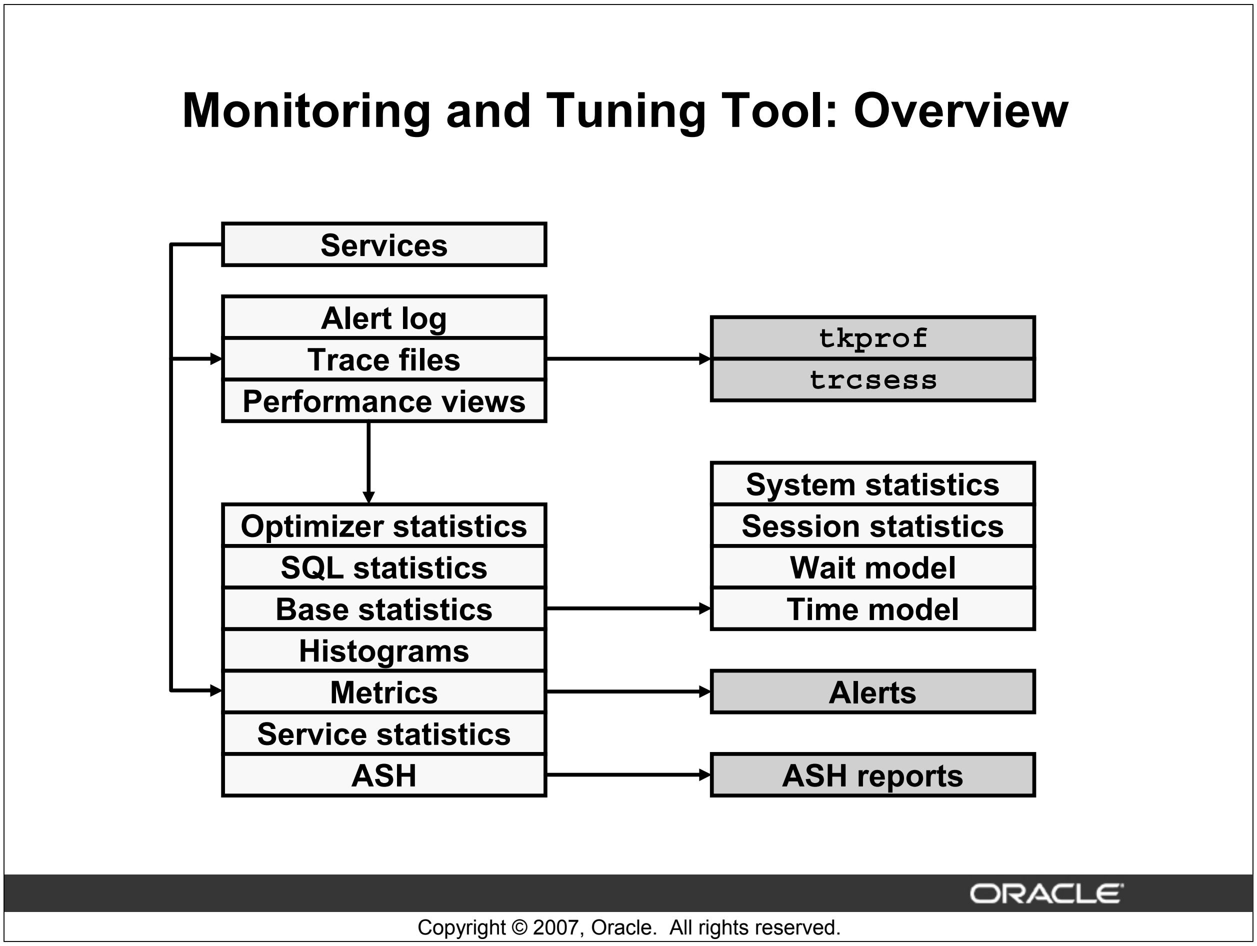
RDA Report



RDA Report

The Remote Diagnostics Agent (RDA) report provides a comprehensive set of information to Oracle Support Services. This report is not required for all service requests, but it is very helpful for performance-related requests and may be requested by the support analyst. RDA is a set of scripts to gather detailed information from an Oracle database environment. The scripts are focused to collect information that aid in problem diagnosis. However, the output is also useful to see the overall system configuration. For more information and download location, see MetaLink Note 330363.1, “Remote Diagnostic Agent (RDA) 4 - FAQ.”

Only a portion of the overview section of the RDA report is shown in the slide. The RDA report is quite large and detailed. It can be easily viewed with common browsers.



Monitoring and Tuning Tool: Overview

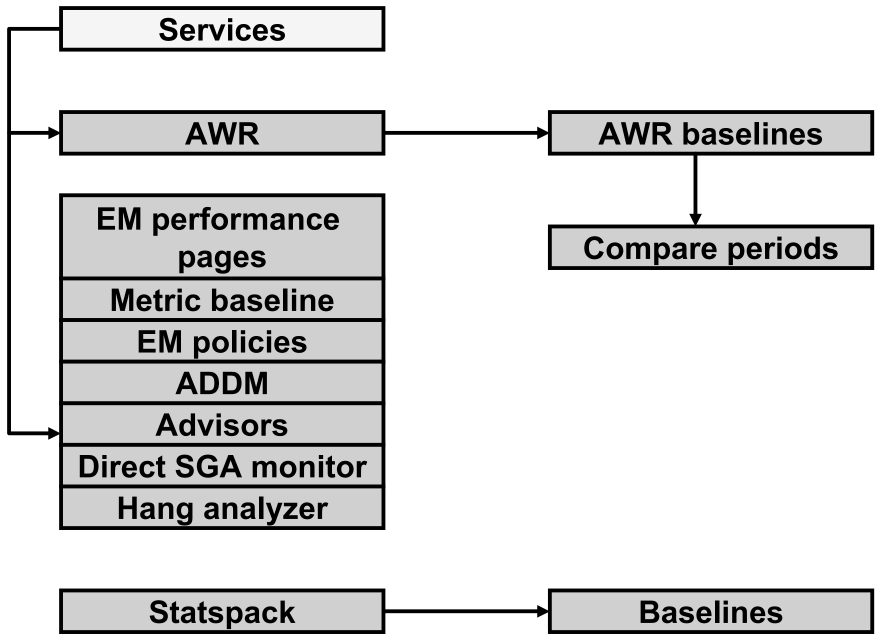
In the following lessons, you use many of the tools listed in the slide. The light-colored boxes indicate tools that contain raw data elements. The darker boxes are tools that use the raw data to derive more useful information. Often the information is formatted in reports such as the Active Session History (ASH) report.

Performance views is another name for the dynamic performance views or `V$` (v-dollar) views that hold raw statistics in memory.

Trace files are very difficult to interpret until they have been formatted with the `tkprof` utility. The `trcsess` utility gives a unique tool for combining and filtering trace files to extract the statistics for a single session, service, or module.

The Services box indicates that the direction of performance monitoring is organized around services. Statistics are being aggregated by services, and several reports can report by services. Statistics gathered by services rather than schema, instance, or session can provide a unique view of application performance.

Monitoring and Tuning Tool: Overview



Monitoring and Tuning Tool: Overview (continued)

You have additional available tools to use in the following lessons. The tools listed in this slide format the data to make it more useful information. Several of the tools analyze the data to provide proactive problem detection and recommendations.

Summary

In this lesson, you should have learned how to:

- Write appropriate tuning goals
- Identify the proper methodology for tuning in different development phases
- Balance performance and safety trade-offs
- Identify common tuning problems
- Log a performance service request with Oracle Support

Practice 2 Overview: Writing Tuning Goals

This practice covers the following topics:

- Transforming a problem definition into a goal
- Making a goal measurable

3 Statistics and Wait Events

Objectives

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

- Identify dynamic performance views that are useful in tuning
- Identify the key tuning components of the alert log file
- Identify the key tuning components of user trace files
- Use dynamic performance views to view statistics and wait events

Performance Tuning Data

Types of data gathered include the following:

- **Cumulative statistics**
  - **Wait events with time information**
  - **Time model**
- **Metrics: Statistic rates**
- **Sampled statistics: Active session history**
  - **Statistics by session**
  - **Statistics by SQL**
  - **Statistics by service**
  - **Other dimensions**

Performance Tuning Data

The Oracle database server software captures information about its own operation. Three major types of data are collected: cumulative statistics, metrics, and sampled statistics.

Cumulative statistics are counts and timing information of a variety of events that occur in the database server. Some are quite important, such as buffer busy waits. Others have little impact on tuning, such as index block split. The events that collect the most time tend to be the most important. The statistics in Oracle Database 10*g* are correlated by the use of a time model. The time model statistics are based on a percentage of DB time, giving them a common basis for comparison.

Metrics are statistic counts per unit. The unit could be time such as seconds, or per transaction, or session. Metrics provide a base to proactively monitor performance. You can set thresholds on a metric causing an alert to be generated. For example, you can set thresholds for when the reads per millisecond exceed a previously recorded peak value or when the archive log area is 95% full.

Sampled statistics are a powerful feature that allow you to look back in time. You can view the statistics that were gathered in the past, in various dimensions, even if you had not thought of specifying data collection for these beforehand.

Performance Tuning Tools

Tools that are available including the following:

- **Basic**
  - **Enterprise Manager pages**
  - **Alert log**
  - **Trace files**
  - **Dynamic performance views and tables**
- **Add-in**
  - **Statspack**
- **Options**
  - **Diagnostics Pack**
  - **Tuning Pack**

Performance Tuning Tools

Oracle database statistics are stored in various tables and views. Some statistics are stored in permanent tables, such as the statistics gathered by `DBMS_STATS` on database objects for use by the optimizer. Many of the statistics used for performance tuning are stored in dynamic tables and views that are memory based. These statistics are not saved when the instance is shut down.

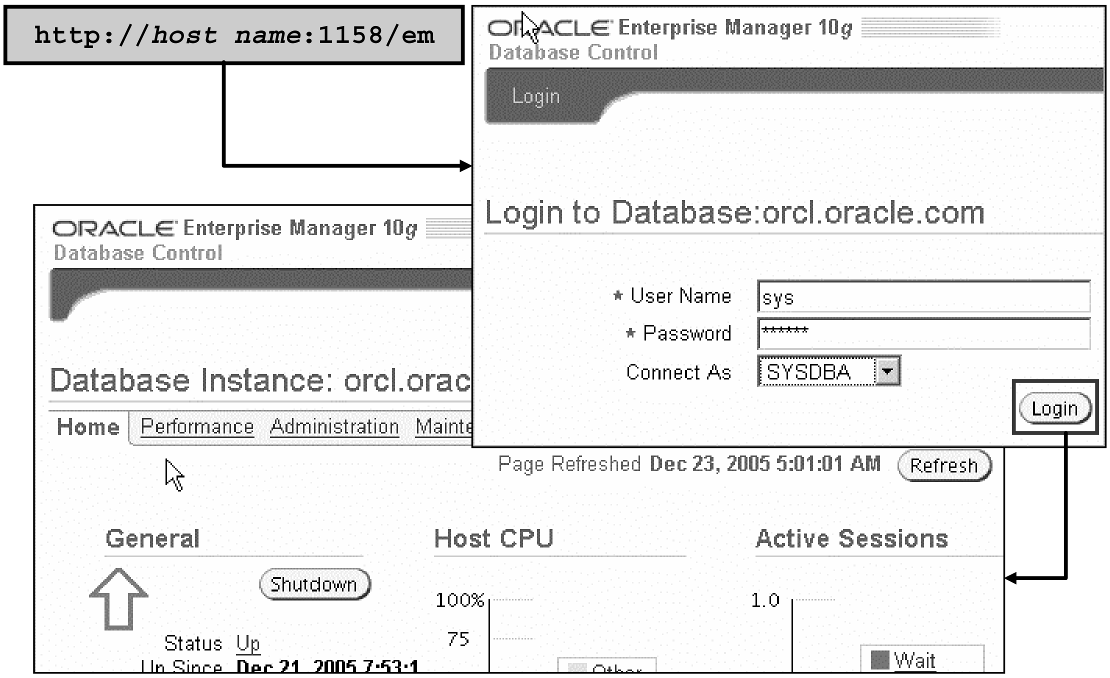
The alert log is a chronological listing of database events and informational messages. The alert log can give important information about the operation of the database, areas that could be tuned, and reference information related to the tuning reports.

Background and user processes create trace files when certain events occur. These trace files can sometimes give you insight into performance problems.

Statspack is a set of procedures and scripts supplied with the Oracle database software. With Statspack, you can collect snapshots of statistics, produce reports comparing snapshots, and create baselines.

Diagnostics Pack is a separately licensed option required for the use of Automatic Workload Repository (AWR) and tools based on AWR. Diagnostics Pack collects statistics, ranks problems, and recommends solutions.

Accessing the Database Home Page



Database Home Page: Overview

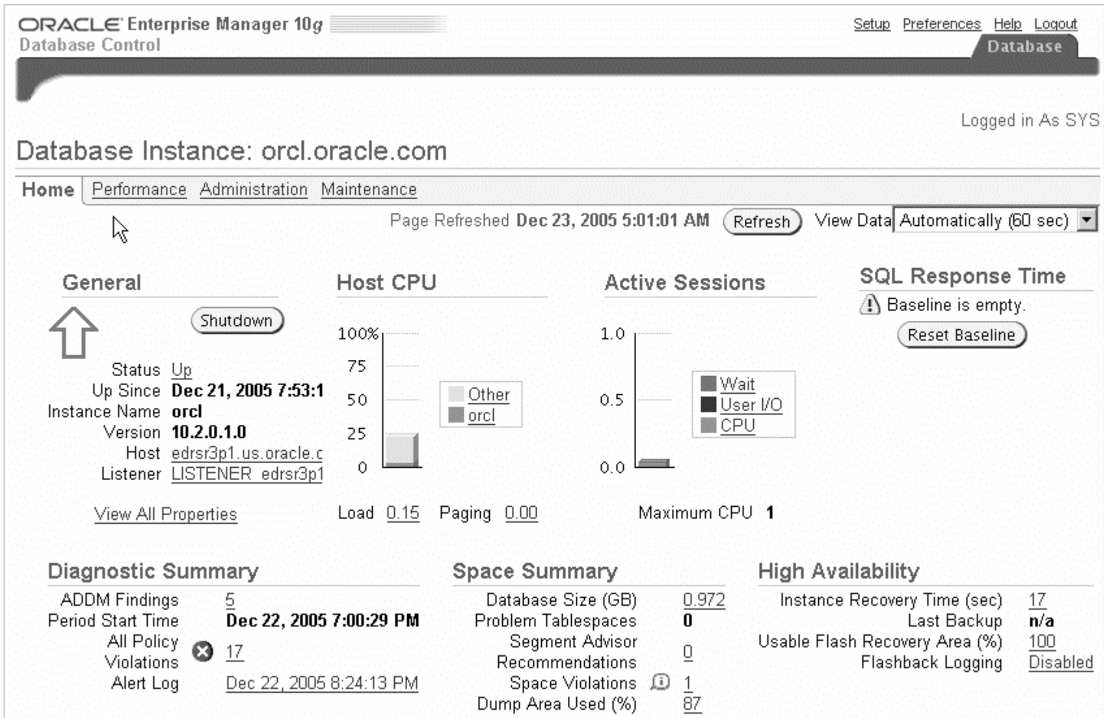
You can access EM Database Control by opening your Web browser and entering the following URL: http://*host name*:*port number*/em.

Host name is the name or address of your computer. *Port number* is the EM Database Control HTTP port number that is specified during installation. The default port is 1158. You can find the value for your system in the `$ORACLE_HOME/install/portlist.ini` file.

The Enterprise Manager Database home page is your starting point to monitor and administer your database. Use the Database Home page to:

- Determine the current status of the database by viewing a series of metrics
- Start or stop the database
- Access the performance, administration, and maintenance of the database environment via three tabs with each page displaying subsections

EM Performance Pages



Enterprise Manager Performance Pages

The Database Home page is the starting point. General database health and performance information is presented on the home page with links and drilldowns to detailed information. Metrics are presented on the Database Home page in the following categories:

- **General:** This category provides a quick view of the status of the database and provides basic information about the database. The status can be Up, Down, Under Blackout, Unmonitored, or Unknown. From this section, you can access other pages for more details, such as the Availability page, the Host Home page, the Listener Home page, and the ASM Home page.
- **Host CPU:** This category displays a bar chart showing relative CPU utilization of the Oracle database host. The 100% indicator (on the chart) represents the total CPU that the host system provides. Two values appear in the bar chart. The darker color at the bottom corresponds to the instance legend and represents how much of the CPU this instance is consuming. The upper, lighter color corresponds to the Other legend and represents all other processes.

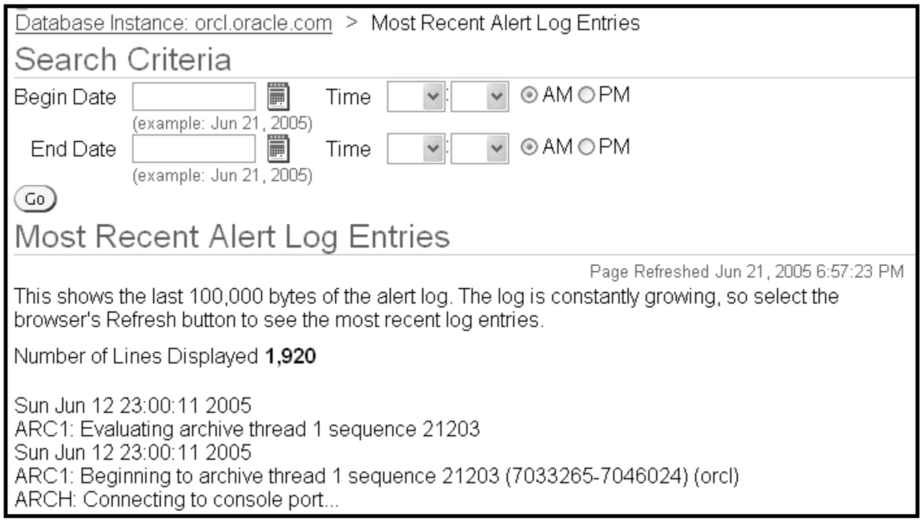
Enterprise Manager Performance Pages (continued)

- **Active Sessions:** The bar chart shows the amount of time the instance consumed using CPU and I/O, and the amount of time it consumed in bottlenecks. The number shown beside the bar chart is a literal number representing the number of active sessions, rather than the total number of sessions. The chart shows the latest value instead of a historical value. The three session categories are always CPU, User I/O, and Wait. The Wait category represents the value for all wait classes combined, excluding User I/O.
- **SQL Response Time:** This category displays the current response of the tracked set of SQL versus the baseline response. If the baseline and response time are equal, the system is running normally. It is possible for the response time to exceed the baseline's response time, which means that one or more SQL statements are performing slower than normal. The lower the response time, the more efficiently the SQL statements execute.
- **Diagnostic Summary:** This category displays information about policy violations, and the latest Automatic Database Diagnostic Monitor (ADDM) finding. The link from Performance Findings takes you to the ADDM page, which provides a performance analysis table with findings that need attention. ADDM uses snapshots of database activity to perform a top-down analysis of your database activity.
- **Space Summary:** Using this category, you can identify storage-related issues and provide recommendations for improved performance.
  The Database Size (GB) number is derived from the Total Size (MB) field at the bottom of the Tablespaces page. For example, if this number is 99,000.0, the number for Database Size (GB) on the home page would be 99.
- **High Availability:** This category displays the time of the last backup for pre–Oracle 10*g* databases, the most recent backup time, and whether or not the backup was successful for Oracle version 10*g* databases. If the last backup failed for a version 10*g* database, the link for Last Backup displays the results of the backup. If the backup was successful, you can drill down to the Manage Current Backups page. Clicking any of the links produces the Configure Recovery Settings page. Instance Recovery Time relates to Current Estimated Mean Time to Recover under Instance Recovery, and Flashback Logging relates to "Enable flashback logging" under Flash Recovery Area.
- **Job Activity:** This category displays a report of the Enterprise Manager job executions, showing the scheduled, running, suspended, and problem executions. If a value other than 0 appears in a field, you can click the number to go to the Job Activity page, where you can view information about all scheduled, currently running, and past jobs.

The Alerts table at the bottom of the page shows all open alerts. To get more information about an alert, click the corresponding message.

Viewing the Alert Log

Database Home page > Related Links region > Alert Log Content



Viewing the Alert Log

Each database has an `alert_`<*sid*>`.log` file. The file is on the server with the database and is stored in the directory specified in the `BACKGROUND_DUMP_DEST` initialization parameter. The alert log file of a database is a chronological log of messages and errors, including the following:

- Any nondefault initialization parameters used at startup
- All internal errors (`ORA-600`), block corruption errors (`ORA-1578`), and deadlock errors (`ORA-60`) that occurred
- Administrative operations, such as the SQL statements `CREATE`, `ALTER`, `DROP DATABASE`, and `TABLESPACE`, and the Enterprise Manager or SQL*Plus statements `STARTUP`, `SHUTDOWN`, `ARCHIVE LOG`, and `RECOVER`
- Several messages and errors relating to the functions of shared server and dispatcher processes
- Errors during the automatic refresh of a materialized view

Enterprise Manager monitors the alert log file and notifies you of critical errors. You can also view the log to see noncritical error and informative messages.

Using Alert Log Information to Aid in Tuning

The alert log file contains the following information that can be used to aid in tuning the database:

- **Incomplete checkpoints**
- **Time to perform archiving**
- **Instance recovery start and complete times**
- **Deadlock and timeout errors**
- **Checkpoint start and end times**

Using Alert Log Information to Aid in Tuning

The information listed in the slide and additional information are written to the alert log. The information written to the alert log changes somewhat with each version of the Oracle database. Some values, such as the checkpoint start and end times, are written only when requested. These values are written into the alert log file only if the LOG_CHECKPOINTS_TO_ALERT parameter has been set to TRUE.

The alert log file can grow to an unmanageable size. You can safely delete the alert log while the instance is started, although you should consider making an archived copy of it first. This archived copy could prove valuable if you should have a future problem that requires investigating the history of an instance.

For example, suppose the DBA noticed a change in performance statistics. The DBA finds that an instance parameter has changed since the last baseline. To confirm that the performance change corresponds to the parameter change, the alert log can be searched. The alert log lists all the nondefault parameter settings on each startup and records ALTER SYSTEM commands with a time stamp.

Using Alert Log Information to Aid in Tuning (continued)

The following sample of the alert log shows the startup parameters, the warning message for `FAST_START_MTTR_TARGET`, and a sequence of log file switches showing the time for each:

```
Starting up ORACLE RDBMS Version: 10.2.0.1.0.
System parameters with non-default values:
  processes                = 150
 __shared_pool_size       = 88080384
  __large_pool_size        = 4194304
  __java_pool_size         = 4194304
  __streams_pool_size      = 0
  sga_target               = 285212672
  control_files            =
/u01/app/oracle/oradata/orcl/control01.ctl,
/u01/app/oracle/oradata/orcl/control02.ctl,
/u01/app/oracle/oradata/orcl/control03.ctl
  db_block_size            = 8192
  __db_cache_size          = 184549376
  compatible               = 10.2.0.1.0
  db_file_multiblock_read_count= 16
  db_recovery_file_dest    =
/u01/app/oracle/flash_recovery_area
  db_recovery_file_dest_size= 2147483648
  undo_management          = AUTO
  undo_tablespace          = UNDOTBS1
  remote_login_passwordfile= EXCLUSIVE
  db_domain                = oracle.com
  dispatchers              = (PROTOCOL=TCP)
(SERVICE=orclXDB)
  job_queue_processes      = 10
…
Tue Dec 20 01:12:16 2005
MTTR advisory is disabled because FAST_START_MTTR_TARGET
is not set
…
Tue Dec 20 01:13:42 2005
Thread 1 advanced to log sequence 6
  Current log# 2 seq# 6 mem# 0:
/u01/app/oracle/oradata/orcl/redo02.log
Tue Dec 20 01:16:32 2005
Thread 1 advanced to log sequence 7
  Current log# 3 seq# 7 mem# 0:
/u01/app/oracle/oradata/orcl/redo03.log
Tue Dec 20 01:43:46 2005
Thread 1 advanced to log sequence 8
  Current log# 1 seq# 8 mem# 0:
/u01/app/oracle/oradata/orcl/redo01.log
…
```

User Trace Files

- **Server-process tracing can be enabled or disabled at the session or instance level.**
- **A user trace file contains statistics for traced SQL statements in that session.**
- **User trace files are created on a per server process basis.**
- **User trace files can also be created by:**
  - **Performing a `BACKUP CONTROL FILE TO TRACE`**
  - **Process errors**

User Trace Files

Server processes can generate user trace files at the request of the user or DBA.

Instance-Level Tracing

This trace logging is enabled or disabled by the `SQL_TRACE` initialization parameter. The default value is `FALSE`.

Session-Level Tracing

The following statement enables the writing to a trace file for a particular session:

```
EXECUTE dbms_monitor.session_trace_enable (8,12,
waits=>TRUE, binds=>TRUE);
```

In the syntax, `8` and `12` are the system identifier and serial number of the connected user.

The `DBMS_MONITOR` package is created when the `catproc.sql` script is run. This script is located in the following directory:

- On UNIX: `$ORACLE_HOME/rdbms/admin`
- On Windows: `%ORACLE_HOME%\rdbms\admin`

To enable the writing of a trace file for your current session, execute the following command:

```
ALTER SESSION SET sql_trace=TRUE;
```

Background Process Trace Files

- The Oracle database server dumps information about errors detected by any background process into trace files.
- Oracle Support uses these trace files to diagnose and troubleshoot.
- These files do not usually contain tuning information.

ORACLE

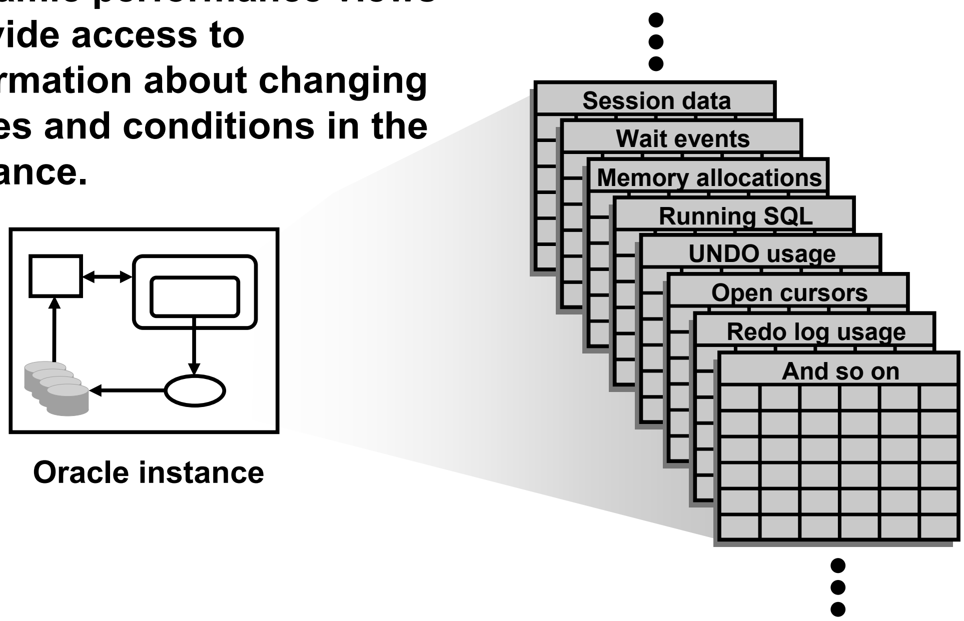
Background Processes Trace Files

The background processes create these files. In general, these files contain diagnostic information rather than information regarding performance tuning. However, by using events, information regarding performance can be written to these files.

Database events can be set by the DBA, but usually only under the supervision of Oracle Support. These files are difficult to read because they are intended for diagnosis and troubleshooting by Oracle Support, but they can contain valuable information that a DBA can use.

Dynamic Performance Views

Dynamic performance views provide access to information about changing states and conditions in the instance.



Dynamic Performance Views

The Oracle database server maintains a dynamic set of data about the operation and performance of the instance. These dynamic performance views are based on virtual tables that are built from memory structures inside the database server. That is, they are not conventional tables that reside in a database. `V$` views externalize metadata contained in memory structures of an Oracle instance. Some of the `V$` views can show you data before a database is mounted or open. The `V$FIXED_TABLE` view lists all the dynamic views.

Dynamic performance views include information about:

- Sessions
- File states
- Wait events
- Locks
- Backup status
- Memory usage and allocation
- System and session parameters
- SQL execution
- Statistics and metrics

Note: The `DICT` and `DICT_COLUMNS` views also contain the names of these dynamic performance views.

Dynamic Performance Views: Usage Examples

a
```
SQL> SELECT sql_text, executions
  2  FROM v$sqlstats
  3  WHERE cpu_time > 200000;
```

b
```
SQL> SELECT * FROM v$session
  2  WHERE machine = 'EDRSR9P1' and
  3  logon_time > SYSDATE - 1;
```

c
```
SQL> SELECT sid, ctime
  2  FROM v$lock WHERE block > 0;
```

ORACLE

Dynamic Performance Views: Usage Examples

Enterprise Manager uses these views and DBAs can also query these views as needed. The three examples shown in the slide answer the following questions:

a. What are the SQL statements and their associated number of executions in which the CPU time consumed is greater than 200,000 microseconds?
b. What sessions logged in from the EDRSR9P1 computer within the last day?
c. What are the session IDs of any sessions that are currently holding a lock that is blocking another user, and how long has that lock been held? (`block` may be 1 or 0; 1 indicates that this session is the blocker.)

Dynamic performance views hold many of the statistics that are used in performance tuning. Most contain information about a specific component of the instance. In this course, you use dynamic performance views to tune specific components. For a complete list of the various dynamic performance views, refer to *Oracle Database Reference 10g Release 2*.

Dynamic Performance Views: Considerations

- These views are owned by `SYS`.
- Different views are available at different times:
  – The instance has been started.
  – The database is mounted.
  – The database is open.
- You can query `V$FIXED_TABLE` to see all the view names.
- These views are often referred to as "v-dollar views."
- Read consistency is not guaranteed on these views because the data is dynamic.

ORACLE

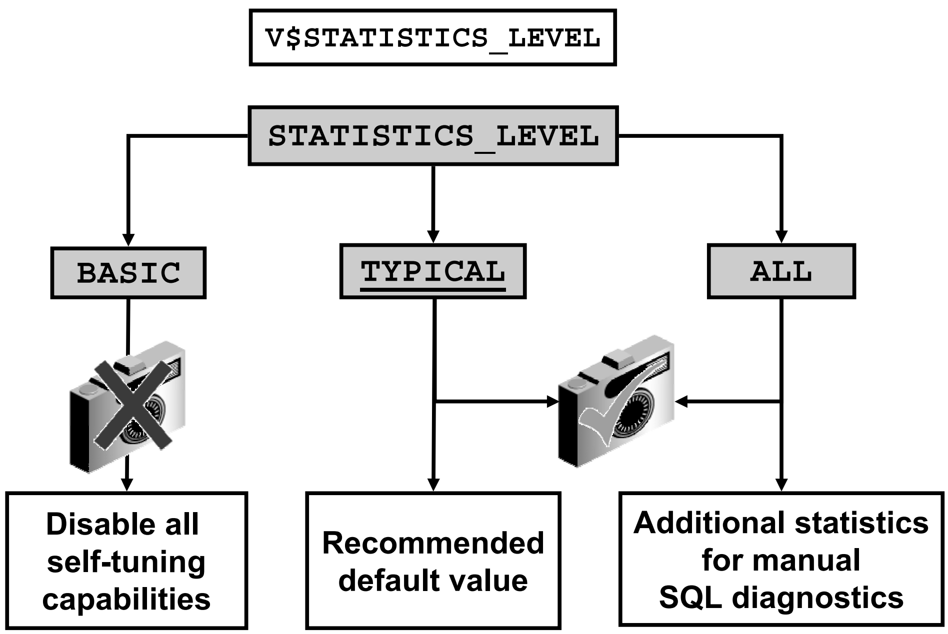
Dynamic Performance Views: Considerations

Some dynamic views contain data that is not applicable to all states of an instance or database. For example, if an instance has just been started but no database is mounted, you can query `V$BGPROCESS` to see the list of background processes that are running. If you query `V$DATAFILE` to see the status of database data files, you receive the error `ORA-01507: database not mounted` because it is the mounting of a database that reads the control file to find out about the data files associated with a database. It is inappropriate to query certain `V$` views at some instance states.

Because these views are based on memory structures, the data in them does not survive a shutdown. Statistics that are recorded before and after a shutdown and startup cannot be compared.

There is no locking mechanism on these views, so the data is not guaranteed to be read consistent. You occasionally see anomalies in the statistics when one or more tables were updated but not all the tables were related to a particular statistic.

Statistic Levels



Statistic Levels

You determine the level of statistic collection on the database by setting the value of the `STATISTICS_LEVEL` parameter. The values for this parameter are:

- **`BASIC`:** No advisory or other statistical data is collected. You can manually set other statistic collection parameters such as `TIMED_STATISTICS` and `DB_CACHE_ADVICE`. Many of the statistics required for a performance baseline are not collected. Oracle Corporation strongly recommends that you do not disable statistic gathering.
- **`TYPICAL`:** This is the default value. Data is collected for segment-level statistics, timed statistics, and all advisories. The value of other statistic collection parameters is overridden.
- **`ALL`:** Collection is made of all the `TYPICAL` level data, the timed operating system statistics, and the row source execution statistics. The value of other statistic collection parameters is overridden.

Query `V$STATISTICS_LEVEL` to determine which other parameters are affected by the `STATISTICAL_LEVEL` parameter.

```
SQL> select statistics_name, activation_level
  2  from v$statistics_level
  3  order by 2;
```

Statistic Levels (continued)

```
STATISTICS_NAME                          ACTIVAT
---------------------------------------- -------
Plan Execution Statistics                ALL
Timed OS Statistics                      ALL
Timed Statistics                         TYPICAL
Segment Level Statistics                 TYPICAL
PGA Advice                               TYPICAL
Shared Pool Advice                       TYPICAL
Modification Monitoring                  TYPICAL
Longops Statistics                       TYPICAL
Bind Data Capture                        TYPICAL
Ultrafast Latch Statistics               TYPICAL
Threshold-based Alerts                   TYPICAL
Global Cache Statistics                  TYPICAL
Active Session History                   TYPICAL
Undo Advisor, Alerts and Fast Ramp up    TYPICAL
Buffer Cache Advice                      TYPICAL
MTTR Advice                              TYPICAL
```

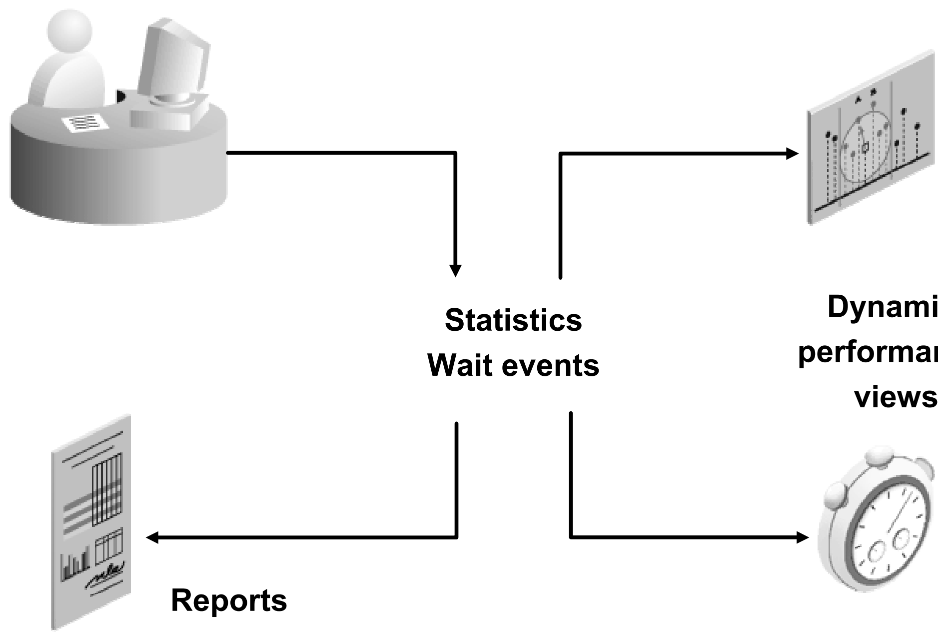
The following statistic parameters can also be set individually.

- `TIMED_STATISTICS`: Is set to `TRUE` to collect time statistics
- `DB_CACHE_ADVICE`: Accepts the following values:
  - `OFF`: No statistics collected and no memory used
  - `READY`: No statistics collected, but memory is allocated. Setting `DB_CACHE_ADVICE` to `READY` prevents memory errors when collecting statistics on buffer cache utilization.
  - `ON`: Statistics collected and memory allocated. Changing the status of `DB_CACHE_ADVICE` from `OFF` to `ON` can raise an error if the required memory is not available.
- `TIMED_OS_STATISTICS`: Specifies the interval (in seconds) at which an Oracle instance collects operating system statistics when a request is made from the client to the server or when a request completes

When `STATISTICS_LEVEL` is modified by `ALTER SESSION`, the following advisories or statistics are turned on or off in the local session only. Their systemwide state is not changed:

- Timed statistics
- Timed OS statistics
- Plan execution statistics

Statistics and Wait Events



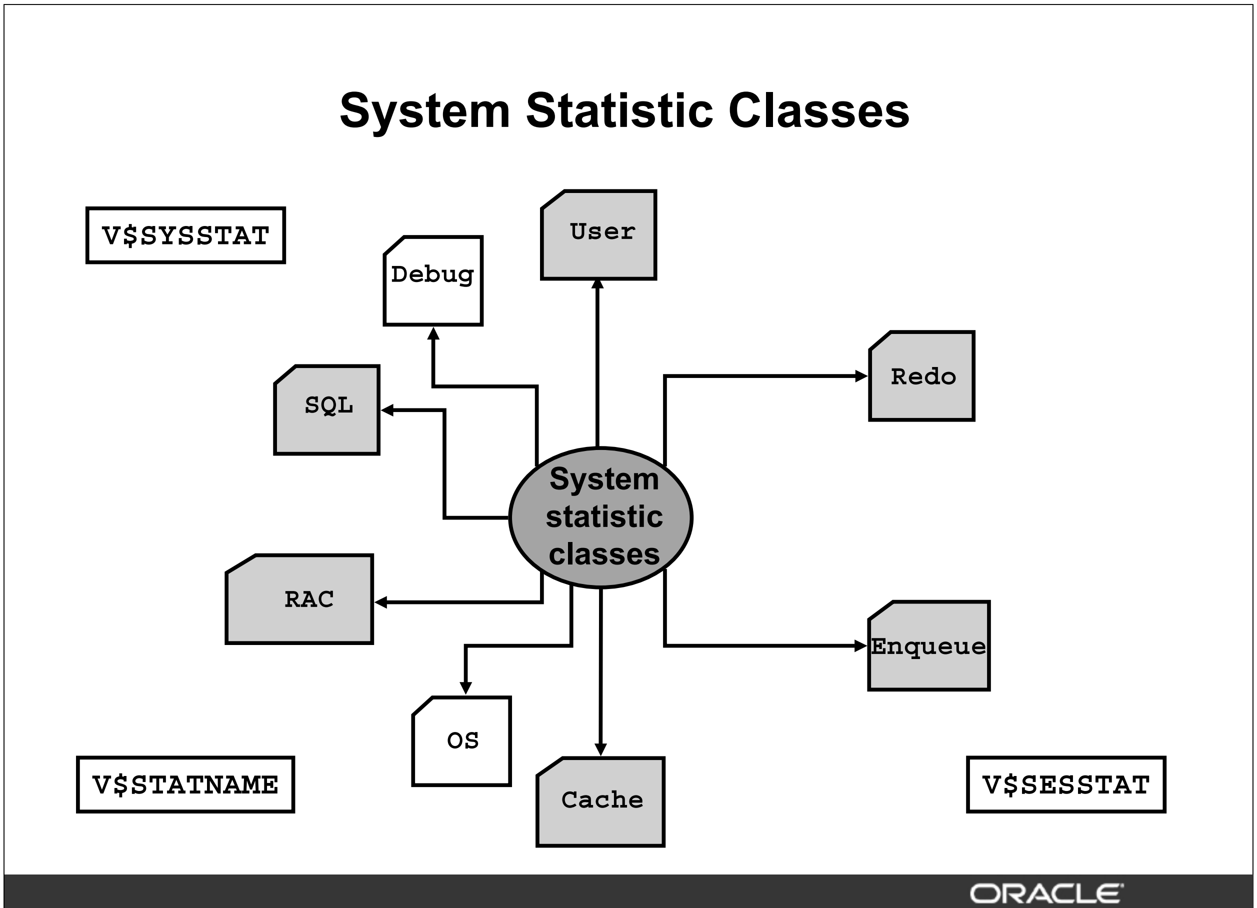
Statistics and Wait Events

Statistics are counters of events that happen in the database. These are indicators of the amount of useful work that is being completed. Statistics include counts such as the number of `user commits` or `db file sequential reads`. Many statistics have corresponding wait events. There are several statistics that can indicate performance problems that do not have corresponding wait events.

Wait events are statistics that are incremented by a server process or thread to indicate that it had to wait for an event to complete before being able to continue processing. Wait event data reveals various symptoms of problems that might be impacting performance, such as latch contention, buffer contention, and I/O contention. Remember that these are only symptoms of problems rather than actual causes.

Most dynamic performance views show the values of the statistics at a point in time (for example, since instance startup).

Statistics and wait events are the raw data that is used to determine what needs to be tuned, and whether the tuning session met the goal. Both Statspack and AWR use statistics and wait events. Statspack makes calculations and reports the derived information. AWR goes a step further and makes recommendations.



System Statistic Classes

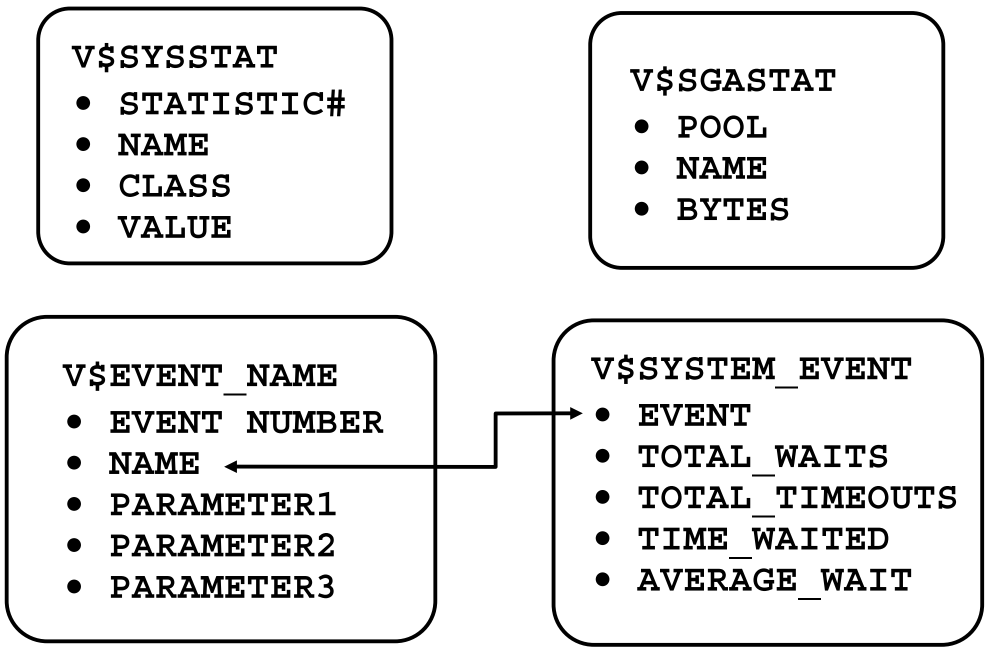
This slide describes the statistic classes stored in the `V$SESSTAT` and `V$SYSSTAT` views. It is necessary to create classes for those statistics because there are more than 360 of them. The `CLASS` column contains a number representing one or more statistic classes. The following class numbers are additive:

- `1, User`
- `2, Redo`
- `4, Enqueue`
- `8, Cache`
- `16, OS`
- `32, Real Application Clusters`
- `64, SQL`
- `128, Debug`

For example, a class value of 72 represents a statistic that relates to SQL statements and caching.

Note: Some statistics are populated only if the `TIMED_STATISTICS` initialization parameter is set to `TRUE`.

Displaying Systemwide Statistics



ORACLE

Displaying Systemwide Statistics

Systemwide Statistics

V$STATNAME displays decoded statistic names for the statistics shown in V$SESSTAT and V$SYSSTAT. The server displays all calculated system statistics in the V$SYSSTAT view. You can query this view to find cumulative totals since the instance started.

Example

```
SQL>  SELECT name, class, value FROM v$sysstat;
NAME                                   CLASS          VALUE
------------------                    ------     ----------
logons cumulative                          1           6393
logons current                             1             10
opened cursors cumulative                  1         101298
table scans (short tables)                64           6943
table scans (long tables)                 64            344
redo entries                               2        1126226
redo size                                  2      816992940
```

The results shown are only a partial display of the output.

Displaying Systemwide Statistics (continued)

SGA Global Statistics

All calculated memory statistics are displayed in the V$SGASTAT view. You can query this view to find cumulative totals of detailed SGA usage since the instance started.

Example

```
SQL>  SELECT * FROM v$sgastat;
POOL            NAME                              BYTES
------          ------------------------     ----------
                fixed_sga                         46136
                db_block_buffers                 409600
                log_buffer                       524288
shared pool     free memory                     8341616
shared pool     SYSTEM PARAMETERS                 42496
shared pool     transaction                       64800
shared pool     dictionary cache                 156524
shared pool     library cache                    358660
shared pool     sql area                         551488
......
```

The results shown are only a partial display of the output.

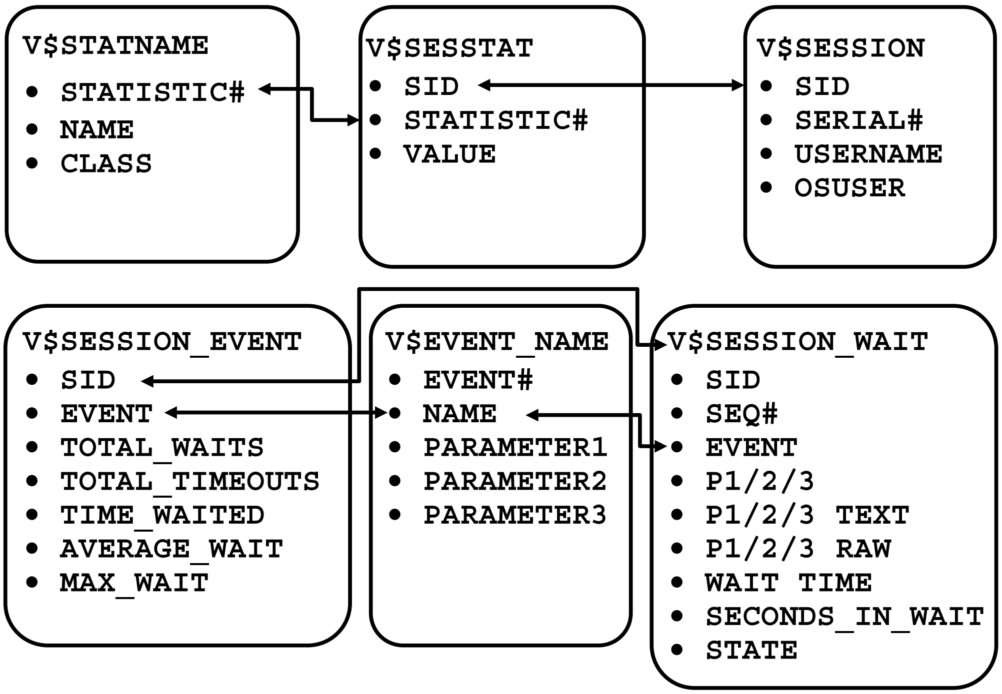
Wait Events Statistics

All kinds of wait events are cataloged in the V$EVENT_NAME view. About 870 events are named.

Cumulative statistics for all sessions are stored in V$SYSTEM_EVENT, which shows the total waits for a particular event since instance startup.

When you are troubleshooting, you need to know if a process has waited for any resource.

Displaying Session-Related Statistics



ORACLE

Displaying Session-Related Statistics

Session data is cumulative from connect time.

You can display current session information for each user who is logged on.

The V$MYSTAT view displays the statistics of the current session.

Example

Determine the type of connection that the users have:

```
SQL> SELECT sid, username, type, server
  2  FROM v$session;
  SID  USERNAME        TYPE            SERVER
-----  ------------    -----------     ----------------
    1                  BACKGROUND      DEDICATED
    2                  BACKGROUND      DEDICATED
    3                  BACKGROUND      DEDICATED
    4                  BACKGROUND      DEDICATED
    5                  BACKGROUND      DEDICATED
    6                  BACKGROUND      DEDICATED
    9  SYSTEM          USER            DEDICATED
```

The Oracle database server displays all calculated session statistics in the V$SESSTAT view. You can query this view to find session cumulative totals since the instance started.

Displaying Session-Related Statistics (continued)

Example

Determine the sessions that consume more than 30,000 bytes of PGA memory.

```
SQL> SELECT username, name, value
  2  FROM v$statname n, v$session s, v$sesstat t
  3  WHERE s.sid=t.sid
  4  AND   n.statistic#=t.statistic#
  5  AND   s.type='USER'
  6  AND   s.username is not null
  7  AND   n.name='session pga memory'
  8  AND   t.value > 30000;

USERNAME        NAME                         VALUE
----------      -------------------    -----------
SYSTEM          session pga memory          468816
```

Session Wait Event Statistics

The V$SESSION_EVENT view shows, by session, the total waits for a particular event since instance startup. The V$SESSION_WAIT view lists the resources or events for which active sessions are waiting. V$SESSION also includes the current wait information.

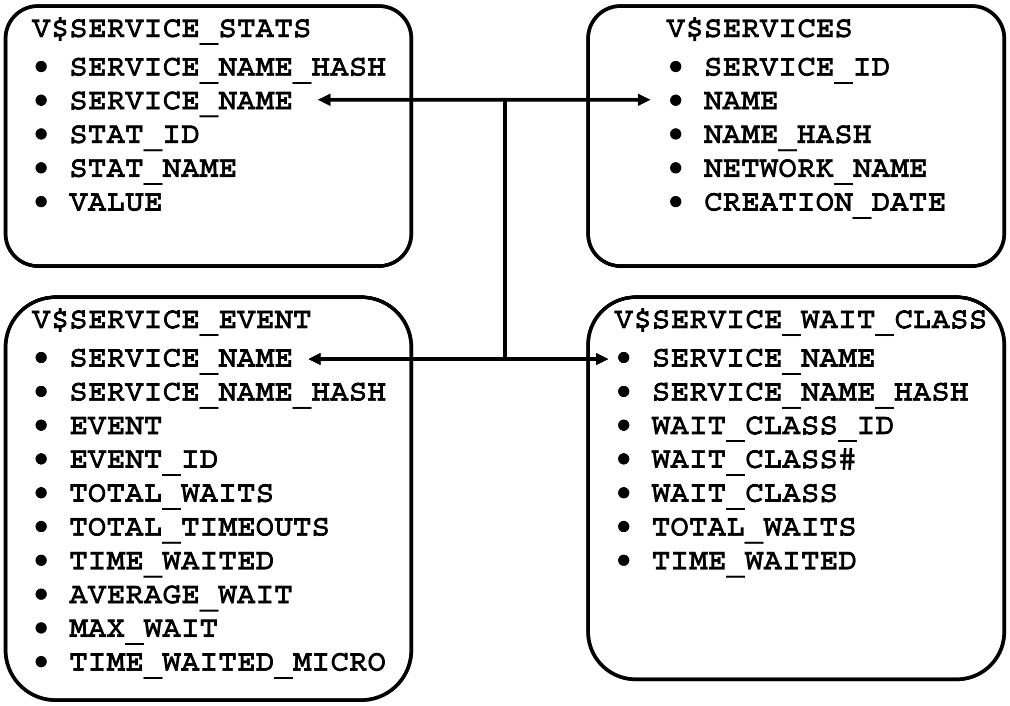
When you are troubleshooting, you need to know whether a process has waited for any resource. The structure of V$SESSION_WAIT makes it easy to check in real time whether any sessions are waiting and why.

```
SQL> SELECT sid, event
  2  FROM v$session_wait
  3  WHERE wait_time = 0;

SID       EVENT
-----     ---------------------------------------
    1     pmon timer
    2     rdbms ipc message
    3     rdbms ipc message
    9     rdbms ipc message
   16     rdbms ipc message
   17     rdbms ipc message
   10     rdbms ipc message
    5     smon timer
    8     rows selected.
```

You can then investigate further to see whether such waits occur frequently and whether they can be correlated with other phenomena, such as the use of particular modules.

Note: The wait events shown above are idle wait class events that always appear. They do not indicate a problem. There are more than 60 such idle events that belong to the wait class “Idle” (wait class number 6).

Displaying Service-Related Statistics



Displaying Service-Related Statistics

Service data is cumulative from instance startup. The service name allows collection of statistics by a connection service name. This is very useful for performance monitoring by application. Every user that connects uses a specific service name for each application.

Example

Determine the names of services that have been defined:

```
SQL> select service_id as id, name from v$services;

        ID NAME
---------- --------------------------------
         5 orclXDB
         6 orcl.oracle.com
         1 SYS$BACKGROUND
         2 SYS$USERS
```

There are always two services defined: `SYS$BACKGROUND` and `SYS$USERS`. Up to 62 additional services can be created based on the `SERVICE_NAMES` parameter or can be set with the `DBMS_SERVICE` package.

Displaying Service-Related Statistics (continued)

The Oracle database server displays all calculated service statistics in the V$SESSTAT view. You can query this view to find service cumulative totals since the instance started.

```
SQL> select service_name, stat_name, value
  2  from v$service_stats;

SERVICE_NAME    STAT_NAME                 VALUE
--------------- -------------------- ----------
SYS$USERS       user calls                 6977
SERV1           user calls                  532
SYS$BACKGROUND  user calls                    0
orcl.oracle.com user calls                18948
orclXDB         user calls                    0
SYS$USERS       DB time                84608280
SERV1           DB time               222965588
SYS$BACKGROUND  DB time                       0
orcl.oracle.com DB time                55877745
orclXDB         DB time                       0
SYS$USERS       DB CPU                 37609828
SERV1           DB CPU                207641916
SYS$BACKGROUND  DB CPU                        0
orcl.oracle.com DB CPU                 35532759
orclXDB         DB CPU                        0
```

Service Wait Event Statistics

The V$SESSION_EVENT view shows, by service, the total waits for a particular event since instance startup. The V$SERVICE_WAIT_CLASS view aggregates the waits by service and wait class.

```
SQL> select service_name, event, average_wait
  2  from v$service_event
  3  where time_waited > 0;

SERVICE_NAME    EVENT                        AVERAGE_WAIT
--------------- ---------------------------- ------------
SERV1           log file sync                           3
SERV1           db file sequential read                 1
orcl.oracle.com log file sync                           1
orcl.oracle.com db file sequential read                 1
orcl.oracle.com db file scattered read                  2
orcl.oracle.com latch: shared pool                      1
orcl.oracle.com latch: library cache                    4
```

Wait Events

- A collection of wait events provides information about the sessions that had to wait or must wait for different reasons.
- These events are listed in the `V$EVENT_NAME` view, which has the following columns:
  - `EVENT#`
  - `NAME`
  - `PARAMETER1`
  - `PARAMETER2`
  - `PARAMETER3`

ORACLE

Wait Events

There are about 870 wait events in the Oracle server, including:

- Free Buffer Wait
- Latch Free
- Buffer Busy Waits
- Db File Sequential Read
- Db File Scattered Read
- Db File Parallel Write
- Undo Segment Tx Slot
- Undo Segment Extension

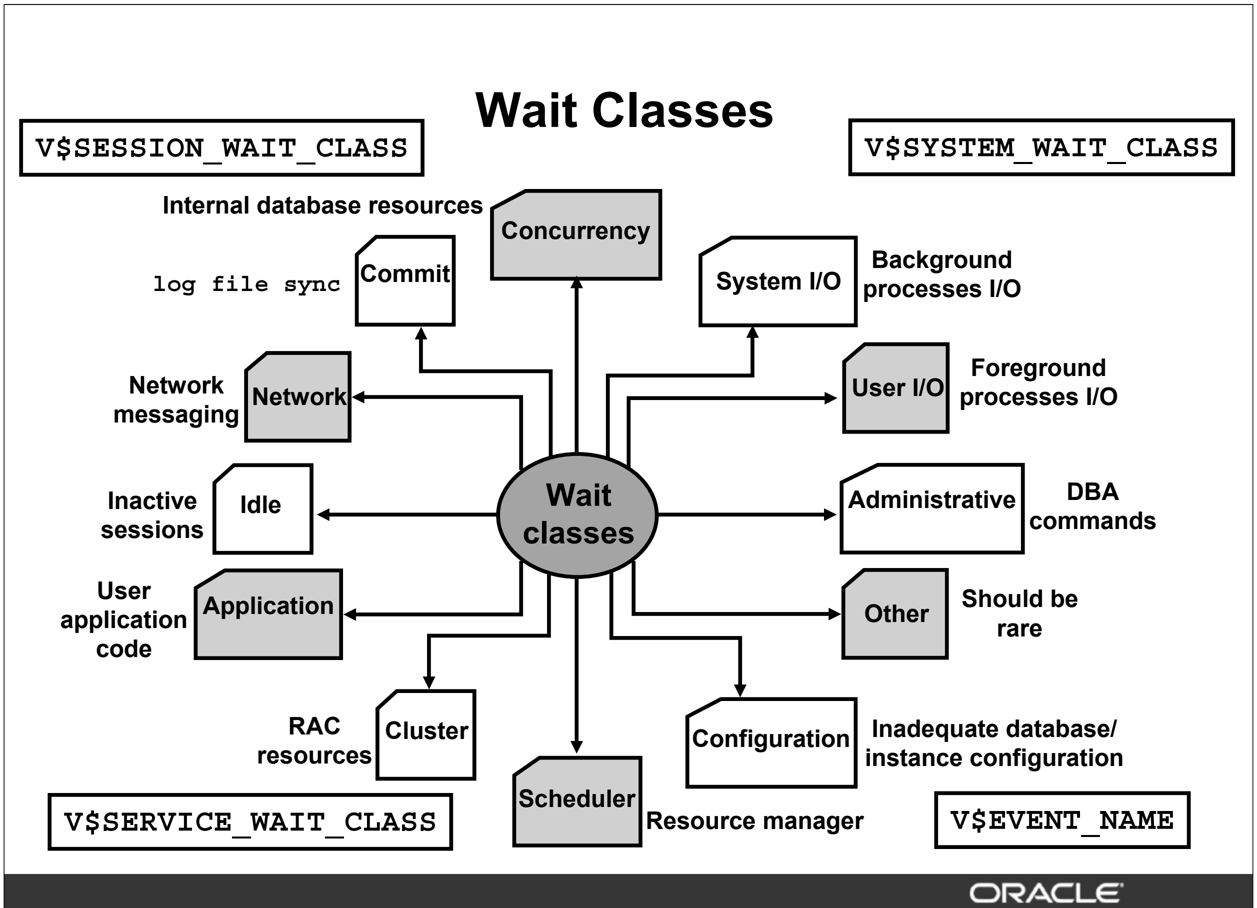
For a description of the common wait events, see the *Oracle Database Reference 10g Release 2*.

Commonly Observed Wait Events

Wait Event	Area
Buffer busy waits	Buffer cache, DBWR
Free buffer waits	Buffer cache, DBWR, I/O
Db file scattered read, Db file sequential read	I/O, SQL Tuning
Enqueue waits (enq:)	Locks
Library cache waits	Latches
Log buffer space	Log buffer I/O
Log file sync	Over-commit, I/O

Commonly Observed Wait Events

The slide shows a list of wait events and component areas that could be the source of these waits.



Wait Classes

The many different wait events that are possible in Oracle Database 10*g* are categorized into wait classes on the basis of the solutions related to that event. This enables high-level analysis of the wait events. For example, exclusive transaction (TX) locks are generally an application-level issue and segment space management (HW) locks are generally a configuration issue. The following are the most commonly occurring wait classes:

- Application: Lock waits caused by row-level locking or explicit lock commands
- Administration: DBA commands that cause other users to wait, such as an index rebuild
- Commit: Waits for redo log write confirmation after a commit
- Concurrency: Concurrent parsing and buffer cache latch and lock contention
- Configuration: Undersized log buffer space, log file sizes, buffer cache size, shared pool size, transaction slot (ITL) allocation, HW enqueue contention, or space allocation (ST) enqueue contention
- User I/O: Waits for blocks to be read off disk
- Network Communications: Waits for data to be sent over the network

Wait Classes (continued)

- Idle: Wait events related to inactive sessions, such as “SQL*Net message from client”

Note: The Other class contains waits that should not typically occur in a system. For example, “wait for EMON to spawn.”

Using the V$EVENT_NAME View

```
SQL> SELECT name, parameter1, parameter2, parameter3
  2  FROM v$event_name;
```

```
NAME                            PARAMETER1  PARAMETER2  PARAMETER3
------------------------------  ----------  ----------  ----------
PL/SQL lock timer               duration
alter system set mts_dispatcher waited
buffer busy waits               file#       block#      id
library cache pin               handle addr pin address
0*mode+name
log buffer space
log file switch (checkpoint incomplete)
transaction                     undo seg#   wrap#       count
...
286 rows selected.
```


Parameters Describing a Wait Event

A wait event may have up to three parameters. The meaning of each of these parameters is listed in the PARAMETER*n* columns.

Buffer Busy Waits Event

The buffer busy waits event records waits for a buffer to become available. This wait indicates that there are some buffers in the buffer cache that multiple processes are attempting to access concurrently.

This event is accompanied by three parameters:

- FILE# and BLOCK#: These parameters identify the block number in the data file that is identified by the file number for the block for which the server needs to wait.
- ID: The buffer busy waits event is called from different places in the session. Each place in the kernel points to a different reason. ID refers to the place in the session calling this event.

Log File Switch (Checkpoint Incomplete) Event

The log file switch (checkpoint incomplete) event records waits for a log switch because the session cannot wrap into the next log. Wrapping cannot be performed because the checkpoint for that log has not completed. This event has no parameter.

Wait Event Statistics

The following views hold session-level wait event statistics:

- **`V$SESSION_EVENT`: Session waits by event for each session that had to wait**
- **`V$SESSION_WAIT`: Session waits by event for current active sessions that are waiting**
- **`V$SYSTEM_EVENT`: Total waits for an event (all sessions together)**

Wait Event Statistics: Statistics for Waiting Sessions

The wait event statistics for sessions that had to wait or are currently waiting for a resource are stored in the `V$SESSION_EVENT` and `V$SESSION_WAIT` views.

Cumulative statistics for all sessions are stored in `V$SYSTEM_EVENT`. This view does not show the same granularity as `V$SESSION_WAIT`.

Using the `db file scattered read` event as an example:

- The `V$SESSION_EVENT` view shows which sessions experience waits for the `db file scattered read` event (only if the session has not logged off the database). However, this view does not give information about which segments were accessed by the session experiencing the wait.
- The `V$SESSION_WAIT` view offers the finest granularity. For the `db file scattered read` event, the view shows the file and the block number that the session is currently waiting for or the last wait event for a session.
- The `V$SYSTEM_EVENT` view informs you that there were waits for the `db file scattered read` event, but does not offer information about which session experienced the waits. This information is calculated from instance startup and represents sessions that may have logged off the database.

Using the V$SESSION_EVENT View

```
SQL> SELECT sid, event, total_waits,average_wait
  2  FROM v$session_event
  3  WHERE sid=10;
```

```
 SID EVENT                         TOTAL_WAITS AVERAGE_WAIT
---- ----------------------------- ----------- ------------
  10 buffer busy waits                      12            5
  10 db file sequential read               129            0
  10 file open                               1            0
  10 SQL*Net message to client              77            0
  10 SQL*Net more data to client             2            0
  10 SQL*Net message from client            76            0
```


Using the V$SESSION_EVENT View

The V$SESSION_EVENT view shows event information totals for individual sessions. It contains the following columns:

- EVENT: Name of the wait event
- TOTAL_WAITS: Total number of waits for the event
- TOTAL_TIMEOUTS: Total number of timeouts for the event
- TIME_WAITED: Total time waited for this event, in hundredths of a second
- AVERAGE_WAIT: Average time waited for this event, in hundredths of a second
- TIME_WAITED_MICRO: Total time waited for this event, in microseconds

The V$SESSION_EVENT view summarizes all session waits for every session listed in V$SESSION. You can join the SID column to V$SESSION.SID to find user details.

Using the V$SESSION_WAIT View

```
SQL> SELECT sid, seq#, event, wait_time, state
  2  FROM v$session_wait;
```

```
SID   SEQ# EVENT                          WAIT   STATE
                                          TIME
--- - --   ---- -------------------------------- -------
  1  1284   pmon timer                        0   WAITING
  2  1697   rdbms ipc message                 0   WAITING
  3   183   rdbms ipc message                 0   WAITING
  4  4688   rdbms ipc message                 0   WAITING
  5   114   smon timer                        0   WAITING
  6    14   SQL*Net message from client      -1   WAITED
                                                  SHORT
                                                  TIME
```


Using the V$SESSION_WAIT View

This view lists the resources or events for which sessions are waiting. This information is also available in the V$SESSION view.

Columns

- SID: Session identifier
- SEQ#: Sequence number identifying the wait
- EVENT: Resource or event waited for. If WAIT_TIME > 0, this is the last event that caused a session wait.
- P1TEXT: Description of the first additional parameter, which corresponds to the PARAMETER1 described for the V$EVENT_NAME view
- P1: First additional parameter value
- P1RAW: First additional parameter value, in hexadecimal
- P2TEXT: Description of the second additional parameter, which corresponds to the PARAMETER2 described for the V$EVENT_NAME view
- P2: Second additional parameter value
- P2RAW: Second additional parameter value in hexadecimal
- P3TEXT: Description of the third additional parameter, which corresponds to the PARAMETER3 described for the V$EVENT_NAME view

Using the V$SESSION_WAIT View (continued)

Columns (continued)

- P3: Third additional parameter value
- P3RAW: Third additional parameter value in hexadecimal
- WAIT_TIME

Value	Explanation
> 0	This is the session's last wait time.
= 0	The session is currently waiting.
= –1	The value is less than 1/100 of a second.
= –2	The system cannot provide timing information.

- SECONDS_IN_WAIT: Number of seconds the event waited
- STATE: Waiting, Waited Unknown Time, Waited Short Time (less than one one-hundredth of a second), or Waited Known Time (the value is stored in the WAIT_TIME column)

Note: Not all of the parameter columns are used for all events.

TIMED_STATISTICS Initialization Parameter

Set the TIMED_STATISTICS parameter to TRUE to retrieve values in the WAIT_TIME column. It is a dynamic initialization parameter.

Using the V$SYSTEM_EVENT View

```
SQL> SELECT event, total_waits, total_timeouts,
  2  time_waited, average_wait
  3  FROM v$system_event;
```

```
EVENT              TOTAL_    TOTAL_   TIME_     AVERAGE_
                    WAITS  TIMEOUTS  WAITED         WAIT
------------------ ------ --------  ------  ----------
latch free              5        5       5           1
pmon timer            932      535  254430  272.993562
process startup         3                8  2.66666667
buffer busy waits      12        0       5           5
...
34 rows selected.
```


Using the V$SYSTEM_EVENT View

The V$SYSTEM_EVENT view shows the total waits for a particular event since instance startup. If you are performing ongoing tuning, you need to know when a process has waited for any resource. Therefore, it becomes useful to query this view each time the system slows down.

The V$SYSTEM_EVENT view contains the following columns:

- EVENT: Name of the wait event
- TOTAL_WAITS: Total number of waits for the event
- TOTAL_TIMEOUTS: Total number of timeouts for the event
- TIME_WAITED: Total time waited for this event, in hundredths of a second
- AVERAGE_WAIT: The average time waited for this event, in hundredths of a second
- TIME_WAITED_MICRO: Total time waited for this event, in microseconds

The output from this view should be ordered by the time waited because this places the most-waited-for events at the top of the list. Remember that this view includes idle events, such as pmon timer, which should be ignored because the process is idle. To collect information about the time an event waits, you must set TIMED_STATISTICS to TRUE or STATISTICS_LEVEL to TYPICAL or ALL.

Precision of System Statistics

- **Views that include microsecond timings:**
  - `V$SESSION_WAIT`, `V$SYSTEM_EVENT`, `V$SERVICE_EVENT`, `V$SESSION_EVENT` (`TIME_WAITED_MICRO` column)
  - `V$SQL`, `V$SQLAREA` (`CPU_TIME`, `ELAPSED_TIME` columns)
  - `V$LATCH`, `V$LATCH_PARENT`, `V$LATCH_CHILDREN` (`WAIT_TIME` column)
  - `V$SQL_WORKAREA`, `V$SQL_WORKAREA_ACTIVE` (`ACTIVE_TIME` column)
- **Views that include millisecond timings:**
  - `V$ENQUEUE_STAT` (`CUM_WAIT_TIME` column)

ORACLE

Precision of System Statistics

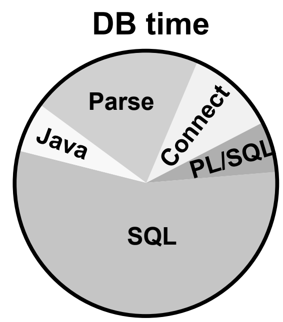
The Oracle database captures certain performance data with millisecond and microsecond granularity. Views that include microsecond and millisecond timings are listed in the slide.

Note: Existing columns in other views capture centisecond times.

The timing information that is gathered on system statistics is cumulative since the instance was started. Some session-level views of statistics record the timing of a single event.

Time Model: Overview

- **The time model is a set of statistics that give an overview of where time is spent inside the Oracle database.**
- **All statistics use the same dimension: time.**
- **The statistics are accessible through:**
  - `V$SYS_TIME_MODEL`
  - `V$SESS_TIME_MODEL`
- **DB time represents the total time spent in database calls.**
- **Tuning goal is to reduce DB time.**
- **Using DB time, you can gauge the performance impact of any entity of the database.**



ORACLE

Time Model: Overview

There are many components involved in tuning an Oracle database system, and each has its own set of statistics. How can you measure the expected benefit from a tuning action on the overall system? For example, would the overall performance improve if you move memory from the buffer cache to the shared pool? When you look at the system as a whole, time is the only common ruler for comparison across components. In the Oracle database server, most of the advisories report their findings in time. There are also statistics called "Time model statistics" that appear as the `V$SYS_TIME_MODEL` and `V$SESS_TIME_MODEL` performance views. This instrumentation helps the Oracle database server identify quantitative effects on the database operations.

The most important of the time model statistics is DB time. This statistic represents the total time spent in database calls and indicates the total instance workload. It is the sum of the CPU and wait times of all sessions not waiting on idle wait events (nonidle user sessions).

The objective for tuning an Oracle database system could be stated as reducing the time that users spend in performing some action on the database, or simply reducing DB time.

Other time model statistics provide quantitative effects (in time) on specific actions, such as logon operations, hard and soft parses, PL/SQL execution, and Java execution.

Time Model Statistics Hierarchy

- DB time
  - DB CPU
  - Connection management call elapsed time
  - Sequence load elapsed time
  - SQL execute elapsed time
    - Repeated bind elapsed time
  - Parse time elapsed
    - Hard parse elapsed time
      - Hard parse (sharing criteria) elapsed time
        - Hard parse (bind mismatch) elapsed time
    - Failed parse elapsed time
      - Failed parse (out of shared memory) elapsed time
  - PL/SQL execution elapsed time
  - Inbound PL/SQL RPC elapsed time
  - PL/SQL compilation elapsed time
  - Java execution elapsed time

- Background elapsed time
  - Background CPU time
    - RMAN CPU time (backup/restore)

Time Model Statistics Hierarchy

The relationships among the time model statistics are listed in the slide. They form two trees: background elapsed time and DB time. All the time reported by a child in the tree is contained within the parent in the tree.

- **DB time:** Amount of elapsed time (in microseconds) spent performing database user-level calls. This does not include the time that is spent on instance background processes such as PMON. DB time is measured cumulatively from the time that the instance was started. Because DB time is calculated by combining the times from all nonidle user sessions, it is possible for the DB time to exceed the actual time elapsed since the instance started up. For example, an instance that has been running for 30 minutes could have four active user sessions whose cumulative DB time is approximately 120 minutes.
- **DB CPU:** Amount of CPU time (in microseconds) spent on database user-level calls
- **Sequence load elapsed time:** Amount of elapsed time spent getting the next sequence number from the data dictionary. If a sequence is cached, this is the amount of time spent replenishing the cache when it runs out. No time is charged when a sequence number is found in the cache. For noncached sequences, some time is charged for every `NEXTVAL` call.

Time Model Statistics Hierarchy (continued)

- Parse time elapsed: Amount of elapsed time spent parsing SQL statements; includes both soft and hard parse time
- Hard parse elapsed time: Amount of elapsed time spent hard-parsing SQL statements
- SQL execute elapsed time: Amount of elapsed time SQL statements are executing. Note that for `SELECT` statements, this also includes the amount of time spent performing fetches of query results.
- Connection management call elapsed time: Amount of elapsed time spent performing session connect and disconnect calls
- Failed parse elapsed time: Amount of time spent performing SQL parses that ultimately fail with some parse error
- Hard parse (sharing criteria) elapsed time: Amount of elapsed time spent performing SQL hard parses when the hard parse resulted from not being able to share an existing cursor in the SQL cache
- Hard parse (bind mismatch) elapsed time: Amount of elapsed time spent performing SQL hard parses when the hard parse resulted from bind type or bind size mismatch with an existing cursor in the SQL cache
- PL/SQL execution elapsed time: Amount of elapsed time spent running the PL/SQL interpreter. This does not include time spent recursively executing or parsing SQL statements or time spent recursively executing the Java Virtual Machine.
- PL/SQL compilation elapsed time: Amount of elapsed time spent running the PL/SQL compiler
- Inbound PL/SQL RPC elapsed time: Time that inbound PL/SQL remote procedure calls (RPCs) have spent executing. It includes all time spent recursively executing SQL and Java, and therefore is not easily related to PL/SQL execution elapsed time.
- Java execution elapsed time: Amount of elapsed time spent running the Java VM. This does not include time spent recursively executing or parsing SQL statements or time spent recursively executing PL/SQL.
- Repeated bind elapsed time: Elapsed time spent on rebinding
- Background CPU time: Amount of CPU time (in microseconds) consumed by database background processes
- Background elapsed time: Total time spent in the database by background sessions (CPU time and nonidle wait time)
- RMAN CPU time (backup/restore): CPU time spent by RMAN backup and restore operations

Summary

In this lesson, you should have learned how to:

- Identify dynamic performance views that are useful in tuning
- Identify key tuning components of the alert log file
- Identify key tuning components of user trace files
- Use dynamic performance views to view statistics and wait events

Practice 3 Overview: Using Basic Tools

This practice covers the following topics:

- Using the alert log information for tuning
- Viewing system statistics and wait events

4

Metrics, Alerts, and Baselines

Objectives

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

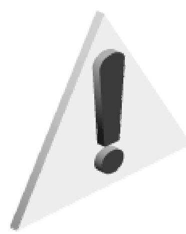
- View metrics by using the metrics history views
- Create metric thresholds
- View alerts
- Create metric baselines
- Enable adaptive thresholds

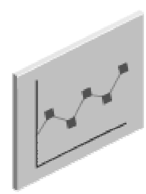
Metrics, Alerts, and Baselines: Definitions

Metric: Rate of change in a cumulative statistic

Alert: Event that is generated when a monitored metric crosses a threshold

Baseline: Data that is gathered from a "normal running database" for performance comparison





ORACLE

Metrics, Alerts, and Baselines: Definitions

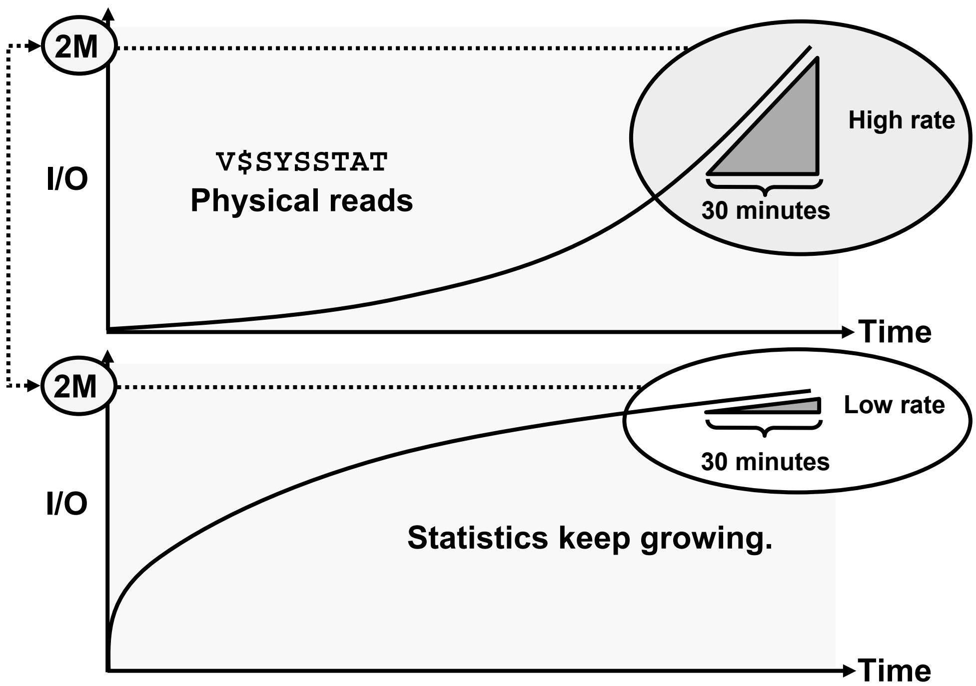
Monitoring for performance requires certain information that goes beyond statistics. To determine whether a particular statistic is important, you need to know how much it has changed over a certain period of time. To be proactive, you need to be notified when certain conditions exist (for example when system response time approaches the agreed maximum). To diagnose performance issues, you need to know what has changed. Metrics, alerts, and baselines provide this information.

A *metric* is a timed rate of change in a cumulative statistic (for example, physical reads per second).

Threshold values can be set for various metrics, and when the value of the metric crosses the threshold value, an *alert* is generated.

A *baseline* comprises stored sets of metrics and statistics. A single set is called a snapshot. A baseline contains two or more snapshots. Usually a baseline is captured during a period of normal or acceptable operation, but it can capture any period of interest. When performance is not as expected, another set of metrics can be captured and compared with the baseline. This method enables the data to clearly point to the performance issues.

Limitation of Base Statistics



Limitation of Base Statistics

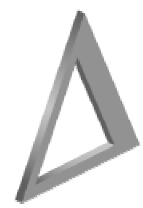
Statistics are counters of events that happen in the database. Statistics and wait events are the raw data. Base statistics are always only a value at a given time.

If you chart the base statistic values over a long period of time, you can see that the values just keep growing. The slide illustrates a possible example of such a graph for the `physical reads` statistic extracted from the `V$SYSSTAT` performance view. As you can see in the slide, although the statistic value is the same for both graphs, the trend is completely different at the end of the observation period. In the upper graph, it is clear that your database is experiencing a much higher rate than in the lower graph. As a result, merely looking at the statistic value is meaningless.

To obtain a better understanding of your database's behavior, you need to be able to see the curves and not just the values. Thus, you need to compute statistic rates over a period of time to determine the trend over that period.

Typical Delta Tools

- **Comparison of statistics at two points in time is needed.**
- **The following tools produce deltas:**
  - **Statspack**
  - **AWR reports**
  - **Customized scripts**



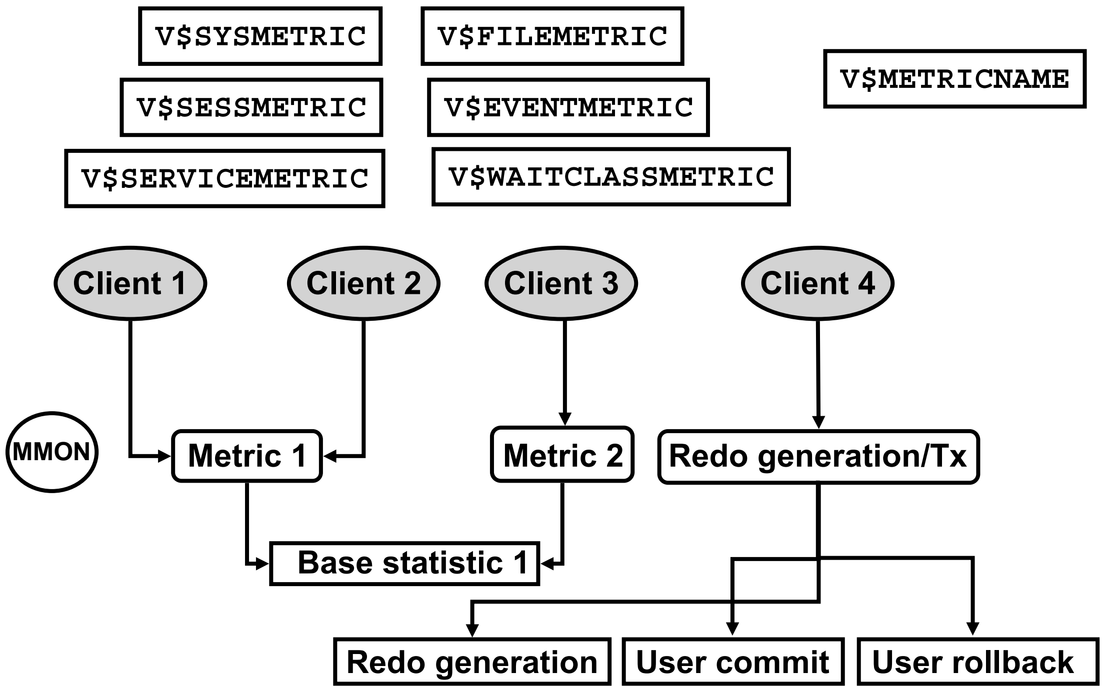
Typical Delta Tools

Base statistics are simply raw numbers that accumulate from the instance startup. A snapshot is a set of statistics captured at a point in time. You can obtain meaningful statistical values by taking snapshots at different times and then computing the difference of the values. The difference is a *delta*.

Several tools can produce a delta. Statspack, Automatic Workload Repository (AWR), and customized scripts can produce reports of the delta over two snapshots.

AWR and Statspack enable you to save snapshot sets for future reference. These saved sets are baselines.

Oracle Database 10*g* Solution: Metrics



Oracle Database 10*g* Solution: Metrics

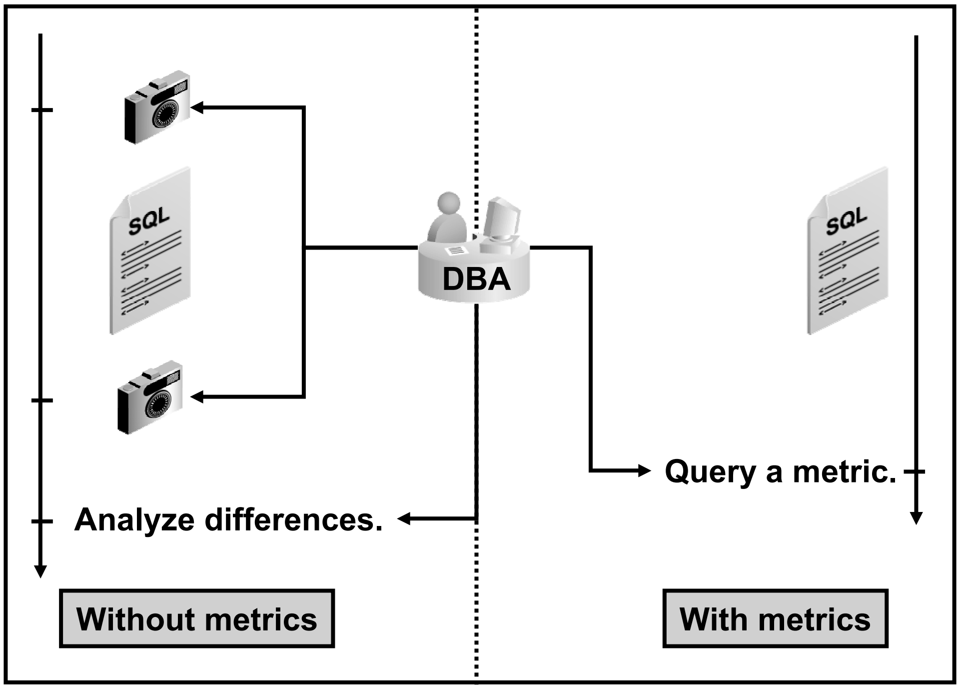
Base statistics are what the Oracle database server collects during normal operations. For example, counting the number of physical reads in the system since startup is a base statistic.

Metrics are secondary statistics that are derived from base statistics. Most metrics track the rates of change of activities in the Oracle server. For example, the average physical reads in the system in the last 60 minutes is a metric. Metrics are used by internal components (clients) for system health monitoring, problem detection, and self-tuning. The Manageability Monitor process (MMON) periodically updates metric data from the corresponding base statistics.

The Oracle database server components use metrics to perform manageability functions. For example, Automatic Database Diagnostics Monitor (ADDM) uses the average physical reads in the system in the last 60 minutes as input. Another component may need a different metric based on the same base statistic: physical reads. For example, the memory advisor may need the physical read counts during peak hours.

Oracle Database 10*g* supports metrics for system, sessions, files, and wait event statistics. Metrics are uniquely identified by a metric number. Each metric is also associated with a metric name. Some of the fixed views that you can access to browse metric data are listed in the diagram. For more information about these views, refer to the *Oracle Database Reference*.

Benefits of Metrics



Benefits of Metrics

The main benefit of keeping metrics is that the data is readily available when a component needs to compute the rate of change of some activity.

In earlier releases, you had to capture statistics before and after running your workload to compute the changed rate for a particular base statistic. With metrics, all you need to do is run your workload and select the corresponding metrics.

Viewing Metric History Information

- **In memory:**
  - `V$SYSMETRIC_HISTORY`
  - `V$FILEMETRIC_HISTORY`
  - `V$WAITCLASSMETRIC_HISTORY`
  - `V$SERVICEMETRIC_HISTORY`
  - `V$SESSION_WAIT_HISTORY` **(last 10 events)**
- **On disk:**
  - `DBA_HIST_SYSMETRIC_SUMMARY`
  - `DBA_HIST_SYSMETRIC_HISTORY` **(alerts only)**
  - `DBA_HIST_SESSMETRIC_HISTORY`
  - `DBA_HIST_SYSTEM_EVENT` **(cumulative)**
  - `DBA_HIST_FILEMETRIC_HISTORY` **(alerts only)**
  - `DBA_HIST_FILESTATXS` **(cumulative)**
  - `DBA_HIST_WAITCLASSMET_HISTORY` **(alert)**

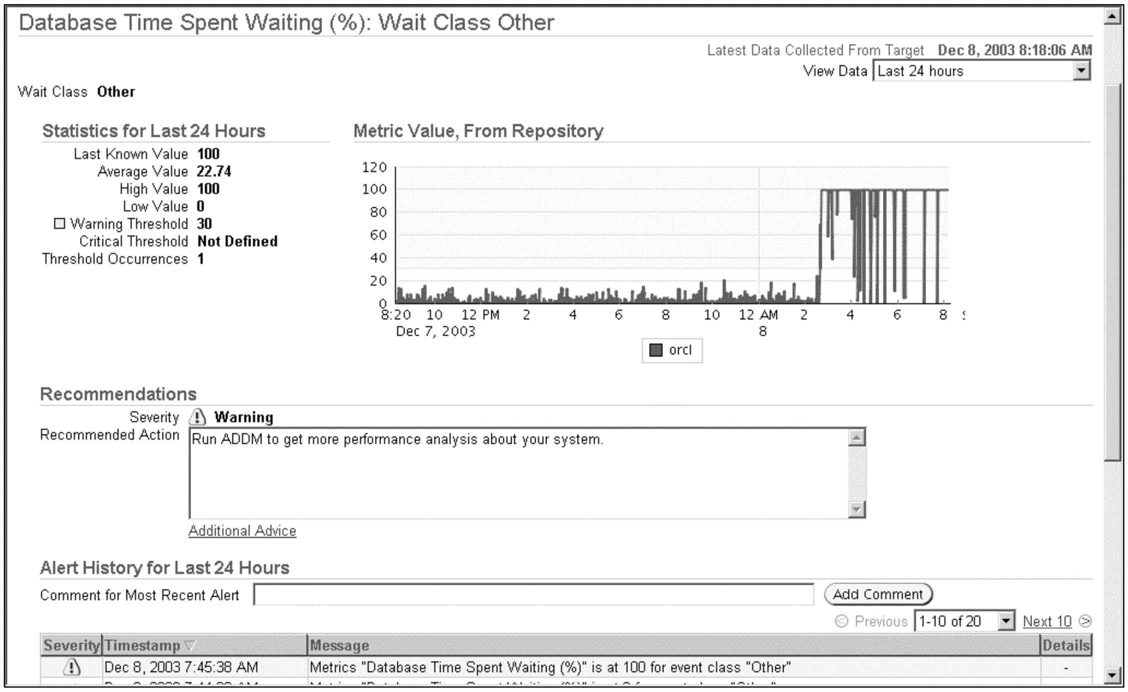
ORACLE

Viewing Metric History Information

Metric values are exposed in some `V$` views where the values are averages over a fairly small time interval (typically 60 seconds). Some of the data also persists in AWR snapshots. The snapshots are kept in the `DBA_HIST` tables.

The slide lists a few of the metric and detail views.

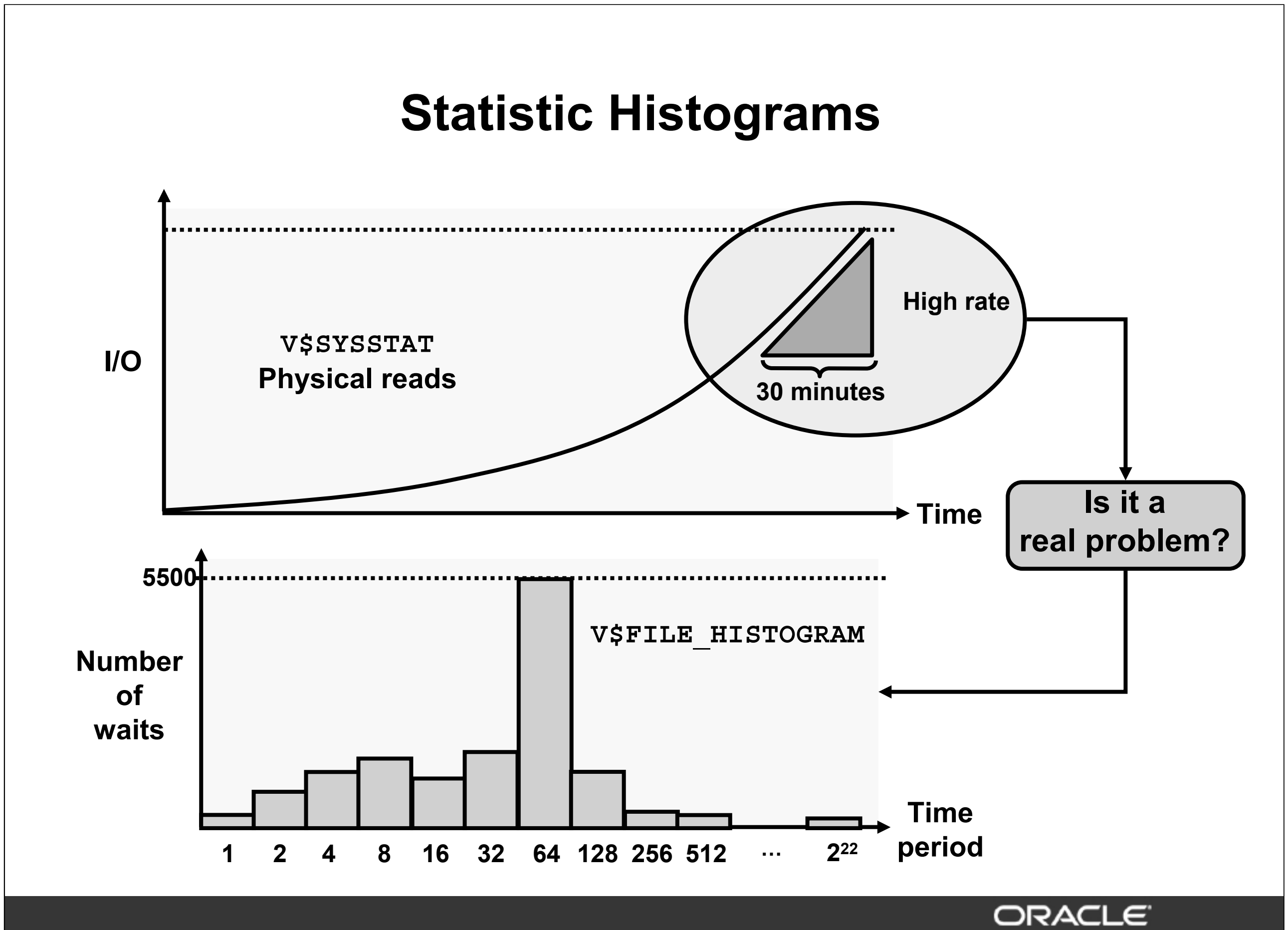
Viewing Detailed Information for a Metric



Viewing Detailed Information for a Metric

Use the All Metrics page to view a list of all the performance metrics available for your database. You can access this page from the Database Control home page by clicking the All Metrics link in the Related Links section.

On the All Metrics page, you can expand all or specific metric groups to see particular metrics. By selecting one metric, you display that metric’s page. You can view the metric’s value over a certain period of time; this view can be customized. The corresponding graphic shows you the metric’s value history.



Statistic Histograms

Although the metrics can give you an idea of the trend for particular statistics, they do not tell you if a particular bottleneck is affecting the whole system or if it is just localized. As an example, you can observe a high metric rate, but this sudden increase could be localized to only one or two sessions in your system. In this case, it might not be worth investigating the issue. However, if the sudden increase is generalized to the whole system, you need to investigate further. This information is available via histogram performance views.

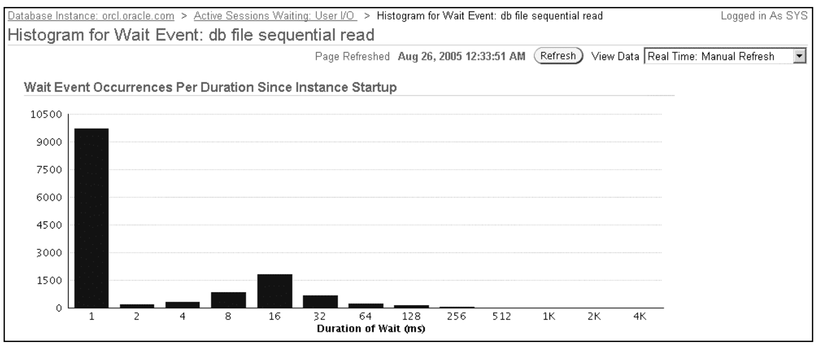
As shown in the slide, you observe a sudden increase in your I/O rate. You can correlate this information to the corresponding I/O histogram found in `V$FILE_HISTOGRAM`. This view displays a histogram of all single block reads on a per-file basis. The histogram has buckets of time intervals, measured in milliseconds, from 1 ms up to 2^{22} ms (69.9 minutes). The value in each bucket is the number of times the system waited for that amount of time.

For example, you can see from the slide that the system was waiting 5,500 times for more than 32 ms and less than 64 ms to read blocks from disks. This is certainly a cause of concern for your system if the access times are normally less than 10 ms. You should thus investigate this further. Had you seen large numbers in shorter wait time periods, you would not have worried much.

The metrics can alert you to potential problems. By drilling down using histograms, you can clearly determine whether there really is a problem.

Histogram Views

- **V$EVENT_HISTOGRAM: For each event such as “db file sequential read”**
- **V$FILE_HISTOGRAM: For single block reads on a per-data-file basis**
- **V$TEMP_HISTOGRAM: For single block reads on a per-temp-file basis**



Histogram Views

V$EVENT_HISTOGRAM displays a histogram of the number of waits on an event basis.

V$FILE_HISTOGRAM displays a histogram of all single block reads on a per-file basis.

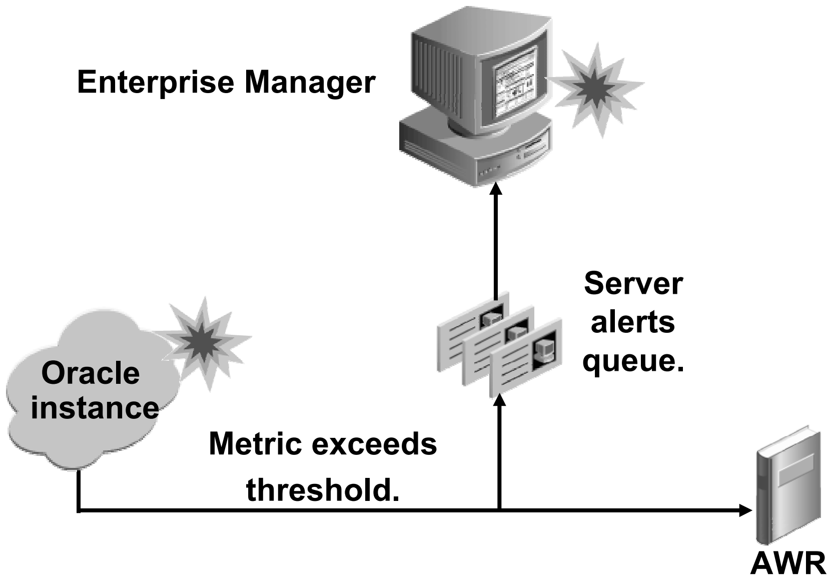
V$TEMP_HISTOGRAM displays a histogram of all single block reads on a per-temp-file basis.

The histogram has buckets of time intervals from < 1 ms, < 2 ms, < 4 ms, < 8 ms, ... < 2^21 ms, < 2^22 ms, >= 2^22 ms.

You can also visualize the histogram statistics from Enterprise Manager. On the Performance page, click one of the legend area items for the Active Sessions graph. For example, click User I/O to drill down to the Active Sessions Waiting:User I/O page. Again, click the wait event name to the right of the Active Sessions graph to reach the corresponding “Histogram for Wait Event” page. The “Histogram for Wait Event: db file sequential read” graph is shown in the slide.

Note: The histogram will not be filled unless the TIMED_STATISTICS initialization parameter is set to TRUE, which is the default value. TIMED_STATISTICS is set automatically when the STATISTICS_LEVEL parameter is set to TYPICAL or ALL.

Server-Generated Alerts



Server-Generated Alerts

Alerts are notifications of when a database is in an undesirable state and needs your attention. By default, the Oracle database sends alerts via Enterprise Manager Database Control, where they are displayed. Optionally, Enterprise Manager can be configured to send an e-mail to the administrator about problem conditions. The Oracle database server also keeps a history of the metric and the alerts in the workload repository.

The alert queue is a persistent multiconsumer queue and is available for users who want to write customized alert handlers.

Thresholds on several key metrics, such as Tablespace Used (%), are set by default. You can set thresholds on the pertinent metrics for your system. If the database deviates from normal readings enough to cross those thresholds, Oracle Database 10*g* proactively notifies you by sending an alert. An early notification of potential problems enables you to respond quickly and often resolve issues before users even notice them.

A few key metrics that can provide early problem notification are the following:

- Average File Read Time (centiseconds)
- Response Time (per transaction)
- SQL Response Time (%)
- Wait Time (%)

Database Control Usage Model

Enable alerts by setting thresholds.

↓

Set up notification rules (paging, e-mail).

↓

Receive notification.

↓

Review alert details and advice.

↓

Correct the problem.

↓

Verify that the problem is resolved.

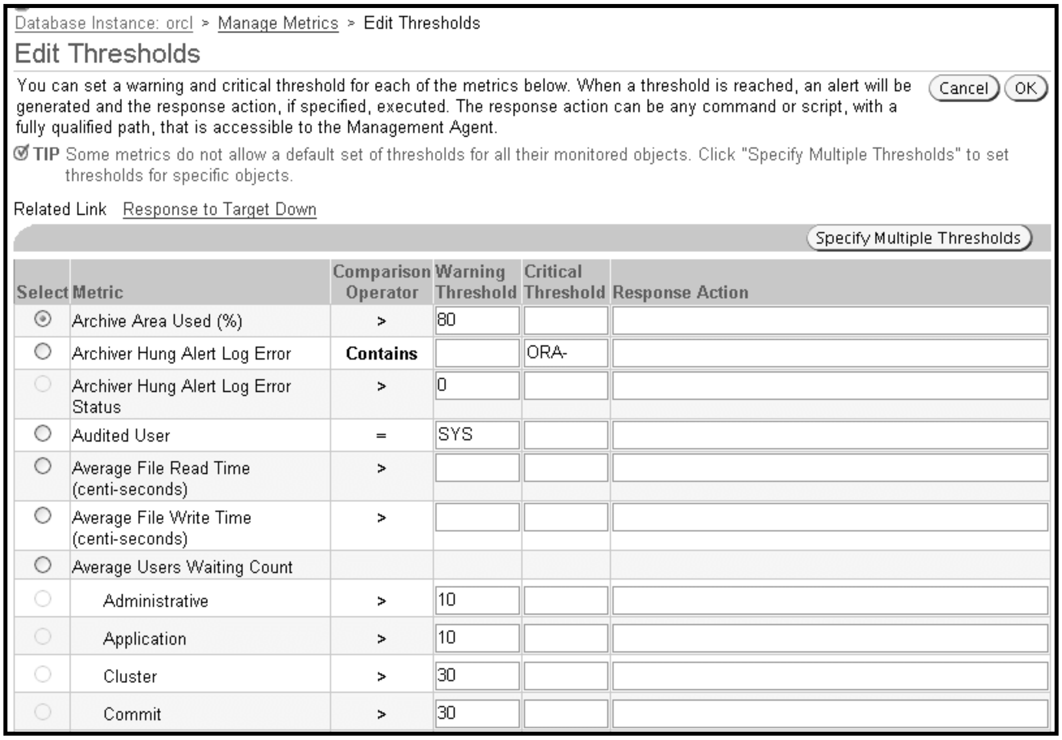
Database Control Usage Model

The following is the basic usage model for server-generated alerts:

- If needed, you can change threshold settings for the server-alert metrics. You can do this by using Database Control or a PL/SQL procedure.
- You set up notification rules (for example, e-mail address or blackout period) by using Database Control.
- When an alert is generated, Database Control displays the alert on the alert pages. Database Control sends out a notification to administrators who have registered for it.
- When you get an alert, you can follow the advice that is given in the alert to correct the problem.

Note: You should ensure that the `STATISTICS_LEVEL` initialization parameter is set to `TYPICAL` or `ALL`.

Setting Thresholds

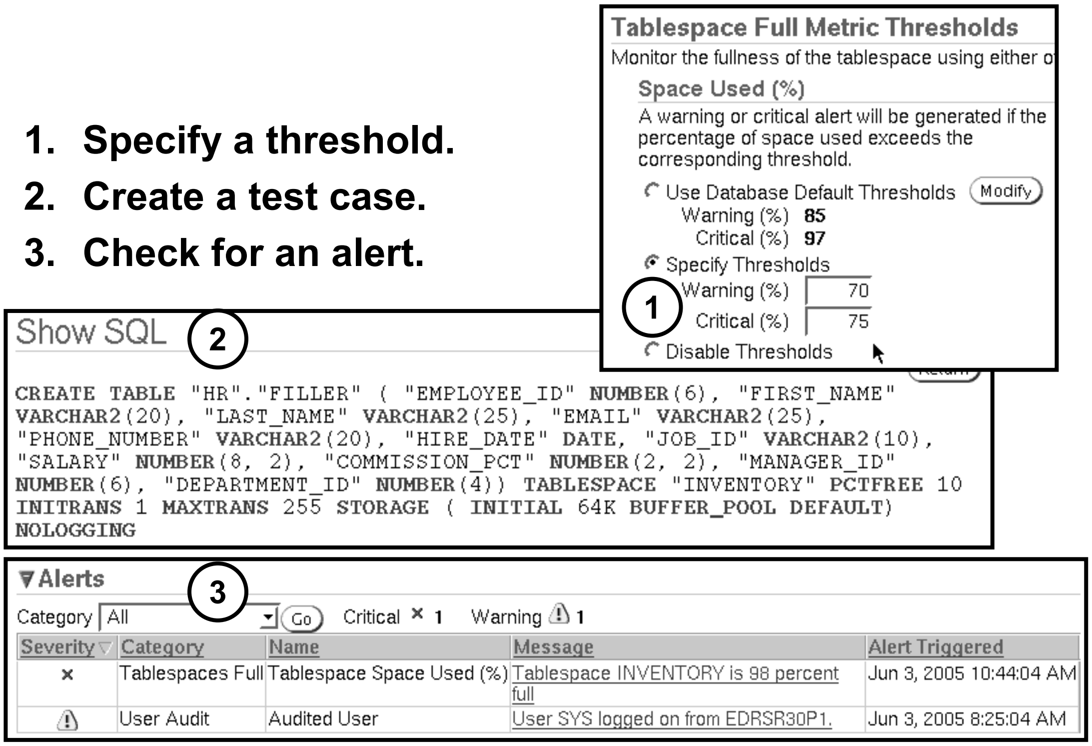


Setting Thresholds

To set or edit a threshold, select Manage Metrics in the Related Links region of the database home page. Click Edit Threshold, and then enter your desired Warning and Critical Threshold values. The appropriate alerts appear when the database reaches your specified values. If required, you can specify an additional response action.

Creating and Testing an Alert

1. **Specify a threshold.**
2. **Create a test case.**
3. **Check for an alert.**



Creating and Testing an Alert

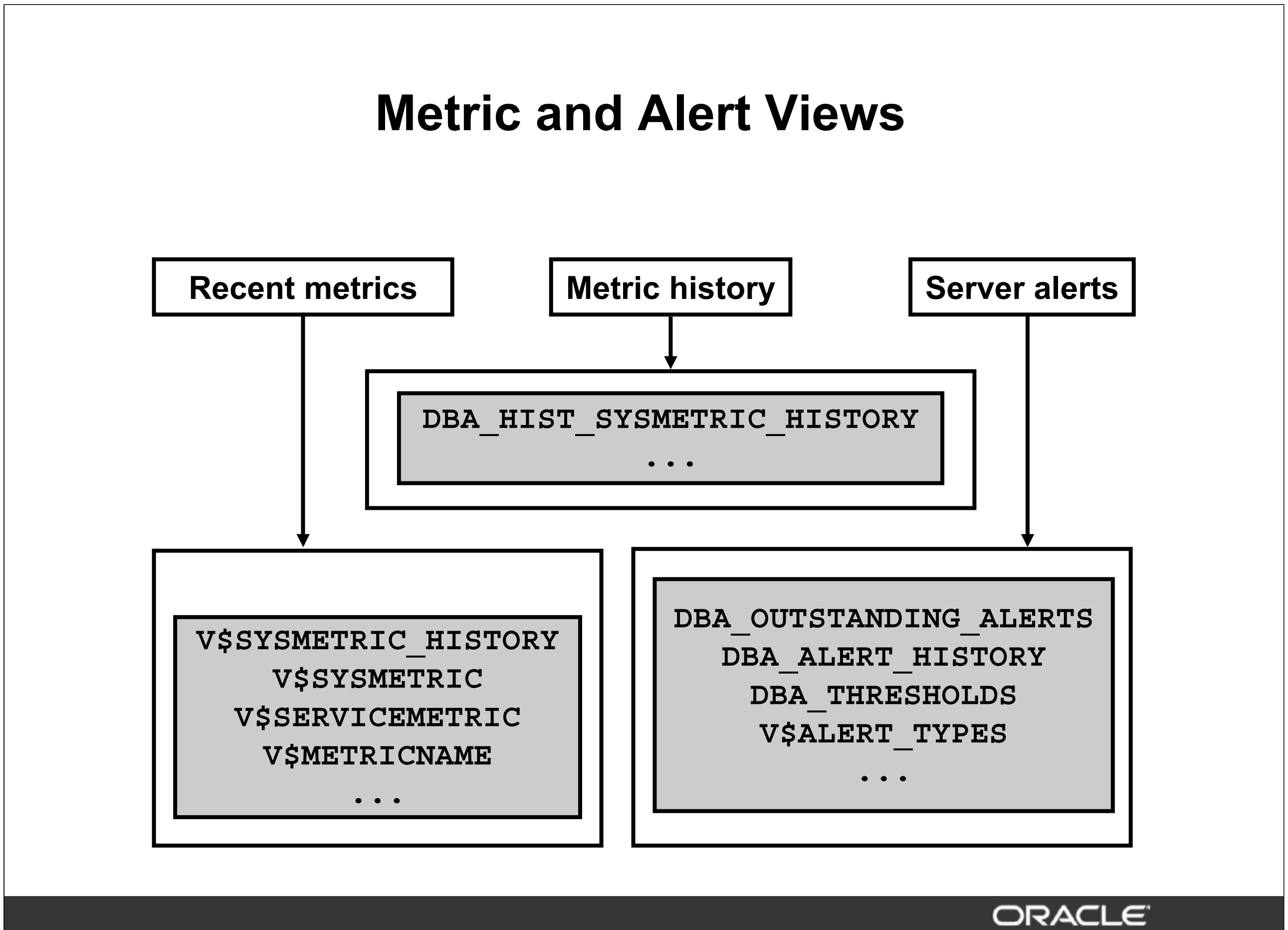
You can also set thresholds for a specific object.

Example: You decide that you need to receive a critical alert if the space used in the `INVENTORY` tablespace exceeds 75%. (This tablespace does not allow its data files to automatically extend.) To create and test the alert, perform the following steps:

1. In Enterprise Manager, navigate to the Tablespace page and set your desired threshold.
2. In SQL*Plus, use the `CREATE TABLE … TABLESPACE … AS SELECT …` command to duplicate an existing table into the tablespace of interest. Continue adding rows to this table to fill the tablespace.
3. After you receive an error that this table is unable to extend, check the Database Instance home page for the associated alert.

Most alerts contain the name of an associated advisor that can be invoked to give you more detailed advice. For each corresponding alert message, Database Control provides a link to invoke the corresponding advisor.

Note: The Tablespace Full alert is evaluated at 10-minute intervals



Metric and Alert Views

When enabled, the metric values are regularly computed by MMON and are kept in memory for one hour. The in-memory values for system-level metrics are viewable through the `V$SYSMETRIC` and `V$SYSMETRIC_HISTORY` views. Similar views are available for service-level metrics.

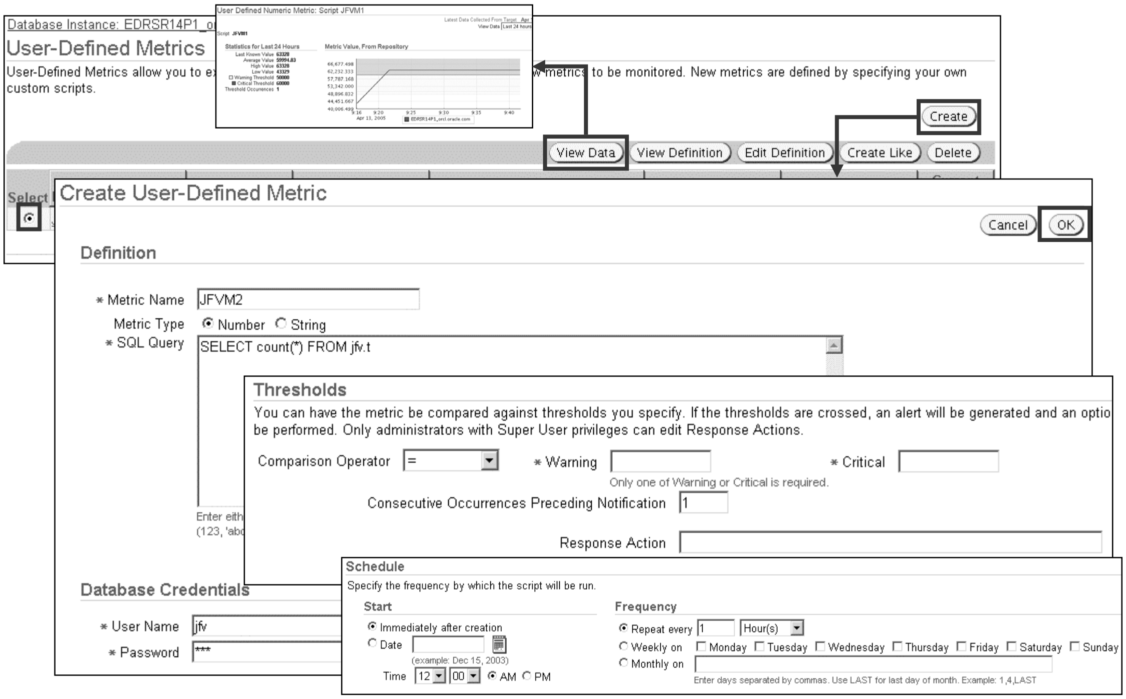
On-disk metric collection for all these metrics is enabled simply by enabling the automatic snapshot mechanism of AWR. The on-disk values for the metrics are viewable through the `DBA_HIST_*` views. The purge policy for metric history is the same as that for other snapshot data.

The following dictionary views enable you to access information about server alerts:

- `DBA_OUTSTANDING_ALERTS` describes alerts that the Oracle database server considers to be outstanding.
- `DBA_ALERT_HISTORY` represents a time-limited history of alerts that are no longer outstanding.
- `DBA_THRESHOLDS` gives you the threshold settings defined for the instance.
- `V$ALERT_TYPES` gives you information about each server alert type.

Note: For more information about these views, see the *Oracle Database Reference*.

User-Defined SQL Metrics



ORACLE

User-Defined SQL Metrics

In addition to server-computed metrics, EM also computes its own set of metrics. You can customize the EM metric set and create your own metrics that are checked by EM. To create your own metrics, open the User-Defined Metrics page. The link is available in the Related Links section of the Database Home page.

The User-Defined Metrics page lists existing SQL user-defined metrics. You can view metric data; view, edit, or delete a metric definition; or create another SQL user-defined metric based on an existing metric. If no SQL user-defined metrics have been created, the list is empty. Click Create to define a new user-defined metric.

The Create User-Defined Metric page enables you to define your SQL metric.

Note: Although user-defined metrics do not appear on the All Metrics page, any threshold violation is reported in the Alerts section of the Home page.

Practice 4-1 Overview: Working with Metrics

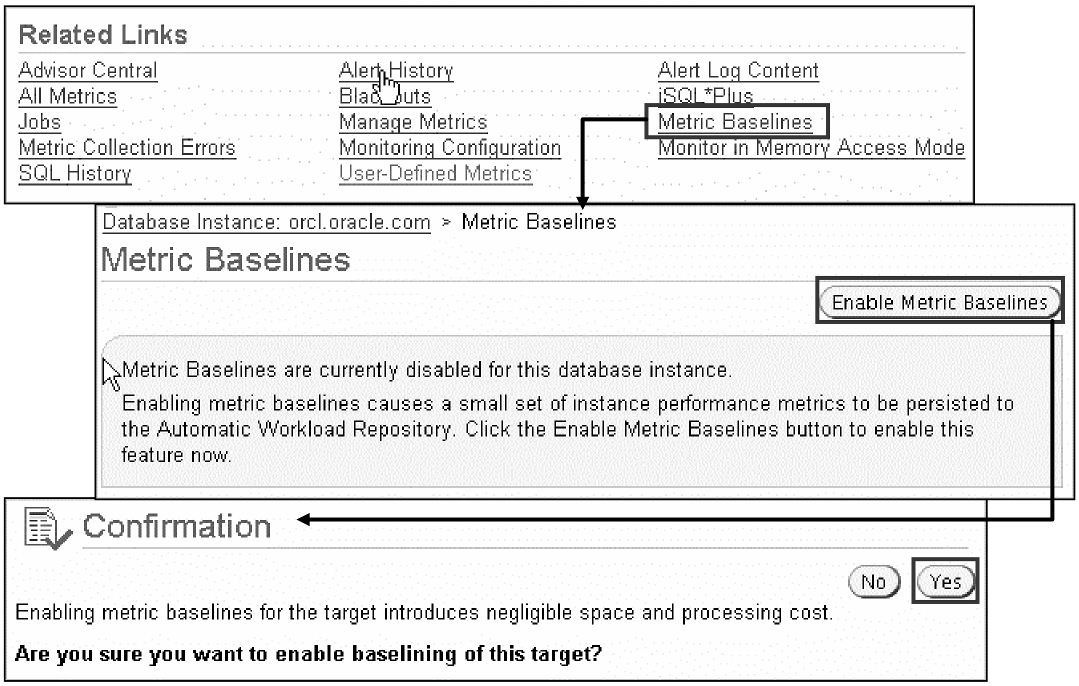
This practice covers the following topics:

- Viewing metrics using the metrics history views
- Creating metric thresholds
- Viewing alerts
- Clearing alerts

Working with Metric Baselines

- Enable metric baselines.
- Create a metric baseline.
  - Static baseline
  - Moving window baseline
- View the baseline statistics.
- Set a threshold on a baseline.
  - Fixed threshold (percent of maximum)
  - Adaptive threshold

Enabling Metric Baselining



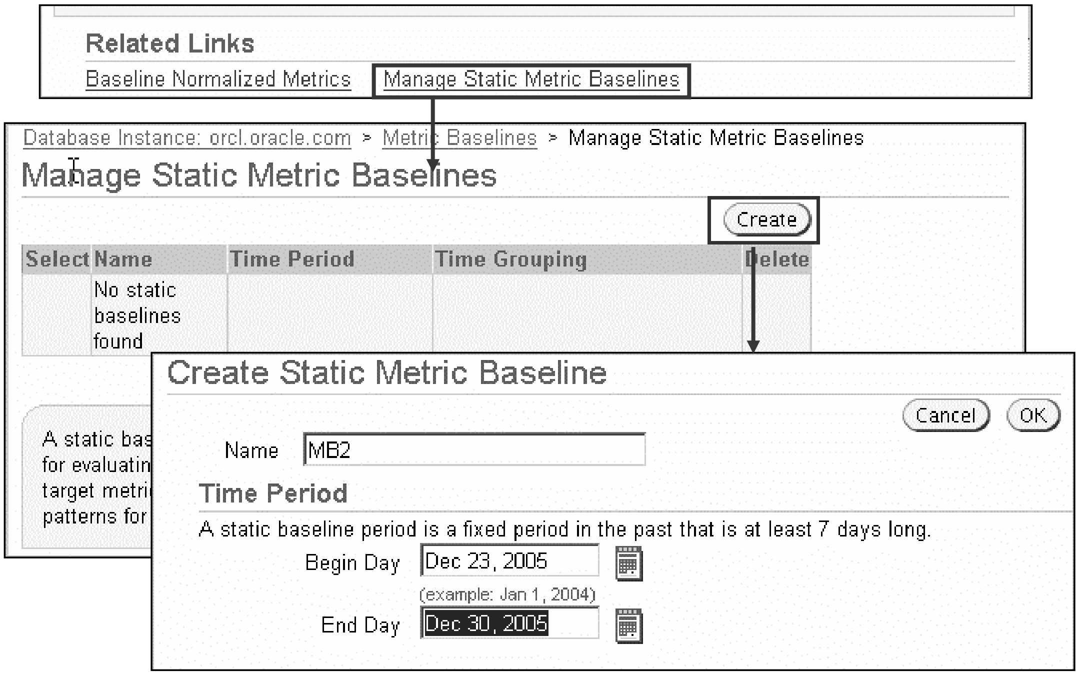
Enabling Metric Baselining

Before you can successfully use metric baselines and adaptive thresholds, you must enable that option by using Enterprise Manager. Internally, Enterprise Manager causes the system metrics to be flushed and submits a job once a day that is used to compute moving-window baseline statistics. It also submits one job once every hour to set thresholds after a baseline is activated.

You can enable metric baselining from the Database Home page by clicking the Metric Baselines link in the Related Links section. On the Metric Baselines page, click Enable Metric Baselines, and on the Confirmation page, click Yes.

Note: Enabling metric baselines for your database introduces negligible space and processing cost.

Creating Static Metric Baselines



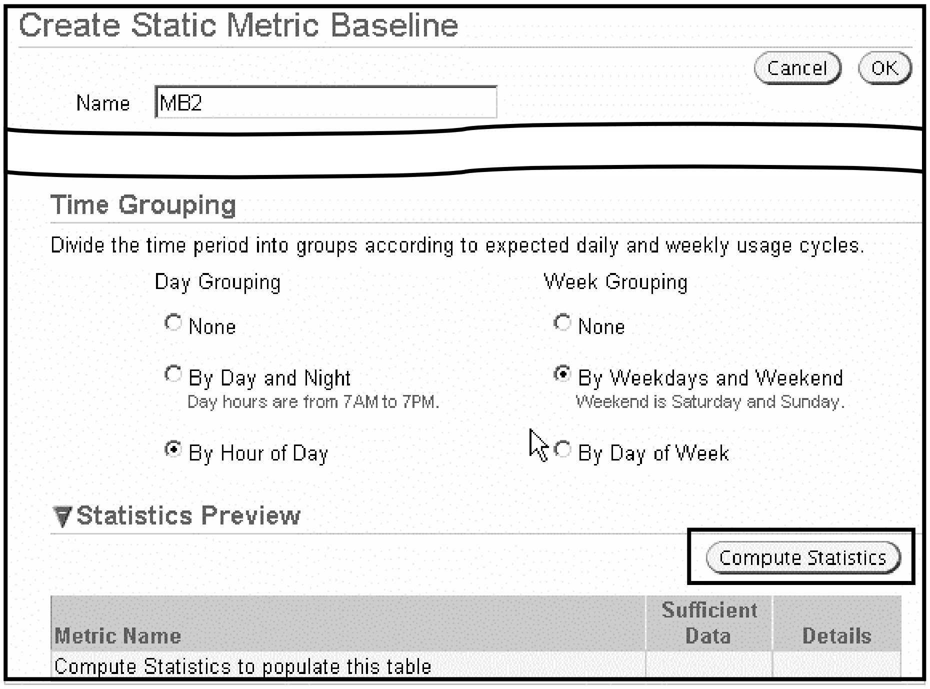
Creating Static Metric Baselines

Use the Create Static Metric Baseline page to create a new static metric baseline.

You can access this page from the Metric Baselines page by clicking the Manage Static Metric Baselines link in the Related Links section.

On the Create Static Metric Baseline page, specify a Name for your static metric baseline. Then select a Time Period by using the Begin Day and End Day fields. These two dates define the fixed interval that calculates metric statistics for later comparisons.

Creating Static Metric Baselines



ORACLE

Creating Static Metric Baselines (continued)

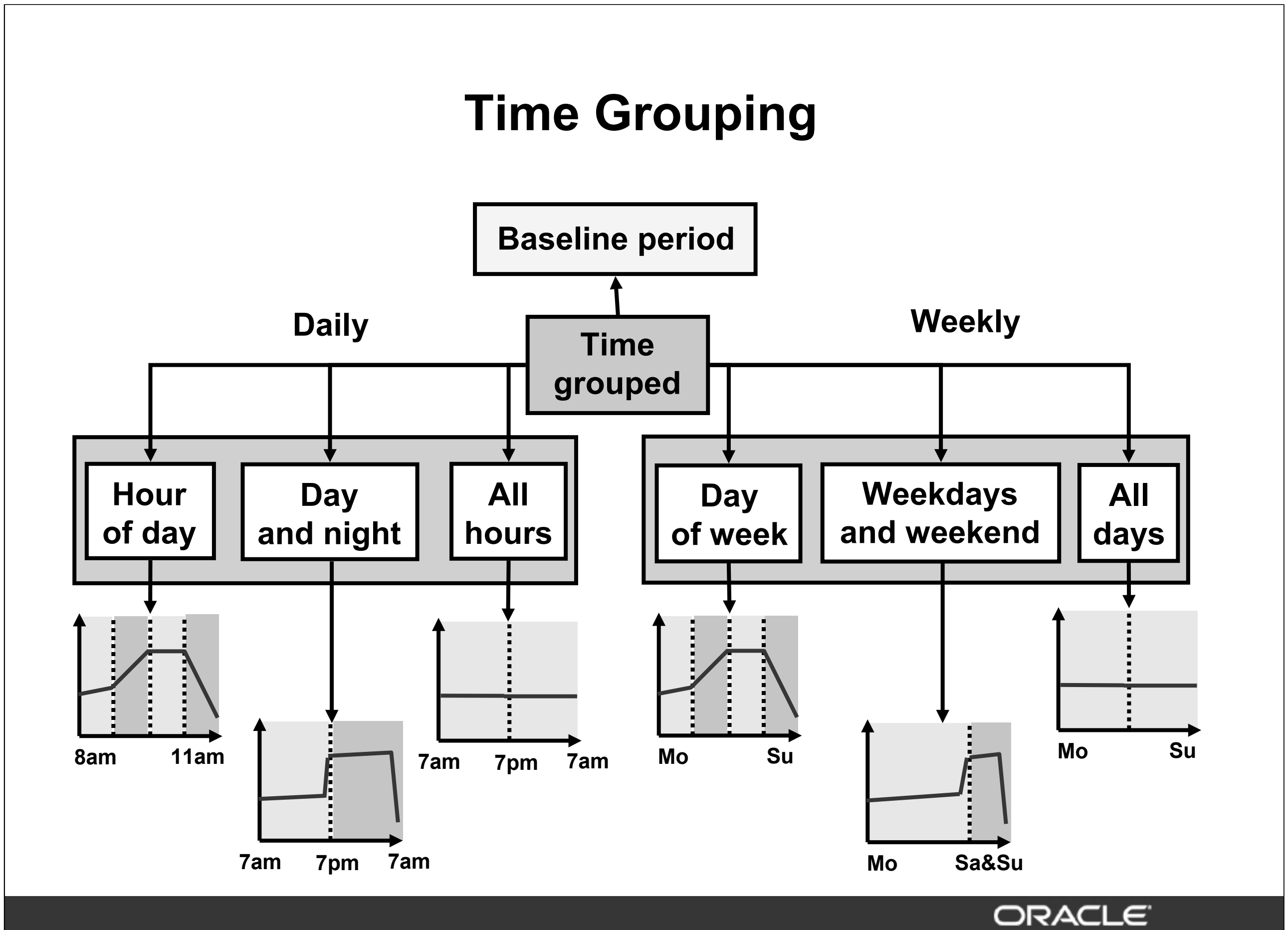
After choosing a time period, select the Time Grouping scheme:

- **By Hour of Day:** Creates 24 hourly groups
- **By Day and Night:** Creates two groups: day hours (7:00 a.m. to 7:00 p.m.) and night hours (7:00 p.m. to 7:00 a.m.)
- **By Day of Week:** Creates seven daily groups
- **By Weekdays and Weekend:** Creates two groups: weekdays (Monday through Friday) together and weekends (Saturday and Sunday) together

You must choose an option for both Day and Week time groupings. You can choose the None option, and you can combine options. For instance, grouping by Day and Night and Weekdays and Weekend produces four groups.

You next click Compute Statistics to compute statistics on all the metrics referenced by the baseline. Enterprise Manager computes statistics only once, which is when the baseline is created.

If an alert message appears in the Model Fit column, either there is insufficient data to perform reliable calculations or the data characteristics do not fit the metric baselines model.



Time Grouping

The supported time grouping schemes are shown in the slide.

The daily options are:

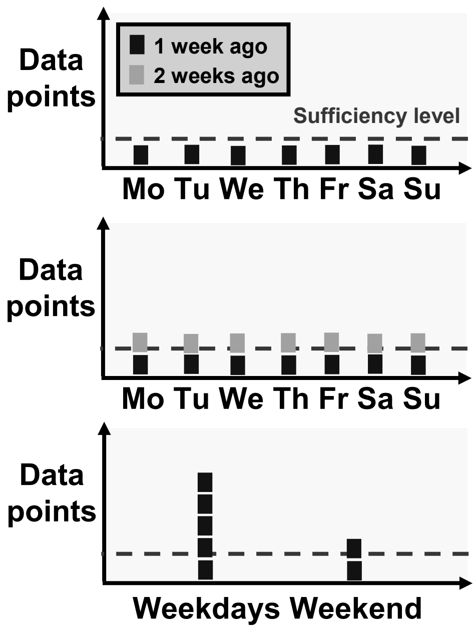
- **By hour of day:** Aggregate each hour separately for strong variations across hours.
- **By day and night:** Aggregate the hours of 7:00 a.m. to 7:00 p.m. as day and 7:00 p.m. to 7:00 a.m. as night.
- **None - By all hours:** Aggregate all hours together when there is no strong daily cycle.

The weekly time grouping options are:

- **By day of week:** Aggregate days separately for strong variations across days.
- **By weekday and weekend:** Aggregate Monday to Friday together and Saturday and Sunday together.
- **None - By all days:** Aggregate all days together when there is no strong weekly cycle.

Note: Baseline time grouping is always done on both daily and weekly dimensions. If you do not choose one or the other, the corresponding *All* aggregation level is chosen.

Time Grouping: Considerations



Moving window size	7 days
Grouped by	Day of week
Sufficient data?	No

Moving window size	14 days
Grouped by	Day of week
Sufficient data?	Yes

Moving window size	7 days
Grouped by	Weekdays and weekend
Sufficient data?	Yes

Time Grouping: Considerations

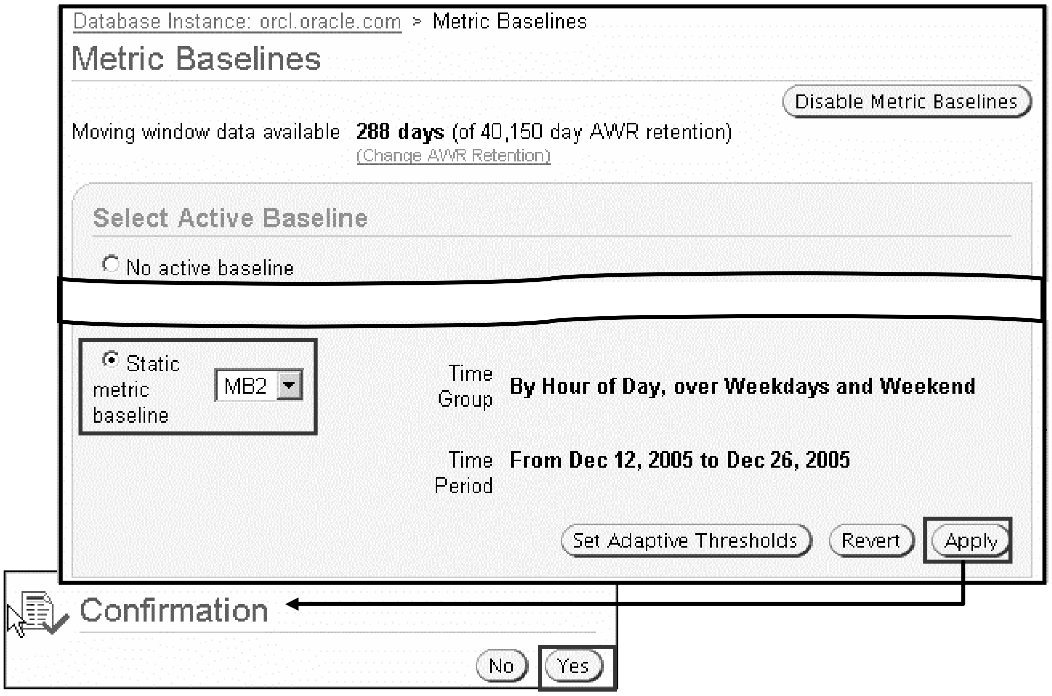
Time grouping allows baselines to adjust for cyclic daily or weekly usage patterns. Select time grouping to aggregate data around expected usage variations—for example, by weekdays versus weekends. Larger groups (more samples) aggregate more data and give stronger statistical estimates. Smaller groups enable finer-grained separation of metric data into statistically similar groups.

If there is insufficient data to reliably use statistical alert thresholds, either extend the time period or make time groups larger to aggregate statistics across more data samples. Each of these actions increases the amount of data per group, resulting in sufficient data to compute more realistic metric baselines statistics.

In the slide example, given a particular amount of collected statistics, a moving window of seven days grouped by day of week does not provide enough data points. It has only one sample per day. The slide shows two ways to solve this problem:

- You can increase the number of days in the moving window, thus having more days' data contributing to the collection of statistics.
- You can choose the broader grouping option of Weekdays and Weekend, making it possible to stay with the seven-day moving-window size. The totals of the two weekend days bring the amount of data up to the required level for one group, and the total data for all of the weekdays provides enough data for the other group.

Activating the Static Metric Baseline



ORACLE

Activating the Static Metric Baseline

Use the Metric Baselines page to configure your active baseline.

After baselining is enabled, you can access the Metric Baselines page directly from the Database Home page by clicking the Metric Baselines link in the Related Links section.

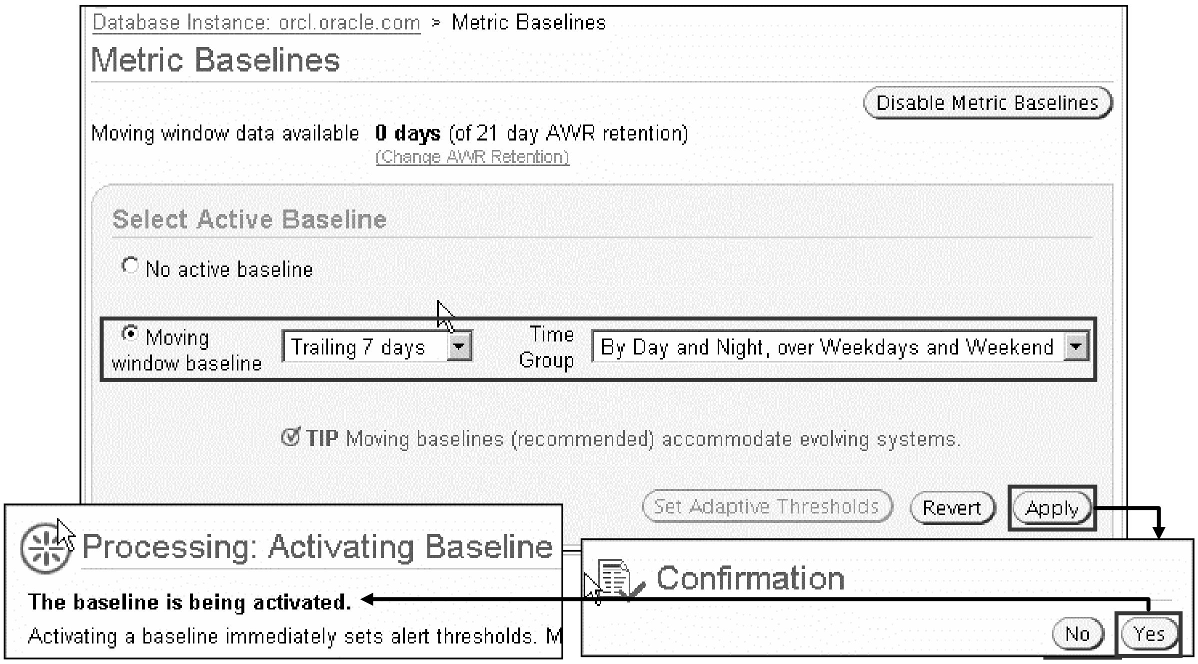
By default, "No active baseline" is selected, meaning that no metric baseline is currently configured.

You can either use one "moving window" metric baseline or select an already defined "static baseline."

The screenshot shows an example of using a static window baseline:

1. Select the static metric baseline you want to define for this baseline. You may select any static baseline that has been created from the pull-down menu.
2. Click Apply. You are asked to confirm your choice.

Activating the Moving Window Metric Baseline



Activating the Moving Window Metric Baseline

Use the Metric Baselines page to configure your active baseline.

After baselining is activated, you can access the Metric Baselines page directly from the Database Home page by clicking the Metric Baselines link in the Related Links section.

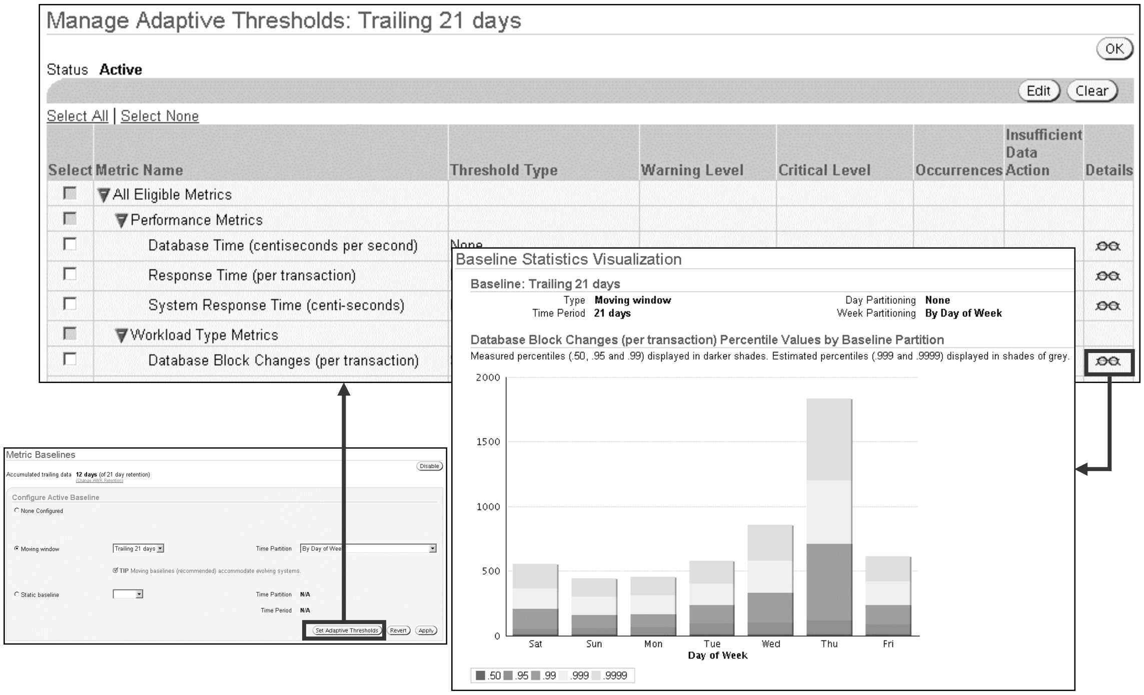
By default, None Configured is selected, meaning that no metric baseline is currently configured.

You can either use one “moving window” metric baseline or select an already defined “static baseline.” A “moving window” metric baseline cannot be predefined.

The screenshot shows you an example of using a moving window baseline:

1. Select the time period you want to define for this baseline, such as “Trailing 7 days.” This period moves with the current time. The most recent seven-day period becomes the baseline period (or reference time) for all metric observations and comparisons today. Tomorrow, this reference period drops the oldest day and picks up today.
2. Define the Time Grouping scheme. Grouping options available for a baseline depend on the size of the time period for the baseline. The system automatically gives you realistic choices. There must be at least two samples for each period.
3. Click Apply. Enterprise Manager computes statistics on all the metrics referenced by the baseline. The computing of statistics is done every day automatically.

Visualizing Metric Baseline Statistics

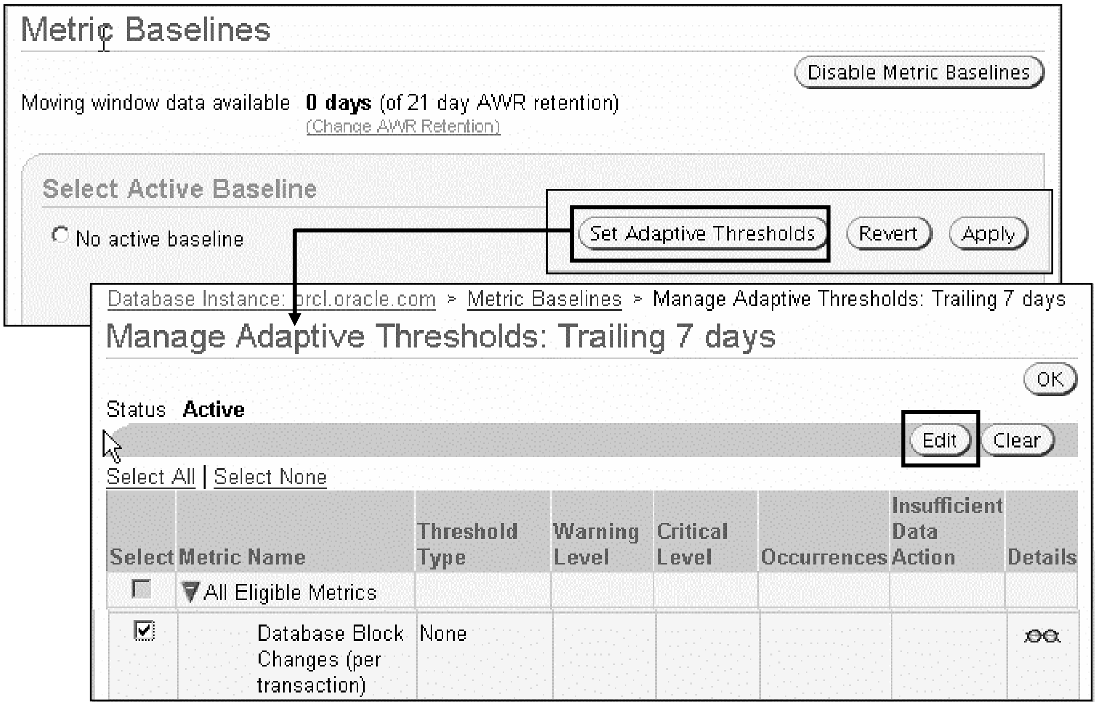


Visualizing Metric Baseline Statistics

You can visualize the collected statistics for your metric baselines by using the Manage Adaptive Thresholds page. You can access the Manage Adaptive Thresholds page from the Metric Baselines page by clicking Set Adaptive Thresholds after selecting the corresponding baseline. On the Manage Adaptive Thresholds page, click the corresponding eyeglass icon in the Details column. You are then directed to the Baseline Statistics Visualization page for the corresponding metric. The slide illustrates what you might see for the Database Block Changes metric. In the example, you can view the data distribution of that metric relative to its usage pattern.

Note: This page displays a stacked bar chart of metric percentiles for each of the hours in a typical week over the baseline period. The baseline time grouping determines how many distinct bars are in the chart from a minimum of one (coarsest grouping: “by all hours” and “by all days”) to a maximum of 168 (finest grouping: “by hour of day” and “by day of week”).

Setting Adaptive Alert Thresholds



ORACLE

Setting Adaptive Alert Thresholds

Use the Edit Baseline Alert Parameters page to do the following:

- View the current status of the 15 metrics that can be set with adaptive thresholds
- Set thresholds for Warning Level, Critical Level, and Occurrences
- Specify threshold action for insufficient statistical data

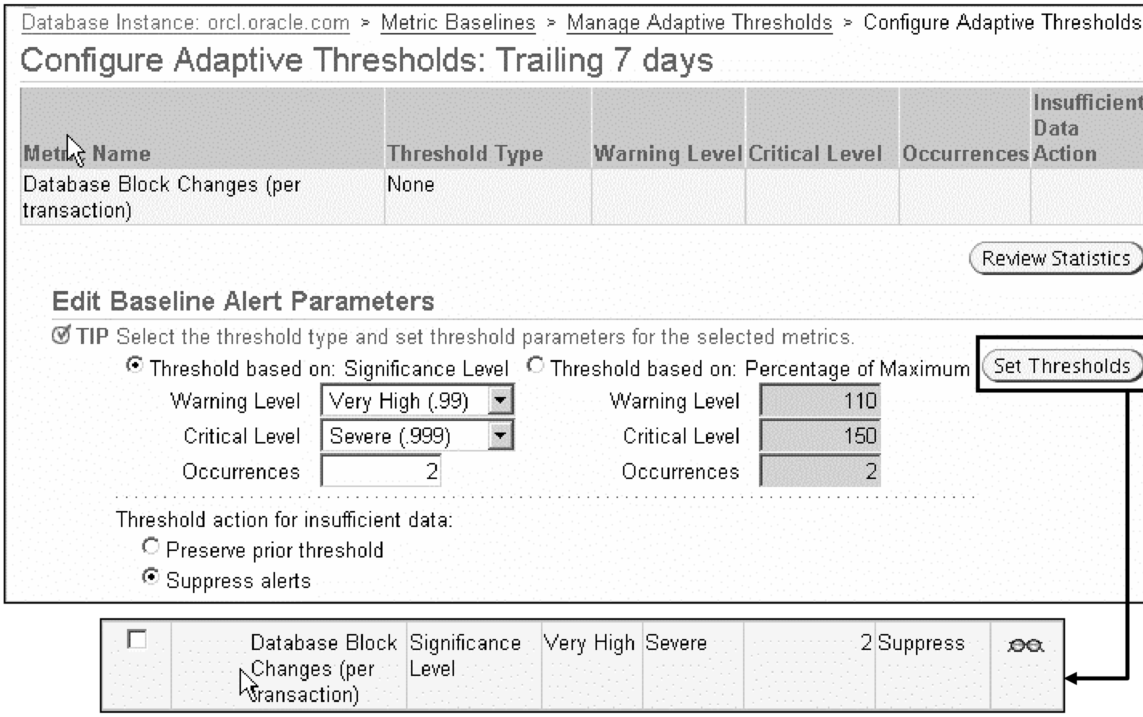
The warning and critical levels control the type of alert to be generated. For Significance Level thresholds, alerts are generated if the significance level of an observation is at or above the level that you select. The following Significance Level thresholds are supported:

- High, significant at 0.95 (5 in 100) level
- Very High, significant at 0.99 (1 in 100) level
- Severe, significant at 0.999 (1 in 1,000) level
- Extreme, significant at 0.9999 (1 in 10,000) level

It is recommended that you use Significance Level thresholds conservatively and experimentally at first. High is the most conservative.

For "Percentage of Maximum" thresholds, percentage of maximum–based alerts are generated if the observation is at or above the percentage of maximum you specify. The measured 99th percentile value is used as the trimmed maximum value.

Setting Adaptive Alert Thresholds



Setting Adaptive Alert Thresholds (continued)

A warning threshold level of 110% of maximum triggers an alert when an observed metric value is 110% of the 99th percentile value as measured over the baseline time group that is assigned to the time of the observation.

The Occurrences parameter indicates how many consecutive occurrences must happen before generating the alert.

For the threshold action choices, allowing alerts leaves the thresholds as they are for a period of time, and suppressing alerts clears the threshold for that period.

Metric Baseline and Adaptive Threshold: Considerations

- **Baselining must be enabled using Enterprise Manager.**
- **Only one moving window baseline can be defined.**
- **Multiple static baselines can be defined.**
- **Only one baseline can be active at a time.**
- **Adaptive thresholds require an active baseline.**

ORACLE

Metric Baseline and Adaptive Threshold: Considerations

Consider the following when using metric baselines:

- Before you can use baselines and adaptive thresholds in your environment, you must enable the feature by using Enterprise Manager.
- There can be only a single moving window baseline and as many static baselines as you want to define.
- There can be at most one baseline designated as active on the target. The active baseline primarily controls the setting of adaptive alert thresholds.

Classification of Eligible Metrics

Performance Metrics

- Database Time (centiseconds per second)
- Response Time (per transaction)
- System Response Time (centiseconds)

Workload Volume Metrics

- Executes (per second)
- Redo Generated (per second)
- Network Bytes (per second)
- Physical Writes (per second)
- Physical Reads (per second)
- Current Logons Count
- User Calls (per second)
- Number of Transactions (per second)

Workload Type Metrics

- Database Block Changes (per transaction)
- Enqueue Requests (per transaction)
- Total Parses (per transaction)
- Session Logical Reads (per transaction)

Classification of Eligible Metrics

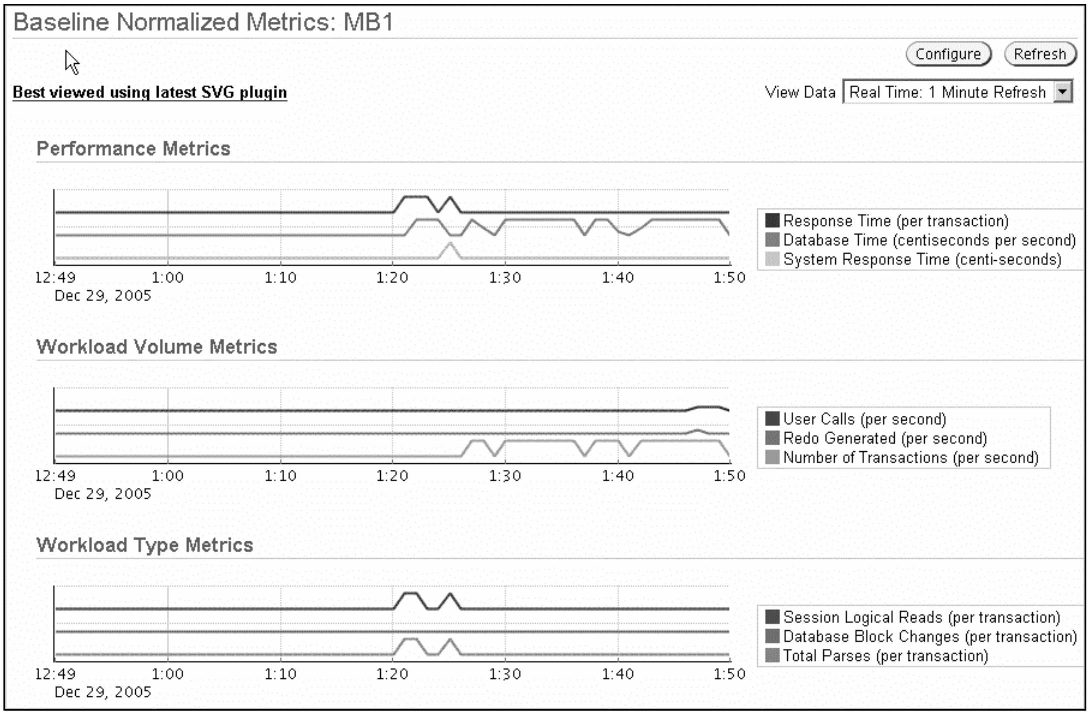
You can assign adaptive thresholds to only a subset of all existing metrics. These metrics are grouped into the following three categories:

- **Performance metrics** typically measure response time and indicate how well the system is performing.
- **Workload Volume metrics** measure the load or demand on the system, or the amount of work tasked to the system.
- **Workload Type metrics** attempt to distinguish the kind or shape of the work tasked to the system.

In Oracle Database 10*g*, there are 15 metrics visible in `V$SYSMETRIC_HISTORY` that are recognized as a reasonable set of system-level indicators of performance and workload demand and shape.

You can create metric baselines for the list of metrics (organized by category) shown in the slide.

Metric Baseline: Normalized View



Metric Baseline: Normalized View

Metric value time series can be normalized against a baseline by converting each observation to some integer measure of its statistical significance relative to the baseline. Each point in the metric time series is compared to the baseline and assigned a value based on a statistical difference called a *variance*. Small variances are called *noise* because they are statistically insignificant. You can select the level of significance by setting Noise Reduction to Low, Medium, or High. High removes all but the most significant events from the display.

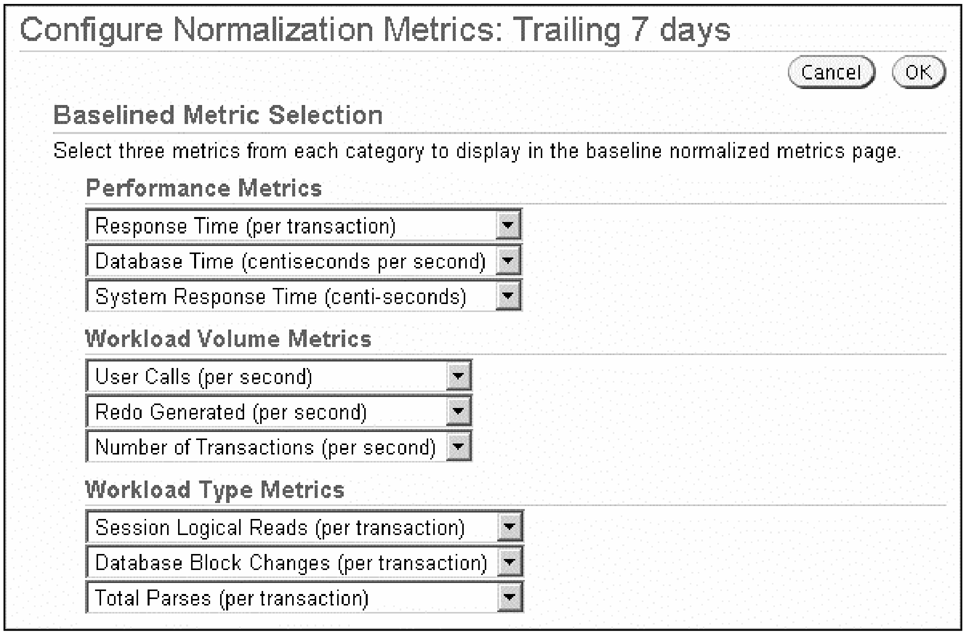
Significant events are aggregated and shown over five-minute intervals in the graphs. The range of values of the variance is the same for all metrics, so there is a common Y-axis. The X-axis is based on time. Thus, the graphs can be easily compared with each other.

You can see the normalized view of your metrics on the Baseline Normalized Metrics page. You access this page on the Metric Baselines page by clicking the Baseline Normalize Metrics link in the Related Links section.

The normalized view separates metrics into groups based on category. Metrics from the same group appear in a graph together and in related shades of color. This encourages you to think about group-level events in addition to individual metric events.

Note: You can view the data in real time, or you can look at historical data.

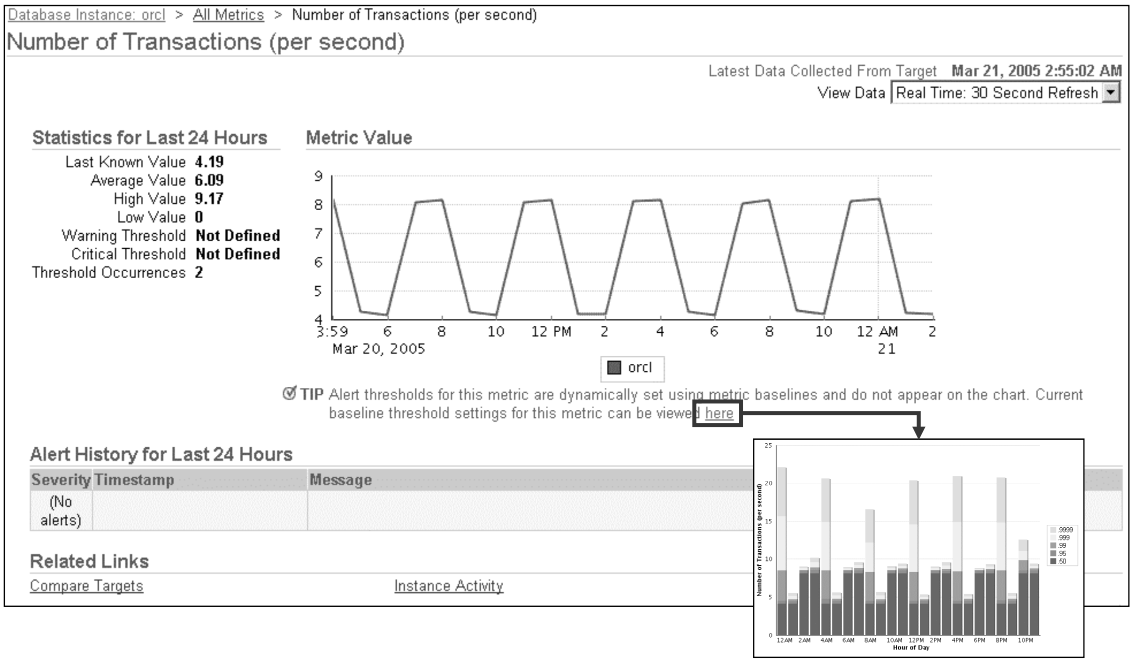
Configuring Normalization Metrics



Configuring Normalization Metrics

If you click Configure on the Baseline Normalized Metric page, you are directed to the Configure Normalization Metrics page (shown in the slide). You can use this page to select three metrics from each category to be displayed on the Baseline Normalized Metrics page.

Adaptive Thresholds and the All Metrics Page



Adaptive Thresholds and the All Metrics Page

Threshold metrics for which adaptive thresholds are set are not reflected on the All Metrics page. Instead, you are asked to go to the Baseline Statistics Visualization page, where you can view how a metric baseline influences alerts. For the corresponding metric, Enterprise Manager automatically sets thresholds according to the statistics calculated over the corresponding time group.

Summary

In this lesson, you should have learned how to:

- View metrics by using the metrics history views
- Create metric thresholds
- View alerts
- Create metric baselines
- Enable adaptive thresholds

Practice 4-2 Overview: Working with Baselines

This practice covers the following topics:

- Creating a static metric baseline
- Viewing the baseline data
- Enabling adaptive thresholds
- Viewing deviations from the baseline

5

Using Statspack

Objectives

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

- Install Statspack
- Create Statspack snapshots
- Generate Statspack reports
- Identify the major sections of a Statspack report

Introduction to Statspack

- Statspack is a set of scripts that capture and report on performance data from within the Oracle database.
- With Statspack scripts, you can:
  – Capture instance data by taking a “snapshot”
  – Store snapshot data in the database in a separate schema
  – Create reports between two snapshots
  – Mark snapshots as baseline information
  – Use the reports as part of a performance tuning method

ORACLE

Introduction to Statspack

Statspack is a set of scripts and PL/SQL packages. Statspack allows you to capture point-in-time performance data called a snapshot into a schema dedicated to Statspack. You can create a report between snapshots and view the database activity and performance for the time period covered by the snapshots. You can preserve sets of snapshots by marking them as baseline information. Marking the snapshots protects them from deletion by the purge procedure.

The reports generated by Statspack provide performance tuning information that can be used as part of a tuning methodology.

Statspack Scripts

- Install Statspack by using `spcreate.sql`.
- Collect statistics with `statspack.snap`.
- Automate the collection of statistics with `spauto.sql`.
- Produce a report by using `spreport.sql` or `sprepsql.sql`.
- Purge Statspack data with `statspack.purge` or `sppurge.sql`.
- Truncate all Statspack tables with `sptrunc.sql`.
- Drop the Statspack repository with `spdrop.sql`.
- Export the Statspack repository with `spexp.par`.

Statspack Scripts

Installation of the Statspack Package

The installation of the Statspack utility creates the PERFSTAT user, which owns all PL/SQL code and database objects created (including the Statspack tables, the constraints, and the Statspack package). During the installation, you are prompted for the PERFSTAT user's default and temporary tablespaces.

The Statspack package has been available with the Oracle database since Oracle 8.1.6. When initially installed, approximately 80 MB of the PERFSTAT user's default tablespace is used. This may increase later when the tables contain snapshot information.

Collecting Statistics

To take a snapshot of performance data, log on to SQL*Plus as the PERFSTAT user, and then execute the STATSPACK.SNAP procedure. This procedure stores the current performance statistics in the Statspack tables, which can be used with another snapshot taken at a later time to produce a report or a base line.

Automatically Collecting Statistics

To compare performance from one day, week, or year to the next, there must be multiple snapshots taken over a period of time. The best method to gather snapshots is to automate the collection at regular time intervals.

Statspack Scripts (continued)

Automatically Collecting Statistics (continued)

The spauto.sql script makes use of the DBMS_JOB package to provide an automated method for collecting statistics. The supplied script schedules a snapshot every hour on the hour. This can be changed to suit the requirements of the system.

Producing a Report

To examine the change in statistics between two time periods, execute the spreport.sql file while being connected to the PERFSTAT user. You are shown a list of collection periods and can select a start and end period. The difference between the statistics at each end point is then calculated and put into a file with a name of your choice.

Truncating Statspack Tables

If you want to truncate all performance data indiscriminately, it is possible to do this using sptrunc.sql. This script truncates all statistics data gathered, including snapshots marked as baselines.

Dropping the Statspack Repository

To drop the Statspack Repository and remove the Statspack package, use the spdrop.sql script. If there are errors during repository creation, you must drop the repository before you attempt to re-create it.

Exporting the Statspack Repository

If you want to share data with other sites, for example if Oracle Support requires the raw statistics, it is possible to export the PERFSTAT user. An export parameter file (spuexp.par) has been supplied for this purpose. To use this file, supply the export command with the userid parameter, along with the export parameter file name:

```
exp userid=perfstat/perfstat_password parfile=spuexp.par
```

This creates a file called spuexp.dmp and the spuexp.log log file.

Note: It is also possible to upgrade Statspack tables to the latest version of the Oracle database that you are using. Scripts are provided to accomplish this task.

Installing Statspack

- **Install Statspack:**
  - **Supplying parameters as requested interactively**
  - **Defining parameters in the session for batch mode**

```
SQL> connect / as sysdba
SQL> define default_tablespace='SYSAUX'
SQL> define temporary_tablespace='TEMP'
SQL> define perfstat_password='erg8oiw'
SQL> @?/rdbms/admin/spcreate
SQL> undefine perfstat_password
```

- **The `spcreate.sql` script creates the `PERFSTAT` user to own the Statspack repository and package.**

ORACLE

Installing Statspack

Installation scripts create the user called PERFSTAT, who owns all PL/SQL code and database objects created including the Statspack tables, constraints, and the STATSPACK package.

There are two ways to install Statspack: interactively or in batch mode (as shown in the slide). Batch mode is useful when you do not want to be prompted for the PERFSTAT user's password and the default and temporary tablespaces. To install in batch mode, you must assign values to the SQL*Plus variables that specify the password and the default and temporary tablespaces before running spcreate.sql. Both methods require the default tablespace to be defined. You cannot use the SYSTEM tablespace as the default tablespace for the PERFSTAT user. SYSAUX is the recommended default tablespace.

If you receive an error during the installation, execute spdrop.sql before you rerun spcreate.sql.

Note: To execute the installation script, you must use SQL*Plus and connect as a user with the SYSDBA privilege.

Capturing Statspack Snapshots

- **Capture a snapshot:**

```
SQL> connect perfstat/erg8oiw
SQL> execute statspack.snap;
```

- **Use the function to determine the snapshot number:**

```
SQL> variable snap number;
SQL> begin :snap := statspack.snap; end;/
SQL print snap
```

- **Automate snapshot collection:**

```
SQL> connect perfstat/erg8oiw
SQL> @spauto
```


Capturing Statspack Snapshots

The simplest and most interactive way to take a snapshot is to log in to SQL*Plus as the `PERFSTAT` user and execute the `statspack.snap` procedure (as shown in the slide). This method stores the current values for the performance statistics in the Statspack tables and can be used as a baseline snapshot for comparison with another snapshot taken at a later time.

If you want to automate the gathering and reporting phases (such as during a benchmark), you may need to know the `snap_id` of the snapshot just taken. To take a snapshot and display the `snap_id`, call the `statspack.snap` function (as shown in the slide).

To use an Oracle-automated method for collecting statistics, you can use `DBMS_JOB`. A sample script is supplied in `spauto.sql`, which schedules a snapshot every hour on the hour. This script should be run as `PERFSTAT`. The `JOB_QUEUE_PROCESSES` initialization parameter should be set to a number greater than zero before automatic statistics gathering can run. The script also reports the corresponding job number. You can use the `DBMS_JOB` package to change the interval of statistics collection, force the job to run immediately, or to remove automatic job collection.

Note: On UNIX systems, you can use utilities such as `cron` or `at` to automate snapshot collection.

Configuring Snapshot Data Capture

- **Temporarily change a capture variable:**

```
SQL>  execute statspack.snap(i_snap_level=>6);
```

- **Change defaults and take a snapshot:**

```
SQL>  execute statspack.snap -
     (i_snap_level=>10, i_modify_parameter=>'true');
```

- **Modify defaults without taking a snapshot:**

```
SQL> execute statspack.modify_statspack_parameter -
     (i_snap_level=>10, i_buffer_gets_th=>10000,-
     i_disk_reads_th=>1000);
```

- **Include session details in a snapshot:**

```
SQL>  execute statspack.snap(i_session_id=>3);
```


Configuring Snapshot Data Capture

The specified snapshot level and thresholds affect the amount of data Statspack captures. You can control the amount of information gathered by the package by specifying a snapshot level. A higher snapshot level gathers more data. The default level set by the installation is level 5. Level details are shown in a later slide.

There are threshold parameters that can be configured when collecting data on SQL statements. Data is captured on any SQL statements that breach the specified thresholds.

To temporarily use a snapshot level or threshold that is different from default snapshot values, simply specify the required threshold or snapshot level when taking the snapshot. This value is used only for the immediate snapshot taken, as shown in the slide.

You can save the new value as the instance default by using `i_modify_parameter`, as shown in the slide. You can also change the defaults without taking a snapshot, using the `STATSPACK.MODIFY_STATSPACK_PARAMETER` procedure.

To gather session statistics and wait events for a particular session, in addition to the instance statistics and wait events, specify the `session id` in the call to Statspack.

Statspack Snapshot Levels

Level =>	Statistics captured
0	General performance
5	SQL statements
6	SQL plans and SQL plan usage
7	Segment-level statistics
10	Parent and child latches

AWR captures them all by default.

Statspack Snapshot Levels

All levels contain general performance statistics, including the following: wait statistics, system events, system statistics, rollback segment data, row cache, SGA, background events, session events, lock statistics, buffer pool statistics, latch statistics, resource limit, and enqueue statistics. Statistics are included for each of the following, if enabled: Automatic Undo Management, buffer cache advisory data, automatic PGA memory management, and Cluster DB statistics.

Levels 5 and higher gather the performance data on SQL statements that exceed one of six predefined threshold parameters for resource usage. The time required for the snapshot to complete depends on the value of `SHARED_POOL_SIZE` and the number of SQL statements in the shared pool at the time the snapshot is taken. You can change the predefined threshold values. The thresholds are:

- Number of executions of the SQL statement (default 100)
- Number of disk reads performed by the SQL statement (default 1,000)
- Number of parse calls performed by the SQL statement (default 1,000)
- Number of buffer gets performed by the SQL statement (default 10,000)
- Size of sharable memory used by the SQL statement (default 1 MB)
- Version count for the SQL statement (default 20)

Statspack Snapshot Levels (continued)

Level 6 includes all statistics gathered at the lower levels. In addition, it gathers optimizer execution plans and plan usage data for each of the high-resource-usage SQL statements that are captured. A level 6 snapshot gathers information that is invaluable when determining whether the execution plan used for a SQL statement has changed. Therefore, level 6 snapshots should be used whenever there is the possibility that a plan may change, such as after large data loads or after gathering new optimizer statistics. The plan for a SQL statement is captured only if the statement is in the shared pool at the time the snapshot is taken and if it exceeds one of the SQL thresholds. You can gather plans for all statements in the shared pool if you temporarily specify the executions threshold (`i_executions_th`) to be zero (`0`) for those snapshots.

Level 7 includes all statistics gathered at the lower levels. In addition, it gathers the performance data on highly used segments. A level 7 snapshot captures segment-level statistics for segments that exceed one of seven threshold parameters that measure access or contention rates. You can change the default threshold values for the following:

- Number of logical reads on the segment (default 10,000)
- Number of physical reads on the segment (default 1,000)
- Number of buffer busy waits on the segment (default 100)
- Number of row lock waits on the segment (default 100)
- Number of ITL waits on the segment (default 100)
- Number of global cache Consistent Read blocks served (default 1,000)
- Number of global cache Current blocks served (default 1,000)

Levels 10 and higher include all statistics gathered at the lower levels. In addition, they gather Parent and Child Latch information. Data gathered at this level can sometimes cause the snapshot to take longer to complete and should be used only when advised by Oracle personnel.

Note: Refer to the `spdoc.txt` file for more information about the various levels and thresholds.

Statspack Baselines and Purging

```
SQL> exec statspack.make_baseline -
  2  (i_begin_snap => 45, i_end_snap   => 50);
```

```
SQL> exec statspack.statspack.make_baseline(2,4,false);
```

```
SQL> exec statspack.make_baseline –
  2  (to_date('31-AUG-2005 09:00','DD-MON-YYYY HH24:MI'),-
  3  to_date('31-AUG-2005 12:00','DD-MON-YYYY HH24:MI'));
```

```
SQL> exec statspack.clear_baseline -
  2  (to_date('13-DEC-2004 23:00','DD-MON-YYYY HH24:MI'),-
  3  to_date('14-FEB-2005 02:00','DD-MON-YYYY HH24:MI'));
```

```
SQL> exec statspack.purge –
  2  (i_begin_date=>to_date('01-JAN-2003','DD-MON-YYYY'),-
  3  i_end_date  =>to_date('02-JAN-2003','DD-MON-YYYY'),-
  4  i_extended_purge=>TRUE);
```

ORACLE

Statspack Baselines and Purging

The PERFSTAT schema can grow very large with many snapshots. You can purge unnecessary data from the PERFSTAT schema by using STATSPACK.PURGE. By default, PURGE removes all snapshots, but you can identify and keep snapshot data by creating baselines. After you have determined which snap ids or times of day most represent a particular workload whose performance data you would like to keep, you can mark the data as baselines. You can later decide to clear one or more baselines. Clearing the baseline does not remove the data; it simply identifies the data as candidates for purging. You cannot roll back a purge operation.

The slide shows examples of managing baselines of Statspack data:

- Making a baseline of snaps 45 and 50 (including the range of snapshots in between)
- Making a baseline of only snapshots 2 and 4 (excluding the snapshots in between)
- Making a baseline of snapshots taken between August 31, 2005, at 9:00 a.m., and August 31, 2005, at 12:00 noon
- Clearing an existing baseline that includes all snapshots between December 13, 2004, at 11:00 p.m., and February 14, 2005, at 2:00 a.m.
- Purging all snapshots that fall between January 1, 2003, and January 2, 2003; also performing an extended purge

Statspack Baselines and Purging (continued)

In earlier releases of Statspack, Statspack identifier tables that contained SQL Text, SQL Execution plans, and Segment identifiers were not purged. Now you can purge unreferenced data in these tables by requesting that the extended purge be performed at the same time as the normal purge. Purging this data can be resource intensive, so you may choose to perform an extended purge less frequently than the normal purge.

Reporting with Statspack

```
SQL> connect perfstat/perfstat_password
SQL> @?/rdbms/admin/spreport
```

```
SQL> connect perfstat/perfstat_password
SQL> define begin_snap=1
SQL> define end_snap=2
SQL> define report_name=batch_run
SQL> @?/rdbms/admin/spreport
```

```
SQL> connect perfstat/perfstat_password
SQL> define begin_snap=39
SQL> define end_snap=40
SQL> define hash_value=1988538571
SQL> define report_name=batch_sql_run
SQL> @sprepsql
```


Reporting with Statspack

You can generate a performance report from two snapshots. There are two reports available:

- The instance report (`spreport.sql` and `sprepins.sql`) is a general instance health report, covering all aspects of instance performance. The instance report calculates and prints ratios, increases, and so on for all statistics between the two snapshot periods.
- The SQL report (`sprepsql.sql` and `sprsqins.sql`) is a report for a specific SQL statement. The SQL report is usually run after you examine the high-load SQL sections of the instance health report. The SQL report displays SQL-specific statistics, the complete SQL text, and, if a level-6 (or higher) snapshot has been taken, information about any SQL plan(s) associated with that statement.

Both reports prompt for the beginning snapshot ID, the ending snapshot ID, and the report name. The SQL report also requests a hash value for the SQL statement to be reported on. You can configure each report by using parameters such as the total number of rows of SQL output to display in each SQL section or the number of days of snapshots to list. Separating the phase of data gathering from producing a report provides the flexibility of basing a report on any data points selected.

Note: The instance must not be shut down between the times that the beginning and ending snapshots are taken.

Statspack Considerations

- Set STATISICS_LEVEL = TYPICAL.
- Set TIMED_STATISTICS = TRUE to collect timing information.
- Avoid overlap between Statspack and AWR.
- Choose a sensible snapshot interval.
- Fix interval when ####### appears in the report output.
- Gather optimizer statistics on the PERFSTAT schema.

Statspack Considerations

In Oracle Database 10*g*, the STATISTICS_LEVEL parameter is set to TYPICAL by default. When STATISTICS_LEVEL is set to TYPICAL or ALL, Time Model data is populated by the server and can be captured by Statspack. To have the most useful data available for tuning, at a minimum you should set TIMED_STATISTICS to TRUE before the creation of a snapshot. Whenever STATISTICS_LEVEL is set to TYPICAL or ALL, TIMED_STATISTICS is automatically set to TRUE.

If STATISTICS_LEVEL is TYPICAL or ALL, and you do not use the Database Diagnostics pack, then you should disable AWR data capture by setting the AWR snapshot schedule interval to zero. If you use both Statspack and AWR, schedule the Statspack snapshots to avoid the times at which AWR is attempting to capture data. By default, AWR snapshots are taken every hour on the hour.

In general, snapshot data should be collected in intervals of an hour. In the unusual situation in which finer-grained analysis is required, the snapshot interval can be changed for the duration of that problem analysis. Snapshot intervals of less than 15 minutes are seldom useful.

Statspack Considerations (continued)

When a value is too large for a Statspack field, it is represented by a series of pound signs, such as `#######`. The main reason for this overflow is an analysis interval that is too large. Reduce the length of time between snapshots selected for the report, or choose snapshots that are closer in time.

For best performance when running the performance reports, optimizer statistics should be gathered on the Statspack schema. In Oracle Database 10*g*, the Oracle server automatically gathers optimizer statistics on database segments when the segments become stale. If you have disabled the automatic job, you should gather statistics manually by using the following command:

```
execute
dbms_stats.gather_schema_stats(ownname=>'PERFSTAT',cascade=>true);
```

Statspack and AWR Reports

The Statspack and AWR reports are designed to be used top-down.

- **Include the following on the first few pages:**
  - **Summary Information**
  - **Load Profile (useful with baseline)**
  - **Instance Efficiency (useful with baseline)**
  - **Top 5 Timed Events**
  - **Time Model**
- **Show detailed data on the following pages:**
  - **SQL**
  - **IO**
  - **Wait Statistics**

Statspack and AWR Reports

The Statspack and AWR reports are designed to be used top-down. The first few pages always indicate the biggest bottleneck to investigate. The Summary Information section shows the time interval being reported. Load Profile shows the level and type activity. This is very useful as a sanity check, to be sure that the workload has not changed significantly. The Instance Efficiency section gives the traditional ratios plus additional efficiency percentages that help identify and confirm the areas that need tuning.

The Top 5 Timed Events section identifies which events account for the most time by percentage. The Time Model section shows the time and the ratio of time consumed by a specific event to `DB_TIME`.

The Time Model entries that relate to the Top 5 Timed Events indicate the possible impact that tuning each event could have.

The subsequent pages of the Statspack and AWR reports give detailed information about a variety of database areas. These detailed sections are used to confirm diagnoses and plan remedies for the symptoms shown on the first pages.

Reading a Statspack or AWR Report

- **Start at summary data at the top:**
  - **Top 5 Timed Events**
  - **Wait Events and Background Wait Events**
  - **Wait Event Histogram (only in Statspack)**
  - **Load Profile (useful with baseline)**
  - **Instance Efficiency (useful with baseline)**
  - **Time Model**
- **Drill down to specific sections.**
  - **Indicated by top wait event**

CONFIRM

ORACLE

Reading a Statspack or AWR Report

When diagnosing a possible database problem, start by checking the host hardware and OS, the middle tier or the network, and other resources outside the database. If these are not being completely consumed, it is time to look at the database.

Start at the Top

Use the Top 5 Timed Events section to identify which events account for the most time by percentage. After these are identified, look at the Wait Events and Background Wait Events sections for the average wait time for the highest-ranking events. This data can sometimes provide insight into the scale of the wait. Compare the average with data in the Wait Event Histogram section. Is the average indicative of the typical wait time, or is the average wait time severely skewed by a small number of atypical waits?

Look at the Load Profile, Instance Efficiency, and Time Model sections. Pay attention to any statistics or ratios that are related to the top wait events. Is there a single consistent picture? If not, note other potential issues to investigate while looking at the top events, but do not be distracted from the top wait events. Scan the other statistics. Are there any statistics in the Load Profile that are unusually high for this site, or any ratios in the Instance Efficiency section that are atypical for this site (when compared to a baseline)? Next, drill down to the appropriate section in the Statspack report for additional data.

Statspack/AWR Report Drilldown Sections

Drilldown	Section
LC	Shared Pool Statistics
CPU	Host CPU
CPU	Instance CPU
MEM	Virtual Memory Paging
CPU	SQL ordered by CPU
IO	SQL ordered by Elapsed
CPU	SQL ordered by Gets
IO	SQL ordered by Reads
LC	SQL ordered by Parse Calls

CONT: Block contention
CPU: CPU consumption
ENQ: Enqueue
IO: I/O consumption
LAT: Latch contention
LC: Library cache
MEM: Memory consumption
PGAM: PGA memory consumption
RBS: Rollback segment
UNDO: Automatic undo
SP: Shared pool

Statspack/AWR Report Drilldown Sections

The relevant sections to examine are indicated by the top wait event.

The slide shows you a partial list of the most important Statspack sections indicating which sections can be used for possible drilldown. Each wait event falls into a type of problem. These types are listed in the slide. Use the type to determine the section to examine for more detailed information.

While drilling down, determine whether the data in these sections is consistent with the wait events. If it is not, identify related areas in the report for an alternative data source.

Analyze the detailed data in the drilldown sections after identifying candidate problems and resources showing contention. Consider whether there is sufficient data to build a plausible theory for the cause and resolution of the problem.

Use the *Oracle Database Performance Tuning Guide* as a resource to identify the cause and resolution of the contention. The manual includes detailed descriptions of how to diagnose causes and solutions of Wait Events and how to optimally configure the instance to avoid problems.

Statspack/AWR Report Drilldown Sections (continued)

Note

In Oracle Database 10*g*, many of the commonly requested resources have their own wait events, such as common latches and all enqueues. This makes it easier to identify contention. For example, library cache latch–related wait events would include "latch: library cache," "latch: library cache lock," and "latch: library cache pin."

Drilldown	Section
LC	SQL ordered by Sharable Memory
LC	SQL ordered by Version Count
IO	IO Sections
IOMEM	Buffer Cache
PGAMIO	PGA Memory Stats
PGAM	Process Memory Stats
ENQ	Enqueue activity
RBS UNDO	Rollback and Undo
LAT	Latch Statistics
CPU	Segments by Logical Reads
IO	Segments by Physical Reads
CONT	Segments by Row Lock Waits
CONT	Segments by ITL Waits
CONT	Segments by Buffer Busy Waits
LC	Dictionary Cache
LC	Library Cache Activity
STREA	Streams
SP LC	SGA breakdown difference
SP LC	SQL Memory Statistics

Report Drilldown: Examples

- If the top timed event is related to I/O waits, look at:
  - SQL ordered by Reads
  - SQL ordered by Elapsed
  - Tablespace IO Stats
  - File IO Stats
  - File IO Histogram
- If the top timed event is related to CPU usage, look at:
  - Load Profile
  - Time Model
  - SQL ordered by CPU
  - SQL ordered by Gets

Report Drilldown: Examples

If the top events are I/O related (db file sequential read and db file scattered read), look at the "SQL ordered by Reads" or "SQL ordered by Elapsed" sections, and the Tablespace IO Stats, File IO Stats, and File IO Histogram sections.

If the top timed event is CPU time, look at the number of logical reads in Load Profile and the top statistics in the Time Model data. Look at "SQL ordered by CPU" or "SQL ordered by Gets." Correlate the output. Does the evidence point to SQL execution?

Note: On a well-performing system, the top events are likely to be CPU time, db file scattered read, and db file sequential read.

Load Profile Section

- **Allows characterization of the application**
- **Can point toward potential problems:**
  - **High hard parse rate**
  - **High I/O rate**
  - **High login rate**
- **Is more useful if you have a comparable baseline**
- **Answers the question, “What has changed?”**
  - **Txn/sec change implies changed workload.**
  - **Redo size/txn implies changed transaction mix.**
  - **Physical reads/txn implies changed SQL or plan.**

```
Load Profile                          Per Second       Per Transaction
~~~~~~~~~~~~                     ---------------       ---------------
                   Redo size:         186,146.10                518.05
               Logical reads:           1,670.36                  4.65
               Block changes:           1,451.40                  4.04
              Physical reads:               0.01                  0.00
             Physical writes:               0.66                  0.00
                  User calls:               8.22                  0.02
                      Parses:              11.20                  0.03
                 Hard parses:               1.03                  0.00
                       Sorts:               9.01                  0.03
                      Logons:               0.20                  0.00
                    Executes:             382.20                  1.06
                Transactions:             359.32

  % Blocks changed per Read:   86.89    Recursive Call %:    99.39
 Rollback per transaction %:    0.01       Rows per Sort:     8.63
```


Load Profile Section

The Load Profile section indicates the nature of the workload on the system. It includes the key `V$SYSSTAT` statistics needed to characterize the system. From the Load Profile, you can determine the rates of:

- Logical and physical writes
- Logical and physical reads
- Hard and soft parses
- Session logons
- User activity

The values are shown per second (for absolute rates) and per transaction (for easier comparison with your application per-transaction rates to determine whether the application code, application use, or execution plans have changed).

It is thus helpful to know what is typical in a system. If, for example, you know the typical number of Logical reads/sec and Redo size/sec when the system performs well, you can look at these values and see whether they are any different when the system performs badly. To continue with the example, is the system doing twice the number of logical I/Os now than when the system performed well last week? Has the workload increased, or has the execution plan of key statements changed?

Time Model Section

- **The Time Model section lists a set of statistics that give an overview of where time is spent inside the Oracle database server.**
- **This section is reported from `V$SYS_TIME_MODEL`.**
- **Gauge the performance impact of any entity of the database, and then drill down.**

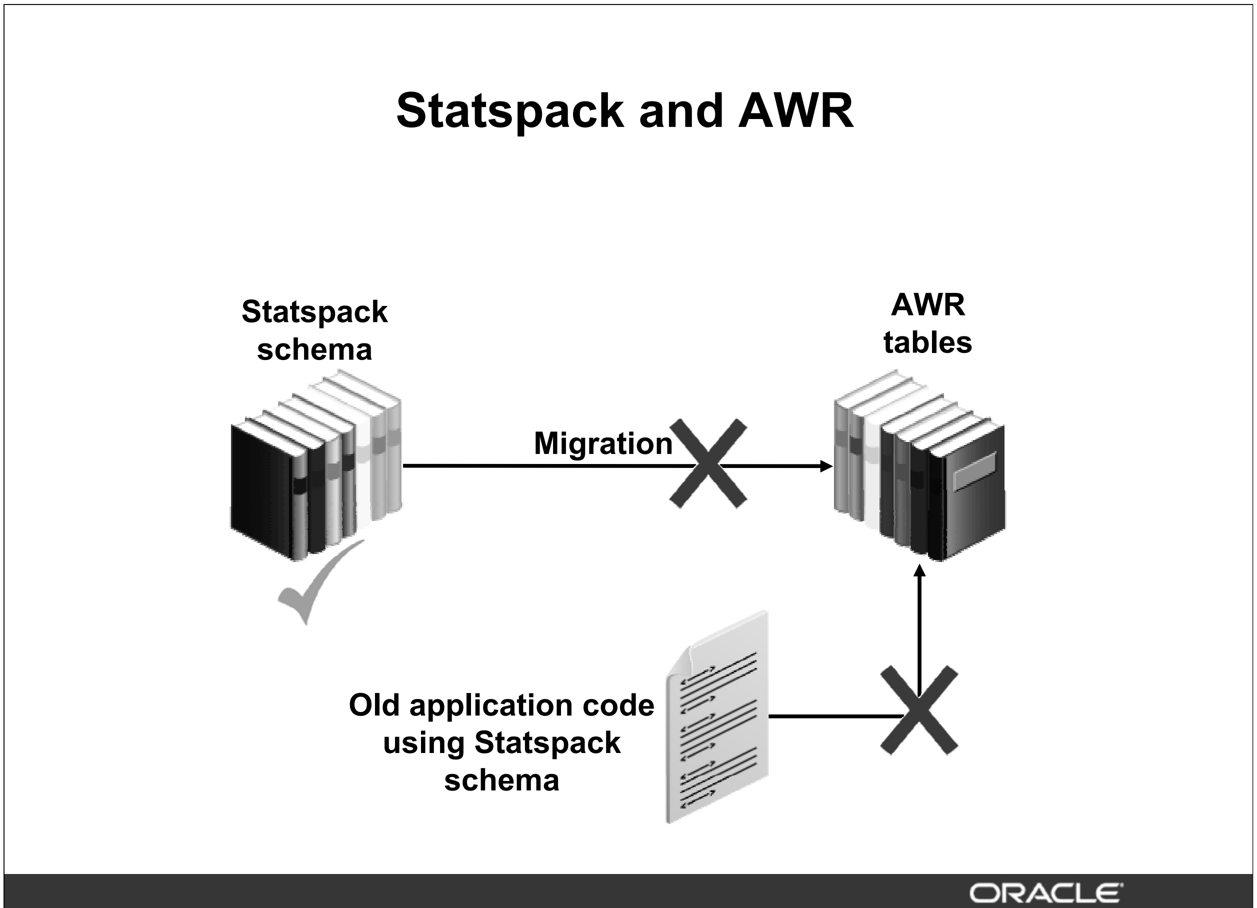
```
Time Model System Stats  DB/Inst: ORCL/orcl  Snaps: 1-2
-> Ordered by % of DB time desc, Statistic name

Statistic                                   Time (s) % of DB time
----------------------------------- ------------------ ------------
sql execute elapsed time                       620.7         98.8
DB CPU                                          78.1         12.4
PL/SQL execution elapsed time                   10.7          1.7
parse time elapsed                               7.9          1.3
hard parse elapsed time                          4.1           .7
connection management call elapsed              2.5           .4
PL/SQL compilation elapsed time                  0.4           .1
hard parse (sharing criteria) elaps              0.0           .0
repeated bind elapsed time                       0.0           .0
sequence load elapsed time                       0.0           .0
DB time                                        628.4
background elapsed time                         62.1
background cpu time                             13.6
          ------------------------------------------------------
```

ORACLE

Time Model Section

The Time Model section is captured from a view called V$SYS_TIME_MODEL. Time Model data is intended to be used to help identify which sections to drill down to. For example, in earlier releases, a strange set of wait events and latches would indicate that the application was performing an excess of connection management (logon/logoff activities). In earlier releases, only by knowing which latches to look for was it possible to track down this inefficient activity. The Time Model tracks time by different types of operations in the database, thus making it simpler to identify the types of operations that the instance spends its time performing.

Some of the statistics measure the CPU time used for an operation, and others measure the total elapsed time for that operation. The name of the statistic indicates which of these two it is measuring. Elapsed time includes wait time.



Statspack and AWR

In the past, historical data could be obtained manually by using Statspack. You can continue to use Statspack in Oracle Database 10*g*; however, if you want to use Automatic Workload Repository instead, you need to change your application code.

Statspack users should switch to Automatic Workload Repository in Oracle Database 10*g*.

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

Summary

In this lesson, you should have learned how to:

- Install Statspack
- Create Statspack snapshots
- Generate Statspack reports
- Identify the major sections of a Statspack report

ORACLE

Practice 5 Overview: Using Statspack

This practice covers the following topics:

- Installing Statspack
- Creating Statspack snapshots
- Generating Statspack reports

6

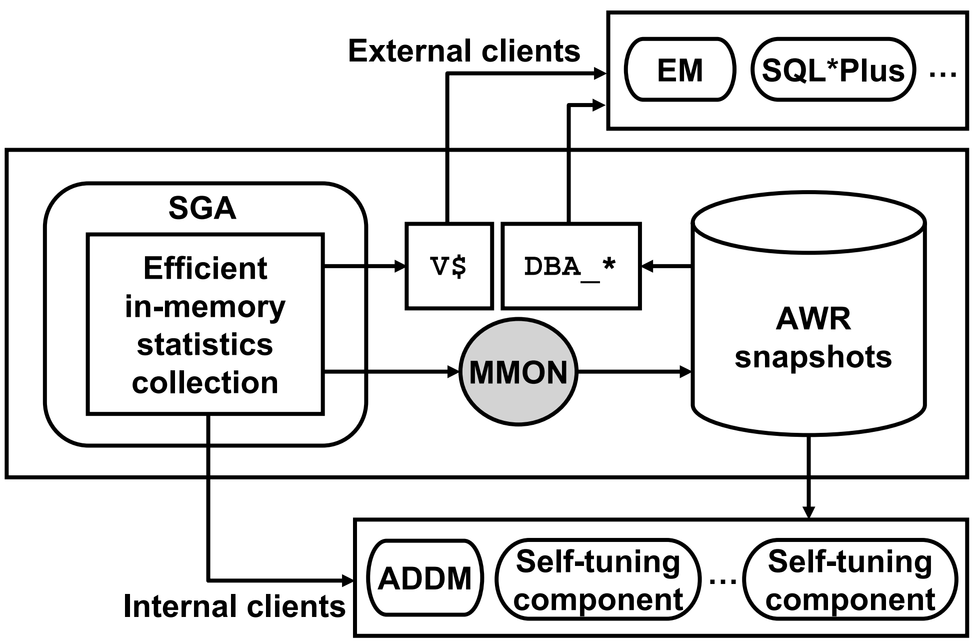
Using Automatic Workload Repository

Objectives

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

- Create and manage Automatic Workload Repository (AWR) snapshots
- Generate AWR reports
- Create snapshot sets and compare periods
- Generate Automatic Database Diagnostic Monitor (ADDM) reports
- Generate Active Session History (ASH) reports

Automatic Workload Repository: Overview



Automatic Workload Repository: Overview

Automatic Workload Repository (AWR) is the infrastructure that provides services to Oracle Database 10*g* components to collect, maintain, and use statistics for problem detection and self-tuning purposes. The AWR infrastructure consists of two major parts:

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

Statistics are stored in persistent storage for several reasons:

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

The memory version of the statistics is transferred to disk on a regular basis by a background process called MMON (Manageability Monitor).

With AWR, the Oracle database server provides a way to capture historical statistics data automatically, without the intervention of DBAs.

Automatic Workload Repository Data

- **Base statistics:**
  - **SQL and optimizer statistics**
  - **OS statistics**
  - **Wait classes**
  - **Time statistics**
- **Metrics**
- **Active Session History**
- **Advisor results**
- **Snapshot statistics**
- **Database feature usage**

```
V$SYSSTAT
V$SQL
V$SEGMENT_STATISTICS
V$SYS_TIME_MODEL
V$SYSMETRIC_HISTORY
V$SYSTEM_WAIT_CLASS
V$OSSTAT
V$ACTIVE_SESSION_HISTORY
```

↓

```
DBA_ADVISOR_*
DBA_HIST_*
DBA_FEATURE_*
DBA_HIGH_WATER_MARK_*
DBA_TAB_STATS_HISTORY
```

ORACLE

Automatic Workload Repository Data

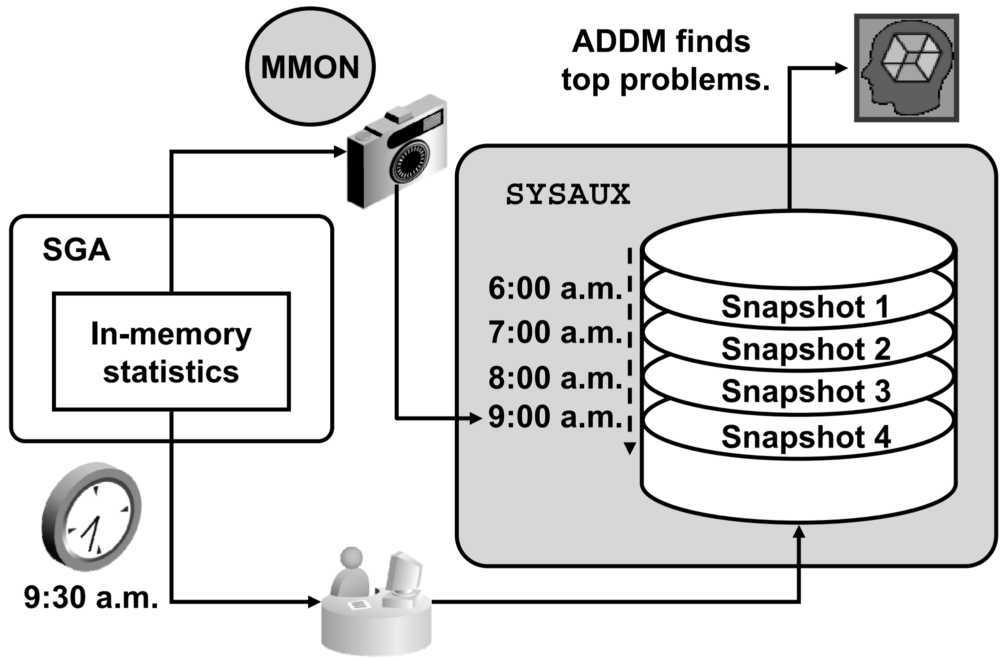
AWR provides a facility for collecting various statistics in memory. The following statistics are collected using the AWR collection facility:

- Object statistics that determine both access and usage statistics of database segments
- SQL statistics that are used to efficiently identify the top SQL statements on the basis of CPU, Elapsed, and Parse statistics
- The wait classes interface that is used for high-level performance analysis
- The time-model statistics that are based on how much time activities have spent
- Some of the statistics currently collected in `V$SYSSTAT` and `V$SESSTAT`
- Some of the Oracle optimizer statistics for self-learning and tuning
- Operating system statistics
- Active Session History (ASH), which represents the history of recent session activity
- Metrics that provides the rate of change for certain base statistics

Parts of these statistics are stored on disk in the form of snapshots. AWR provides dictionary views to look at those snapshot statistics. In addition, AWR stores the various advisor session results as well as database feature usage statistics.

Note: The list of views given in the slide is not an exhaustive list of statistics captured by AWR. For more information about these views, see the *Oracle Database Reference*.

Workload Repository



Workload Repository

The workload repository is a collection of persistent system-performance statistics owned by SYS. The workload repository resides in the SYSAUX tablespace and is one of the main SYSAUX occupants.

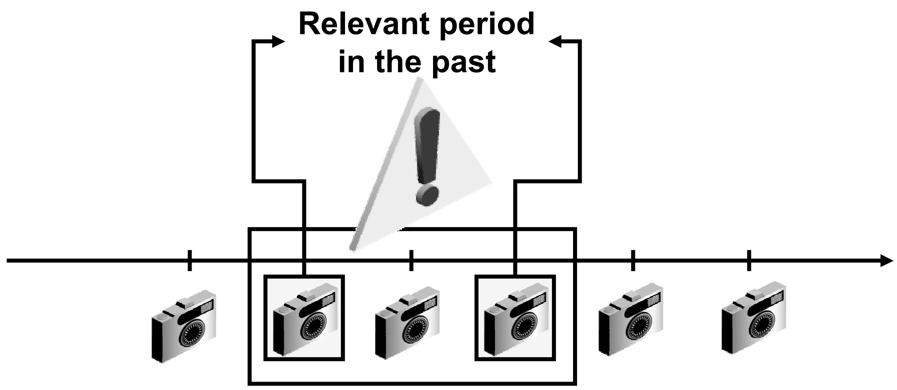
A *snapshot* is a set of performance statistics captured at a certain time. Snapshots are used for computing the rate of change of a statistic. Each snapshot is identified by a snapshot sequence number (snap_id) that is unique in the workload repository.

By default, snapshots are generated every 60 minutes. You can adjust this frequency by changing the snapshot INTERVAL parameter. Because internal advisories rely on these snapshots, be aware that adjustment of the interval setting can affect diagnostic precision. For example, if INTERVAL is set to 4 hours, you may miss spikes that occur within 60-minute intervals.

In a Real Application Clusters environment, each snapshot spans all nodes in a cluster. Snapshots for data in each node share the same snap_id, differentiated by their instance IDs. Snapshots in Real Application Clusters are captured at roughly the same time.

You can take manual snapshots by using Database Control. Taking manual snapshots is supported in conjunction with the automatic snapshots that the system generates. Manual snapshots are expected to be used when you want to capture the system behavior at two specific points in time that do not coincide with the automatic schedule.

AWR Snapshot Sets




```
DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE (
        start_snap_id IN NUMBER  ,
        end_snap_id   IN NUMBER  ,
        baseline_name IN VARCHAR2);
```


AWR Snapshot Sets

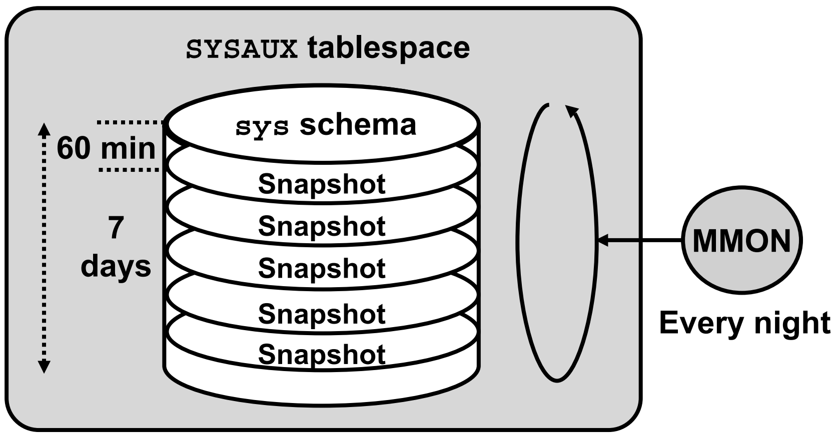
A snapshot set is the mechanism for you to tag and retain sets of snapshot data for important periods. A snapshot set is defined on a pair of snapshots; the snapshots are identified by their snapshot sequence numbers (`snap_ids`). Each snapshot set corresponds to one and only one pair of snapshots. Snapshot sets are used to retain snapshot data. Therefore, snapshots belonging to snapshot sets are retained until the snapshot sets are dropped.

A snapshot set can be identified by either a user-supplied name or a system-generated identifier. Execute the `DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE` procedure to create a snapshot set, and specify a name and a pair of snapshot identifiers. A snapshot set identifier that is unique for the life of a database is assigned to the newly created snapshot set. Usually you set up snapshot sets from some representative periods in the past, to be used for comparisons with current system behavior. You can also set up threshold-based alerts by using snapshot sets from Database Control.

You can get the `snap_ids` directly from `DBA_HIST_SNAPSHOT` or from Database Control.

Note: For more information about the `DBMS_WORKLOAD_REPOSITORY` package, see the *Oracle Database PL/SQL Packages and Types Reference*.

AWR Snapshot Purging Policy



AWR Snapshot Purging Policy

You control the amount of historical AWR statistics by setting a retention period and a snapshot interval. In general, snapshots are removed automatically in chronological order. Snapshots that belong to baselines are retained until their baselines are removed. On a typical system with 10 active sessions, AWR collections require 200 MB to 300 MB of space if the data is kept for seven days. The space consumption depends mainly on the number of active sessions in the system. A sizing script `utlsyxsz.sql` includes factors such as the size of the current occupants of the `SYSAUX` tablespace, number of active sessions, frequency of snapshots, and retention time. The `awrinfo.sql` script produces a report of the estimated growth rates of various occupants of the `SYSAUX` tablespace. Both scripts are located in the `$ORACLE_HOME/rdbms/admin` directory.

AWR handles space management for the snapshots. Every night the MMON process purges snapshots that are older than the retention period. If AWR detects that `SYSAUX` is out of space, it automatically reuses the space occupied by the oldest set of snapshots by deleting them. An alert is then sent to the DBA to indicate that `SYSAUX` is under space pressure.

AWR Snapshot Settings

```
DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS (
        retention IN NUMBER DEFAULT NULL,
        interval  IN NUMBER DEFAULT NULL,
        topnsql   IN NUMBER DEFAULT NULL);
```

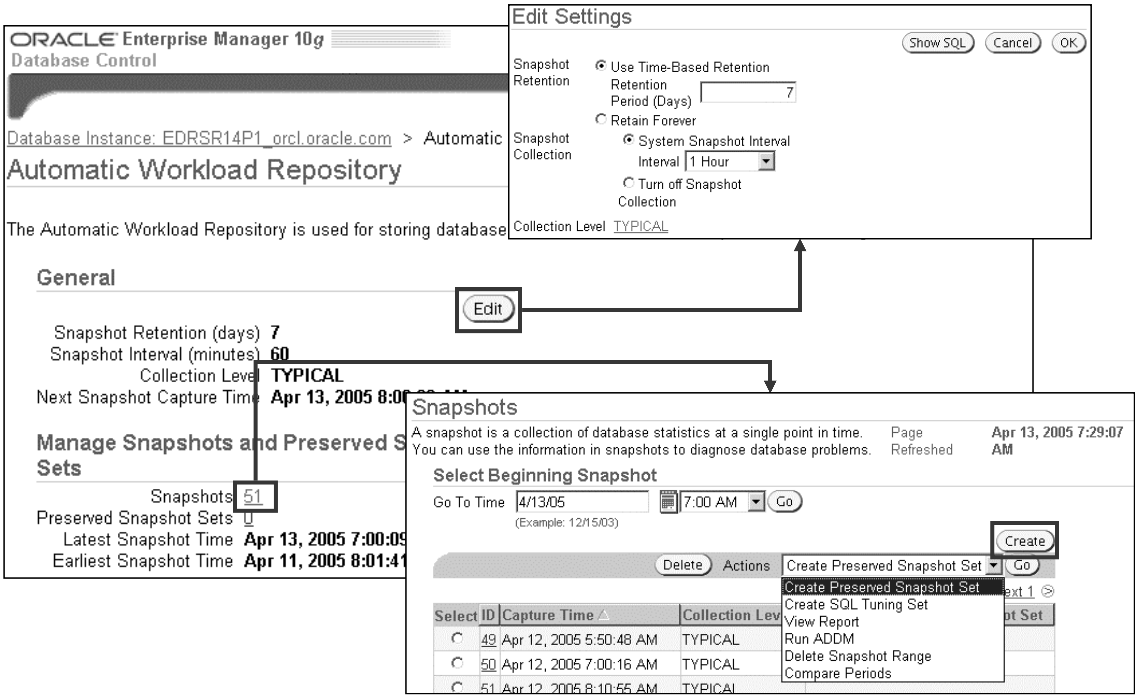

AWR Snapshot Settings

With the MODIFY_SNAPSHOT_SETTINGS procedure, you can control the snapshot parameters. You can use this procedure to change:

- **The retention period.** RETENTION is specified in minutes. The default is seven days (10,080 minutes); the minimum is one day. Setting RETENTION to the value 0 disables automatic purging.
- **The INTERVAL between snapshots.** The minimum value is 10 minutes, the maximum is 100 years, and the default value is 60 minutes.
- **The number of Top SQL statements for which to capture performance data.** You are allowed to specify the following values: DEFAULT, MAXIMUM, and *n* (where *n* is the number of Top SQL statements to flush for each SQL criterion such as Elapsed Time and CPU Time). Specify DEFAULT to capture the Top 30 for the TYPICAL level and Top 100 for the ALL level of STATISTICS_LEVEL. Specify MAXIMUM to capture the complete set of SQL in the cursor cache. Specify NULL to keep the current setting.

Note: Under exceptional circumstances, automatic snapshot collection can be completely disabled setting the snapshot interval to 0. The automatic collection of the workload and statistical data is stopped and much of the Oracle self-management functionality is not operational. In addition, you are unable to manually create snapshots. For this reason, Oracle Corporation strongly recommends that you do not disable automatic snapshot collection.

Database Control and AWR



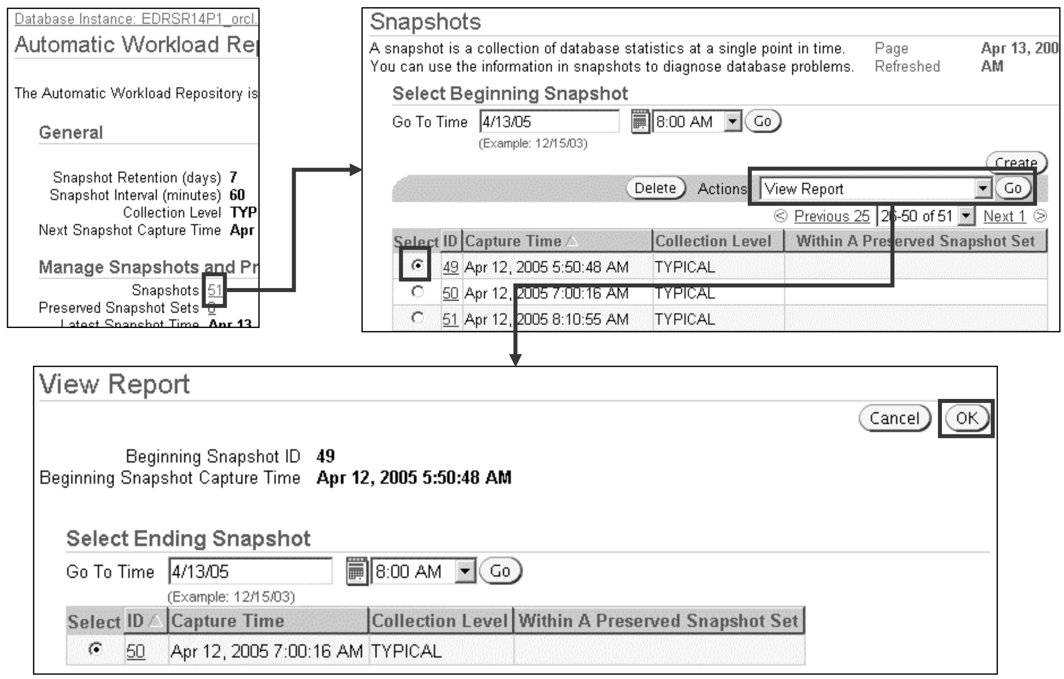
Database Control and AWR

Using Database Control, you can configure the `RETENTION` and `INTERVAL` parameters for capturing snapshots. To access the Automatic Workload Repository page, click the Administration tab on the Database Control home page. Then click the Automatic Workload Repository link in the Statistics Management section.

Using the Automatic Workload Repository page, you can:

- Edit the workload repository settings
- Look at the detailed information of created snapshots and manually create new ones
- Create baselines (also called preserved snapshot sets)
- Generate an AWR report

Generating AWR Reports in EM



Generating AWR Reports

AWR can produce a summary report on statistics stored in the workload repository similar to Statspack. The report contains general information about the overall behavior of the system over a time period defined by two snapshots.

To generate an AWR report, you can use the Automatic Workload Repository page in Database Control. On this page, click the link corresponding to the number of snapshots. This opens the Snapshots page. On the Snapshots page, select the beginning snapshot, select View Report from the Actions drop-down list, and then click Go. On the View Report page, select the ending snapshot and click OK.

Generating AWR Reports in SQL*Plus

WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
ORCL	1090770270	orcl	1	10.2.0.0.0	NO	edrsr14p1

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	140	21-Mar-05 23:00:46	25	10.8
End Snap:	141	22-Mar-05 00:00:13	27	12.4
Elapsed:		59.46 (mins)		
DB Time:		0.78 (mins)		

Report Summary

Cache Sizes

	Begin	End		
Buffer Cache:	44M	44M	Std Block Size:	8K
Shared Pool Size:	208M	208M	Log Buffer:	256K

Load Profile

ORACLE

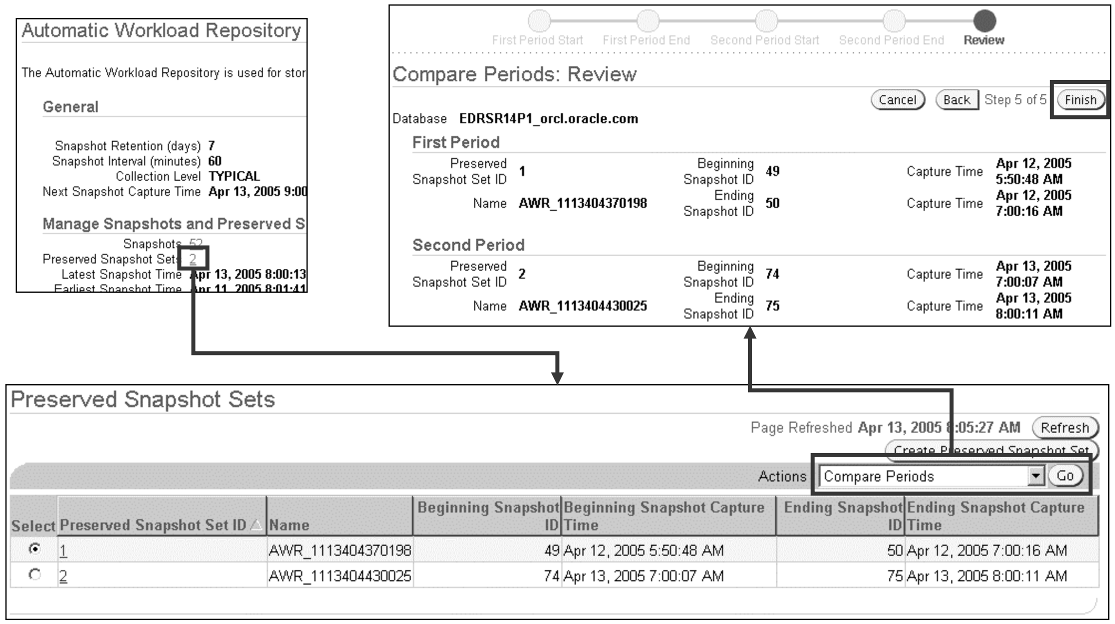
Generating AWR Reports in SQL*Plus

Alternatively, you can use the awrrpt.sql SQL*Plus script in the ORACLE_HOME/rdbms/admin directory. The script should be run when connected as SYSDBA. The script prompts you for report options:

- HTML or a text report
- The number of days of snapshots to choose from: Entering the number of days shows you the most recent snapshots being taken. You can also determine which SNAP_IDs you should use by querying the DBA_HIST_SNAPSHOT table to retrieve the mapping between a SNAP_ID and the actual wall-clock time.
- Begin SNAP_ID, end SNAP_ID: A snapshot pair that defines the reporting time period
- File name: The user-specified file into which the report is written

Next, use your favorite browser to view the report. The report holds the same information whether it is produced as a text or as an HTML report. The advantage of an HTML report is the presence of links to the detailed sections of the report.

Snapshot Sets and Period Comparisons



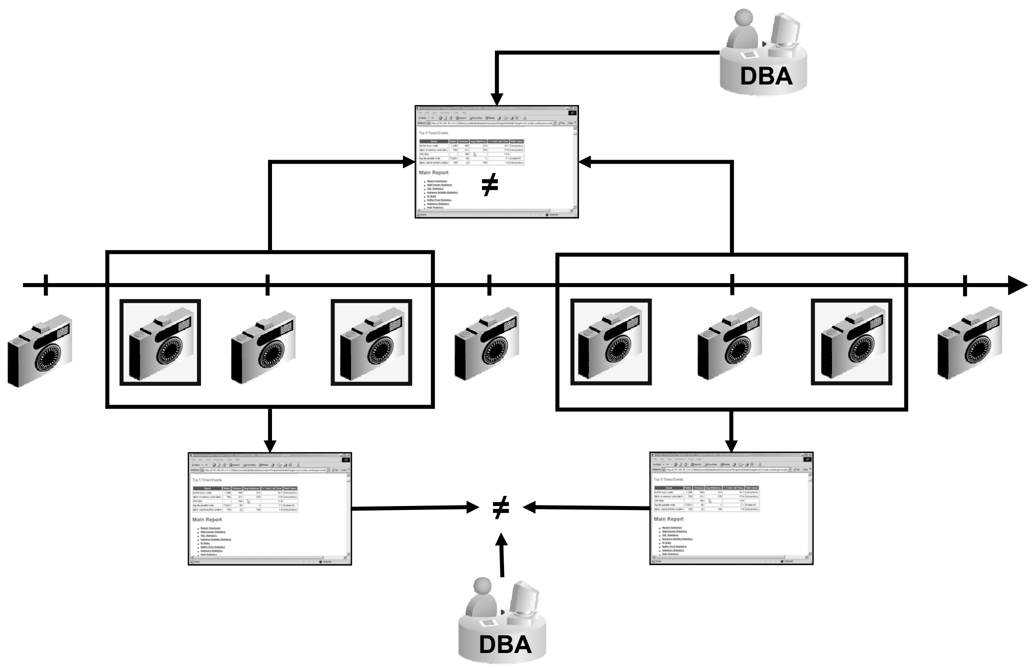
ORACLE

Snapshot Sets and Period Comparisons

Oracle Database 10*g* includes preserved snapshot sets, which are simply pairs of snapshots. In Enterprise Manager, navigate to the Administration tabbed page, and then click Automatic Workload Repository. On the Automatic Workload Repository page, you can click either the number of snapshots to choose snapshots to preserve as sets or the number of snapshot sets. By clicking the snapshot number, you can perform various operations on the snapshots, including performing a Compare Periods operation.

A Compare Periods operation enables you to define two different time periods and compare their respective Active Session History (ASH) data sets. You can simplify the process if you define snapshot sets first. After at least two snapshot sets are created, you can click the number of snapshot sets on the Automatic Workload Repository page. From this point, you can perform the Compare Periods operation. Just follow the Compare Periods Wizard to select both the snapshot sets, and then click Finish.

Compare Periods: Benefits



Compare Periods: Benefits

You can use the Workload Repository Compare Period Report to compare two periods in AWR. Whereas an AWR report shows AWR data between two snapshots (or two points in time), the Workload Repository Compare Period Report shows the difference between two periods (or two AWR reports, which equates to four snapshots).

Use the Workload Repository Compare Period Report to identify detailed performance attributes and configuration settings that are different between two time periods. For example, if the application workload is known to be stable for a given time of day, but the performance on Tuesday was poor between 10:00 a.m. and 11:00 a.m., then generating a Workload Repository Compare Period Report for Tuesday from 10:00 a.m. to 11:00 a.m. and Monday from 10:00 a.m. to 11:00 a.m. should identify configuration settings, workload profile, and statistics that were different between these two time periods. Based on the changes reported between these two time periods, the cause of the performance degradation can be accurately diagnosed. The two time periods selected for the Workload Repository Compare Period Report can be of different duration because the report normalizes the statistics by the amount of time spent on the server for each time period and presents statistical data ordered by the largest difference between the periods.

Compare Periods: General

First Period

Beginning Snapshot ID **538** Beginning Snapshot Capture Time **Sep 2, 2005 2:10:56 AM**
Ending Snapshot ID **539** Ending Snapshot Capture Time **Sep 2, 2005 2:20:20 AM**

Second Period

Beginning Snapshot ID **543** Beginning Snapshot Capture Time **Sep 2, 2005 3:00:57 AM**
Ending Snapshot ID **544** Ending Snapshot Capture Time **Sep 2, 2005 3:10:05 AM**

General Report

View Data: Per Second

Previous 1-27 of 27 Next

Name △	First Period Metric Ratio	Second Period Metric Ratio	First Period Value	Second Period Value	First Period Rate Per Second	Second Period Rate Per Second
DB cpu (seconds)			0.00	0.00	0.00	0.00
DB time (seconds)			2,560.81	7,643.77	4.54	13.95
db block changes			1,477,241.00	1,931,544.00	2,619.22	3,524.72
execute count			374,188.00	488,518.00	663.45	891.46
global cache cr block receive time (seconds)			0.00	0.00	0.00	0.00
global cache cr blocks received			0.00	0.00	0.00	0.00
global cache current block receive time (seconds)			0.00	0.00	0.00	0.00
global cache current blocks received			0.00	0.00	0.00	0.00
global cache get time (seconds)			0.00	0.00	0.00	0.00
global cache gets			0.00	0.00	0.00	0.00
opened cursors cumulative			8,962.00	6,151.00	15.89	11.22
parse count (total)			7,569.00	4,143.00	13.42	7.56
parse time cpu (seconds)			6.58	7.18	0.01	0.01
parse time elapsed (seconds)			11.25	21.41	0.02	0.04
physical reads			64.00	51.00	0.11	0.09
physical writes			7,513.00	6,219.00	13.32	11.35

Compare Periods: General

This slide shows a portion of the results of the Compare Periods operation, which identifies statistical differences between two snapshot periods. The bar graphs indicate the proportional number for that metric compared to the other time period. For example, this report shows that there were significantly more "db block changes" in the second period. On the General tabbed page, you can also display the general statistics per transaction instead of per second as shown in the slide. Simply select the corresponding value in the View Data field.

Clicking the Report link on this page displays an HTML report comparing the two periods, showing differences in areas such as wait events, OS statistics, services, SQL statistics, instance activity, IO statistics, segment statistics, and so on.

Note: If the sizes of the two time periods are different, the data is normalized over `DBTIME` before calculating the difference so that periods of different lengths can be compared.

Compare Periods: Report

General | Report

WORKLOAD REPOSITORY COMPARE PERIOD REPORT

Snapshot Set	DB Name	DB Id	Instance	Inst num	Release	Cluster	Host
First (1st)	ORCL	1090770270	orcl	1	10.2.0.1.0	NO	edrsr14p1
Second (2nd)	ORCL	1090770270	orcl	1	10.2.0.1.0	NO	edrsr14p1

Snapshot Set	Begin Snap Id	Begin Snap Time	End Snap Id	End Snap Time	Elapsed Time (min)	DB Time (min)	Avg Active Users
1st	538	02-Sep-05 02:10:56	539	02-Sep-05 02:20:20	9.40	15.58	1.66
2nd	543	02-Sep-05 03:00:57	544	02-Sep-05 03:10:05	9.15	100.20	10.66

Configuration Comparison

	1st	2nd	%Diff
Buffer Cache:	180M	180M	0.00
Std Block Size:	8K	8K	0.00
Shared Pool Size:	80M	80M	0.00
Log Buffer:	2,904K	2,904K	0.00
SGA Target:	285M	285M	0.00
PGA Aggregate Target:	94M	94M	0.00
Undo Management:	AUTO	AUTO	

Load Profile

	1st Per Sec	2nd Per Sec	%Diff	1st Per Txn	2nd Per Txn	%Diff
Redo size:	334,301.82	426,211.90	27.49	519.72	487.27	-6.24
Logical reads:	2,841.36	3,963.36	39.49	4.42	4.53	2.49
Block changes:	2,618.95	3,519.02	34.37	4.07	4.07	0.00
Physical reads:	0.11	0.09	-18.18	0.00	0.00	0.00
Physical writes:	13.32	11.33	-14.94	0.02	0.01	-50.00
User calls:	4.52	5.23	15.71	0.01	0.01	0.00
Parses:	13.42	7.55	-43.74	0.02	0.01	-50.00
Hard parses:	0.62	0.67	8.06	0.00	0.00	0.00
Sorts:	3.46	4.46	28.90	0.01	0.01	0.00
Logons:	0.11	0.11	0.00	0.00	0.00	0.00
Executes:	663.38	890.02	34.16	1.03	1.02	-0.97
Transactions:	643.24	874.69	35.98			

	1st	2nd	Diff
% Blocks changed per Read:	92.17	88.79	-3.38
Recursive Call %:	99.78	99.81	0.03
Rollback per transaction %:	0.00	0.00	-0.00
Rows per Sort:	7.83	7.49	-0.34
Avg Length of Calls (sec):	0.37	0.37	0.00

Top 5 Timed Events

1st					2nd				
Event	Waits	Time(s)	Percent Total DB Time	Wait Class	Event	Waits	Time(s)	Percent Total DB Time	Wait Class
CPU time		373.1	39.90		buffer busy waits	6,773	2,491.0	41.43	Concurrency
log file parallel write	253,437	194.9	20.85	System I/O	latch: In memory undo latch	4,272	1,123.2	18.68	Concurrency
buffer busy waits	754	83.9	8.98	Concurrency	CPU time		429.1	7.14	
latch: In memory undo latch	637	79.8	8.53	Concurrency	log file parallel write	26,705	377.6	6.28	System I/O
latch: cache buffers chains	87	10.0	1.07	Concurrency	latch: cache buffers chains	655	144.0	2.39	Concurrency

ORACLE

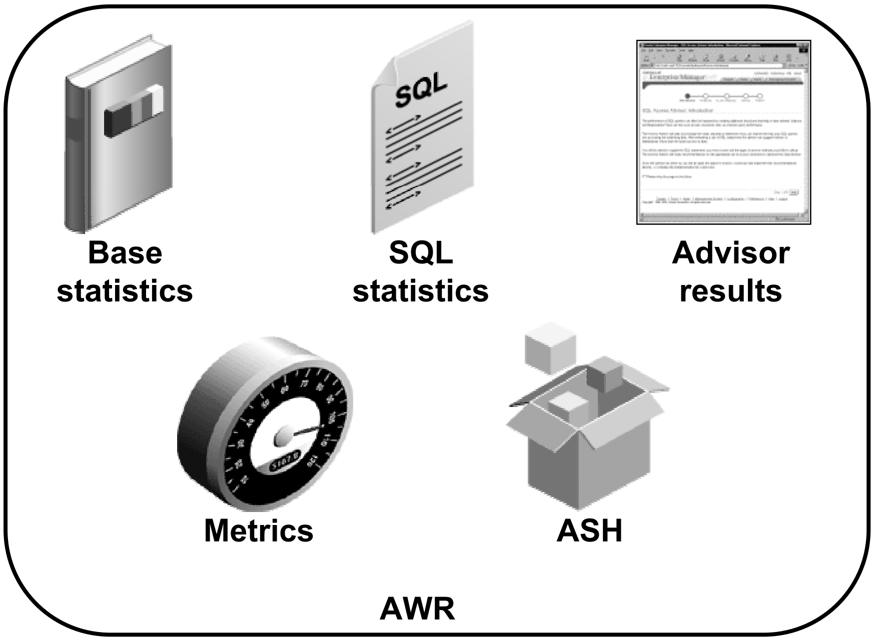
Compare Periods: Report

When you click the Report tab on the Compare Period: Results page, you generate the Workload Repository Compare Period Report. This report contains the same sections as the ones you can find in a Statspack/AWR report. However, the Compare Period report shows you a configuration comparison for both time periods in each of the sections.

Some of the sections of the report are shown in the slide.

Note: You can also generate such a report by using the `awrddrpt.sql` script located in your `$ORACLE_HOME/rdbms/admin` directory.

AWR Data



AWR Data

AWR captures a variety of statistics. AWR stores base statistics, that is, counters and value statistics (for example, log file switches and process memory allocated). AWR captures SQL statistics such as disk reads per SQL statement. Metrics such as physical reads per minute are also captured.

The Active Session History (ASH) data is captured first to memory at one-second intervals for only sessions that are currently active (performing a database call). Then the ASH data is reduced by a factor of ten by storing to disk a random sample of the in-memory data. The ASH data is heavily used by Automatic Database Diagnostic Monitor (ADDM) to identify root causes of performance issues.

The advisor reports produced by ADDM, the Segment Advisor, and other advisors are also stored in AWR for later viewing.

Note: The examples on this page do not represent the complete list.

DBMS_WORKLOAD_REPOSITORY Package

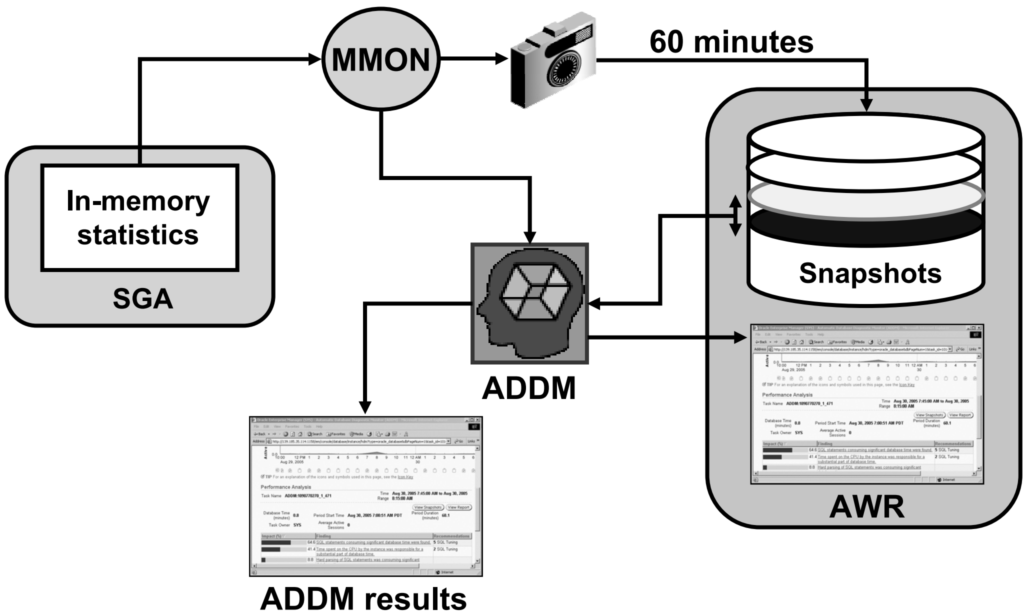
Procedure Name	Description
CREATE_SNAPSHOT	Procedure to create a manual snapshot immediately
DROP_SNAPSHOT_RANGE	Procedure to drop a range of snapshots
CREATE_BASELINE	Procedure to create a single AWR baseline
DROP_BASELINE	Procedure to drop a single AWR baseline
MODIFY_SNAPSHOT_SETTINGS	Procedure to modify the snapshot settings

DBMS_WORKLOAD_REPOSITORY Package

The DBMS_WORKLOAD_REPOSITORY PL/SQL package contains procedures that enable you to manage the workload repository. For example, you can find procedures for managing snapshots and baselines in this package.

Note: For more information about these procedures, refer to the *Oracle Database PL/SQL Packages and Types Reference*.

ADDM Performance Monitoring



ORACLE

ADDM Performance Monitoring

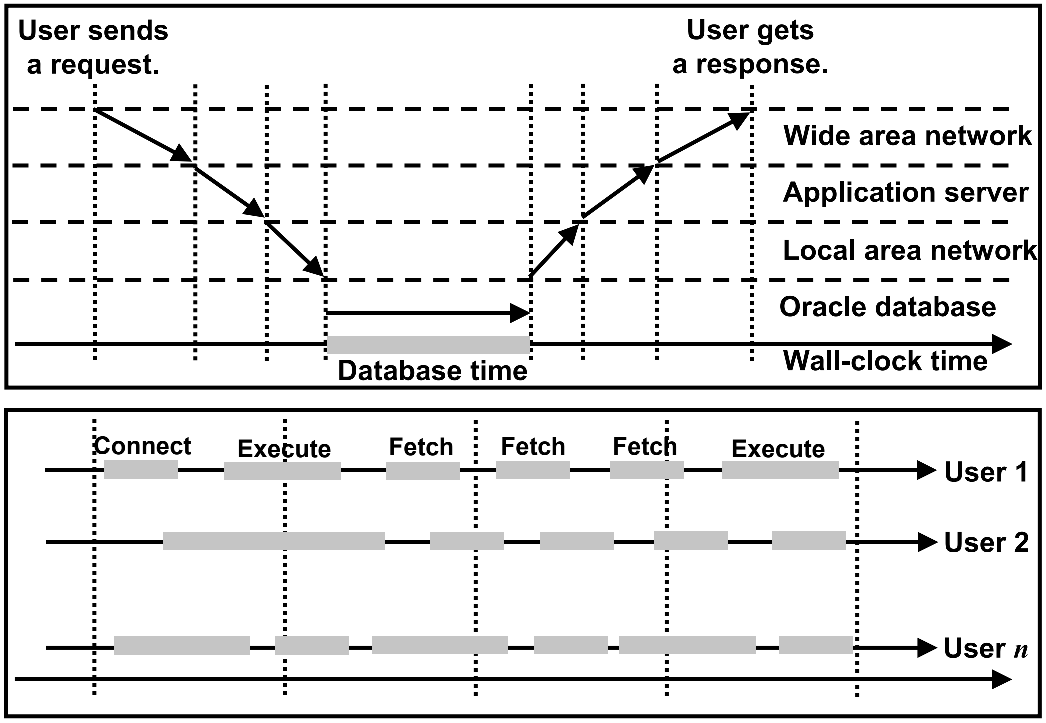
By default, the Oracle database server automatically captures statistical information from the SGA every 60 minutes and stores it in Automatic Workload Repository (AWR) in the form of snapshots. These snapshots are stored on disk and are similar to Statspack snapshots. However, they contain more precise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

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

The results of each ADDM analysis are stored in Automatic Workload Repository and are also accessible through Database Control.

Note: Although ADDM analyzes Oracle database performance over the period defined by the last two snapshots, it is possible to manually invoke an ADDM analysis across any two snapshots.

ADDM and Database Time



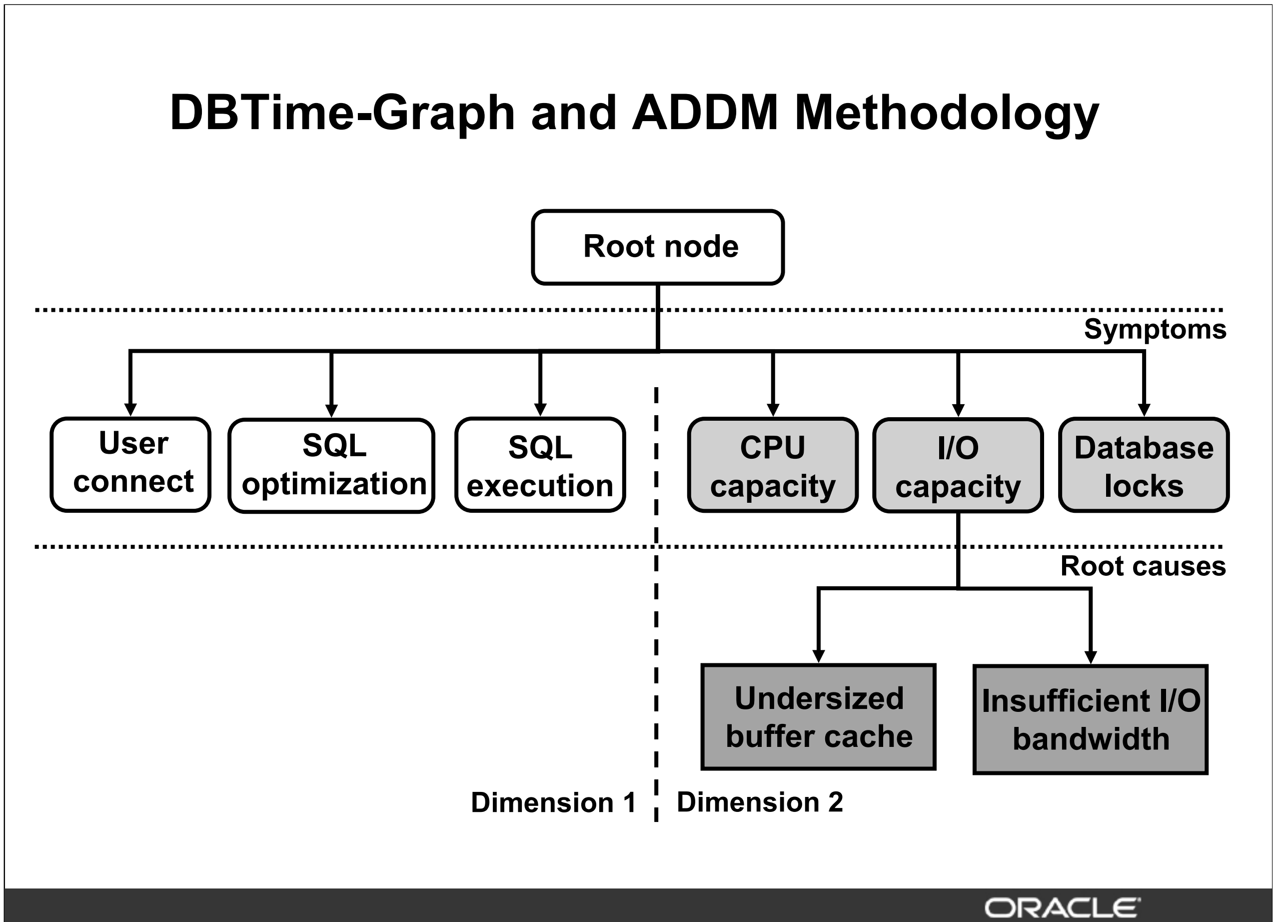
ADDM and Database Time

Database time is defined as the sum of the time spent inside the database processing user requests. The upper graphic in the slide illustrates a simple scenario of a single user submitting a request. The user's response time is the time interval between the instant the request is sent and the instant the response is received. The database time involved in that user request is only a portion of that user's response time that is spent inside the database.

The lower graphic in the slide illustrates database time as it is summed over multiple users, and each user is performing a series of operations resulting in a series of requests to the database. You can see that the database time is directly proportional to the number and duration of user requests and can be higher or lower than the corresponding wall-clock time.

Using the database time as a measure, you can gauge the performance impact of any entity of the database. For example, the performance impact of an undersized buffer cache would be measured as the total database time spent in performing additional I/O requests that could have been avoided if the buffer cache was larger.

Database time is simply a measurement of the total amount of work done by the database server. The rate at which the database time is consumed is the database load average, measured as database time per second. The objective of ADDM is to reduce the amount of database time spent on a given workload, which is analogous to consuming less energy to perform the same task.



DBTime-graph and ADDM Methodology

Identifying the component that contributes the most database time is equivalent to finding the single database component that provides the greatest benefit when tuned. ADDM uses database time to identify database components that require investigation and also to quantify performance bottlenecks. The first step in automatic performance tuning is to correctly identify the causes of performance problems. You can explore effective tuning recommendations to solve or alleviate the issue only when the root cause of the performance problem is correctly identified. ADDM looks at the database time spent in two independent dimensions:

- The first dimension looks at the database time spent in various phases of processing user requests. This dimension includes categories such as “connecting to the database,” “optimizing SQL statements,” and “executing SQL statements.”
- The second dimension looks at the database time spent using or waiting for various database resources used in processing user requests. The database resources considered in this dimension include both hardware resources, such as CPU and I/O devices, and software resources, such as database locks and application locks.

To perform automatic diagnosis, ADDM looks at the database time spent in each category under both these dimensions and drills down to the categories that had consumed significant database time. This two-dimensional drilldown process can be represented by using the DBTime-graph.

DBTime-graph and ADDM Methodology (continued)

Performance problems often distribute database time across many categories in one dimension but not in the other. For example, a database with insufficient CPU capacity slows down all phases involved in processing user requests, which is in the first dimension of the ADDM analysis. However, it is evident from the second dimension that the top performance problem affecting the database is insufficient CPU capacity. This two-dimensional view of determining where database time is consumed gives ADDM very good judgment in zooming in to the more significant performance issues.

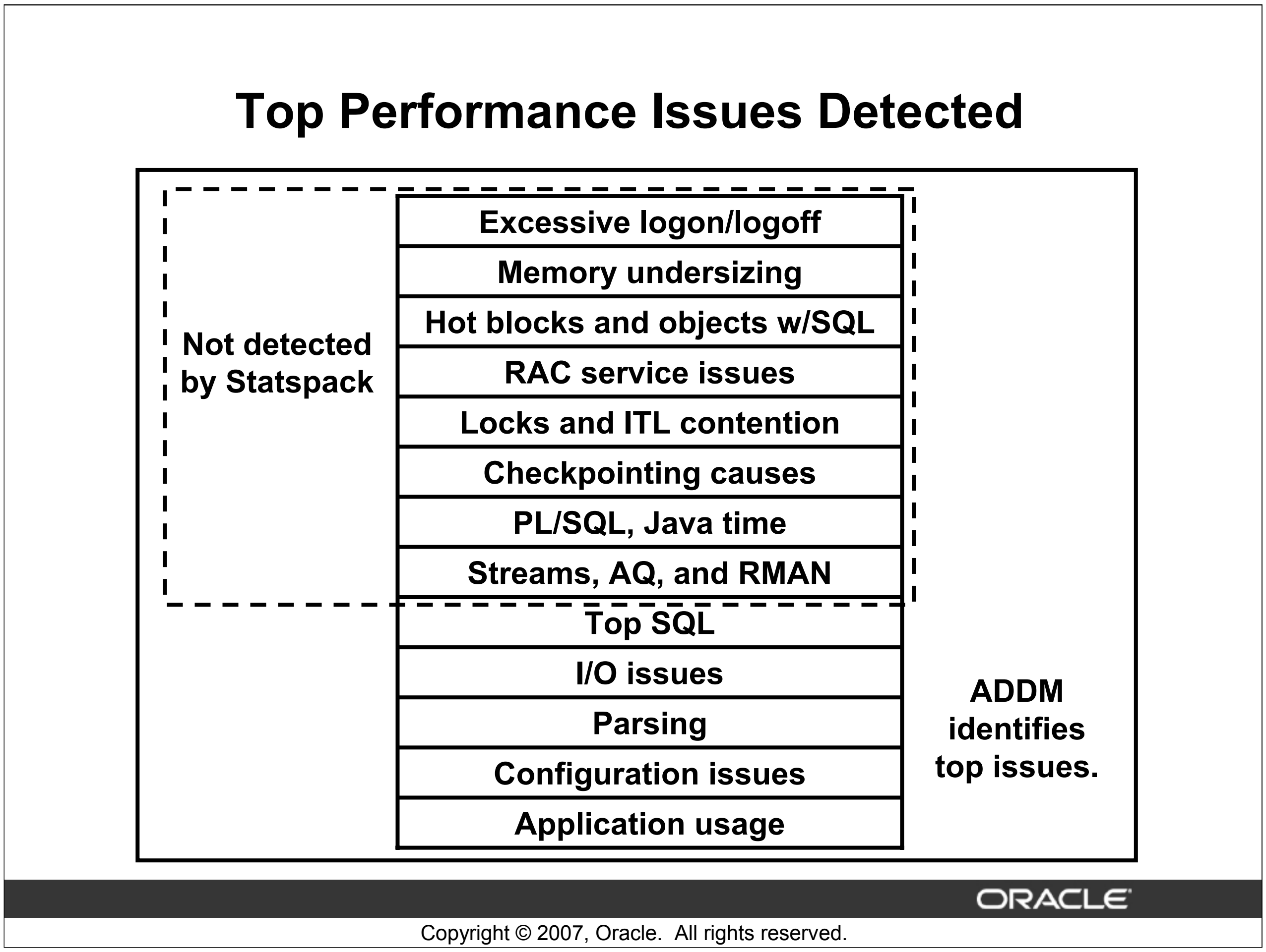
ADDM explores this DBTime-graph starting at the root node and all the children of any node where the database time consumed is significant. Branch nodes in this graph identify the performance impact of what is usually a symptom of some performance bottleneck. The terminal nodes identify particular root causes that can explain all the symptoms that were significant along the path in which the terminal node was reached. For example, in the slide, the branch node "I/O Capacity" measures database time spent in all I/O requests, which is significant due to various bottlenecks. Whenever significant database time is spent in I/O requests, all the children of the "I/O Capacity" node are explored. They correspond to the two terminal nodes on the slide. The "Undersized Buffer Cache" node points to a particular root cause: The data-block buffer cache is undersized causing excessive number of I/O requests. The "Insufficient I/O Bandwidth" node looks for hardware issues that can slow down all I/O requests.

After a terminal node identifies a root cause, ADDM measures the impact of the root cause in database time. It then explores ways that can solve or alleviate the problem identified. ADDM uses the various metrics and statistics measurements collected by AWR to come up with tuning recommendations. The system maintains measurements at various granularities, so an ADDM analysis can go from the symptoms (for example, commit operations consumed significant database time) to the root cause (for example, write I/O to one of the log files was very slow—possibly a hardware issue). The nodes also allow ADDM to estimate the maximum possible database time that can be saved by the suggested tuning recommendations, which is not necessarily equal to the database time attributed to the root cause.

In addition to identifying bottlenecks, ADDM also identifies key components that are not experiencing any performance bottlenecks. The idea is to prevent you from tuning components that have marginal effect on the total database throughput.

It is interesting to note that ADDM need not traverse the entire DBTime-graph. It can prune the uninteresting subgraphs. This can be achieved only because the DBTime-graph is constructed in a way that a node's database time is contained in the database time attributed to its parents. By pruning and not traversing uninteresting subgraphs (which represent database components that are not consuming significant database time), the cost of an ADDM analysis depends only on the number of actual performance problems that were affecting the database. The cost of the analysis does not depend on the actual load on the database or on the number of issues that ADDM could potentially diagnose.

At the end of the analysis, ADDM reports the top root causes identified, ranked by the performance impact attributed with each root cause along with the respective tuning recommendations.



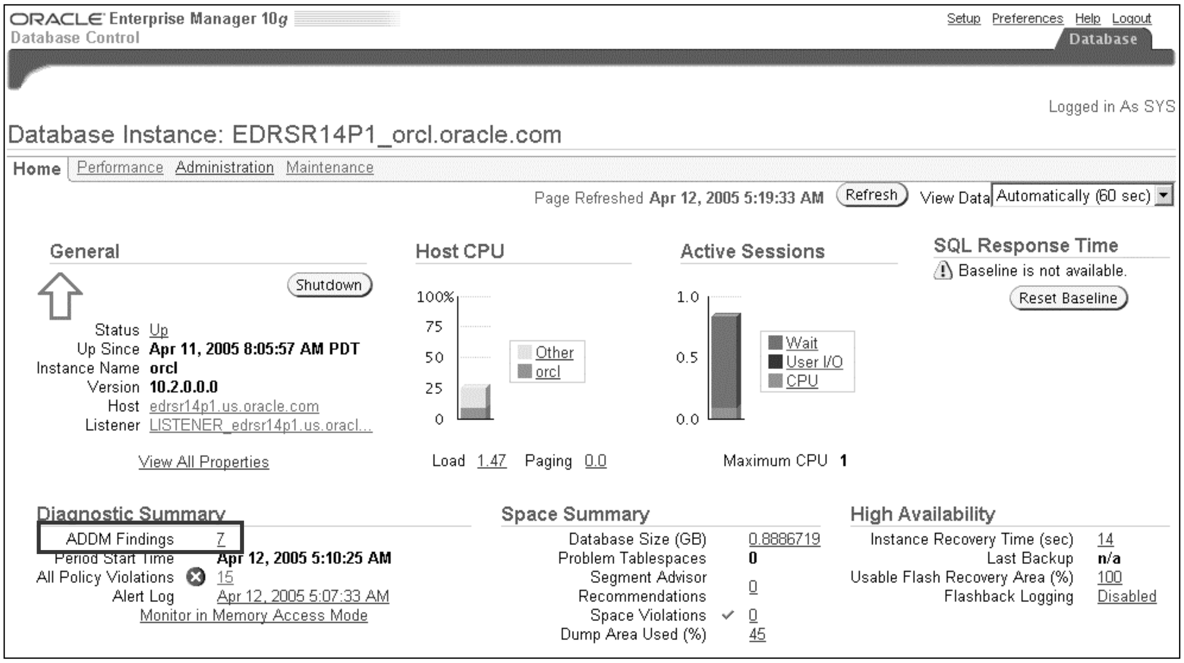
Top Performance Issues Detected

With previous releases of the Oracle database, Statspack was unable to identify some of the problems listed in the slide because of lack of granularity in the statistics. With the introduction of the wait-and-time statistics model in Oracle Database 10*g*, ADDM can now identify the top performance issues listed. Another benefit of ADDM over Statspack is that ADDM concentrates its analysis on top problems (the ones that impact the system most).

In Oracle Database 10*g* Release 2, ADDM has been further enhanced to make its analysis more accurate in areas related to CPU, Paging, and Integrated Cache. In addition, the scope of ADDM has now been broadened to add more server components, such as Streams, AQ, and RMAN.

Note: The list of performance issues in the slide is not a comprehensive list of issues that are automatically detected by ADDM. In fact, it represents only a subset of the potential issues that ADDM can discover.

Database Control and ADDM Findings



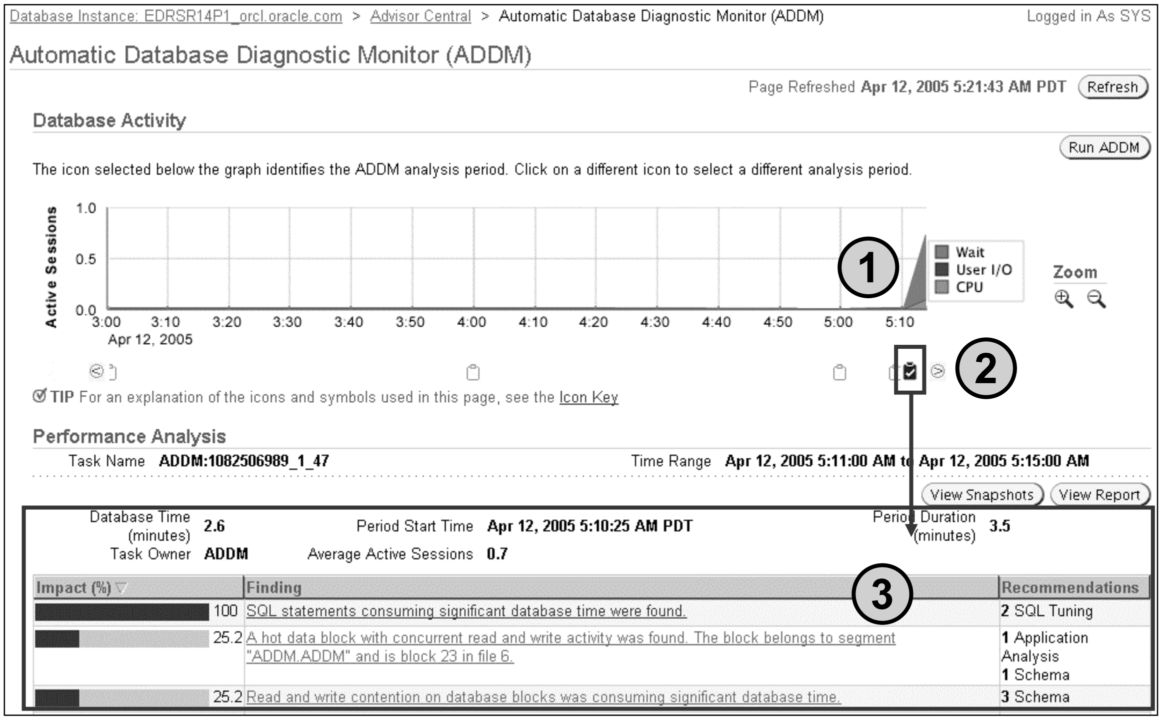
ORACLE

Database Control and ADDM Findings

On the Database Control home page for your database, you can see the Diagnostic Summary section, which gives you the number of ADDM findings from the previous automatic run.

By clicking the Performance Findings link, you are directed to the Automatic Database Diagnostic Monitor (ADDM) page, where you can access the details of the latest ADDM run.

ADDM Analysis Results



ORACLE

ADDM Analysis Results

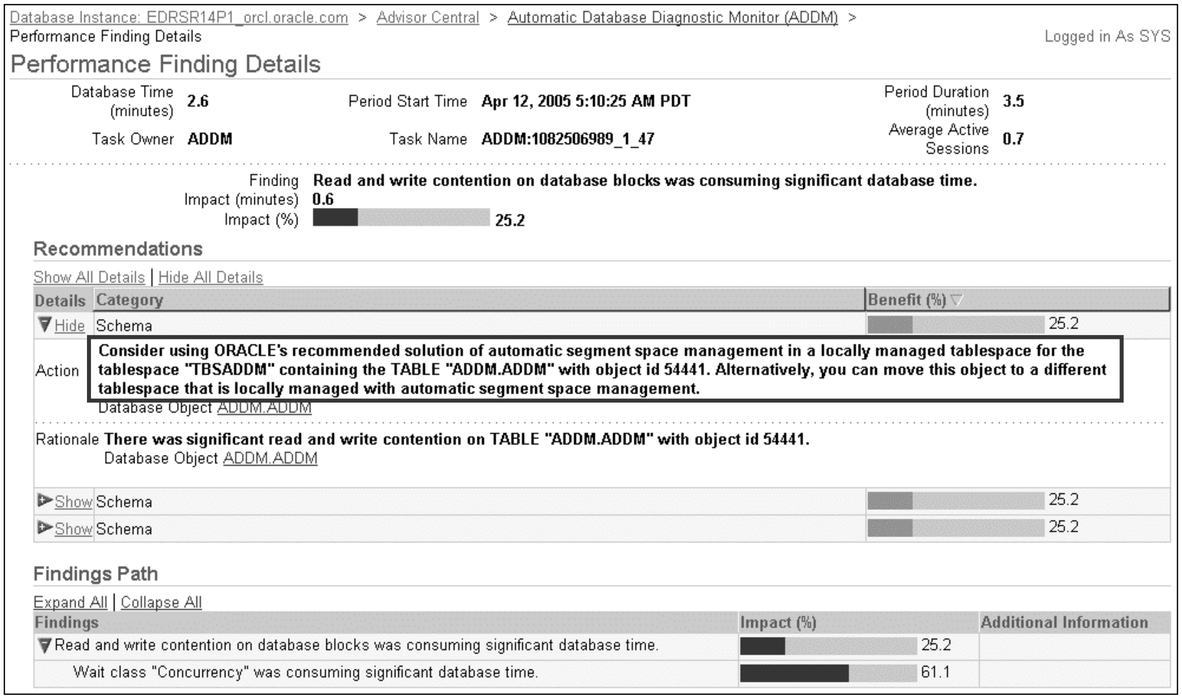
On the Automatic Database Diagnostic Monitor (ADDM) page, you can see the detailed findings for the latest ADDM run. Database Time represents the sum of the nonidle time spent by sessions in the database for the analysis period. A specific Impact percentage is given for each finding. The impact represents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

The numbers in the slide refer to the following:

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

You can click the View Report button to get details of the performance analysis in the form of a text report.

ADDM Recommendations



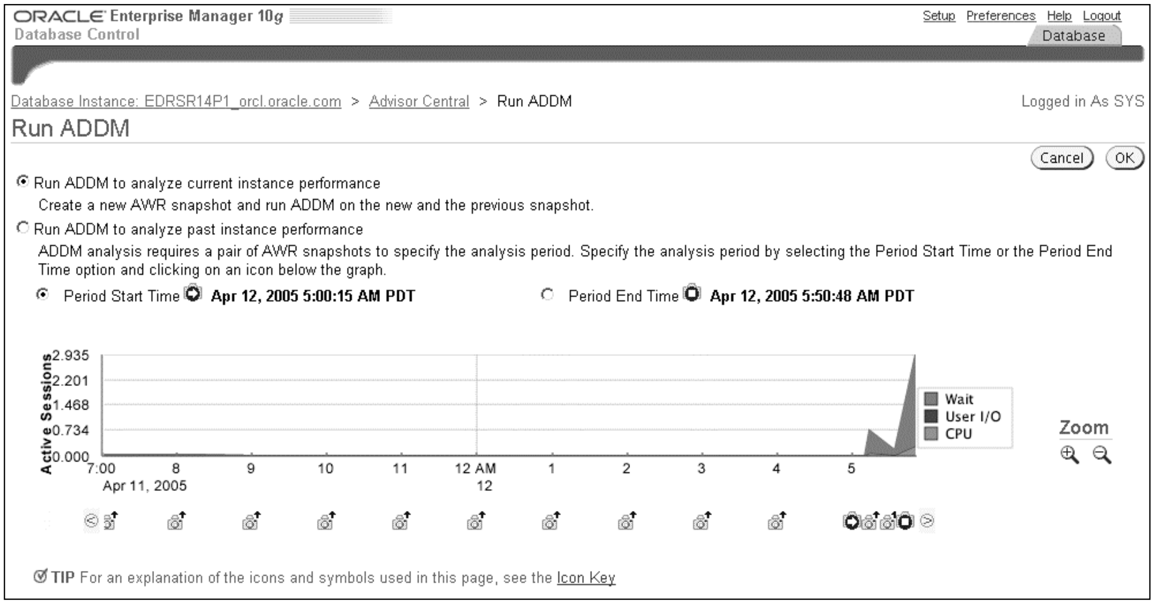
ADDM Recommendations

On the Performance Finding Details page, you are given some recommendations to solve the corresponding issue. Recommendations are grouped into Schema, SQL Tuning, DB Configuration, and other categories. The Benefit (%) column gives you the maximum reduction in database elapsed time if the recommendation is implemented.

ADDM considers a variety of changes to a system. Its recommendations can include:

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

Database Control and ADDM Task



Database Control and ADDM Task

By default, ADDM tasks are run for every Oracle database snapshot that is stored in the workload repository. However, you can create a custom ADDM task to analyze a period of time that you identify with a starting snapshot and an ending snapshot.

To create an ADDM task, navigate to the Database Home page. In the Related Links section, click the Advisor Central link. On the Advisor Central page, click the ADDM link.

Choose the Period Start Time option, and then click the snapshot you want to use as the start point of the period of time. Then choose the End Time option and click the snapshot to use as the terminating point of the time period. Click the OK button to view the Automatic Database Diagnostic Monitor (ADDM) page, on which you get the confirmation that a new task has been created.

In the Performance Analysis section of this page, you get the result of your manually created task.

Note: The results of a manually created ADDM task are also accessible on the Advisor Central page. You can search for your task by using the Search section of this page.

Changing ADDM Attributes

1. **Ensure that `STATISTICS_LEVEL` is set to `TYPICAL` or `ALL`.**
2. **ADDM analysis of I/O performance depends on the expected speed of the I/O subsystem:**
   a. **Measure your I/O subsystem speed.**
   b. **Set the expected speed.**

```
SQL> exec DBMS_ADVISOR.SET_DEFAULT_TASK_PARAMETER(-
          'ADDM', 'DBIO_EXPECTED', 8000);
```

```
SELECT parameter_value, is_default
FROM   dba_advisor_def_parameters
WHERE  advisor_name = 'ADDM' AND
       parameter_name = 'DBIO_EXPECTED';
```

ORACLE

Changing ADDM Attributes

ADDM is enabled by default and is controlled by the STATISTICS_LEVEL initialization parameter. ADDM does not run automatically if STATISTICS_LEVEL is set to BASIC. The default setting for STATISTICS_LEVEL is TYPICAL.

ADDM analysis of I/O performance partially depends on the ADDM parameter DBIO_EXPECTED. This parameter describes the expected performance of the I/O subsystem. The value of DBIO_EXPECTED represents the average time to read a single database block in microseconds. ADDM uses the default value of 10,000 microseconds (10 milliseconds), which is an appropriate value for most modern hard drives. If your hardware is significantly different, consider using a different value.

To determine the correct setting for DBIO_EXPECTED, perform the following steps:

a. Measure the average read time of a single database block read for your hardware. Note that this measurement is for random I/O, which includes seek time if you use standard hard drives. Typical values for hard drives are between 5,000 and 20,000 microseconds.
b. Set the DBIO_EXPECTED value. For example, if the measured value is 8,000 microseconds, you should execute the first command shown in the slide connected as SYS user. The query shown in the slide gives you the current value of this parameter.

Retrieving ADDM Reports by Using SQL

```
SELECT dbms_advisor.GET_TASK_REPORT(task_name)
FROM   dba_advisor_tasks
WHERE  task_id = (
       SELECT max(t.task_id)
       FROM   dba_advisor_tasks t,
              dba_advisor_log l
       WHERE  t.task_id = l.task_id   AND
              t.advisor_name = 'ADDM' AND
              l.status = 'COMPLETED');
```

```
SQL> @?/rdbms/admin/addmrpt
…
Enter value for begin_snap: 8
Enter value for end_snap: 10
…
Enter value for report_name:
Generating the ADDM report for this analysis ...
```


Retrieving ADDM Reports by Using SQL

The first example in the slide shows you how to display the most recent ADDM report by using a SQL command.

To diagnose database performance issues, ADDM analysis can be performed across any two AWR snapshots if the following requirements are met:

- Both snapshots did not encounter any errors during creation, and both have not yet been purged.
- There were no shutdown or startup actions between the two snapshots.

The second example uses the `addmrpt.sql` script. This SQL*Plus script can be used to run ADDM on any two AWR snapshots that are provided. The two snapshots must have been taken by the same instance.

The script identifies your `DBID` and lists the snapshot identifiers for the last three days. This can help you determine the pair of snapshots on which you want to perform the analysis.

Active Session History: Overview

- Stores the history of database time
- Samples session activity in the system including:
  - SQL identifier of a SQL statement
  - Object number, file number, and block number
  - Wait event identifier and parameters
  - Session identifier and session serial number
  - Module and action name
  - Client identifier of the session
  - Service hash identifier
  - Blocking session
- Is always on for first fault analysis
- No need to replay the workload

ORACLE

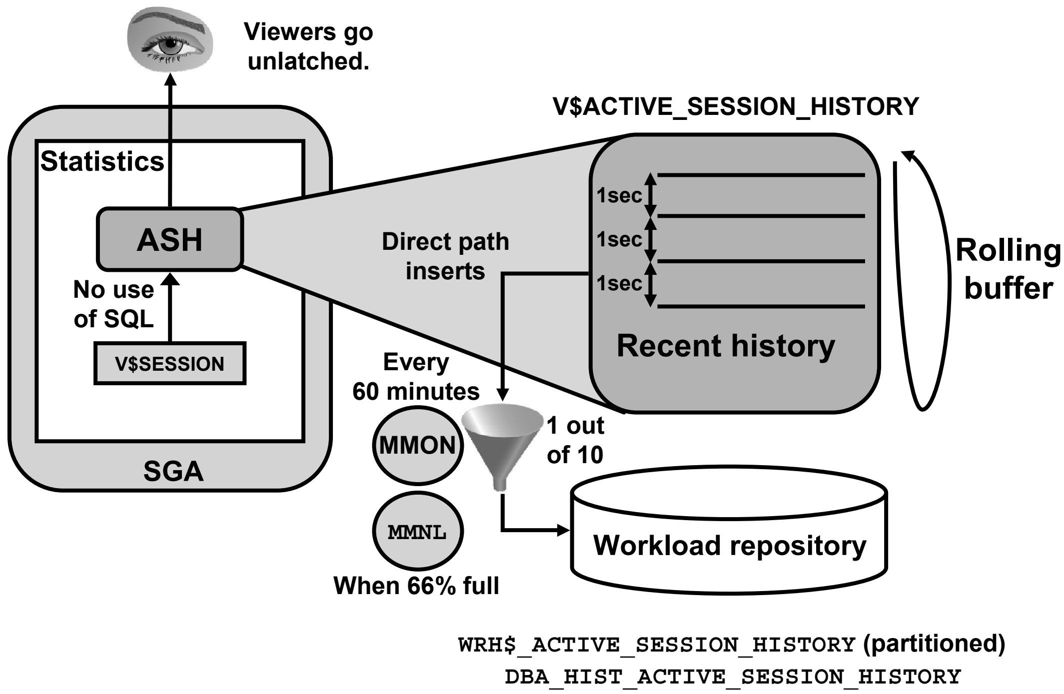
Active Session History: Overview

The V$ACTIVE_SESSION_HISTORY view provides sampled session activity in the instance. This can be used as a first fault system analysis. Any session that is connected to the database and is waiting for an event that does not belong to the Idle wait class is considered as an active session. This includes any session that was on the CPU at the time of sampling.

Active session samples are stored in a circular buffer in SGA. As the system activity increases, the number of seconds of session activity that can be stored in the circular buffer decreases. The time of a session sample is retained in the V$ view. The number of seconds of session activity displayed in the V$ view is completely dependent on database activity.

As part of the Automatic Workload Repository (AWR) snapshots, the content of V$ACTIVE_SESSION_HISTORY is flushed to disk. By capturing only active sessions, a manageable set of data is represented with the size being directly related to the work being performed rather than the number of sessions allowed on the system. You can examine the current Active Session History (ASH) data in the V$ACTIVE_SESSION_HISTORY view and historical data in the DBA_HIST_ACTIVE_SESS_HISTORY view. You can perform detailed analysis over this data repeatedly, avoiding the need to replay the workload to gather additional performance-tracing information. The data in the ASH can be rolled up on various dimensions that it captures (as shown in the slide).

Active Session History: Mechanics



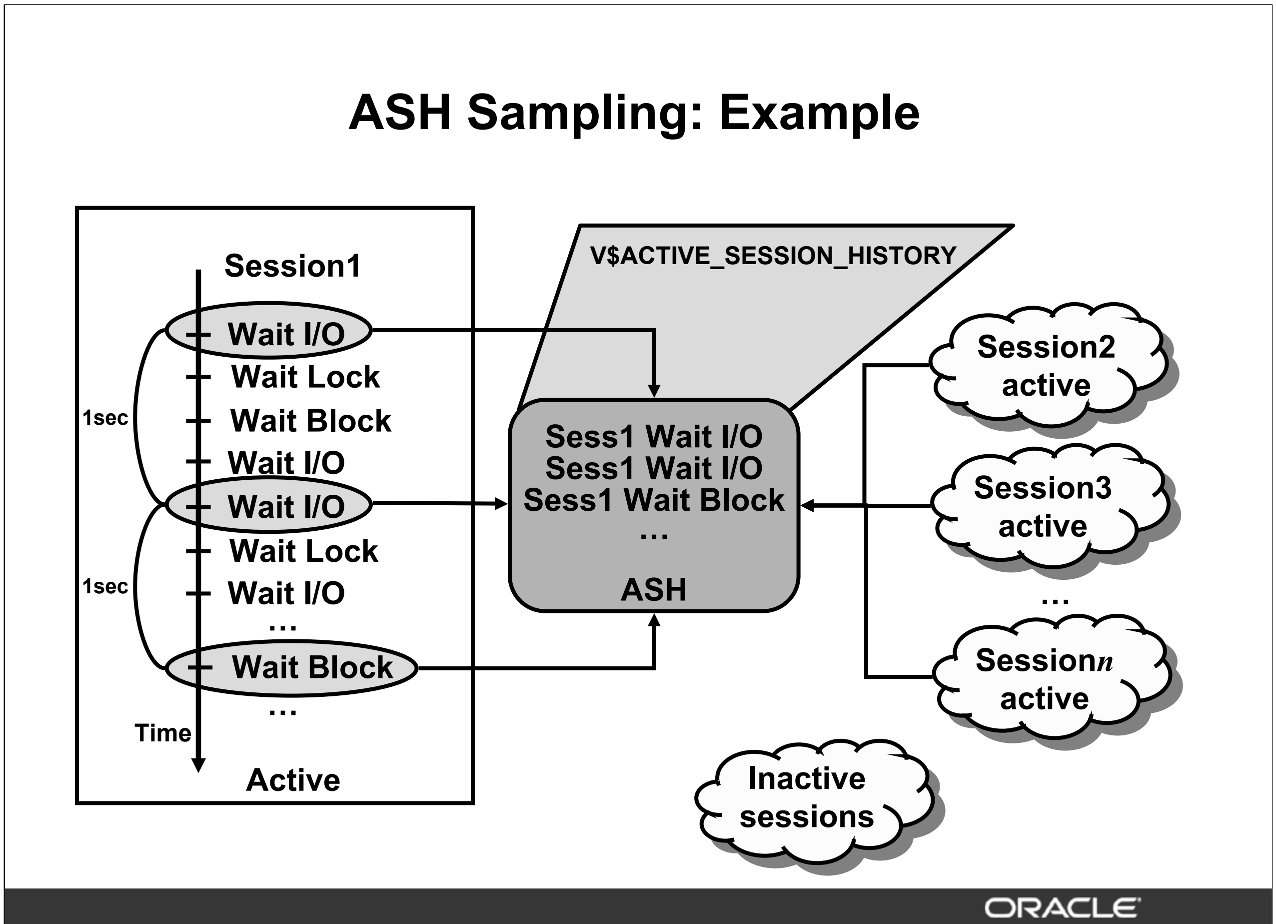
Active Session History: Mechanics

It would be very expensive to record all activities of all sessions. The ASH extracts only samples of information from `V$SESSION`. One sample is extracted every second. This is efficiently done without using SQL.

ASH is designed as a rolling buffer in memory, and previous information is overwritten when needed. The volume of the data in the ASH buffer can be very large, and flushing it all to disk is unacceptable. A more efficient approach is to filter the history data while flushing it to the workload repository. This is done automatically by the manageability monitor (MMON) process every 60 minutes and by the manageability monitor light (MMNL) process whenever the buffer is full.

The slide shows you all the nonintrusive implementation features used to capture ASH data.

Note: The memory for the ASH comes from the System Global Area (SGA), and it is fixed for the lifetime of the instance. It represents 2 MB of memory per CPU. The ASH cannot exceed a maximum bound of 5% of the shared pool size, or 5% of the `SGA_TARGET`.



ASH Sampling: Example

As explained earlier, ASH represents the history of recent sessions activity. The diagram shows you how sessions are sampled when active. Each second, the Oracle database server looks at active sessions and records the events these sessions are waiting for. Inactive sessions are not sampled. The sampling facility is very efficient because it directly accesses the Oracle database internal structures. The following is the sampled information:

- SQL identifier of SQL statement
- Object number, file number, and block number
- Wait event identifier and parameters
- Session identifier and session serial number
- Module and action name
- Client identifier of the session
- Service hash identifier

ASH statistics are available through the `V$ACTIVE_SESSION_HISTORY` fixed view. This view contains one row for each active session per sample. All the columns that describe the session in the ASH are present in the `V$SESSION` view.

Accessing ASH Data

- Dump to trace file
- `V$ACTIVE_SESSION_HISTORY`
- `DBA_HIST_ACTIVE_SESS_HISTORY`
- ASH report
- EM Diagnostic Pack performance pages
- ADDM

Accessing ASH Data

As shown in the slide, there are many ways in which you can access ASH-collected data. The following pages provide more details for each possibility.

Dump ASH to File

The generated file contains ASH samples for the last number of minutes specified:

```
SQL> oradebug setmypid
SQL> oradebug dump ashdump 10
```

```
<<<ACTIVE SESSION HISTORY - PROCESS TRACE DUMP HEADER BEGIN>>>
DBID, INSTANCE_NUMBER, SAMPLE_ID, SAMPLE_TIME, SESSION_ID, SESSION_SERIAL#, USER_ID,
SQL_ID, SQL_CHILD_NUMBER, SQL_PLAN_HASH_VALUE, SERVICE_HASH, SESSION_TYPE, SQL_OPCODE,
BLOCKING_SESSION, BLOCKING_SESSION_SERIAL# QC_SESSION_ID, QC_INSTANCE_ID, XID,
CURRENT_OBJ#, CURRENT_FILE#, CURRENT_BLOCK#, EVENT_ID, SEQ#, P1, P2, P3, WAIT_TIME,
TIME_WAITED, FORCE_MATCHING_SIGNATURE, PROGRAM, MODULE, ACTION, CLIENT_ID
<<<ACTIVE SESSION HISTORY - PROCESS TRACE DUMP HEADER END>>>
<<<ACTIVE SESSION HISTORY - PROCESS TRACE DUMP BEGIN>>>
1090770270,1,1317127,"08-26-2005 01:11:40.505471000", 162, 1, 0, "", 0, 0, 165959219, 2,
0, 4294967295, 0,0,0,,5129,1,11242,86156091,1664,0,0,0,0,64620,0,"oracle@edrsr14p1
(CJQ0)","","","",1090770270,1,1317116,"08-26-2005 01:11:29.505471000", 142,20,24,
"0hbv80w9ypy0n",0,3304045827, 3427055676, 1,3,4294967291, 0,0,0,,8751, 3,2486,
1421975091,24975,1413697536,1,0,620,0,17258348159868772889,"emagent@edrsr14p1 (TNS V1-
V3)","emagent@edrsr14p1 (TNS V1-V3)","",""
…
<<<ACTIVE SESSION HISTORY - PROCESS TRACE DUMP END>>>
```


Dump ASH to File

You can dump the ASH content to a trace file by using the `oradebug` command as shown in the slide. You can control the amount of information dumped in the trace file by specifying the number of minutes of history you are interested in.

The structure of the trace file is shown in the slide. You basically get one row in the trace file for each row that exists in the ASH for the corresponding period of time. You can use SQL*Loader to load its content to a database table. However, you need to create the corresponding SQL*Loader control file.

Extracting Data from the ASH

- `GROUP BY` and `COUNT`
  - Proxy for nonidle elapsed time
  - Proportions of actual time spent
- Can analyze any time slice
- Example: Returns most active SQL in the past minute

```
SELECT   sql_id, count(*),
         round(count(*)/sum(count(*)) over (), 2) pctload
FROM     v$active_session_history
WHERE    sample_time > sysdate -1/24/60 and
         session_type <> 'BACKGROUND'
GROUP BY sql_id
ORDER BY count(*) desc;
```

ORACLE

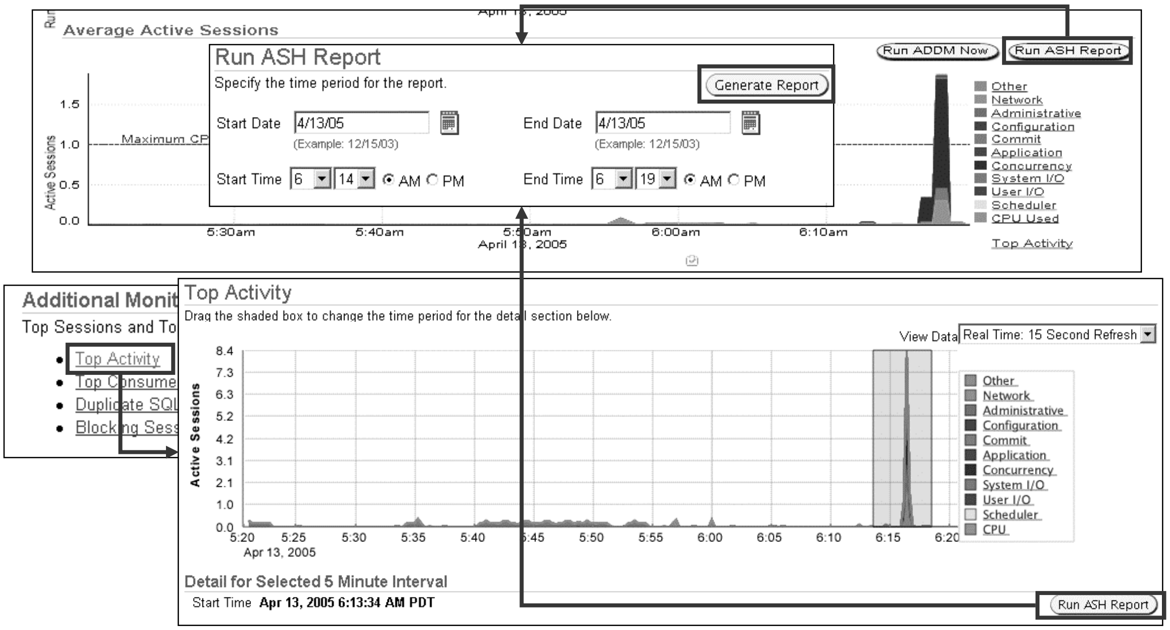
Extracting Data from the ASH

You can use regular SQL statements to analyze any time slice of your choice using `V$ACTIVE_SESSION_HISTORY`. However, you can obtain more accurate results by including more samples in your analysis. As you have already seen, you can extract your data from the ASH by using different dimensions such as a `SQL_ID`, an `ACTION`, or an object number.

The SQL statement shown in the slide returns the list of the most active SQL statements executed on your instance for the last minute.

Note: You can also use the `DBA_HIST_ACTIVE_SESSION_HISTORY` view from the workload repository whenever you want to access data no longer in memory. However, the number of samples stored in the ASH on-disk version is less than its corresponding in-memory version: one sample every 10 seconds.

Generating ASH Reports



Generating ASH Reports

The ASH Report is a digest of the ASH samples that were taken during a time period. Some of the information it shows are top wait events, top SQL, top SQL command types, and top sessions, among others. The ASH Report, which is run against collected ASH data, can be focused on a time period of days, hours, or minutes. The start and end times are not restricted to when the AWR snapshots were taken; the period can span snapshot boundaries. This makes it possible for you to focus your analysis on a very small time period—even as small as just a few minutes.

Using Enterprise Manager, you can generate the ASH Report by navigating to the Performance page and then clicking Run ASH Report. On the Performance page, you can also click the Top Activity link. On the Top Activity page, you can then choose a five-minute interval by dragging the shaded area and then clicking Run ASH Report.

ASH Report Script

```
SQL> define dbid        = '';
SQL> define inst_num    = '';
SQL> define report_type = 'html';
SQL> define begin_time  = '09:00';
SQL> define duration    = 480;
SQL> define report_name = '/tmp/sql_ashrpt.txt';
SQL> define slot_width  = '';
SQL> define target_session_id   = '';
SQL> define target_sql_id       = 'abcdefghij123';
SQL> define target_wait_class   = '';
SQL> define target_service_hash = '';
SQL> define target_module_name  = '';
SQL> define target_action_name  = '';
SQL> define target_client_id    = '';
SQL> @?/rdbms/admin/ashrpti
```

ORACLE

ASH Report Script

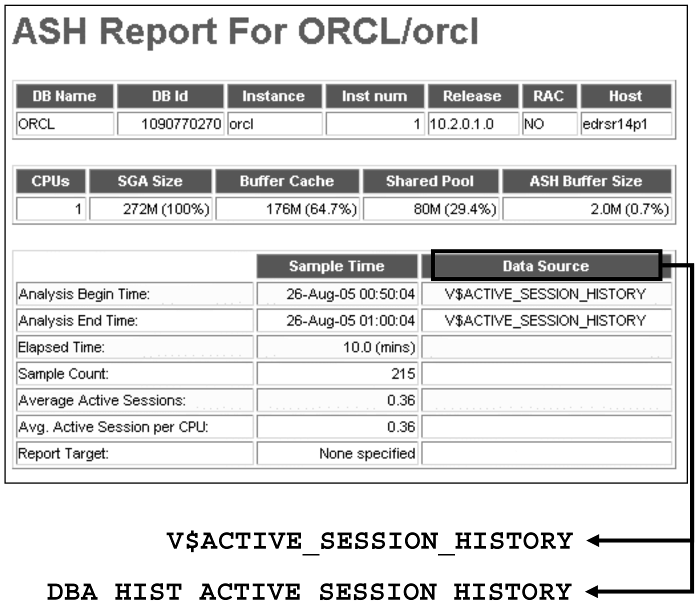
Another way to generate the ASH Report is to run the `$ORACLE_HOME/rdbms/admin/ashrpt.sql` SQL*Plus script.

By simply pressing [Enter] when prompted for each variable value, you can quickly generate an ASH Report that covers the last 15 minutes of active history. Alternatively, you can define SQL*Plus variables before invoking the script.

As shown in the slide, you can target the report for a particular ASH dimension with the `ashrpti.sql` script. You can define variables or enter them when prompted.

To target a specific ASH dimension, you define any number of the variables shown in the slide to the value you want reported on. This enables you to focus the analysis on a single item or combination of items during a particular time span.

ASH Report: General Section



ASH Report For ORCL/orcl

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
ORCL	1090770270	orcl	1	10.2.0.1.0	NO	edrsr14p1

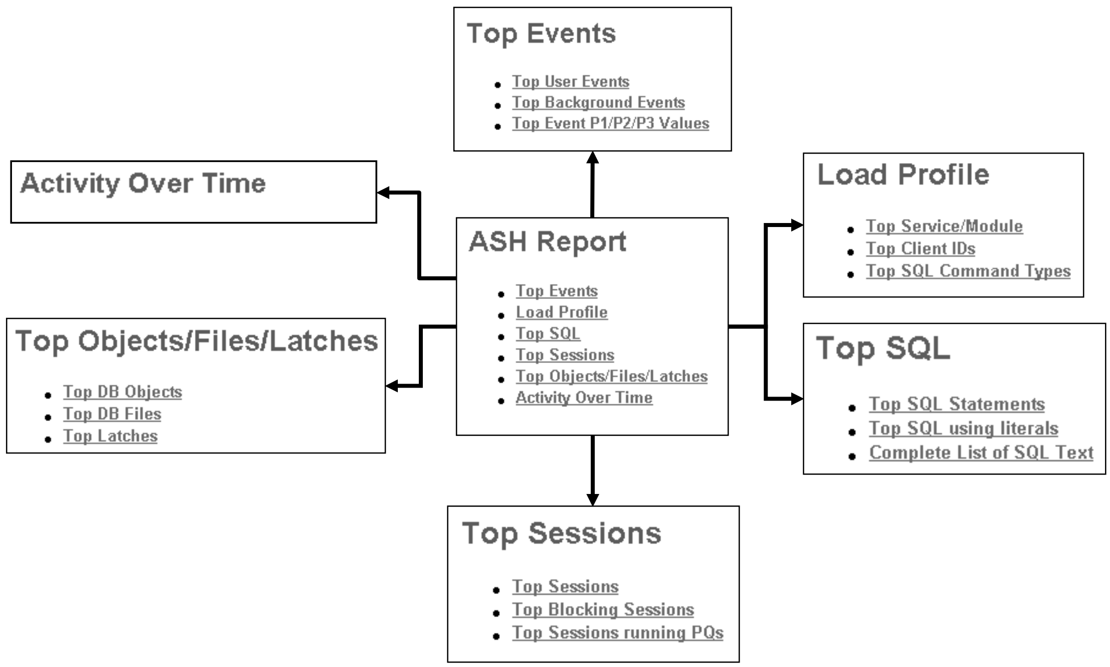
CPUs	SGA Size	Buffer Cache	Shared Pool	ASH Buffer Size
1	272M (100%)	176M (64.7%)	80M (29.4%)	2.0M (0.7%)

	Sample Time	Data Source
Analysis Begin Time:	26-Aug-05 00:50:04	V$ACTIVE_SESSION_HISTORY
Analysis End Time:	26-Aug-05 01:00:04	V$ACTIVE_SESSION_HISTORY
Elapsed Time:	10.0 (mins)	
Sample Count:	215	
Average Active Sessions:	0.36	
Avg. Active Session per CPU:	0.36	
Report Target:	None specified	

ASH Report: General Section

The Data Source column of the report summary shows where the data came from. This can have two different values because the data from V$ACTIVE_SESSION_HISTORY, which is in-memory data, regularly gets migrated to DBA_HIST_ACTIVE_SESS_HISTORY, which is on-disk data. This migration helps to relieve the memory requirements. During migration, the data is reduced from 1-second samples to 10-second samples, thus taking up 1/10 the space on disk. Because of that difference in granularity, data that comes from memory can possibly be more accurate than data from the history view.

ASH Report Structure



ASH Report Structure

The slide shows the various sections of the ASH Report (which follows the pattern of the AWR report). Starting from the top of the slide, the sections are:

- **Top Events**
  Reports the user events, background events, and event parameter values
- **Load Profile**
  Reports the top service and top clients and identifies the type of SQL commands
- **Top SQL**
  Reports top SQL statements, top SQL using literals, and the SQL text for these SQL statements
- **Top Sessions**
  Reports the top sessions found waiting, top blocking sessions, and aggregates for PQ sessions
- **Top Objects/Files/Latches**
  Reports the top objects, files, and latches that were involved in a wait
- **Activity Over Time**
  Reports the top three wait events for 10 equally sized time periods during the report period

This report enables you to see very detailed activity over the past hour.

ASH Report: Activity Over Time

Activity Over Time

- Analysis period is divided into smaller time slots
- Top 3 events are reported in each of those slots
- 'Slot Count' shows the number of ASH samples in that slot
- 'Event Count' shows the number of ASH samples waiting for that event in that slot
- '% Event' is 'Event Count' over all ASH samples in the analysis period

Slot Time (Duration)	Slot Count	Event	Event Count	% Event
00:50:04 (56 secs)	3	CPU + Wait for CPU	2	0.93
		db file sequential read	1	0.47
00:51:00 (1.0 min)	6	CPU + Wait for CPU	5	2.33
		db file sequential read	1	0.47
00:52:00 (1.0 min)	1	CPU + Wait for CPU	1	0.47
00:53:00 (1.0 min)	17	CPU + Wait for CPU	13	6.05
		log file parallel write	2	0.93
		log file sync	1	0.47
00:54:00 (1.0 min)	171	buffer busy waits	64	29.77
		CPU + Wait for CPU	57	26.51
		latch: In memory undo latch	23	10.70
00:55:00 (1.0 min)	5	CPU + Wait for CPU	3	1.40
		log file parallel write	1	0.47
		os thread startup	1	0.47
00:56:00 (1.0 min)	11	CPU + Wait for CPU	8	3.72
		log file parallel write	3	1.40
00:57:00 (1.0 min)	3	CPU + Wait for CPU	3	1.40

ORACLE

ASH Report: Activity Over Time

One of the most informative sections of the ASH Report is the Activity Over Time section. In this section, the ASH report time span is divided into 10 time slots. If the time period is too short or if the data is too sparse, the resulting report has fewer slots. Slots 2 through 9 are defined as an integer number of minutes that are all the same size for easy comparison. As a result, it is best to compare the inner slots to one another.

Using the Activity Over Time section, you can perform skew analysis by looking for spikes in the Event Count column. This indicates an increase in the number of waiters for a particular event. A spike in the Slot Count indicates an increase in active sessions because ASH data is sampled from active sessions only.

In the pictured example, note the circled event. The number of active session samples has increased, and the number of sessions associated with the `buffer busy waits` event has also spiked. This kind of skew in a slot possibly represents the root cause of the problem that is being investigated.

Summary

In this lesson, you should have learned how to:

- Create and manage AWR snapshots
- Generate AWR reports
- Create snapshot sets and compare periods
- Generate ADDM reports
- Generate ASH reports

ORACLE

Practice 6 Overview: Using AWR-Based Tools

This practice covers the following topics:

- Generating an AWR report
- Creating a preserved set on snapshots
- Generating an ADDM report
- Generating an ASH Report

7

Reactive Tuning

Objectives

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

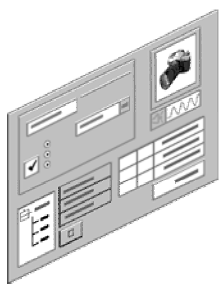
- Use Enterprise Manager pages to identify performance issues
- Eliminate operating system issues

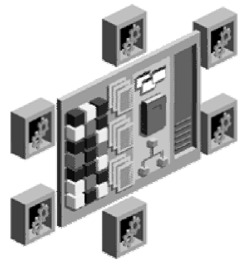
ORACLE

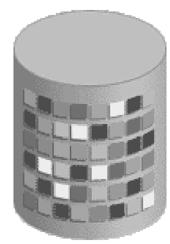
Where Is the Problem?

Performance issues can be located in the following areas:

- **Operating system (OS)**
- **Application**
- **Instance**







Where Is the Problem?

Performance is affected when you have contention for one or more resources. There are hardware resources (for example, CPU, memory, network, and disk) and software or application resources (row lock, latches, and so on). OS utilities allow you to monitor hardware usage whereas Oracle statistics can tell you about internal usage. After a critical resource is identified, finding the remedy becomes the next challenge. Tuning the SQL generally yields the greatest benefits, but you may not be allowed to change the application code. The choice then becomes whether to tune the instance, or to add more hardware.

The first step in tuning is to identify the problem. Eliminating the areas that are not problems is as important as identifying the areas that are problems. The operating system (OS) should be the first area checked. Make sure that there is no problem with the OS. The application is the next level to check. Then tune the database instance.

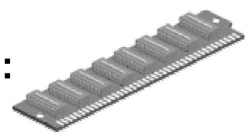
When you are using Enterprise Manager and the AWR-based tools, the most significant performance issues are categorized automatically and presented in order of the highest impact. The Statspack and AWR reports present the summary pages, and from those you can identify the highest-impact issues.

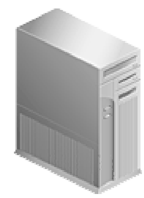
Operating System Issues

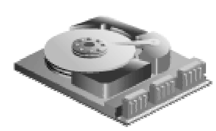
Operating System
Application
Instance

Categories of operating system issues:

- **Memory**
- **CPU**
- **I/O (disk and network)**







ORACLE

Operating System Issues

Operating system (OS) issues fall into three categories: memory, CPU, and I/O. The I/O category includes disk and network accesses.

You can check the memory by using OS tools such as `vmstat`. The key measurement is the paging rate. When there is more demand for memory than there is memory available, the OS moves pages of memory into a swap area or page file on disk. The remedy is to reduce the demand for memory, or increase the available memory. Reducing demand can include rescheduling processes, moving processes to other machines, or limiting the memory that the process is allowed to use. When database processes use large amounts of memory, the memory demands may be reduced by making the SQL more efficient. Another way is to reduce memory demands when there are thousands of processes is to use connection pooling.

The demand for CPU resources is measured by the run queue length (load) and `%busy`. Every OS has a scheduler that determines when a process runs and how long it runs before another process gets a turn on the CPU. The scheduler places processes that are ready to run on the run queue. Usually the average length of the queue is the load; on some systems the load includes the running processes as well. When the run queue is longer than two times the number of CPUs, CPU may be your limiting resource. The cause may be poorly

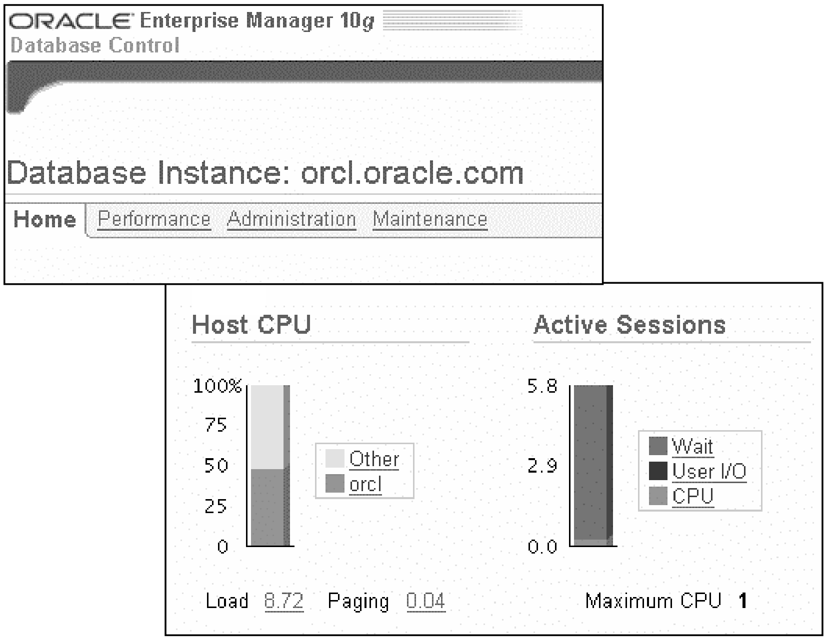
Operating System Issues (continued)

written applications, which use a large amount of CPU or too many services on the machine. The cure is to increase the CPU available or reduce the demand. Tune the application or move services such as file servers or DNS to other machines.

The symptoms of an I/O problem are long device queues, longer than normal wait times, or high request rates. Again, the cure is to reduce demand or increase the capacity. Increasing the capacity could mean spreading the requests across all the available disks by striping, or adding more disks and controllers. When the database processes incur high I/O rates, or high I/O wait times, tuning the SQL statements to make fewer I/O requests is usually the most cost-effective method to reduce the I/O.

All of these measurements are available through Enterprise Manger. These OS measurements are also available through OS-specific tools. On UNIX-like systems, `vmstat`, `sar`, and `iostat` are the main tools. On Windows-based systems, the task manager and optional OS monitoring tools are available.

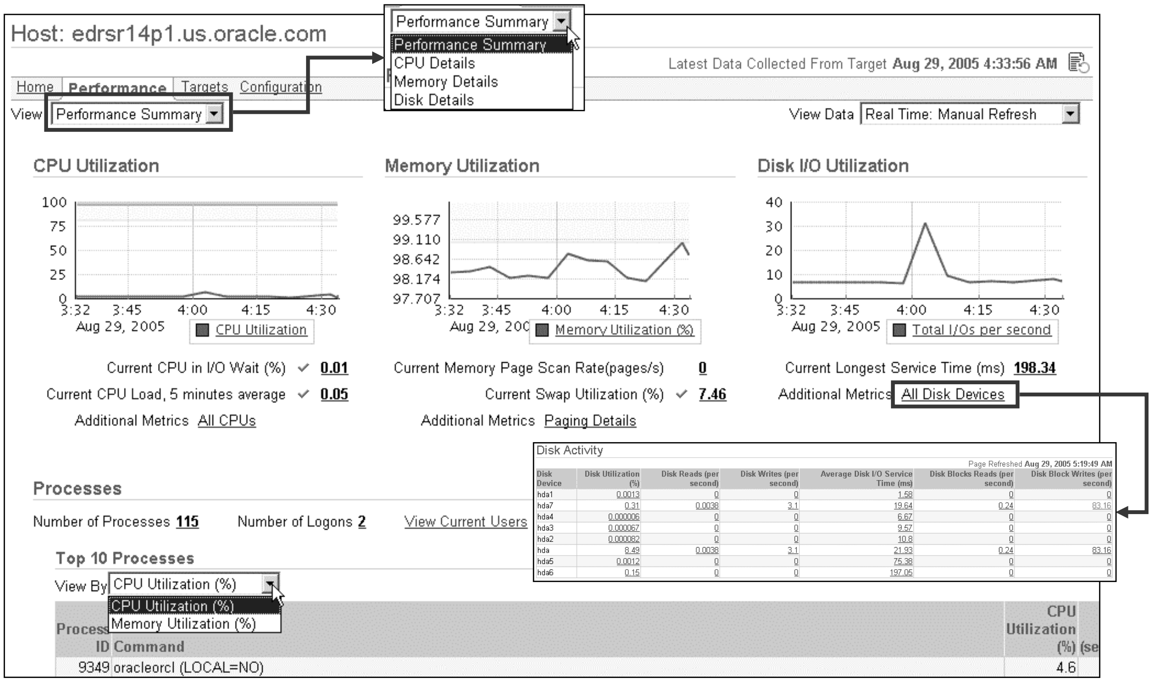
Database Home Page



Database Home Page

The Database Home page displays a summary of the OS statistics. From the example in the slide, you can see that the instance is using approximately 50% of the CPU and some other processes are using the remainder. CPU %busy is 100% and the load is 8.72. The CPU is very busy and there are processes waiting for the CPU to be available. The run queue has an average length of 8.72. The memory demand is low, and the paging is only 0.04 pages per second. In this example, you can see that there is an OS problem outside the database. Click the Load value to drill down to details about the Host processes.

OS Statistics and Enterprise Manager



OS Statistics and Enterprise Manager

You can view OS statistics captured by Enterprise Manager from the Host Performance page. You can access this page from the Database Control Home page by clicking the Host link in the General section. On the Host Home page, click the Performance tab. Clicking the CPU Load value drilldown link brings you directly to the Performance page.

By default, the Performance page displays the Performance Summary report, where you can see the metric values for CPU Utilization, Memory Utilization, and Disk I/O Utilization. These metrics correspond to the key measurements that are gathered by OS tools. In each of the utilization sections, you can drill down to see the utilization statistics gathered over the last 24 hours.

The Processes section of the Performance page lists the Top 10 Processes, and you can view the list ordered by CPU Utilization or by Memory Utilization. Under each chart on the Performance page, you can drill down to more detail about the corresponding metrics, as shown by the Disk Activity report in the slide. This report is similar in content to the UNIX `top` command or Windows Task Manager.

Note: You can set thresholds for the host metrics (similar to the database metrics).

OS Statistics

Statistic Name	Description
NUM_CPUS	Number of CPUs or processors available
IDLE/BUSY/USER/NICE/SYS_TIME	# hundredths of a second that a processor has been idle/busy…
IOWAIT_TIME	The average # hundredths of a second that a processor has been waiting for I/O to complete
OS_CPU_WAIT_TIME	# hundredths of a second that processes have been in a ready state, waiting to be scheduled
RSRC_MGR_CPU_WAIT_TIME	# hundredths of a second Oracle processes have been in a ready state, waiting for CPU to be available for their consumer group in the currently active resource plan
VM_PAGE_IN/OUT_BYTES	# bytes of data paged in/out due to paging
PHYSICAL_MEMORY_BYTES	Total number of bytes of physical memory
LOAD	# processes either running or in the ready state over the the last 60 seconds
NUM_CPU_SOCKETS	Number of CPU sockets available
NUM_CPU_CORES	Number of CPU cores available

OS Statistics

The instance captures the statistics listed in the slide in the `V$OSSTAT` view. Many CPU timers have several orders of magnitude finer granularity than instance-elapsed timers. This can result in underestimating CPU time, because many operations report zero CPU usage. As CPUs get faster, the likelihood of underestimating CPU time increases.

Paging and swapping activity as a result of pressure on memory can affect the performance of the database. Before Oracle Database 10*g* Release 2, you were unable to diagnose it by looking at any of the available database statistics. The `V$OSSTAT` view captures machine-level information in the database so that the system can determine whether there are hardware-level resource issues. The slide briefly describes most of the statistics that can be captured.

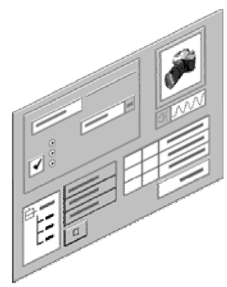
Note: The availability of all statistics (except for `NUM_CPUS` and `RSRC_MGR_CPU_WAIT_TIME`) is subject to the operating system platform on which the Oracle database is running.

Application Issues

Operating System
Application
Instance

Categories of application issues:

- Resource serialization
- Relational design
- Session management
- Cursor management
- Program design



Application Issues

Most application issues can be placed in the categories of resource serialization, relational design, session management, or cursor management.

Resource serialization issues can come from a variety of sources. As an example, there is a business requirement that all printed checks must have consecutive numbers and no numbers can be skipped. This requirement cannot be met without using a single source for check numbers. Multiple processes trying to print checks will have to serialize through this one resource. A large order entry table can enter new rows in a variety of blocks spread across multiple disks, but an `ORDER_ID` generated from a single sequence populating a primary key will cause the last block of the primary key index to be a point of serialization. All the inserting processes will try to update that last block with a new `ORDER_ID` key.

Relational design issues include over-normalization, which causes unnecessary table joins, and under-normalization, which forces many additional constraints or application checks to keep data consistent.

Session management relates to middleware. An example is a Web page that continually logs on and off a database. The logon procedures cost the end user too much time.

Application Issues (continued)

Cursor management issues are often caused by programming errors, and they limit scalability. An example is an application that does not make use of bind variables in the WHERE clause. Consider the following two statements:

```
SELECT * FROM hr.employees WHERE employee_id = 7900;
SELECT * FROM hr.employees WHERE employee_id = 7369;
```

If CURSOR_SHARING is set to SIMILAR or FORCE, both of the preceding statements parse to a single cursor if there is an index on EMPLOYEE_ID, even though the SQL text is different. Even with no index on EMPLOYEE_ID, these two statements may share a single cursor if you assign a value of FORCE to CURSOR_SHARING. Some application issues can be mitigated by changes to the instance parameters such as CURSOR_SHARING.

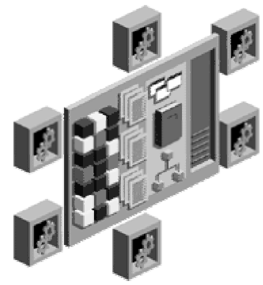
A well-designed application often runs well on an untuned database instance. But a poorly designed application runs poorly even on a well-tuned instance. Inefficient resource utilization is the primary characteristic of a poorly designed program.

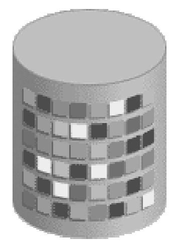
Instance Issues

Operating System
Application
Instance

Categories of instance issues:

- **Memory**
  - SGA
  - PGA
- **I/O**
  - Physical I/O
  - Data placement
- **Contention**
  - Locks
  - Latches
  - Serialized resources



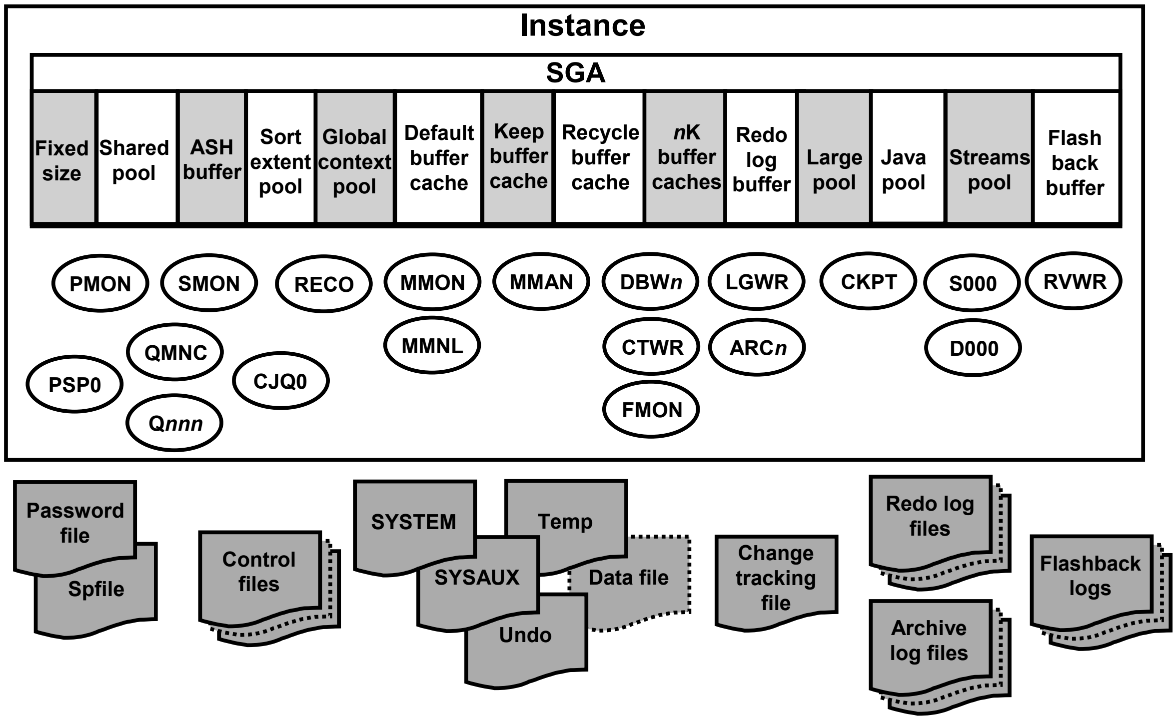


ORACLE

Instance Issues

The top-level categories are listed in the slide. The rest of the course is devoted to identifying and tuning the most common issues in each area.

Architecture

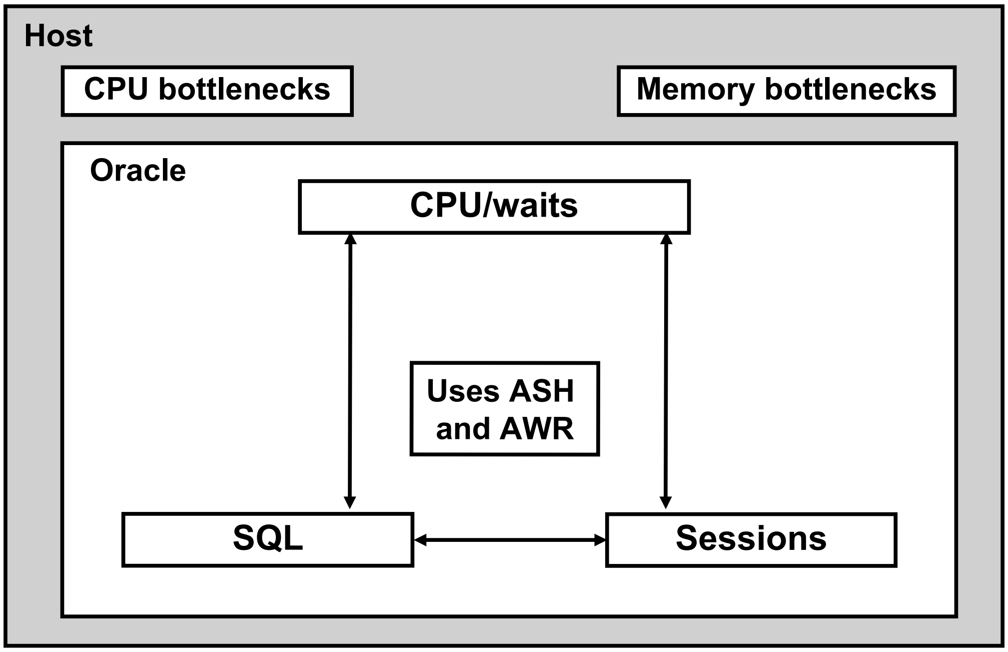


Instance Architecture

The Oracle Database 10*g* architecture is complex. The complexity is balanced by the management tools that are built into the Oracle database server and that automate many of the tedious and repetitive tasks formerly performed by DBAs. Support for these tools is provided by processes such as Manageability Monitor (MMON), Memory Manager (MMAN), and Manageability Monitor Light (MMNL). The overhead for these manageability tools is generally less than 2%. You use several of these manageability tools and advisors in this course.

The diagram in the slide is a road map for the rest of the course. Various areas and processes are highlighted in each lesson of the course.

Performance Management Approach



Performance Management Approach

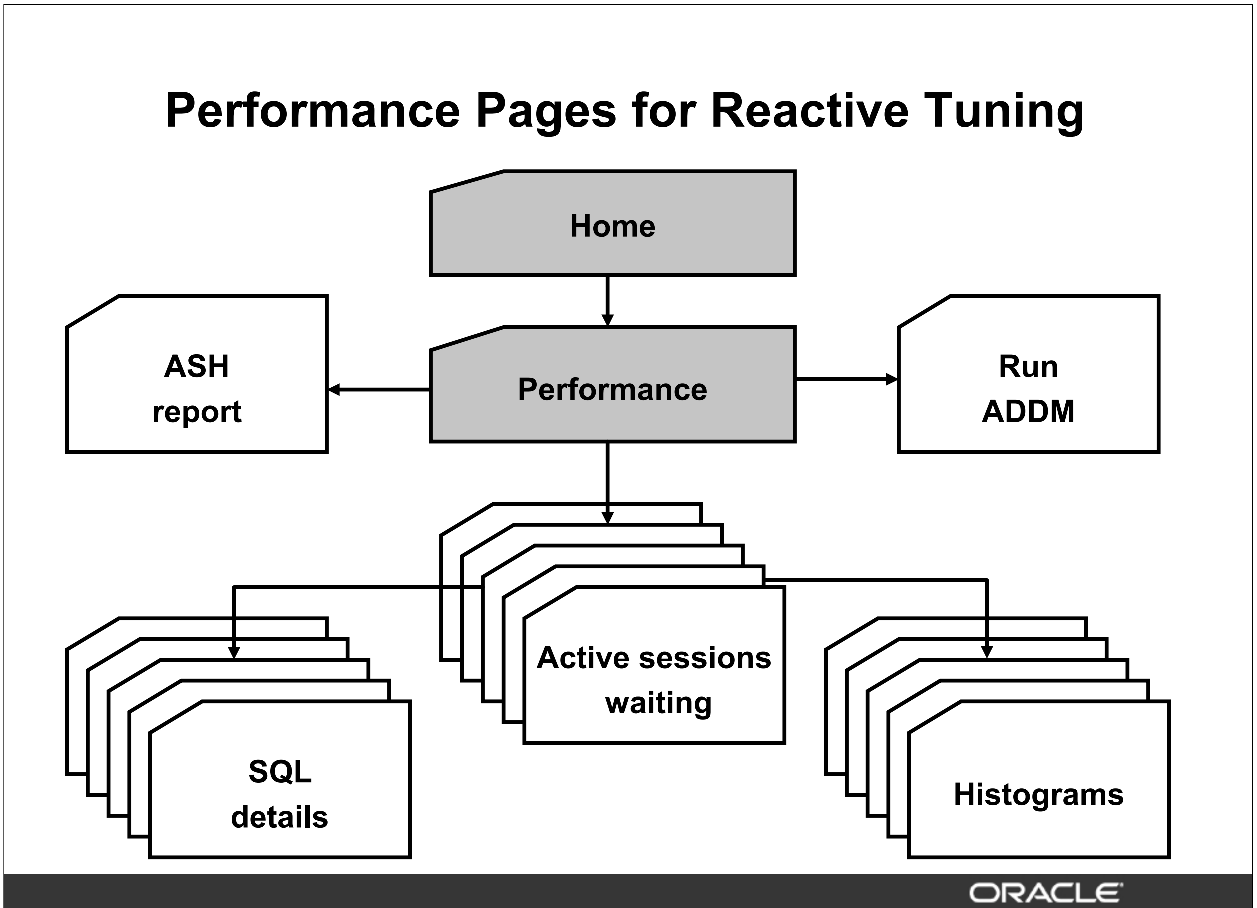
Database Control enables you to correlate both host and instance bottlenecks by giving you detailed statistics from both sides. This enables you to differentiate the work executed by the Oracle database server from the work executed outside the Oracle database server.

The approach used by Enterprise Manager is to provide you with information about three major tuning areas:

- CPU and wait classes
- Top SQL statements affecting the Oracle instance
- Top sessions affecting the Oracle instance

For each of these dimensions, Database Control gives you current data by extracting statistics from the Active Session History (ASH), or historical data by extracting statistics from Automatic Workload Repository (AWR).

On most of the performance-related pages, you can navigate within these three dimensions.

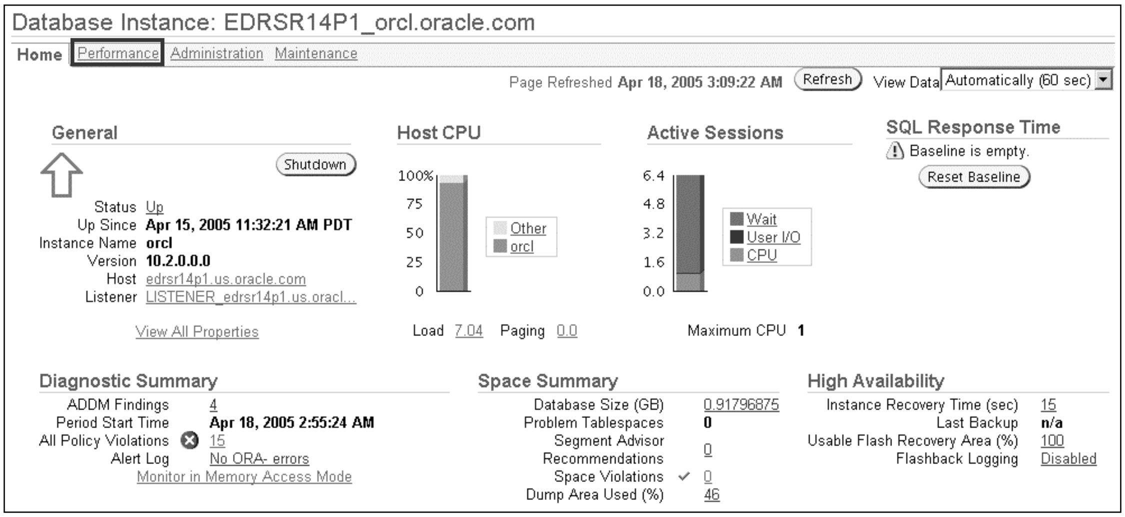


Performance Pages for Reactive Tuning

There are cases in which real-time problem diagnosis needs to be performed. An irate user calls you, or you see a sudden spike in the activity of the system on the monitor. The Enterprise Manager Performance pages use the same data sources as AWR and ADDM to display information about the running of the database and the host system; this information is easily absorbed and allows for rapid manual drilldown to the source of the problem. The slide shows only a subset of the pages that are available.

Other pages that are helpful for manual reactive tuning will be shown in the context of identification and diagnosis of various problems.

Database Home Page



Database Home Page

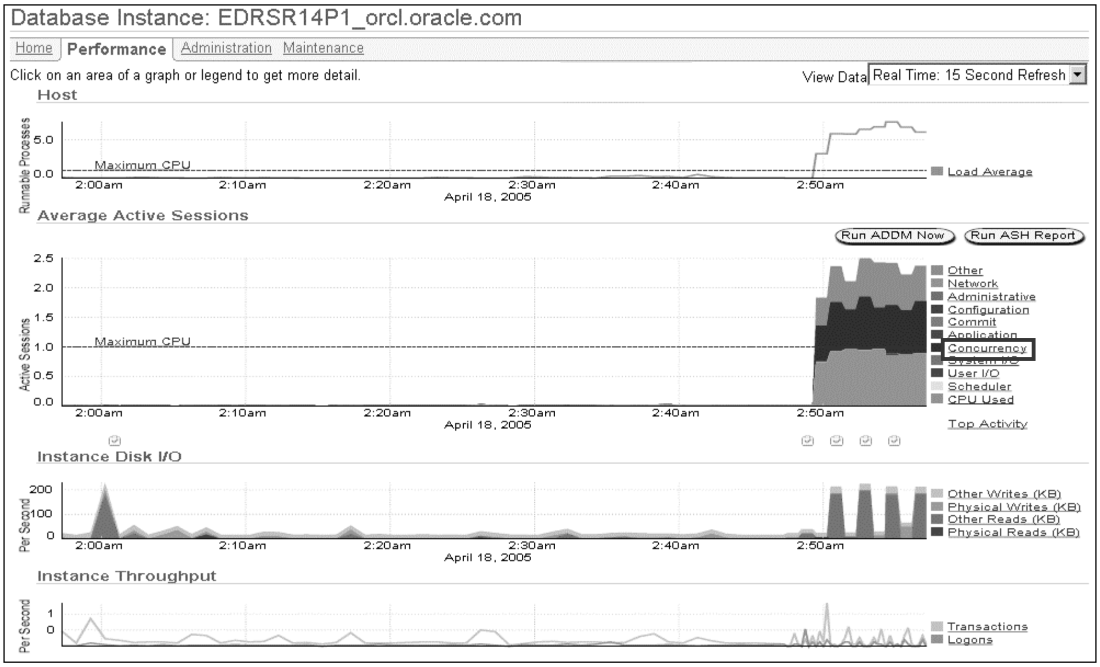
This is the Database Control Home page. This page gives you an overview of the database instance health. The Automatic Database Diagnostic Monitor (ADDM) findings for the latest run are also shown on this page (but do not appear in the slide).

This page gives you statistics about:

- Host CPU: On the graph, host CPU is being used at around 100%, with 90% of it being used by this database instance.
- Active sessions waiting for I/Os, non-I/O events, and using CPU: On the graph, there are 6.5 active sessions over the last sample that (on average) are waiting for resources.
- SQL response time (which can be used to identify key SQL statements): This metric is computed as an increased percentage compared to a baseline of representative SQL statements.
- Important space-related alerts

Click the Performance link to access the Database Performance page.

Database Performance Page



Database Performance Page

The Database Performance page consists of three sections displaying host information, user activity, and throughput information. With this information, you can first verify that the machine has ample CPU and memory resources available before analyzing the database. You can then assess the database health from the Average Active Sessions graph, which shows how much CPU the users are consuming and whether there are users waiting for resources instead of running on the CPU. This graph illustrates the most important classes of wait time spent by the instance.

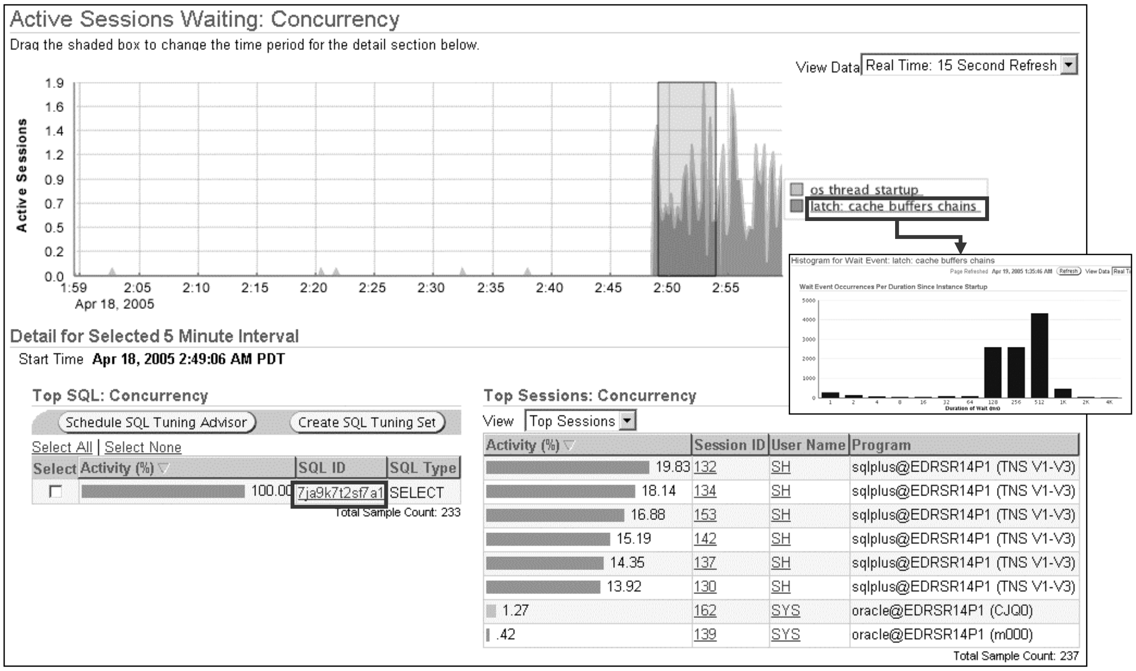
Basically, wait events are categorized in different classes. You can click the most important area corresponding to a particular wait class. In this example, the Concurrency wait class has been predominant on this instance since 2:50 a.m.

The graph also shows you the frequency at which snapshots are taken. The Maximum CPU line is an important reference point. When the green CPU Used value reaches the Maximum CPU line, the database instance is running at 100% CPU of the host machine. All values other than CPU Used represent users waiting and contention for resources.
In this case, one of the biggest contention areas is Concurrency. Click either the colored area of the graph or the legend to drill down for more detailed information.

Database Performance Page (continued)

Click the corresponding snapshot that corresponds best to the time at which the problem was discovered. This directs you to the corresponding ADDM analysis page. Alternatively, you can instantaneously create a snapshot and run ADDM by clicking Run ADDM Now. For transient performance issues that typically last less than 15 minutes, you can also click Run ASH Report to analyze activity by different dimensions.

Active Sessions Waiting Pages



ORACLE

Active Sessions Waiting Pages

These pages give you more detail about the corresponding wait problem. In the slide, the upper part gives you detail about the wait events for the Concurrency class. The “Cache buffer chains” waits in the Concurrency class are the most important ones. If you click the “latch: cache buffer chains” link, the histogram for that wait event is displayed.

You can move the slider box along the time line to go back in the past. After the time window is set, the Detail sections are automatically updated to reflect the corresponding period.

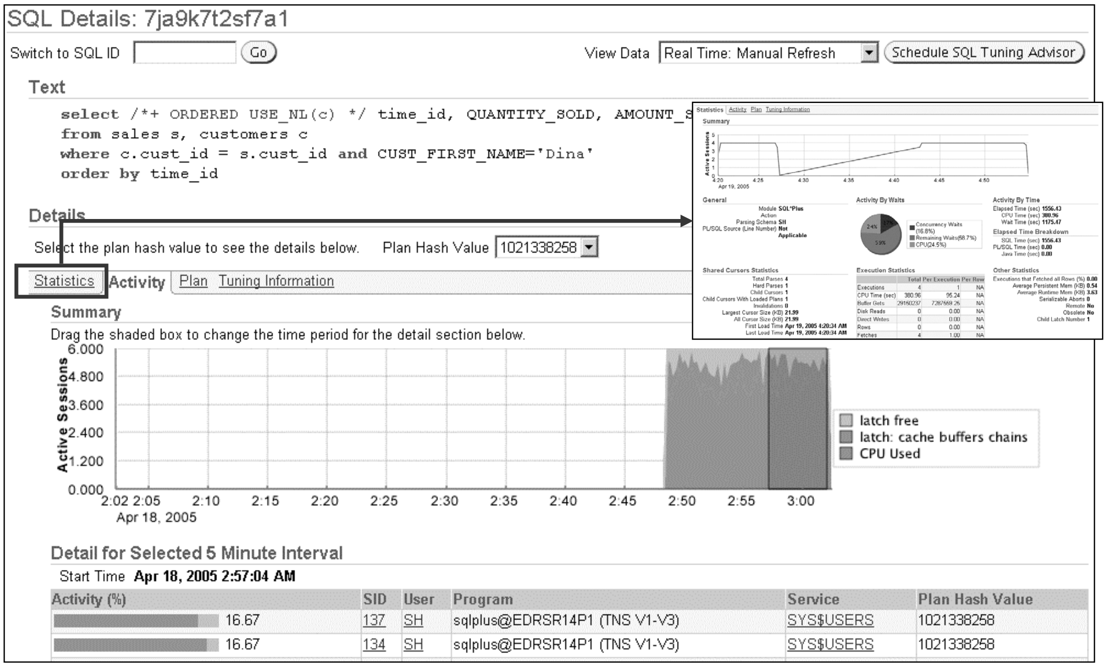
The Top SQL section shows you the corresponding top SQL statements waiting for the waits of the chosen class. This section displays the SQL statements that were found waiting the most times during the sample interval. The idea is that if one statement has the majority of the wait time, then it should be looked into. This is the case here, so you drill down on this statement.

By default, the Top Sessions section is displayed, but you can also look at Top Services, Top Modules, and Top Actions. In the slide, the Top Sessions section displays the sessions that were found waiting the most during an interval. In this case, the waits are fairly well balanced. But if one session were to stand out, it should be looked at in more detail.

Active Sessions Waiting Pages (continued)

The basic idea behind these graphics is to show the same information aggregated on different dimensions. Click the SQL ID link or the Session ID link to see more details about the problematic SQL statements or sessions.

SQL Details



Activity (%)	SID	User	Program	Service	Plan Hash Value
16.67	137	SH	sqlplus@EDRSR14P1 (TNS V1-V3)	SYS$USERS	1021338258
16.67	134	SH	sqlplus@EDRSR14P1 (TNS V1-V3)	SYS$USERS	1021338258

SQL Details

Click a SQL ID in the Top SQL section to access the SQL Details information page. The Activity tabbed page displays the users executing the SQL statements and the breakdown of CPU and wait events by each user and for all users cumulatively.

Use the Statistics tab to:

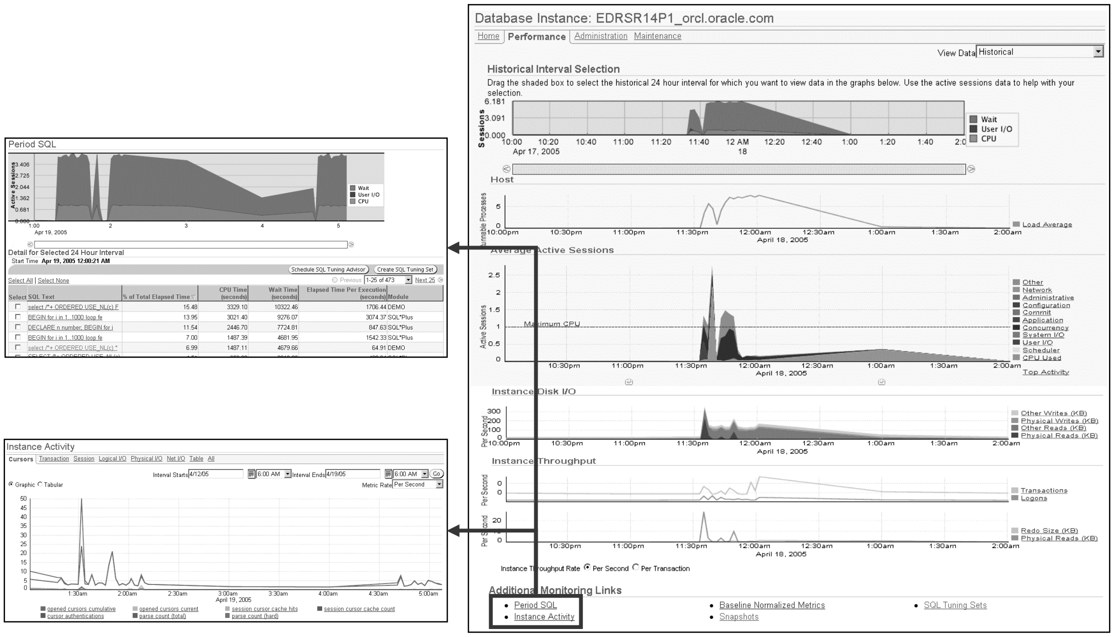
- View the SQL statement for the SQL ID
- View real-time or historical execution activity and statistics
- Select another SQL ID to view
- Select a plan hash value if more than one plan is available for this SQL statement
- View several different types of charts for historical data

Only a few clicks are necessary to identify which SQL statement is causing the problem.

To understand how to fix this problem, you need to find the corresponding ADDM analysis. Retrieve this analysis by clicking the Database tab at the top of the page to access the Performance page.

Note: You can also view historical information on this page by selecting Historical from the View Data drop-down list. In this mode, the Statistics tabbed page offers information such as Executions Per Hour, Elapsed Time Per Execution, Disk Reads Per Execution, and Buffer Gets Per Execution.

Historical Data View



Historical Data View

Enterprise Manager provides a Historical view. You can view the history of database session activity for both real-time (less than one hour) and historical-time contexts. This feature greatly increases your ability to diagnose problems related to a particular database session because it permits you to go back in time and view the history of the session activity.

When you select Historical from the View Data drop-down list, another page appears with a Historical Interval Selection chart. You drag the shaded box to the desired 24-hour interval to update the charts on the page. The historical view provides the following monitoring links:

- **Period SQL:** Shows up to 810 SQL statements in a selected 24-hour period. You can change the period, access the SQL Details page for one of the statements, run the SQL Tuning Advisor to initiate corrective action for one or more statements, or group selected statements into a SQL Tuning Set.
- **Instance Activity:** Enables you to view database activity for specific data about groups of metrics, such as cursors, sessions, and transactions

Diagnosis of Hung or Extremely Slow Databases

- **New functionality for problem analysis when the database is performing very slowly or is hung:**
  - **Direct access to SGA for performance monitoring (memory access mode)**

    `V$SESSION`

    `V$SESSION_WAIT`

    `V$SYSTEM_EVENT`

    `V$SYSSTAT`
  - **Hang analysis using Enterprise Manager**
- **Also supported for Oracle9*i* databases**

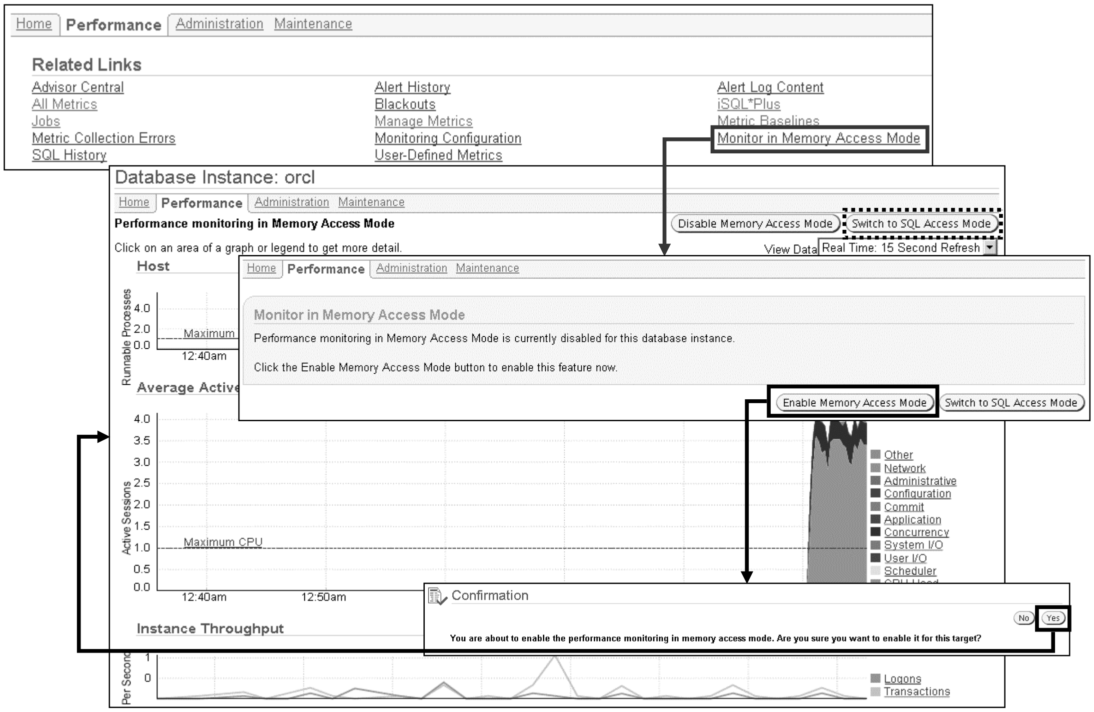
Diagnosis of Hung or Extremely Slow Databases

This functionality for problem analysis is available when the database is performing very poorly or is actually hung.

The system supports the collection of real-time performance statistics directly from the System Global Area (SGA) by using optimized or lightweight system-level calls as an alternative to SQL. In Enterprise Manager (EM), this is referred to as the *memory access mode*. There is one SGA collector thread per Oracle instance; it is automatically started by the EM agent when it starts monitoring a database instance. The `V$` views shown in the slide are the main performance views used for high-level performance diagnostics and, therefore, are the ones for which direct access to SGA is available. If more extensive drilldown information is required, you must use SQL to retrieve it. Host information, such as the number of CPUs and the host name, are also collected and made visible through the EM interface.

Enterprise Manager's hang analysis feature builds on the current `ORADEBUG` Hang Analysis utility to provide a graphical user interface.

Using Memory Access Mode



Using Memory Access Mode

You can access the memory access mode graphic page in the Related Links section of the Home page by clicking the Monitor in Memory Access Mode link. The link takes you to the Performance page in Memory Access view mode. As shown in the slide, you have to Enable Memory Access Mode for the first time, and you can disable it later if you want.

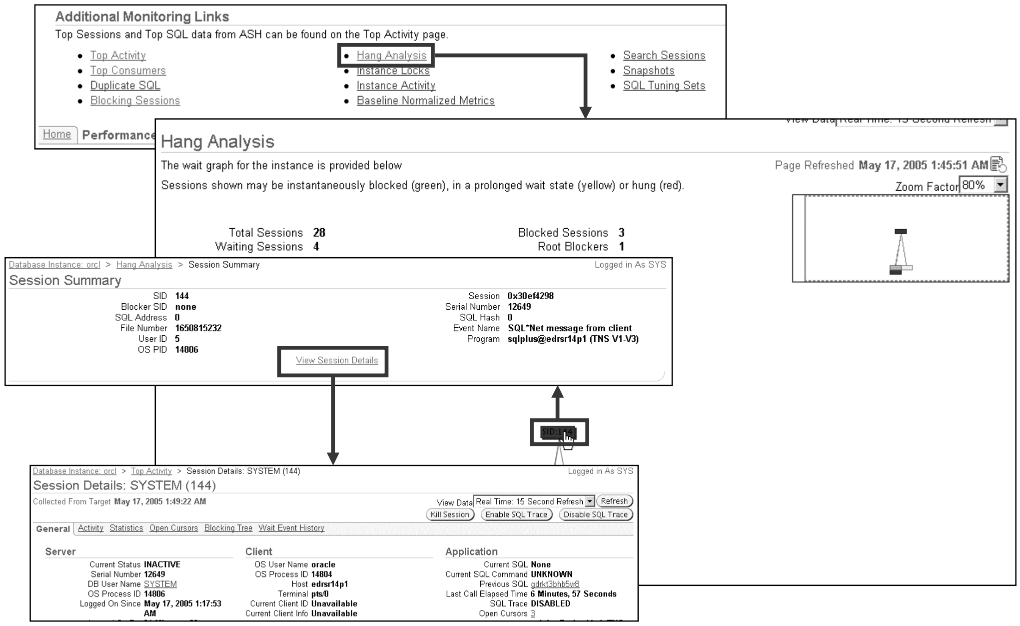
When you want to return to SQL Access view mode, click the “Switch to SQL Access Mode” button on the Performance page.

Memory Access mode avoids the computation associated with parsing and executing SQL statements, thereby making it robust for severe cases of library cache contention that can prevent the instance from being monitored using SQL. You should switch to Memory Access mode for slow or hung systems.

Pages in Memory Access mode contain data sampled at a higher frequency than pages in SQL mode. Charts may appear to be slightly different from SQL mode for this reason. Consequently, the Memory Access mode page provides better information about where events begin and end, and you may also detect short-duration events that might otherwise be missed.

Note: Memory access mode is also called *direct SGA attach mode*.

Using the Hang Analysis Page



Using the Hang Analysis Page

The ORADEBUG Hang Analysis utility is provided by the database to capture the instantaneous system wait-for-graph. This is the building block for the hang analysis graphical user interface in Enterprise Manager (EM). This functionality exists in all post-Oracle9*i* databases and can be used to perform both instancewide and clusterwide analysis of waiting sessions.

To analyze hangs and system slowdowns, you can use the Hang Analysis page in the Additional Monitoring Links section of the Performance page in EM. You can also use the Blocking Sessions page to display a list of all sessions currently blocking other sessions.

You use the Hang Analysis page to:

- Determine which sessions are causing bottlenecks
- Examine session summaries for blocking or blocked sessions

Using the Hang Analysis Page (continued)

This page provides you with a graphical topology of waiting sessions in the system, with blocked sessions appearing below blocking sessions. Enterprise Manager assesses the situation on the basis of historical activity and determines which sessions are hung or likely to be hung, rather than being only instantaneously in a wait state. To view summary information about a session, click a session ID in the topology. The Session Summary page appears, which displays general information about the selected session. You can click View Session Details in the Session Summary page to get further session information and determine whether terminating the session is beneficial. The sessions are displayed in green, yellow, or red depending on the perceived seriousness of the state of the session within the topology.

You can also select a Zoom Factor percentage on the basis of the desired size of the topology. In cases where there are many blocked sessions, you can use a smaller factor to visualize the overall wait information for the system. You can then zoom in on relevant portions and read the topology or drill down further.

You click in an area of the Zoom Factor microwindow to move the topology to the desired viewing position. For example, if you click the left side of the window, the topology shifts to the right to enable you to fully view the left side of the topology.

Summary

In this lesson, you should have learned how to:

- Use Enterprise Manager pages to identify performance issues
- Eliminate and identify operating system issues

Practice 7 Overview: Reactive Tuning

This practice covers the following topics:

- Using Enterprise Manager performance pages to identify or eliminate OS issues
- Using Enterprise Manager performance pages to identify database performance issues in an application workload

ORACLE

8

Tuning the Shared Pool

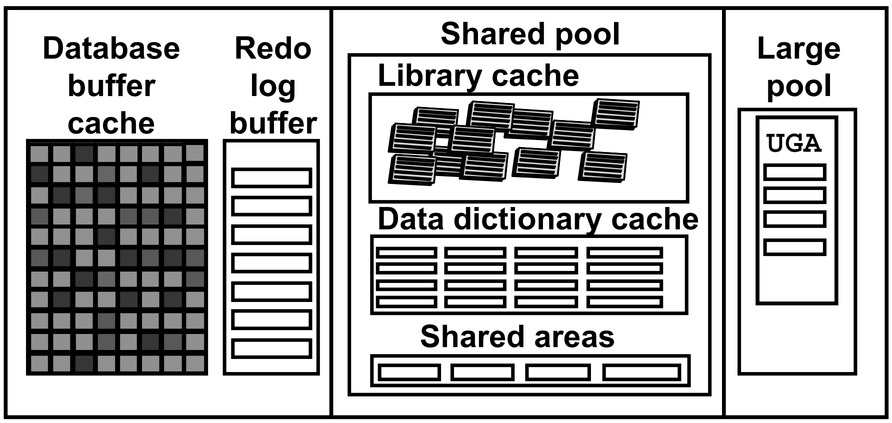
Objectives

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

- Diagnose shared pool problems
- Size the shared pool
- Size the reserved area
- Keep objects in the shared pool

Shared Pool Architecture

Major components of the shared pool are:
- **Library cache**
- **Data dictionary cache**
- **User global area (UGA) for shared server sessions**



Shared Pool Architecture

The fundamental purpose of the shared pool is to serve as a metadata cache. Much of the shared pool usage is to support the execution of shared SQL and PL/SQL packages. For example, to build a cursor for a simple `SELECT` statement, the optimizer must be aware of the table, column, and index metadata and optimizer statistics. This data is cached in the shared pool independent of the cursor, so that it can be reused. The shared pool holds many named areas and fixed areas that are allocated on startup. The `V$SGASTAT` view lists these areas. The major components of the shared pool are:

- The library cache, which stores shared SQL and PL/SQL code and object metadata in areas that are distinguished by namespaces
- The data dictionary cache, which holds row images from the data dictionary tables

Space allocation in the shared pool is managed by a least recently used (LRU) algorithm. The DBA is most concerned about shared pool allocations that are re-creatable. The re-creatable allocations can be aged out to make space for other re-creatable allocations. For best performance, you must strike a balance between the amount of memory dedicated to the shared pool and the processing required to reload these re-creatable objects.

Shared Pool Operation

The shared pool is managed by an LRU algorithm.

- **New objects require memory allocations.**
- **Re-creatable objects are aged out of the cache.**
- **Objects are made up of chunks of memory.**
- **A memory allocation is a whole chunk.**
- **A chunk is contiguous.**

Shared Pool Operation

The Oracle database server removes releasable objects from the shared pool when a new object requires memory. These new objects are most often new statements that require memory in the library cache, but they can be data dictionary objects in the row cache or other objects. The Oracle database server uses an LRU algorithm to determine the objects to be removed. The server processes scan the LRU list removing the least recently used objects and checking for sufficient contiguous memory to satisfy the request. If sufficient contiguous memory cannot be found, an additional granule of memory is requested if the automatic shared memory manager is enabled; otherwise, you receive an error (`ORA-04031`).

SQL and PL/SQL statements do not require contiguous memory for the entire statement or block. Memory heaps that are used to represent the SQL statement in the shared pool are allocated in chunks. The chunks do not have to be contiguous, but each chunk allocation must be contiguous. The chunk sizes can vary, but memory is allocated (where possible) in 1 KB or 4 KB chunks to create more uniform memory allocations. As a result, the same kinds of objects and chunks sizes are allocated and aged out at a steady state. This allocation method reduces fragmentation. Practice shows that, in most cases, it reduces total memory usage as well.

Library Cache

- Stores complex object metadata associated with cursors
- Stores SQL statements and PL/SQL blocks that are to be shared by users
- Prevents statement reparsing

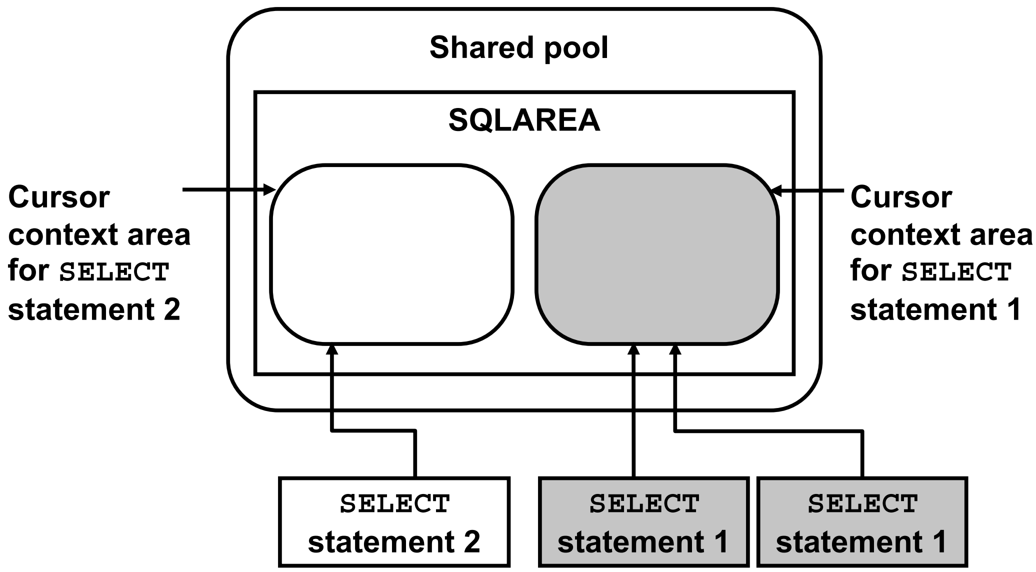
ORACLE

Library Cache

The library cache is the most important part of the shared pool from a tuning perspective. If the library cache is well tuned, the other components of the shared pool usually work efficiently.

Complex object metadata that is associated with the cursors is stored in the library cache. The Oracle database server uses the library cache to store SQL statements and PL/SQL blocks so that users can share statements, thus reducing the need to parse a similar statement. Each of these types of data associated with the cursors has a name (or namespace) assigned to the allocation. This naming allows the memory usage to be displayed as a sum of a type of allocation. For example, the `sqlarea` namespace shows all of the allocation for SQL and PL/SQL executable code. The tables, procedures, and index namespaces show the allocations for each of these object types.

SQL and PL/SQL Storage



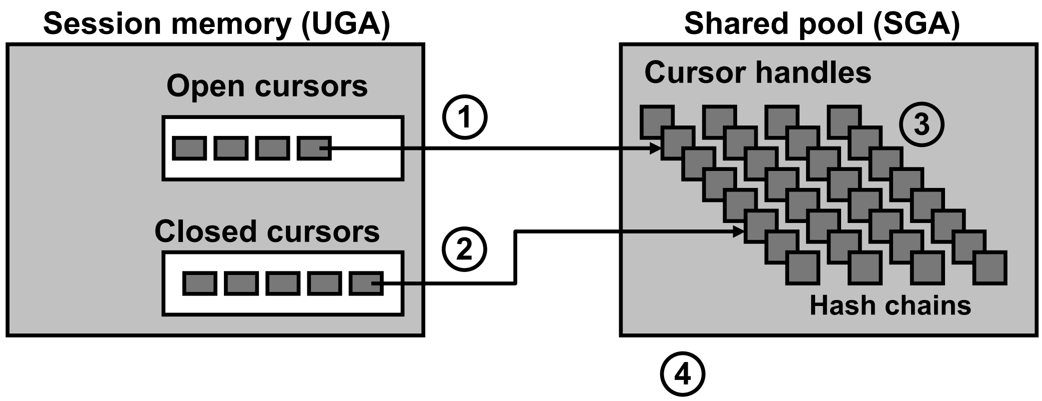
SQL and PL/SQL Storage

The Oracle server uses the library cache and `sqlarea` to store SQL statements and PL/SQL blocks. To determine whether a statement is already cached, the Oracle server:

- Reduces the statement to the numeric value of the ASCII text
- Uses a hash function of this number

The hash value is not a unique value, and several statements may hash to the same value. The cursor contexts for these statements are all stored in the same hash chain. The hash chain is searched for the correct statement. When a statement is submitted the first time the cache is searched, the cursor is constructed from the statement. When the statement is submitted subsequently, the cursor handle is found and the cursor is reused.

Cursor Usage and Parsing



Parse procedure:

1. **Find and execute an open cursor.**
2. **Find a closed cursor in the session cache.**
3. **Search the hash chains (soft parse).**
4. **Construct the cursor (hard parse).**

ORACLE

Cursor Usage and Parsing

All developers want their code to run as fast as possible. For code with SQL statements, that means cursor access must be fast. The fastest possible access of a cursor is through the open cursor cache in the session memory of the server session. Every open cursor in the open cursor cache has a pointer to the SGA memory location of that cursor handle. To execute the cursor, the pointer is used; parsing is not required. An open cursor has already been parsed and the cursor handle is in the library cache.

When a cursor is closed, the cursor information is moved into the session's closed cursor cache, if the `SESSION_CACHED_CURSORS` parameter has been set to some value. (The default is 0 for database versions earlier than 10.2.0.2; the value is 50 for later versions.)

When a cursor is opened, the session hashes the SQL statement and performs a hash lookup in the closed cursor cache. If the cursor is found, it is moved to the open cursor cache, then the pointer to the cursor handle in the shared pool is used to execute the cursor. No parse is required.

If the cursor is not found in the session, the hash value is used to search the hash chains in the shared pool for the cursor handle. The search is registered as a parse. If the cursor handle is found and the rest of the cursor has not aged out, the cursor is executed. This is a soft parse.

Cursor Usage and Parsing (continued)

If some part of the cursor has aged out of the shared pool, or if the cursor does not exist in the shared pool, then the cursor is constructed. This is a hard parse. The cursor construction requires lookups of the metadata for dependent objects such as tables, indexes, extents, and sequences. If the metadata for these objects is not already cached in the shared pool, recursive SQL is generated to fetch the information from the data dictionary.

Important Shared Pool Latches

- `latch: shared pool` **protects memory allocations in the shared pool.**
- `latch: library cache` **locates matching SQL in the shared pool.**

Important Shared Pool Latches

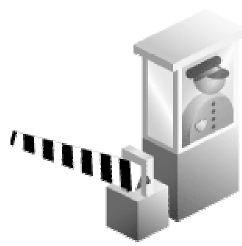
Latched operations are protected by a latch. The process must obtain a latch to perform the operation. Latches are created in memory. Typically, one latch protects multiple objects or a class of objects. Latches protect (serialize) operations on memory objects to prevent multiple processes from accessing the same object at the same time. This prevents corrupting memory objects, and assures that the object does not change between the time it is read and updated.

The `latch: shared pool` latch must be obtained for any shared pool memory allocation. Hard parses require memory allocations. Contention for the shared pool latch generally indicates a high rate of memory allocations and deallocations. This is often associated with an undersized shared pool.

When a cursor handle is requested for an execute, it must be pinned; this is a latched operation protected by the `latch: library cache` latch. When the hash chains are searched for either a soft parse (the statement is located) or a hard parse (the statement is not located), the operation is protected by `latch: library cache`. If the hash chains are frequently searched, the latch is held more frequently and there is more contention. Select operations from `V$SQL` and `V$OPEN_CURSORS` also require the `latch: library cache` latch.

Mutex

A mutual exclusion object enables:
- Sharing of a resource without corruption
- Shared access for reads
- Exclusive access for update
- Each object to have its own mutex



Mutex

Mutex is the short form of *mutual exclusion object*. A mutex, similar to a latch, is a low-level serialization mechanism used to control access to a shared data structure in the SGA. Serialization is required to avoid an object being:

- Deallocated while someone is accessing it
- Read while someone is modifying it
- Modified while someone is modifying it
- Modified while someone is reading it

Mutexes can be defined and used in different ways, as in the following examples:

- Each structure being protected by a mutex can have its own mutex (for example, a parent cursor has its own mutex, and each child cursor has its own mutex).
- Each structure can be protected by more than one mutex, with each mutex protecting a different part of the structure.
- A mutex can protect more than one structure.

The mutex can be stored either within the structure it protects or elsewhere. Mutexes are usually dynamically created at the same time as the structure they protect. If a mutex is embedded within the structure it protects, it is implicitly destroyed when the owning structure is freed.

Benefits of Mutexes

Performance is improved with mutexes because they:

- **Are smaller and faster**
- **Have less potential for contention**
- **Replace latches and pins**

Benefits of Mutex

Although mutexes and latches are both serialization mechanisms, mutexes have certain features that latches do not.

- **Smaller and Faster**
  Mutexes are an alternative to latches because they are smaller and much faster to get. A mutex get uses fewer instructions compared to a latch get. A mutex takes less memory space compared to a latch.
- **Less Potential for False Contention**
  Latches typically protect multiple objects. When a latch protects one or more hot objects, the latch itself can become a serialization point when accessing any of the objects protected by that latch. This can be a false contention point, where the contention is for the protection mechanism (that is, latch), rather than the target object you are attempting to access. Unlike latches, with mutexes it is possible to create a mutex for each structure protected. This means that false contention is much less likely because each structure can be protected by its own mutex.

Benefits of Mutexes (continued)

- **Replace Latches and Pins**
  A mutex can be concurrently referenced by many sessions, providing all sessions reference the mutex in S (Shared) mode. The total number of sessions referencing a mutex in S mode is called the reference count ("ref count"). The ref count for a mutex is stored within the mutex itself. A mutex can also be held in X (eXclusive) mode by one session only.
  Mutexes have a dual nature; they can act as a serialization mechanism (for example, latch) and also as a pin (for example, preventing an object from aging out). For example, the ref count of a mutex is a replacement for a library cache pin. Instead of each session creating and then deleting a library cache pin when executing a cursor, each session increments and decrements the ref count (so the ref count replaces *n* distinct pins).

Note: Latches and mutexes are independent mechanisms; that is, a process can hold a latch and a mutex at the same time.

Mutex Views and Statistics

- **Mutex views**
  - `V$MUTEX_SLEEP`
  - `V$MUTEX_SLEEP_HISTORY`
- **Mutex wait events**
  - `cursor:mutex X`
  - `cursor:mutex S`
  - `cursor:pin X`
  - `cursor:pin S`
  - `cursor:pin S wait on X`

Mutex Views and Statistics

Mutex operations are faster and have less contention than latches, but mutex operations still have waits associated with them. Two `V$` views provide details of mutex sleeps:

- `V$MUTEX_SLEEP` shows a summary of sleeps and wait time for a particular `mutex_type/location` combination.
- `V$MUTEX_SLEEP_HISTORY` shows sessions sleeping for a particular `mutex_type/location` combination by time while it is held by a specific holding session.

Mutex wait events have two categories:

- `cursor:mutex` indicates that the mutex waits on parent cursor operations and statistics block operations.
- `cursor:pin` events are waits for cursor pin operations, where a mutex has replaced the `latch:library cache pin`.

Mutex wait events are of two types:

- Short-duration events that should rarely be seen. These occur when one process attempts to update the mutex while it is being changed by another process. The waiting process will spin waiting for the mutex to be available. For example, `cursor:pin S` is incremented when another process is updating the reference count (pin) of a shared cursor.

Mutexes Views and Statistics (continued)

- Long-duration events occur when a process must wait for other processes to finish their operation. For example, `cursor: mutex X` is incremented when a process wants an exclusive access but the mutex is being held exclusive or shared by another process.

Mutex-Protected Operations

In Oracle Database 10.2.0.2 and later versions, mutexes protect:

- **Selects from `V$SQLSTAT`**
- **Searches of child cursor lists**

Mutex-Protected Operations

A mutex is another protection mechanism that can protect critical operations. From Oracle database version 10.2.0.2 and later, a `SELECT` from the `V$SQLSTAT` view is protected by mutexes. The use of mutex-protected operations is significantly faster than latched operations. The child cursor lists are protected by mutexes.

Statspack and AWR Indicators

Statspack and AWR reports include indicators in the following sections:

- Load Profile
- Instance Efficiencies
- Top Wait Events
- Time Model

ORACLE

Load Profile

```
Load Profile              Per Second       Per Transaction
~~~~~~~~~~~~         ---------------       ---------------
User calls:                     4.29                 21.78
     Parses:                  188.09                954.20
Hard parses:                   80.00                405.85

 % Blocks changed per Read:     0.18
          Recursive Call %:    99.78
Rollback per transaction %:     9.76
             Rows per Sort:     8.47
```

ORACLE

Load Profile

The Load Profile section (in both Statspack and AWR reports) shows the parses and hard parses. In the example in the slide, hard parses are approximately 40% of the total number of parses.

A high percentage of hard parses indicates that the SQL cursors are not being shared. On an OLTP system or any environment where users are running an application, the percentage of hard parses should be very small. In Decision Support Systems (DSS) and Data Warehouse (DWH) systems where users are accessing the database with ad hoc tools, the percentage of hard parses is much higher. Whether this number is meaningful depends on the type of system you are running. Without a baseline, this number has little meaning. With a baseline, all you can say is that the workload has changed.

Instance Efficiencies

```
Instance Efficiency Percentages
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Buffer Nowait %:  100.00       Redo NoWait %:  100.00
Buffer  Hit   %:   99.87    In-memory Sort %:  100.00
Library Hit   %:   87.90        Soft Parse %:   57.47
Execute to Parse %:   22.50      Latch Hit %:  100.00
Parse CPU to Parse Elapsd %:   83.16
            % Non-Parse CPU:   17.88

 Shared Pool Statistics        Begin   End
                               ------  ------
             Memory Usage %:   92.75   92.97
    % SQL with executions>1:   72.62   72.28
  % Memory for SQL w/exec>1:   92.35   92.17
```


Instance Efficiencies

In the Instance Efficiencies section of the reports, there are several indicators that there is a shared pool issue:

`Library Hit%` shows the percentage of times a requested object was found in the cache.

`Soft Parse%` shows the percentage of times that a requested cursor was found and reused.

`Execute to Parse%` is the percentage of the difference in number of executions and the number of parses divided by the number of statements. This number could be negative if executions have not occurred before the end of a snapshot or if the shared pool is too small and queries are aging out of the shared pool and need to be reparsed before they have finished executing.

`Parse CPU to Parse Elapsed%` shows that the CPU time is taking 83% of the total elapsed time for parsing. This indicates that there are some waits during parsing.

Top Waits

```
Top 5 Timed Events                                  Avg %Total
~~~~~~~~~~~~~~~~~~                                 wait   Call
Event                        Waits    Time (s)     (ms)   Time
----------------- ------------ ----------- ------ ------
CPU time                                  190           98.0
latch: library cache            76          1      19      .8
latch: shared pool              49          1      16      .4
db file sequential read      1,343          0       0      .2
log file parallel write         54          0       5      .1
-------------------------------------------------------
```


Top Waits

Contention in the shared pool shows up as waits on the latches.

`Latch: library cache` indicates that a significant amount of time was spent scanning the SQL hash chains for cursors. Note that the scan takes place even if the cursor is not in the cache.

`Latch: shared pool` indicates that there were a large number of memory allocations that required holding the shared pool latch.

Note: In this example, the total time waited for both latches is only 2% of the total time. Therefore, there will not be a noticeable performance improvement, even if all the contention on these latches were removed. The contention on these latches does point to a shared pool issue and, in combination with the other indicators, shows that cursors are not being shared.

Time Model

```
Time Model System Stats
-> Ordered by % of DB time desc, Statistic name
                                                          % of
Statistic                                     Time (s) DB time
----------------------------------- ---------- ---------
sql execute elapsed time                         399.5      95.6
DB CPU                                           269.3      64.4
parse time elapsed                               126.8      30.3
hard parse elapsed time                          111.9      26.8
PL/SQL compilation elapsed time                    6.6       1.6
PL/SQL execution elapsed time                      5.9       1.4
connection management call elapsed                 4.9       1.2
failed parse elapsed time                          4.0       1.0
hard parse (sharing criteria) elaps                2.3        .5
sequence load elapsed time                         0.5        .1
hard parse (bind mismatch) elapsed                 0.2        .0
repeated bind elapsed time                         0.1        .0
DB time                                          417.9
```


Time Model

The Time Model section is one of the few statistics that can be directly compared between two reports. The `% of DB time` column normalizes the time to the `DB time` of the report.

Recall the statistics rollup, so that parse time elapsed includes hard parse elapse time. Many of these statistics apply directly to tuning the shared pool. The report section shown is based on the `V$SYS_TIME_MODEL` view. For information at the session level, you can `SELECT` from `V$SESS_TIME_MODEL`.

```
SQL> SELECT stat_name, value FROM V$SYS_TIME_MODEL;

STAT_NAME                                               VALUE
------------------------------------------------- ----------
DB time                                            4974523220
DB CPU                                             2066671570
background elapsed time                             956815973
background cpu time                                 277464093
sequence load elapsed time                            9040910
parse time elapsed                                  461952552
hard parse elapsed time                             378922357
```

Time Model (continued)

```
sql execute elapsed time                                4792104462
connection management call elapsed time                   70987475
failed parse elapsed time                                    97761
failed parse (out of shared memory) elapsed time                 0
hard parse (sharing criteria) elapsed time                13683106
hard parse (bind mismatch) elapsed time                    1924234
PL/SQL execution elapsed time                            370402811
inbound PL/SQL rpc elapsed time                                  0
PL/SQL compilation elapsed time                           24675951
Java execution elapsed time                                      0
repeated bind elapsed time                                 2637182
RMAN cpu time (backup/restore)                                   0

19 rows selected.
```

```
SQL> SELECT sid, stat_name, value FROM V$SESS_TIME_MODEL
  2  order by sid;

SID STAT_NAME                                        VALUE
---------- ---------------------------------------- ----------
Lines deleted …
       141 DB time                                           111429752
       141 DB CPU                                             95541752
       141 background elapsed time                                   0
       141 background cpu time                                       0
       141 sequence load elapsed time                             2567
       141 parse time elapsed                                  3189864
       141 hard parse elapsed time                             2789939
       141 sql execute elapsed time                           86099325
       141 connection management call elapsed time               13882
       141 failed parse elapsed time                                 0
       141 failed parse (out of shared memory) elapsed time          0
       141 hard parse (sharing criteria) elapsed time           531073
       141 hard parse (bind mismatch) elapsed time                   0
       141 PL/SQL execution elapsed time                      26029287
       141 inbound PL/SQL rpc elapsed time                           0
       141 PL/SQL compilation elapsed time                      630044
       141 Java execution elapsed time                               0
       141 repeated bind elapsed time                            11853
       141 RMAN cpu time (backup/restore)                            0
       142 DB time                                            19107141
       142 DB CPU                                             17568564
       142 background elapsed time                                   0
Lines deleted …

456 rows selected.
```

Library Cache Activity

```
Library Cache Activity  DB/Inst: ORCL/orcl  Snaps: 61-71
->"Pct Misses"  should be very low
                 Get  Pct        Pin        Pct             Invali-
Namespace  Requests  Miss     Requests     Miss  Reloads  dations
---------- --------- ------ -------------- ------ ------ --------
BODY              33   27.3            450   14.0      46        0
CLUSTER           49    0.0            200    3.0       6        0
INDEX              5  100.0         35,300    0.0       0        0
SQL AREA      18,880   96.9         89,904   36.5     496        0
```


Library Cache Activity

The SQL area is the most important part of the library cache. This is the area where the memory for compiled cursors is allocated. In this example, the `Pct Miss` value is very high, indicating that cursors are not being reused or reloaded (because of age).

This section of the Statspack and AWR reports is based on the `V$LIBRARY_CACHE` view.

Terminology

- Gets: (Parse) The number of lookups for objects of the namespace
- Pins: (Execution) The number of reads or executions of the objects of the namespace
- Reloads: (Parse) The number of library cache misses on the execution step, thereby causing an implicit reparsing of the SQL statement
- Invalidations: The number of times an object is modified by a DDL, causing all dependent objects to be marked invalid

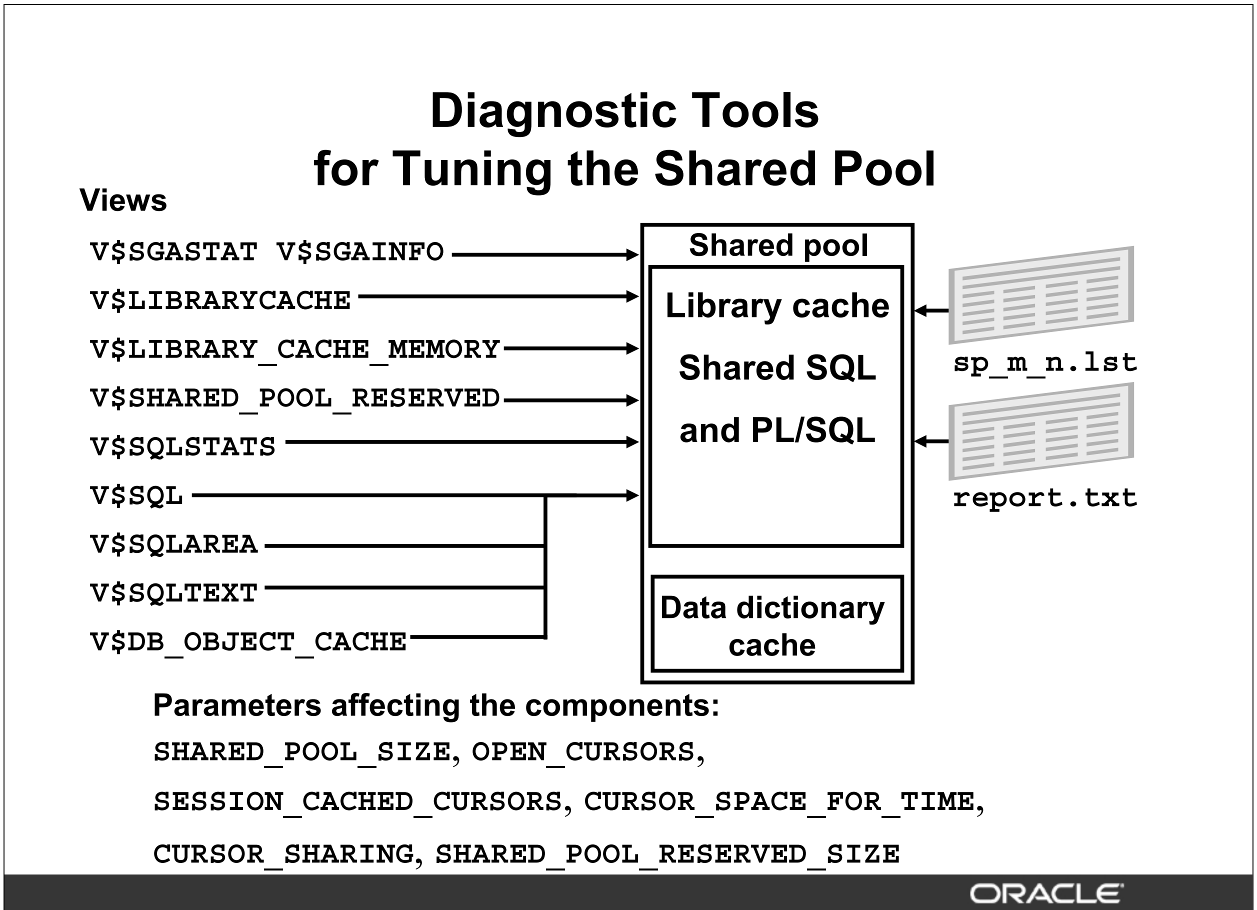
Terminology

Each row in V$LIBRARYCACHE contains statistics for one type of item kept in the library cache, grouped by the value of the NAMESPACE column. The statistics shown are for the entire namespace. Rows of the table with the following NAMESPACE values reflect library cache activity for SQL statements and PL/SQL blocks: SQL AREA, TABLE/PROCEDURE, BODY, and TRIGGER.

Rows with other NAMESPACE values reflect library cache activity for object definitions that the server uses for dependency maintenance: INDEX, CLUSTER, OBJECT, PIPE, JAVA SOURCE, JAVA RESOURCE, and JAVA DATA.

Keywords related to the namespace are:

- GETS: Shows the total number of requests for information about the corresponding item
- PINS: Shows the number of executions of SQL statements or procedures
- RELOADS: Shows the number of times, during the execution phase, that the shared SQL area containing the parsed representation of the statement is aged out of the library cache to make room for another statement. The Oracle server implicitly reloads the statement and parses it again.
- INVALIDATIONS: Shows the number of statements that have been made invalid due to the modification of a dependent object. Invalidations also cause reloads.



Diagnostic Tools for Tuning the Shared Pool

The V$SGASTATS or V$SGAINFO view displays the sizes of all SGA structures. V$SGASTATS is more detailed for the various pieces of the shared pool. V$SGAINFO provides a summary including the granule size and whether the memory area is resizable.

The contents of the shared pool are not aged out if free memory is available in the shared pool. Use the following dynamic views to diagnose performance issues related to the library cache:

- V$LIBRARYCACHE: Statistics on library cache management
- V$LIBRARY_CACHE_MEMORY: Statistics on memory use for each of the namespaces
- V$SQLAREA: Full statistics about all shared cursors and the first 1,000 characters of the SQL statement
- V$SQL: Statistics about shared SQL area; contains one row for each child of the original SQL text entered. The V$SQL view is a similar view to V$SQLAREA, except that it does not include a GROUP BY clause that can make the V$SQLAREA view expensive to query. A SELECT of V$SQL is a latched operation.

Diagnostic Tools for Tuning the Shared Pool (continued)

- V$SQLSTATS: A low impact view for use instead of the other V$SQL* views. V$SQLSTATS uses a mutex rather than a latch.
- V$SQLTEXT: The full SQL text without truncation, in multiple rows
- V$DB_OBJECT_CACHE: Database objects cached, including packages; also objects such as tables and synonyms, where these are referenced in SQL statements

The Statspack and AWR reports contain information from these views and also contain information about SQL statements found in the library cache when either of the snapshots was taken.

Library Cache Reloads

- **Reloads should be less than 1% of the pins:**

```
SQL> SELECT SUM(pins) "Executions",
  2         SUM(reloads) "Cache Misses",
  3         SUM(reloads)/SUM(pins)
  4   FROM V$LIBRARYCACHE;
```

- **If the reloads-to-pins ratio is greater than 1%, increase the value of the `SHARED_POOL_SIZE` parameter.**

Executes `PROC1` —> 1st pin, 1 load
Executes `PROC1` —> 2nd pin, no reload
Executes `PROC1` —> 3rd pin, no reload
Executes `PROC1` —> 4th pin, no reload

Four pins and no reloads

How to Get the Reloads-to-Pins Ratio

The `V$LIBRARYCACHE` view indicates whether statements that have already been parsed have been aged out of the cache. The number of reloads should not be more than 1% of the number of pins.

There are two possible reasons that the reloads-to-pins ratio is greater than 1%:

- The shared parsed areas have been aged out because of lack of space. This can happen even when the cursor is being used frequently.
  Solution: Avoid the frequent reloads by increasing the `SHARED_POOL_SIZE` initialization parameter.
- Shared parsed areas are invalidated.
  Solution: Perform housekeeping (such as creating indexes) during periods of light load.

Statspack and AWR reports: The Library Cache Activity section contains summarized information for the instance. The Instance Activity Statistics section contains the detailed statistics of the activities. Some of the statistics to look for include opened parse count (hard) and parse count (total).

Invalidations

The number of times that objects of the namespace were marked invalid and thus caused reloads:

```
SQL> SELECT COUNT(*) FROM hr.employees;
SQL> SELECT namespace,pins,reloads,
  2  invalidations
  3  FROM V$LIBRARYCACHE;
SQL> execute DBMS_STATS.GATHER_TABLE_STATS -
>    ('HR','EMPLOYEES');
SQL> SELECT COUNT(*) FROM hr.employees;
SQL> SELECT namespace,pins,reloads,
  2  invalidations
  3  FROM V$LIBRARYCACHE;
```

ORACLE

Invalidations

If a schema object is referenced in a SQL statement and that object is later modified in any way, the shared SQL area becomes invalidated (marked as invalid). The statement must be reparsed, and therefore reloaded, the next time it is executed.

For example, when a table, sequence, synonym, or view is re-created, altered, or dropped, or if a procedure or package specification is recompiled, all dependent shared SQL areas are invalidated.

Example

```
SQL> SELECT COUNT(*)
  2  FROM hr.employees;

SQL> SELECT namespace, pins, reloads, invalidations
  2  FROM V$LIBRARYCACHE;

NAMESPACE             PINS       RELOADS   INVALIDATIONS
--------------------- ---------- --------- -------------
SQL AREA              1616         12                0
…
```

Invalidations (continued)

Example (continued)

```
SQL> execute DBMS_STATS.GATHER_TABLE_STATS -
>       ('HR','EMPLOYEES');

PL/SQL procedure successfully completed.

SQL> SELECT COUNT(*) FROM hr.employees;

SQL> SELECT namespace, pins, reloads, invalidations
  2  FROM V$LIBRARYCACHE;

NAMESPACE             PINS    RELOADS INVALIDATIONS
--------------- ---------- ---------- -------------
SQL AREA              1688         14             3
…
```

In a production environment, DDL statements against your tables will be infrequent. The most frequent cause of invalidation may be the scheduled gathering of statistics for the optimizer.

Avoid Hard Parses

In an OLTP system, reduce misses by keeping hard parsing to a minimum:

- **Make sure that users can share statements.**
- **Prevent frequently used statements from being aged out by allocating enough space.**
- **Avoid invalidations that induce reparsing.**

ORACLE

Avoid Hard Parses

In an OLTP environment, you should minimize reparsing.

- If an application makes a parse call for a SQL statement and the parsed representation of the statement does not already exist in a shared SQL area in the library cache, the Oracle server parses the statement and allocates a shared SQL area.
- Ensure that SQL statements can share a shared SQL area whenever possible by using as much reusable code as possible—for example, by having bind variables rather than constants.
- Frequently used statements should always be found in the shared pool. If the pool is too small, such statements may be aged out between uses. This means that the subsequent execution requires a reparse. Allocating more memory to the library cache reduces library cache misses on execution calls.
- If a schema object is referenced in the execution plan of a SQL statement and that object is later modified in any way, then the parsed SQL statement held in memory is rendered invalid and must be parsed again when used.

Are Cursors Being Shared?

- **Check `GETHITRATIO` in `V$LIBRARYCACHE`:**

```
SQL> SELECT gethitratio
  2  FROM V$LIBRARYCACHE
  3  WHERE namespace = 'SQL AREA';
```

- **Determine which statements can be shared:**

```
SQL> SELECT plan_hash_value, count(*)
  2  FROM V$SQL
  3  GROUP BY plan_hash_value ORDER BY 2 DESC
```

```
SQL> SELECT sql_text, executions
 2   FROM V$SQLAREA
 3   WHERE plan_hash_value = NNNNNNNNN
```

ORACLE

Are Cursors Being Shared?

The V$LIBRARYCACHE view GETHITRATIO determines the percentage of parse calls that find a cursor to share (GETHITS/GETS). This percentage should be in the high 90s in OLTP environments. If not, you can try to:

- Improve the efficiency of your application code. This might not be an option if you do not have access to the application code.
- Increase the size of the shared pool:

```
SQL> SELECT namespace, gethitratio
  2  FROM V$LIBRARYCACHE;

NAMESPACE          GETHITRATIO
---------------    -----------
SQL AREA           .86928702
TABLE/PROCEDURE    .80073801
BODY               .6
```

Applications that use dynamic SQL and submit literal values create cursors that vary only in the literals. These applications have high hard parse rates and low GETHIT ratios. There are also child cursors that are identical but cannot be shared. The V$SQL_SHARED_CURSOR view explains why a particular child cursor is not shared with existing child cursors, with about 50 different reasons.

Are Cursors Being Shared? (continued)

Determine Which Statements Can Be Shared

Consider the statement in the slide:

```
SQL> SELECT plan_hash_value, count(*)
  2  FROM V$SQL
  3  GROUP BY plan_hash_value ORDER BY 2 DESC;
```

This statement returns the count of the `plan_hash_value` for unshared cursors ordered by the number of cursors using the same plan. Note that all cursors using the same plan can be shared.

Consider the second statement:

```
SQL> SELECT sql_text, executions
 2   FROM V$SQLAREA
 3   WHERE plan_hash_value = NNNNNNNNN;
```

Here, NNNNNNNNN is the `plan_hash_value` with the highest counts. It denotes the SQL statements that can be shared.

Sharing Cursors

Values for `CURSOR_SHARING` are:

- **`EXACT`**
- **`SIMILAR`**
- **`FORCE`**

Sharing Cursors

Examine the results of the queries from the previous page to determine if you have SQL statements that are identical, except for literal values in their `WHERE` clauses. If you find statements with these characteristics, you may be able to reduce the number of statements requiring space in the library cache by setting the `CURSOR_SHARING` parameter.

The value of the `CURSOR_SHARING` parameter determines the level at which SQL statements will share cursors:

- `EXACT`: This is the default value. With the `CURSOR_SHARING` parameter set to `EXACT`, SQL statements must be identical to share cursors.
- `SIMILAR`: SQL statements that are similar share cursors if their respective execution plans are the same. SQL statements do not share a cursor if the execution plan is not optimal for both statements.
- `FORCE`: SQL statements that are similar share cursors regardless of the impact on the execution plan.

What is a similar SQL statement? Statements that are identical, except for the values of some literals, are called *similar* statements.

Note: It is recommended to set `CURSOR_SHARING` to `EXACT` in a DSS environment or if you are using complex queries. Set `CURSOR_SHARING` to `FORCE` for OLTP.

Avoiding Soft Parses

- **Reducing soft parses reduces library cache latch contention.**
- **Keep soft parsing to a minimum by:**
  - **Setting `SESSION_CACHED_CURSORS`**
  - **Setting `HOLD_CURSOR` in the application precompiler**
  - **Setting `CURSOR_SPACE_FOR_TIME`**

Avoiding Soft Parses

In an OLTP environment, you should minimize reparsing. Soft parses are less expensive than hard parses, but still take some time. In an OLTP environment, the number of soft parses is often in orders of magnitude greater than hard parses.

`SESSION_CACHED_CURSORS` causes the session to keep a cache of recently closed cursors. The session searches the session cache first. If the cursor is found, the soft parse is avoided. To verify that your setting is efficient, compare the `session cursor cache hits` and `parse count` session statistics in `V$SESSTAT` for a typical user session. If few parses result in hits, you may increase the number. Remember that this increases overall demands on memory. Before version 10.2.0, the default for `SESSION_CACHED_CURSORS` is `0`, which means no caching. In version 10.2, it is 50.

The `CURSOR_SPACE_FOR_TIME` parameter causes the cursor to be held open in the shared pool and prevents it from aging out until the cursor is explicitly closed. PL/SQL cursors are always cached in an open state even when they are explicitly closed, so PL/SQL cursors can pin large amounts of memory when this parameter is set.

The `HOLD_CURSOR` parameter in the precompiler applications is equivalent to the `CURSOR_SPACE_FOR_TIME` parameter, but it can be set at the application level.

Avoiding Fragmentation

Avoid fragmentation by:

- **Upgrading to 10.2.0.*x***
- **Keeping frequently required large objects**
- **Reserving space for large objects**
- **Eliminating large anonymous PL/SQL blocks**
- **Enabling use of the large pool**

Avoiding Fragmentation

Reduce memory fragmentation by:

- Upgrading to 10.2.0.*x*. The memory allocation algorithm in Oracle Database 10*g* Release 2 (10.2) reduces the fragmentation by allocating chunks in 1 KB and 4 KB increments and larger.
- Keeping frequently required large objects such as SQL and PL/SQL areas in memory, instead of aging them out with the normal LRU mechanism
- Ensuring availability of contiguous space for large memory requirements through the allocation of reserved space in the shared pool area
- Using small PL/SQL packaged functions instead of large anonymous blocks
- Enabling the large pool. If the large pool is configured, it is used with:
  - Oracle Shared Server connections
  - Parallel query operations
  - RMAN parallel backup and recovery

Sizing the Shared Pool

- **Use Automatic Shared Memory Management.**
- **Use the Shared Pool Advisor and confirm using other diagnostics when data has operational history.**
- **Use 40% of the SGA size to start (when there is no history). Monitor and adjust as needed.**
- **Do not increase the size when free memory is available.**

```
SQL> SELECT * FROM V$SGASTAT
  2  WHERE NAME = 'free memory'
  3  AND POOL = 'shared pool';
```


Sizing the Shared Pool

The SHARED_POOL_SIZE parameter is the only parameter available to influence the library cache size. The dictionary cache takes some portion of the shared pool determined by an internal algorithm. The calculations that have been used to determine the size of the shared pool are based on many estimates about which objects were cached, the number of concurrent users, and concurrent open cursors. These estimates can lead to greatly different values.

Oracle recommends using Automatic Shared Memory Management (ASMM). This is enabled by setting the SGA_TARGET parameter to the value that the entire SGA will use. The Memory Monitor (MMON) process adjusts the memory allocated to dynamic pools to obtain the best performance within the memory allowed.

When the database is already in operation, the Shared Pool Advisor gives an indication of an estimated optimal size. As with any estimate, check other indicators such as latch waits, hard parses, and reloads for confirmation.

When there is no baseline, set the SHARED_POOL_SIZE to approximately 40% of the available SGA, and then monitor. Adjust as needed.

Shared Pool Advisory

```
SQL> SELECT shared_pool_size_for_estimate AS
  2    pool_size, estd_lc_size,
  3    estd_lc_time_saved
  4  FROM V$SHARED_POOL_ADVICE;
```

```
POOL_SIZE  ESTD_LC_SIZE ESTD_LC_TIME_SAVED
---------- ------------ ------------------
        32            8               7868
        40           15               7868
        48           17               7868
        56           17               7868
        64           17               7868
        72           17               7868
        80           17               7868
        88           17               7868
```

ORACLE

Shared Pool Advisory

The STATISTICS_LEVEL initialization parameter controls the shared pool advisory. The shared pool advisory statistics track the library cache's use of shared pool memory and predict the change in total instancewide parse time for different sizes of the shared pool. Two views, V$SHARED_POOL_ADVICE and V$LIBRARY_CACHE_MEMORY, provide information to help you determine how much memory the library cache is using, how much is currently pinned, how much is on the shared pool's LRU list, and how much time might be lost or gained by changing the size of the shared pool. These statistics are reset if the STATISTICS_LEVEL parameter is set to BASIC or when the instance is restarted.

The V$SHARED_POOL_ADVICE view displays information about the estimated time saved during parsing using different shared pool sizes. The sizes range from 50% to 200%, in equal intervals of the current shared pool size, and are not configurable. If rows in the V$SHARED_POOL_ADVICE view have the same values of parse time savings (given in the ESTD_LC_TIME_SAVED column), there would be no additional hits on those size ranges for library cache objects. However, if time saving values increase for larger pool sizes, it may help to increase the shared pool size.

Shared Pool Advisory (continued)

The `V$SHARED_POOL_ADVICE` view should be the first tool to use when sizing the shared pool. If the advisory indicates that a larger pool is not useful for the library cache memory objects, then you can drill down to see whether changing the SQL (to improve use of shared cursors) or other activity (such as deferring DDL to off hours) improves performance.

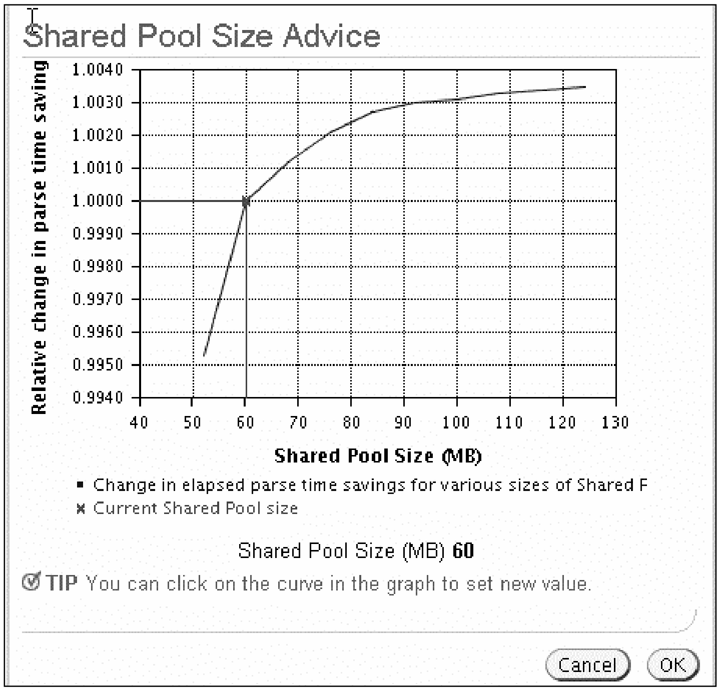
The `V$LIBRARY_CACHE_MEMORY` view displays information about memory allocated to library cache memory objects in different namespaces. A memory object is an internal grouping of memory for efficient management. A library cache object may consist of one or more memory objects.

The Statspack and AWR reports also present the shared pool advisory:

```
Shared Pool Advisory  DB/Inst: ORCL/orcl  End Snap: 71
-> SP: Shared Pool     Est LC: Estimated Library Cache   Factr: Factor
-> Note there is often a 1:Many correlation between a single logical object
   in the Library Cache, and the physical number of memory objects associated
   with it.  Therefore comparing the number of Lib Cache objects (e.g. in
   v$librarycache), with the number of Lib Cache Memory Objects is invalid

                                          Est LC Est LC  Est LC Est LC
    Shared    SP   Est LC                   Time   Time    Load   Load       Est LC
      Pool  Size     Size       Est LC     Saved  Saved    Time   Time          Mem
  Size (M) Factr      (M)      Mem Obj       (s)  Factr     (s)  Factr     Obj Hits
---------- ----- -------- ------------ ------- ------ ------- ------ -----------
        52    .9        8        1,515 #######    1.0   2,843    1.3   4,501,229
        60   1.0       15        2,632 #######    1.0   2,125    1.0   4,536,705
        68   1.1       22        3,890 #######    1.0   1,952     .9   4,546,706
        76   1.3       29        5,201 #######    1.0   1,826     .9   4,555,355
        84   1.4       36        6,505 #######    1.0   1,730     .8   4,560,858
        92   1.5       43        7,811 #######    1.0   1,691     .8   4,562,861
       100   1.7       50        9,114 #######    1.0   1,668     .8   4,564,104
       108   1.8       57       10,631 #######    1.0   1,645     .8   4,565,205
       116   1.9       64       11,319 #######    1.0   1,629     .8   4,565,788
       124   2.1       71       12,077 #######    1.0   1,618     .8   4,566,453
          -------------------------------------------------------------
```

This advisory indicates that only small time improvements would be gained by increasing the shared pool size.

Shared Pool Advisor



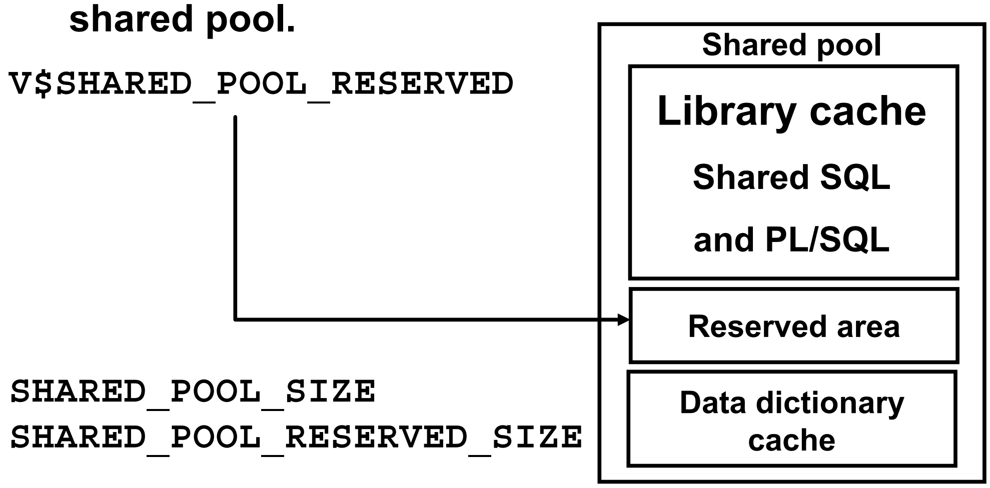
ORACLE

Shared Pool Advisor

You can access the Shared Pool Advisor from Enterprise Manager Database Control or Grid Control when Automatic Shared Memory Management is enabled. Click the Administration tab, the Memory Parameters link in the Database Configuration section, and the Advice button for Shared Pool Size.

Large Memory Requirements

- **Satisfy requests for large contiguous memory.**
- **Reserve contiguous memory within the shared pool.**



Large Memory Requirements

A large object and a large allocation are not equivalent. Most large objects are made up of 4 KB chunks. But some allocations, such as SESSION_PARAM_VALUES allocated at the start of each session, may require 30 KB. A small percentage of allocations exceed 4 KB, but hundreds of objects of 4 KB chunks could be freed without satisfying the request.

Why Reserve Space in the Shared Pool?

A small amount of space is reserved for large allocations; by default, this is 5% of the shared pool size. When an allocation for more than 4 KB is requested, the request will be satisfied from the shared pool, if possible, and then from the reserved space. Smaller allocations are not allowed in the reserved area. This means that the Shared Pool Reserved Area has large contiguous chunks of memory. The DBA can change the size of this reserved memory to satisfy large allocations during operations such as PL/SQL and trigger compilation.

Initialization Parameter

The size of the Shared Pool Reserved Area is set by the SHARED_POOL_RESERVED_SIZE initialization parameter. This parameter specifies how much of the value of SHARED_POOL_SIZE to reserve for large allocations. The default value should be adequate for most installations.

If SHARED_POOL_RESERVED_SIZE is greater than half of SHARED_POOL_SIZE, then the server signals an error.

Large Memory Requirements (continued)

`V$SHARED_POOL_RESERVED` View

This view helps in tuning the reserved pool within the shared pool. The first set of columns of the view are valid only if the `SHARED_POOL_RESERVED_SIZE` parameter is set to a valid value:

```
SQL> desc V$SHARED_POOL_RESERVED

Name                    Null?      Type
--------------------    --------   --------
FREE_SPACE              NUMBER
AVG_FREE_SIZE           NUMBER
FREE_COUNT              NUMBER
MAX_FREE_SIZE           NUMBER
USED_SPACE              NUMBER
AVG_USED_SIZE           NUMBER
USED_COUNT              NUMBER
MAX_USED_SIZE           NUMBER
REQUESTS                NUMBER
REQUEST_MISSES          NUMBER
LAST_MISS_SIZE          NUMBER
MAX_MISS_SIZE           NUMBER
```

The following columns contain values that are valid even if the parameter is not set:

```
REQUEST_FAILURES             NUMBER
LAST_FAILURE_SIZE            NUMBER
ABORTED_REQUEST_THRESHOLD    NUMBER
ABORTED_REQUESTS             NUMBER
LAST_ABORTED_SIZE            NUMBER
```

- `FREE_SPACE`: Total free space in the reserved pool
- `AVG_FREE_SIZE`: Average size of the free memory in the reserved pool
- `MAX_FREE_SIZE`: Size of the largest free piece of memory in the reserved pool
- `REQUEST_MISSES`: The number of times the reserved pool did not have a free piece of memory to satisfy the request and flushed objects from the LRU pool
- `REQUEST_FAILURES`: The number of times that no memory was found to satisfy a request
- `LAST_FAILURE_SIZE`: The size of the last failed request
- `ABORTED_REQUEST_THRESHOLD`: Minimum size of a request that signals an `ORA-04031` error without flushing objects
- `ABORTED_REQUESTS`: Number of requests that signaled an `ORA-04031` error without flushing objects
- `LAST_ABORTED_SIZE`: Last size of the request that returned an `ORA-04031` error without flushing objects from the LRU list

Tuning the Shared Pool Reserved Space

IF REQUEST_FAILURES			Action
>0 and increasing	AND	Request_misses > 0	Increase SHARED_POOL_ RESERVED_SIZE
>0 and increasing	AND	Free_memory => 50% of SHARED_POOL_RESERVED_SIZE	Increase SHARED_POOL_ SIZE
=0	OR	Free_memory => 50% of SHARED_POOL_RESERVED_SIZE	Decrease SHARED_POOL_ RESERVED_SIZE

Tuning the Shared Pool Reserved Space

Diagnostics with the `V$SHARED_POOL_RESERVED` View

Statistics from the `V$SHARED_POOL_RESERVED` view can help you set the `SHARED_POOL_RESERVED_SIZE` parameter. On a system with ample free memory to increase the SGA, the goal is to have `REQUEST_MISSES` equal `0` to prevent request failures or at least to prevent this value from increasing.

`SHARED_POOL_RESERVED_SIZE` Too Small

When `REQUEST_FAILURES` is greater than `0` and increasing, the reserved pool is too small. `REQUEST_MISSES` indicates the number of allocation requests that could not be satisfied. To resolve this, you can increase `SHARED_POOL_RESERVED_SIZE` and `SHARED_POOL_SIZE` values accordingly. Increase the size of both parameters by the same amount because the reserved area comes out of the shared pool. The settings you select for these parameters depend on the SGA size constraints of your system. This option increases the amount of memory available on the reserved area without having an effect on users who do not allocate memory from the reserved pool.

Tuning the Shared Pool Reserved Space (continued)

SHARED_POOL_SIZE Is Too Small

The V$SHARED_POOL_RESERVED view can also indicate when the value for SHARED_POOL_SIZE is too small. This may be the case if REQUEST_FAILURES > 0 is increasing and FREE_MEMORY is a large portion of SHARED_POOL_RESERVED_SIZE and is not decreasing. This indicates that the requests are below the 4 KB threshold. If you have enabled the reserved pool, decrease the value for SHARED_POOL_RESERVED_SIZE. If you have not enabled the reserved pool, you can increase SHARED_POOL_SIZE.

SHARED_POOL_RESERVED_SIZE Is Too Large

Too much memory may have been allocated to the reserved pool if:

- REQUEST_MISS = 0 or is not increasing. The goal is to have this value remain constant using the smallest amount of memory for the Shared Pool Reserved Area.
- FREE_MEMORY = > 50% of the SHARED_POOL_RESERVED_SIZE

If either of these is true, decrease the value for SHARED_POOL_RESERVED_SIZE.

Diagnostics with the ABORTED_REQUEST_THRESHOLD Procedure

By default, the heap manager iterates through the LRU list five times, trying to free and coalesce a contiguous chunk of memory to satisfy the request. The ABORTED_REQUEST_THRESHOLD procedure in the DBMS_SHARED_POOL package sets the minimum size of a request that reports an ORA-4031 error without trying to flush objects and free memory. This procedure limits the extent of a flush that can occur due to a large allocation.

Note: With Automatic Shared Memory Management, an attempt to allocate another granule of memory is made before reporting an ORA-4031 error.

Large Allocations Reported to Alert Log

With Oracle Database 10*g* Release 2, an attempted very large (> 2 MB) allocation of shared pool memory is reported in the alert log:

```
Wed Sep 7 15:56:56 2005
Memory Notification: Library Cache Object loaded into SGA
Heap size 3072K exceeds notification threshold (2048K)
Details in trace file /oracle/rdbms/log/main_ora_27395.trc
KGL object name :select xxx from yyy where …
```

Keeping Large Objects

- **Find those PL/SQL objects that are not kept in the library cache:**

```
SQL> SELECT * FROM V$DB_OBJECT_CACHE
  2  WHERE sharable_mem > 10000
  3  AND (type='PACKAGE' OR type='PACKAGE BODY' OR
  4       type='FUNCTION' OR type='PROCEDURE')
  5  AND  kept='NO';
```

- **Pin large packages in the library cache:**

```
SQL> EXECUTE DBMS_SHARED_POOL.KEEP('package_name');
```


Keeping Large Objects

The loading of large objects is the primary source of shared pool memory fragmentation. User response time increases when a large number of small objects need to be aged out of the shared pool to make room. To prevent these situations, keep these large or frequently required objects in the shared pool to make sure that they are never aged out of the shared pool.

- Which objects to keep:
  - Frequently required large procedural objects such as `STANDARD` and `DIUTIL` packages, and those for which sharable memory exceeds a defined threshold
  - Compiled triggers that are executed often on frequently used tables
  - Sequences, because cached sequence numbers are lost when the sequence is aged out of the shared pool
- When to keep them: Startup time is best, because that prevents further fragmentation.
- Objects marked as `KEPT` are not removed when flushing the shared pool by using the `ALTER SYSTEM FLUSH SHARED_POOL` command.

Keeping Large Objects (continued)

How to Keep Objects

Use the supplied DBMS_SHARED_POOL package and the KEEP procedure to keep objects.

To create the package, run the dbmspool.sql script. The prvtpool.plb script is automatically executed at the end of the dbmspool.sql script. These scripts are not run by catproc.sql.

Use the UNKEEP procedure to unpin objects from the shared pool.

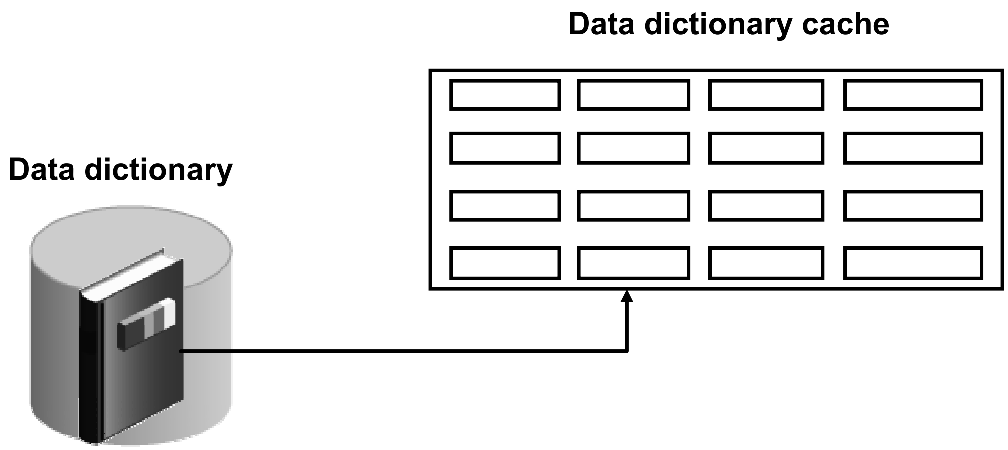
Considerations

Large objects that are constantly being used do not age out of the cache. The objects that are used frequently but are unused long enough age out. These objects may occasionally raise an ORA-4031 error when loading because of fragmentation, or users may complain about the command that calls these objects as being slower. By executing the query shown in the slide periodically, you can choose candidate objects. These are objects that always appear in the cache. Keeping an object that never ages out does not affect the performance.

Keeping objects in the shared pool prevents fragmentation, but it also reduces the available memory for re-creatable objects. After keeping objects in the shared pool, check the reloads and hard parses. The SHARED_POOL_SIZE may need to be increased.

Data Dictionary Cache

The data dictionary cache holds row images of data dictionary rows.



Data Dictionary Cache

The data dictionary cache or row cache holds copies of some of the rows of the data dictionary tables. The row cache is divided into sections, such as segment, extent, histogram, and sequence, depending on the object. Statistics are gathered on the usage of each section and reported in `V$ROWCACHE`.

Three key columns of `V$ROWCACHE` are:

- `PARAMETER`: Gives the name of the data dictionary cache that is being reported
- `GETS`: Shows the total number of requests for information about the corresponding item
- `GETMISSES`: Shows the number of data requests resulting in cache misses

Misses on the row cache are not as expensive as misses in the library cache. A row cache miss results only in a row fetch from the data dictionary. Misses on the data dictionary cache are to be expected in some cases. Upon instance startup, the data dictionary cache contains no data, so any SQL statement issued will result in a cache miss. As more data is read into the cache, the number of cache misses should decrease. Eventually, the database should reach a “steady state” in which the most frequently used dictionary data is in the cache. At this point, very few cache misses should occur. To tune the cache, examine its activity only after your application has been running for some time.

Dictionary Cache Misses

The Statspack and AWR reports output the following:

```
                      Get    Pct       Final
Cache            Requests   Miss       Usage
--------------- -------- ------     -------
dc_histogram_data   5,587   26.8         411
dc_histogram_defs  92,501    3.7         899
dc_object_grants      240   55.0          26
dc_object_ids     959,293    0.1         289
dc_objects         19,385    5.7         314
```

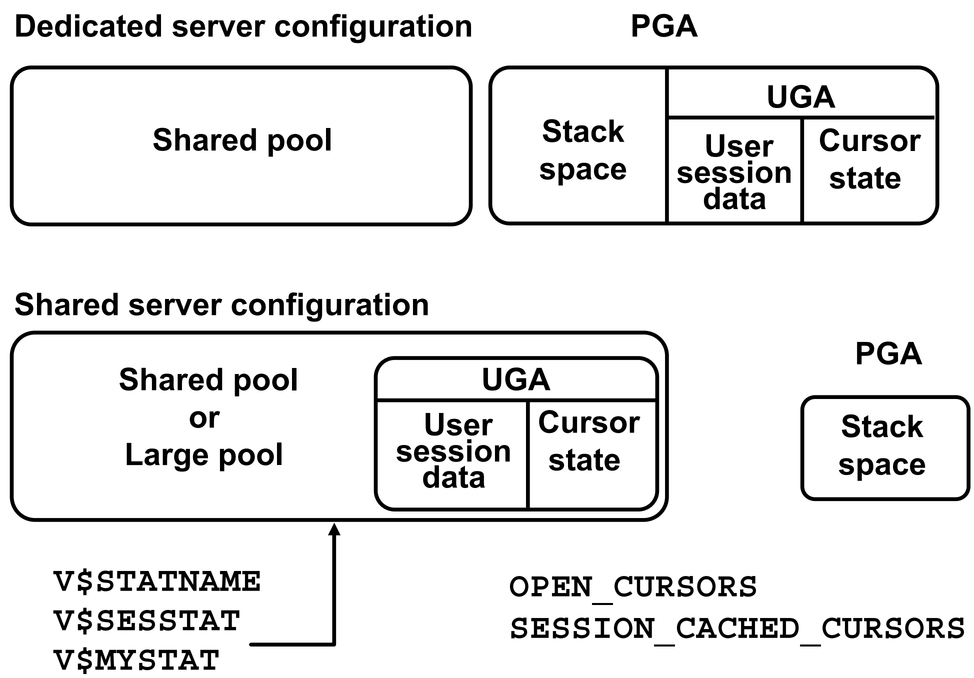
If there are too many cache misses, increase the `SHARED_POOL_SIZE` parameter.

Dictionary Cache Misses

In most cases, the dictionary cache does not need tuning if the library cache is properly tuned. `Pct Miss` should be less than 2% of `Get Requests` if there are a large number of requests. For example, in the slide, `dc_object_grants` has a `Pct Miss` of 55%, but only 240 requests. This is not a concern. `dc_object_ids` is a typical row with a large number of requests and a small `Pct Miss`. If the Statspack report output indicates a high `Pct Miss` and a high `Get Requests` for a number of items, then the `SHARED_POOL_SIZE` should be increased.

Resizing the shared pool is not always necessary. Large OLTP systems, where users log in to the database using their own user ID, can benefit from explicitly qualifying the segment owner, rather than using public synonyms. This significantly reduces the number of entries in the dictionary cache. Also, an alternative to qualifying table names is to connect to the database through a single user ID, rather than individual user IDs. User-level validation can take place locally on the middle tier. Reducing the number of distinct user IDs also reduces the load on the dictionary cache.

UGA and Oracle Shared Server



User Global Area

In a dedicated server configuration, the user global area (UGA) does not use any memory within the SGA. If you use Oracle Shared Server, then the UGA (which includes the user session and cursor state information) is stored in the SGA instead of in a private user memory. Sort areas and private SQL areas are included in the session information. This is because shared servers work on a per-call basis, so any server may need access to any user's information.

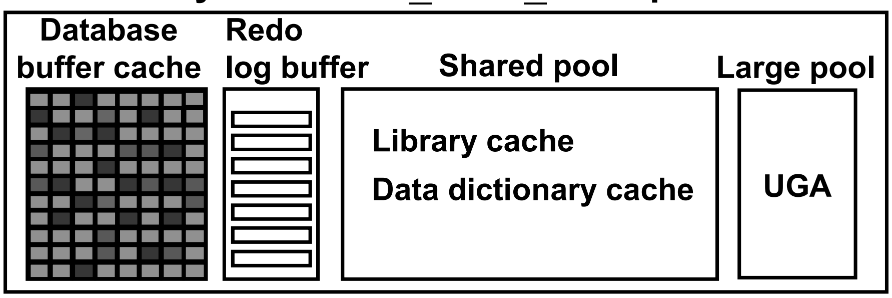
The memory requirement for the UGA with Oracle Shared Server is no larger than if you use dedicated servers, but it is allocated from the SGA instead of process memory. You may need to increase the SGA size, but your private user memory is lower.

In a shared server environment, it is recommended to configure the large pool. If a large pool has not been configured, then the UGA is stored in the shared pool. Increase the `SHARED_POOL_SIZE` parameter to compensate. Allocate approximately 250 KB for each concurrent session as a starting point.

If you set the large pool, then the UGA for shared servers is allocated from the large pool. By allocating session memory from the large pool, the shared pool is used primarily for caching shared SQL and avoids additional contention, with the dependent performance reduction, due to the reduced space used by the UGA.

Large Pool

- **Can be configured as a separate memory area in the SGA, used for memory with:**
  - **I/O server processes: `DBWR_IO_SLAVES`**
  - **Backup and restore operations**
  - **Session memory for the shared servers**
  - **Parallel query messaging**
- **Is used to avoid performance overhead caused by shrinking the shared SQL cache**
- **Is sized by the `LARGE_POOL_SIZE` parameter**



Large Pool

Oracle recommends that you configure the large pool anytime you configure shared servers, parallel query server, `DBWR_IO_SLAVES`, or ASM. In Oracle Database 10*g* Release 2, the large pool participates in the Automatic Shared Memory Management (ASMM). ASMM is the recommended method for managing the memory requirements of the main portions of the SGA.

Existence of the Large Pool

The memory of the large pool is part of the SGA, and adds to the amount of shared memory the Oracle server needs for an instance at startup.

Advantages of the Large Pool

The large pool is used to provide large allocations of session memory for:

- I/O server processes
- Backup and restore operations

The memory for backup and restore operations, and for I/O server processes is allocated in buffers of a few hundred kilobytes. The large pool is better able to satisfy such requests than the shared pool.

Tuning the Large Pool

- **The large pool has one parameter, `LARGE_POOL_SIZE`.**
- **`V$SGASTAT` shows used and free memory.**

```
SELECT * FROM V$SGASTAT
WHERE pool = 'large pool';

POOL         NAME                             BYTES
------------ -------------------------- ----------
large pool   free memory                   2814112
large pool   session heap                  1380192
```

ORACLE

Tuning the Large Pool

The large pool is not governed by a least recently used algorithm. Memory allocations are released by the processes that allocated them or by PMON as part of a cleanup operation.

The only tuning that is available is to monitor the free memory and adjust the LARGE_POOL_SIZE parameter as needed.

The large pool area shown as `session heap` is the User Global Area used by Oracle Shared Servers.

Summary

In this lesson, you should have learned how to:

- Diagnose shared pool problems
- Size the shared pool
- Size the reserved area
- Keep objects in the shared pool

Practice 8 Overview: Tuning the Shared Pool

This practice covers the following topics:

- Tuning a hard parse workload
- Sizing the shared pool
- Keeping objects in the shared pool

ORACLE

9

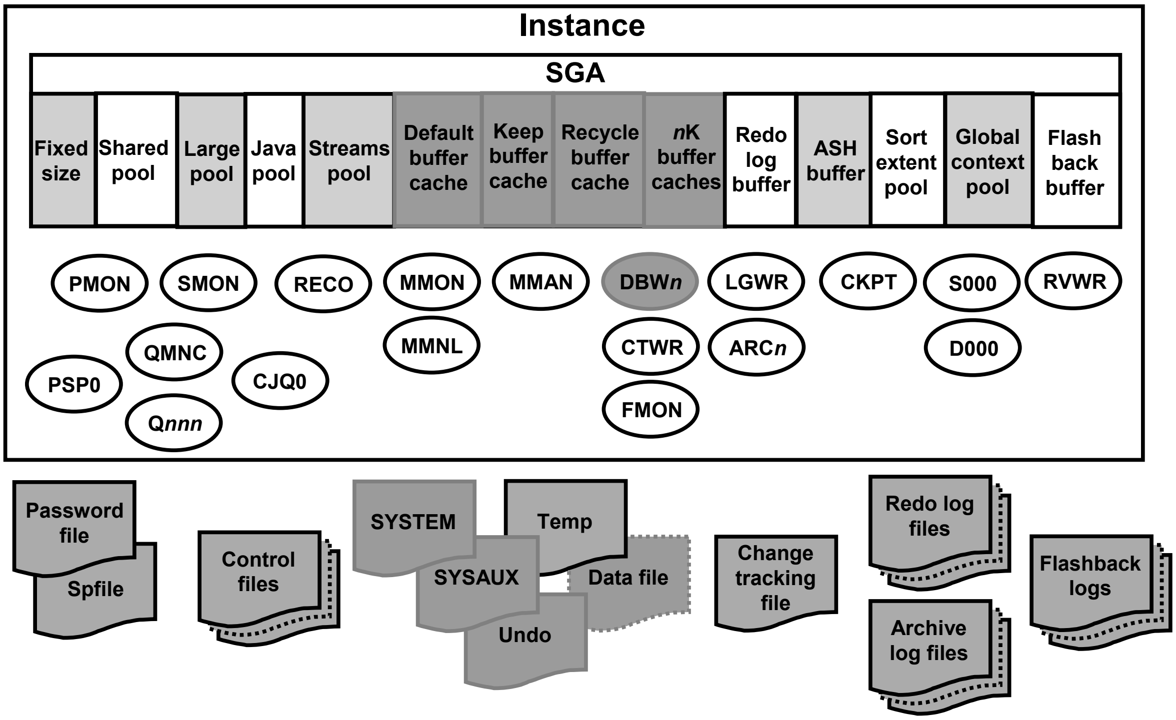
Tuning the Buffer Cache

Objectives

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

- Describe the buffer cache architecture
- Size the buffer cache
- Resolve common performance issues related to the buffer cache
- Use common diagnostic indicators to suggest a possible solution

Oracle Database Architecture



Buffer Cache: Highlights

- Scalable architecture:
  - Multiversion concurrency control
  - Proprietary LRU-based replacement policy
  - Cache fusion
- Incremental checkpointing mechanism
- Advisors
- Private pool for I/O intensive operations

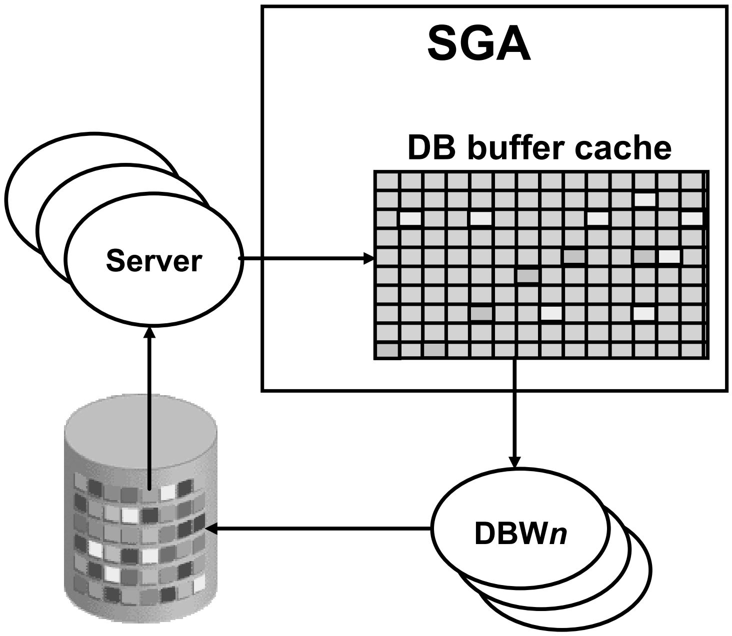
ORACLE

Buffer Cache: Highlights

The buffer cache is an extremely important component for the performance, scalability, and reliability of the database. Some of the highlights of the buffer cache architecture include the following:

- The least recently used (LRU)–like replacement algorithm provides excellent hit rates on a wide range of workload with little overhead.
- A multiversion concurrency control algorithm makes it possible for readers and writers to access different versions in time of the same block.
- Cache fusion is a distributed cache coherency protocol that allows the Oracle database to transparently run on a cluster of machines.
- Incremental checkpointing is a mechanism that tries to maintain a smooth I/O rate for achieving predictable recovery time.
- Where appropriate, the buffer cache advisors assist in configuration and eliminate trial-and-error in performance tuning.
- Direct-path I/O operations use private buffer pools and asynchronous prefetch mechanisms to achieve high bandwidths for decision support systems (DSS) and scan-oriented workloads.

Database Buffers



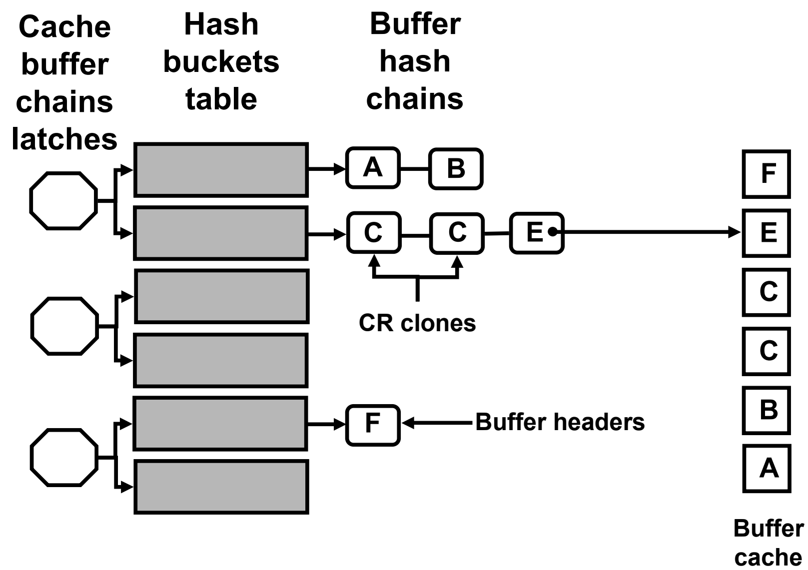
Database Buffers

The database buffer cache is a set of memory buffers in the System Global Area (SGA). Each buffer is sized to hold one database block. The default size of a database block is set at database creation to the size specified in the `DB_BLOCK_SIZE` parameter. Server processes are responsible for moving blocks into the buffers and the DBW*n* processes are responsible for writing blocks that have changed out to the data files.

Every buffer can be in one of three states: free, dirty, or pinned. A free buffer is either empty or identical to the block on disk. A free buffer is available to be overwritten with a different database block at any time. A dirty buffer holds a block that has changed; a server process has made some modification to this block. Only one process at a time is allowed to access a buffer, either to read or write. While the buffer is being accessed, it is pinned. The pin is a very short operation and is not related to a lock. When another process tries to access a buffer that is pinned, a `buffer busy wait` is recorded.

There may be several versions of a block in different buffers of the cache at the same time. When a block is changed by a DML statement, the change is immediately written to the block in memory. A `SELECT` statement requires a view of that block consistent with all committed changes. A Consistent Read (CR) clone block is created by copying the block to another buffer and applying undo data, to make it consistent with the time the `SELECT` statement started.

Buffer Hash Table for Lookups



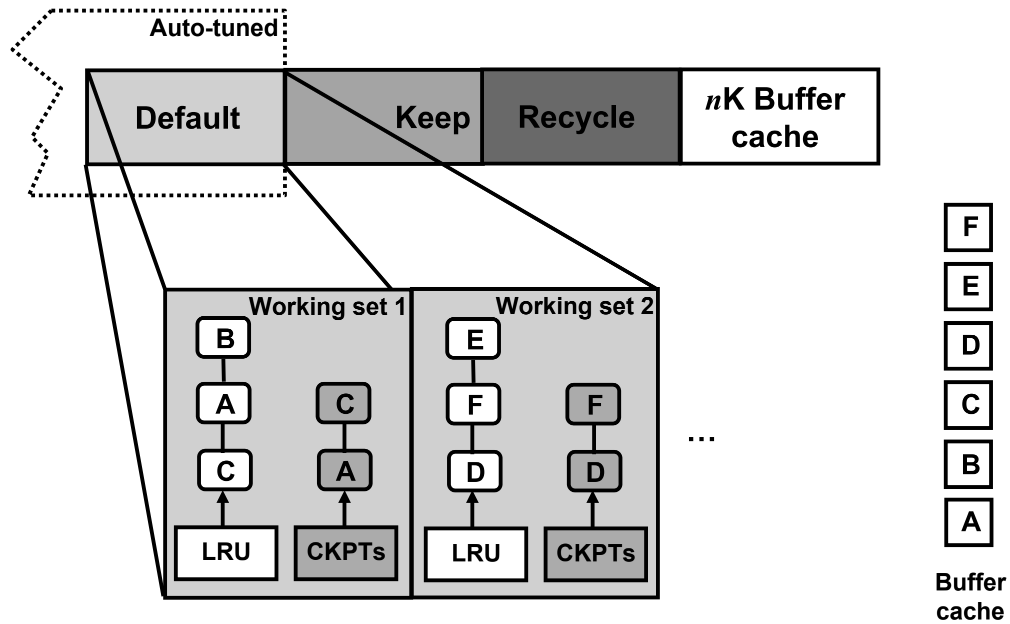
Buffer Hash Table for Lookups

The buffer hash table facilitates buffer lookup. When a server process needs to access a data block in the cache, it requests the block by the data block address (DBA), which is the physical address of a database block. The DBA is hashed to produce an index into a table of hash buckets. For scalability, multiple hash latches protect access to this hash table. These latches are referred to as `cache buffer chains latch` in `V$LATCH`. The hash index is hashed to produce the index of the corresponding latch. As depicted in the slide, each latch protects several buckets, and each hash bucket contains a linked list of all the buffers whose DBAs produce the same hash value. Consistent Read copies (CR clones) of the same block all have the same DBA, thus they are all in the same hash chain. The lookup algorithm returns either the current buffer for the requested DBA or an existing CR clone, or a new CR clone created through the Consistent Read mechanism.

Readers acquire the hash latch in shared mode so all readers can concurrently scan for buffers. Only when a process needs to modify the hash chain does it need to acquire the latch in exclusive mode.

Note: The hashing algorithm provides a very good distribution of blocks across buckets with a very low probability of systematic collisions. Although it is possible to manually change the number of buckets and hash latches, it is not recommended to do so.

Working Sets



Working Sets

Working sets are subsets of buffers in the buffer cache that are created to improve scalability of common buffer cache operations. Each working set contains a least recently used (LRU) list and two checkpoint queues as illustrated in the slide. Each buffer in the cache is assigned to a working set at instance startup. The number of working sets depends on the number of CPUs on the system as well as the number of configured database writer processes (DBW*n*). Concurrent operations by processes on different CPUs can proceed without conflicting with one another by using different working sets.

A *cache miss* occurs when a requested block is not found in the buffer cache. When this happens, an available buffer must be found by searching the LRU list. The LRU list is a linked list of buffer headers arranged by time since last access and frequency of access. The frequency of access is measured by the number of times a buffer has been accessed. This is called touch count. A buffer that is frequently accessed and has been recently accessed is called a hot buffer or block. A buffer that has not been accessed recently is a cold buffer or block. The LRU list is arranged so that the cold buffers are at one end and hot buffers are at the other. Accesses to the LRU lists are protected by the LRU latch known as the `cache buffer lru chains` latch in `V$LATCH`. There is one latch per working set. On a cache miss, a user process picks a random working set and replaces the coldest buffer in that set with the copy read from disk.

Working Sets (continued)

The replacement policy is not strictly LRU based because this would require a manipulation of the list each time a buffer is accessed. In fact, a touch count mechanism is used for every buffer in the list, and a high touch count buffer is moved from the cold part to the hot part only during a replacement operation. This approximate LRU scheme is considerably faster while producing comparable cache hit rates.

To allow buffer replacements to occur, it is necessary to write cold dirty buffers that have aged out to the cold part of the list. Such writes are referred to as aging writes and are performed by DBW*n* processes. The DBW*n* processes work to write enough cold buffers to guarantee a uniform supply of free buffers for replacement purposes. However, because a hot buffer located in the hot part can have the first change in the redo log, aging writes might not be able to advance the thread checkpoint used for crash recovery purposes. That is why another list called the checkpoint queue, which orders buffers in lowest redo block address (RBA) order, is used in addition to the LRU list. Each working set contains two checkpoint queues each protected by a separate checkpoint queue latch of type checkpoint queue. This allows user processes to add buffers to one list while DBW*n* is writing buffers from the other list.

Note: Although it is possible to manually configure the number of working sets, it is not recommended to do so.

Tuning Goals and Techniques

- **It is recommended to use Automatic Shared Memory Management (ASMM).**
- **Tuning goals:**
  - **Servers find data in memory**
  - **No waits on the buffer cache**
- **Diagnostic measures:**
  - **Wait events**
  - **The `V$DB_CACHE_ADVICE` view and cache hit ratio**
- **Tuning techniques:**
  - **Review ADDM recommendations if ASMM is used.**
  - **Reduce the number of blocks required by SQL statements.**
  - **Increase buffer cache size, use multiple buffer pools.**

Tuning Goals and Techniques

Oracle recommends automatic memory configuration for your system using the `SGA_TARGET` and `PGA_AGGREGATE_TARGET` initialization parameters. However, you can manually adjust the memory pools on your system. The manual tuning process is discussed in the rest of this lesson.

Tuning Goals

Because physical I/O takes significant time and CPU, Oracle server performance is improved when the servers find the blocks that they need in memory. The cache hit ratio measures the performance of the database buffer cache. The cache hit ratio is the ratio of the number of blocks found in memory to the number of blocks accessed. A low cache hit ratio indicates that the system may be slow because it is performing unnecessary I/Os.

Diagnostic Measures

To effectively monitor the usage of the buffer cache, you can use the following mechanisms:

- Check the wait events in `V$SYSTEM_EVENT`, `V$SESSIONS_EVENT`, and `V$SESSION_WAIT`.
- Use the `V$DB_CACHE_ADVICE` view.
- Measure the cache hit ratio: Use the `V$SYSSTAT` view, or the Statspack or AWR utilities.

Tuning Goals and Techniques (continued)

Tuning Techniques

If you are using Automatic Shared Memory Management, you should review the recommendations from Automatic Database Diagnostic Monitor for modifying the total size of the shared memory.

If manual SGA tuning is in use, you monitor the buffer cache by monitoring the wait events and calculating the cache hit ratio from statistics collected by the Oracle server.

The first step in tuning the buffer caches is to minimize the wait events recorded in `V$SYSTEM_EVENT` and `V$SESSION_WAIT`. After the wait events have been minimized, examine the utilization of the buffer cache. To improve the cache utilization, you can:

- Ensure that correctly tuned SQL statements are executed, thus minimizing the number of blocks that have to be accessed
- Examine the need for indexes to ensure that blocks are not needlessly read into the buffer cache. Adding indexes could reduce full tables scans.
- Examine the `V$DB_CACHE_ADVICE` view to determine whether the sizes of the buffer caches should be modified
- Use multiple buffer pools to separate blocks by access characteristics
- Configure the tables to be cached in memory

Increasing the size of the data buffer cache does not always improve performance. The characteristics of the application may prevent further improvement of the cache hit ratio. For example, in large data warehouses or decision support systems, which routinely use many scans of large tables, most of the data is read from disk. For such systems, tuning the buffer cache is less important and tuning I/O is vital.

Technical Note

You need to consider the impact of operating system caching. For example, the Oracle server may show a high rate of physical I/O that does not appear at the operating system level. This could mean that database blocks, aged out of the buffer cache, are kept in the operating system cache and can be accessed very quickly. However, as a general rule it is best to bypass the operating system cache because of the following reasons:

- More memory may be required to maintain duplicate blocks in memory (one block in the operating system cache and one in the database buffer cache).
- There is the CPU overhead of copying blocks from the operating system cache to the database buffer cache.

OS caching can be bypassed by using Automatic Storage Management (ASM) or raw devices.

Symptoms

The symptoms that indicate a buffer cache problem:

- **Latch:cache buffer chains**
- **Latch:cache buffer LRU chains**
- **Buffer busy waits**
- **Read waits**
- **Free buffer waits**
- **Cache hit ratio**

Symptoms

There are several symptoms that point to the buffer cache as a problem. If the following are observed in the Top 5 Timed Events, then investigate further:

- Latch:cache buffer chains: This generally points to hot blocks in the cache.
- Latch:cache buffer LRU chains: Contention indicates that the buffer cache is too small, and block replacement is excessive.
- Buffer busy waits: This is almost always an application tuning issue. Multiple processes are requesting the same block.
- Read waits: There are a variety of read waits and multiple causes.
- Free buffer waits: When the DBW*n* processes do not make sufficient free buffer to meet the demand, free buffer waits are seen.

A low cache hit ratio, below 80%, is another symptom of a small buffer cache. More investigation is needed. A low cache hit ratio seldom occurs by itself, and should be used as a confirming symptom.

Cache Buffer Chains Latch Contention

Contention for this latch indicates:

- **Multiple processes attempting to access the same "hot" block**
- **Excessive block replacement**

Cache Buffer Chains Latch Contention

Because each process that accesses the block must obtain the latch, multiple processes accessing the same block will produce contention. This block is called a "hot" block. Hot blocks are often caused by application design choices or inefficient SQL that rereads blocks. This usually calls for application tuning. If the hot block is in an index, a reverse key index may be a solution.

Contention for cache buffer chains latch may indicate that the caller (server process) is attempting to modify the cache buffer chain. This happens when a buffer in the chain must be replaced with another block. Contention can occur when many more blocks are required by the application than there is space in the buffer cache. Check the size of the buffer cache.

Finding Hot Segments

- **Characteristics of cache buffer chains latch contention:**
  - **Many accesses to one or more block under the same latch**
  - **Worse with larger block sizes**
- **To find hot segments:**

```
SQL> SELECT * FROM ( SELECT owner, object_name,
  2       object_type, statistic_name, sum(value)
  3     FROM V$SEGMENT_STATISTICS
  4     GROUP BY owner, object_name, object_type,
  5              statistic_name
  6     ORDER BY SUM(value) DESC)
  7  WHERE ROWNUM < 10;
```


Finding Hot Segments

If a particular block is looked up or updated by a large number of processes concurrently, you may see latch contention on the cache buffer chain latches. In Oracle Database 10*g*, waits for this latch type appear in `V$SESSION_EVENT` and `V$SYSTEM_EVENT` as `latch: cache buffer chains`. This contention can be caused by one or more extremely hot buffers, such as application metadata blocks, undo header blocks, or index root or branch blocks.

There is very little you can do at the block level to alleviate this contention. You must identify the segment to which these blocks belong. The `V$SEGMENT_STATISTICS` view provides this information whenever the `STATISTICS_LEVEL` parameter is set to `TYPICAL` or `ALL`. A very efficient, but not as user-friendly, version of this view is `V$SEGSTAT`. The query in the slide shows the top 10 statistic values, by object and statistic name. The Statspack report shows similar information if you use level 7 snapshots. ADDM reports these types of contention automatically.

Most of the time, by tuning your application SQL, you should be able to remove this type of contention.

Buffer Busy Waits

- **Application-level contention for buffers in the buffer cache**
- **Identify buffer busy waits contention:**

```
SELECT class, count
FROM V$WAITSTAT
WHERE count>0
ORDER BY count DESC;
```

```
SELECT object_name, value
FROM V$SEGMENT_STATISTICS
WHERE statistic_name 'buffer busy waits' AND
      value > 20000;
```

ORACLE

Buffer Busy Waits

In Oracle Database 10*g*, buffer busy waits occur when one session needs to use a buffer in an incompatible mode with the current user of the buffer. For example, one session needs to read a buffer that a second session is modifying. This is, therefore, a case of application-level contention for buffers.

You can identify this type of contention by looking at the third parameter (P3) of the wait event, which represents the class of block that is being impacted (DATA, UNDO, …). Another way to find the kind of blocks that are causing buffer busy waits is to query `V$WAITSTAT`. This gives a breakdown of buffer busy waits by block type. You can find the object that is the cause of most of the contention using segment-level statistics published in `V$SEGMENT_STATISTICS`. See the rows corresponding to `STATISTIC_NAME='buffer busy waits'`.

Index maintenance can add to `buffer busy waits` in transactional systems. All indexes on a table have to be maintained for inserts and deletes.

Note: In Oracle Database 10*g*, when one session needs a buffer that is presently being read into the buffer cache by a second session, the first session is waiting on an event called `read by other session`. This is a read-sharing case, and not a contention case.

Calculating the Buffer Cache Hit Ratio

```
SELECT name, value
FROM v$sysstat
WHERE name IN ('db block gets from cache',
               'consistent gets from cache',
               'physical reads cache');
```

```
PHYSICAL READS = 'physical reads cache')

LOGICAL READS = ('consistent gets from cache' +
                 'db block gets from cache')

HIT RATIO = 1 - PHYSICAL READS/LOGICAL READS
```


Calculating the Buffer Cache Hit Ratio

The buffer cache hit ratio calculates how often a requested block has been found in the buffer cache without requiring disk access. This ratio is computed using data selected from the dynamic performance view `V$SYSSTAT`. The buffer cache hit ratio can be used to verify the physical I/O as predicted by `V$DB_CACHE_ADVICE`. The following statistics are used to calculate the hit ratio:

- `consistent gets from cache`: Number of times a Consistent Read is requested for a block from the buffer cache
- `db block gets from cache`: Number of times a `CURRENT` block is requested from the buffer cache
- `physical reads cache`: Total number of data blocks read from disk into buffer cache

The example shown in the slide has been simplified by using values selected directly from the `V$SYSSTAT` table, rather than over an interval. It is best to calculate the delta of these statistics over an interval while your application is running, and then use them to determine the hit ratio.

Buffer Cache Hit Ratio Is Not Everything

- A badly tuned database can still have a hit ratio of 99% or better.
- Hit ratio is only one part in determining tuning performance.
- Hit ratio does not determine whether a database is optimally tuned.
- Use the wait interface to examine what is causing a bottleneck:
  - `V$SESSION_WAIT`
  - `V$SESSION_EVENT`
  - `V$SYSTEM_EVENT`
- Tune SQL statements.

ORACLE

Buffer Cache Hit Ratio Is Not Everything

A common pitfall is to base cache resizing decisions on hit ratios. Many users mistakenly believe the hit ratio to be an absolute indicator of performance. In fact, the only thing that matters is how much time is being spent on performing disk I/Os due to cache misses.

An application could have a good hit ratio but have many more physical reads than necessary. A logical read is less expensive than a physical read, but there is still an associated cost to retrieving the block. Besides the access cost, there is the cost of memory space for extra blocks that are kept in memory but not used to resolve the query.

For example, suppose two applications (A and B) return the same result set. Application A has a cache hit ratio of 99%, whereas Application B has a hit ratio of 60 %. Which application has the best performance? On further investigation you notice that Application A requires 100,000 logical reads and 1,000 physical reads, whereas Application B has 100 logical reads and 40 physical reads. Which application is better tuned now? With the additional information, the answer is B. There is clearly something wrong with Application A, when it does so many logical reads to return a small result set.

Therefore, having a good cache hit ratio is only a part of the tuning sequence. Examine the wait statistics to determine which areas require your attention.

Interpreting Buffer Cache Hit Ratio

- **Hit ratio is affected by data access methods:**
  - **Full table scans**
  - **Repeated scans of the same tables**
  - **Large table with random access**
  - **Data or application design**
- **Investigate increasing the cache size if:**
  - **Hit ratio is low**
  - **Application is tuned to avoid full table scans**

ORACLE

Interpreting Buffer Cache Hit Ratio

To interpret the buffer cache hit ratio, you should consider the following effects of data access methods:

- Database blocks accessed during a long full table scan are put on the tail end of the least recently used (LRU) list and not on the head of the list. Therefore, the blocks are aged out faster than blocks read when performing indexed lookups or short table scans. When interpreting the buffer cache data, remember that poor hit ratios should be expected when valid large full table scans are occurring.
- Repeated scanning of the same large table or index can artificially inflate a poor cache hit ratio. Examine frequently executed SQL statements with a large number of buffer gets to ensure that the execution plan for such SQL statements is optimal.
- In any large database running OLTP applications in any given unit of time, most rows are accessed either one or zero times. On this basis, there is little purpose in keeping the block in memory following its use.
- If possible, avoid requerying the same data by caching frequently accessed data in the client program or middle tier.

Interpreting Buffer Cache Hit Ratio (continued)

A common mistake is to continue increasing the buffer cache size. Such increases have no effect if you are performing full table scans or operations that do not use the buffer cache. Consider increasing the buffer cache when:

- The application is tuned to avoid full table scans
- The hit ratio is low and the previous access methods have been corrected

Note: Short table scans are scans performed on tables under a certain size threshold. A short table is either less than 20 blocks or less than two percent of the buffer cache.

Read Waits

- **List of wait events performing disk reads into the buffer cache:**
  - `db file sequential read`
  - `db file parallel read`
  - `db file scattered read`
- **If wait time for reads is high:**
  - **Tune the SQL statement that issues most disk reads by sorting `V$SQL` by `DISK_READS` and `BUFFER_GETS`.**
  - **Grow buffer cache if needed.**
  - **Reduce writes due to checkpointing.**
  - **Add more disk capacity.**

ORACLE

Read Waits

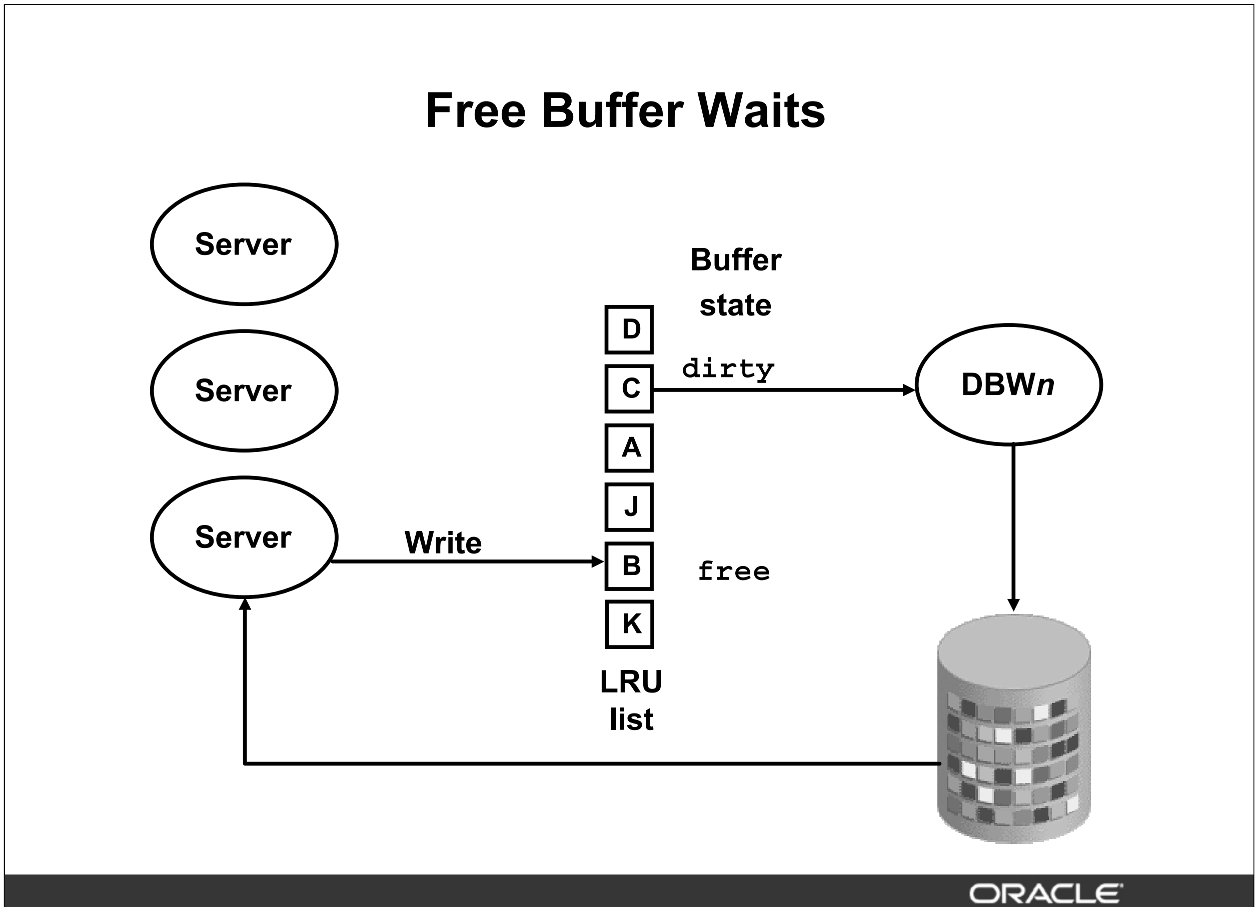
Wait events such as `db file sequential read` (wait for a single block to be read synchronously from disk), `db file parallel read` (wait for a synchronous multiblock read), and `db file scattered read` (wait for multiple blocks to be read concurrently from disk) constitute the set of wait events in `V$SYSTEM_EVENT` for performing disk reads into the buffer cache. These are normal waits and are always present even in well-tuned databases. Excessive waits on read wait events can be determined only by comparing the waits to a baseline when the database was performing acceptable. If the total time for these events appears to be an excessive fraction of the total database time, there are two possible investigative paths:

- When the average time for each read is normal (< 10 ms or the average read time for your disk subsystem):
  - Your workload is simply issuing too many disk reads and it is necessary to try to reduce this number. Find the SQL statements that are issuing the most logical and physical reads and tune them.
  - Additional disk reads may be caused by an undersized buffer cache. Use the buffer cache advisor.

Read Waits (continued)

- When the average time for each read is abnormally high (> 15 ms or the average read time for your disk subsystem):
  - Consider reducing the number of writes due to an aggressive checkpointing policy. Reads are delayed when your system performs writes because the disks are busy servicing writes.
  - Add disk bandwidth by striping the existing disks to spread the I/O load, or add disk devices to reduce disk contention.

Note: ADDM can notify you when average read times are excessive. And you can adjust the expected I/O time by setting the `DBIO_EXPECTED` value as shown in the lesson titled "Using Automatic Workload Repository."



Free Buffer Waits

For every block moved into the buffer cache, a server process must scan the LRU list to find a free buffer that can be used to hold the data block. Ideally, the server process starts from the "cold" end of the LRU list and finds that the first buffer examined is a free buffer. If the first buffer is not free, the next buffer is examined, and so forth, until a free buffer is found or four buffers have been examined. If a free buffer is not found, then the server process requests that DBW*n* write more blocks out to disk, resulting in more free buffers, and the server process goes into a free buffer wait. A large number of free buffers waits indicates that the DBW*n* processes are not creating free buffers fast enough. You can resolve this problem by:

- Reducing the number of buffers required (tuning the application SQL)
- Increasing the total number of buffers (allowing DBW*n* more time to write)
- Increasing the rate that blocks are written to disk

Suppose you have already tuned the application, or you are not allowed to tune the application. Making the buffer cache larger does not always help. The `DB_CACHE_ADVICE` view shows that adding memory to the cache will only give slight benefit. The next step is to speed up the rate at which DBW*n* writes blocks to disk.

Solutions

The buffer cache solutions are applied depending on the symptoms:

- Properly size the buffer cache.
- Cache objects.
- Use the keep and recycle pools.
- Increase the writing speed of DBW*n*.
- Use private I/O pool.

Sizing the Buffer Cache

Buffer cache size affects several tuning diagnostics. If the cache is too small:

- **Extra reads due to block replacement**
- **Extra writes to move dirty blocks to disk**
- **Buffer cache LRU chains contention**

ORACLE

Sizing the Buffer Cache

When the buffer cache is too small, several tuning diagnostics can be affected. There will be more physical reads than would be required for a larger cache. There may be additional writes to move the dirty blocks to disk making room for the other blocks to come into the cache. The buffer cache LRU chains latch will probably show contention because they are being frequently searched for free blocks.

The buffer cache size is controlled by a set of parameters that in the past were set primarily by trial and error. You can use Automatic Shared Memory Management to automatically size the SGA and the buffer cache advisor to assist in choosing an efficient cache size for manual tuning.

Buffer Cache Size Parameters

- **Set the primary block size for the recycle, keep, and default buffer pools: `DB_BLOCK_SIZE`**
- **Set the size of the buffer pools:**
  - `DB_CACHE_SIZE`
  - `DB_KEEP_CACHE_SIZE`
  - `DB_RECYCLE_CACHE_SIZE`
- **Represent all memory for the buffer cache**
- **Are required to use buffer cache features:**
  - **Dynamic grow/shrink**
  - **Buffer cache advice**
  - **Multiple block sizes**

Buffer Cache Size Parameters

The DB_BLOCK_SIZE parameter defines the default database block size. It is set for the database during database creation. This is the block size of the SYSTEM tablespace. Other tablespaces may be of different block sizes.

Three parameters set the size of three buffer pools defined with the default block size. DB_CACHE_SIZE sets the size of the default buffer pool. The DB_KEEP_CACHE_SIZE and DB_RECYCLE_CACHE_SIZE parameters set the size of the keep and recycle buffer pools, respectively. Only the default pool is required to be configured.

These values are specified in units of memory (KB or MB), not in number of blocks. These values represent all of the memory used by the corresponding pools, including the memory used for the buffer cache metadata such as buffer headers. The *_CACHE_SIZE parameters must be used to enable dynamic memory sizing, buffer cache advice, or the use of nondefault block sizes.

Note: The DB_BLOCK_BUFFERS parameter is used only on a 32-bit system with memory in excess of 4 GB to enable the Very Large Memory (VLM) option.
The DB_BLOCK_BUFFERS and DB_CACHE_SIZE parameters cannot be used together. Using DB_BLOCK_BUFFERS disables Automatic Shared Memory Management.

Dynamic Buffer Cache Advisory Parameter

- **The buffer cache advisory feature enables and disables statistics gathering for predicting behavior with different cache sizes.**
- **Use the information provided by these statistics to size the buffer cache optimally for a given workload.**
- **The buffer cache advisory is enabled by means of the `DB_CACHE_ADVICE` initialization parameter:**
  - **This parameter is dynamic and can be changed using `ALTER SYSTEM`.**
  - **Three values are allowed: `OFF`, `ON`, and `READY`.**

Dynamic Buffer Cache Advisory Parameter Values

The recommended tool for sizing the buffer cache is called the Buffer Cache Size Advisor. Set the DB_CACHE_ADVICE parameter to enable the Buffer Cache Size Advisor as follows:

- OFF: The advisor is turned off and the memory for the advisor is not allocated.
- READY: The advisor is turned off, but the memory for the advisor remains allocated. Allocating the memory before the advisor is actually turned on avoids the risk of an ORA-4031 error (inability to allocate from the shared pool). If the parameter is switched to this state from OFF, an ORA-4031 error may be generated.
- ON: The advisor is turned on and the advisor publishes its results through the V$DB_CACHE_ADVICE view. Attempting to set the parameter to this state when it is in the OFF state may lead to an ORA-4031 error. If the parameter is in the READY state, it can be set to ON without error because the memory is already allocated. The advisor mechanism incurs less than 0.1% overhead in terms of CPU and memory utilization and shows very accurate predictions. Before querying V$DB_CACHE_ADVICE, make sure that you are running a representative workload.

Note: If STATISTICS_LEVEL is set to TYPICAL or ALL, then DB_CACHE_ADVICE is automatically set to ON. It is set to OFF if STATISTICS_LEVEL is set to BASIC.

View to Support Buffer Cache Advisory

- Buffer cache advisory information is collected in the `V$DB_CACHE_SIZE` view.
- The view contains different rows that estimate the number of physical reads for cache sizes between 10% and 200% of the current cache size.
- The rows also compute a physical read factor, which is the ratio of the number of estimated reads to the number of actual reads.
- Simulation is done for all buffer pools.

ORACLE

View to Support Buffer Cache Advisory

In the `V$DB_CACHE_ADVICE` view, each cache size simulated has its own row, with the predicted physical I/O activity that would take place for that size. The columns identify the buffer pool, the block size of the pool, and the size of the pool for the estimate. The estimates are:

- `ESTD_PHYSICAL_READ_FACTOR`: Physical read factor for this cache size, which is a ratio of the number of estimated physical reads to the number of reads in the real cache. If there are no physical reads into the real cache, then the value of this column is null.
- `ESTD_PHYSICAL_READS`: Estimated number of physical reads for this cache size
- `ESTD_PHYSICAL_READ_TIME`: Estimated disk read time
- `ESTD_PCT_OF_DB_TIME_FOR_READS`: Estimated disk time as a percentage of the total time

Note: If multiple buffer pools or multiple block size caches are in use, the advisor contains similar entries for all nonstandard buffer caches.

Using the V$DB_CACHE_ADVICE View

```
SELECT size_for_estimate, buffers_for_estimate,
       estd_physical_read_factor, estd_physical_reads
FROM V$DB_CACHE_ADVICE
WHERE name = 'DEFAULT' AND
      advice_status = 'ON' AND
      block_size = (SELECT value FROM V$PARAMETER
                    WHERE name = 'db_block_size');
```

```
                                   Estd Phys    Estd Phys
 Cache Size (MB)      Buffers Read Factor        Reads
---------------- ------------ ----------- ------------
(10%)          30        3,802       18.70  192,317,943
…
              243       30,416        1.33   13,720,149
              273       34,218        1.13   11,583,180
(Current)     304       38,020        1.00   10,282,475
              334       41,822         .93    9,515,878
…
              577       72,238         .67    6,895,122
(200%)        608       76,040         .66    6,739,731
```

ORACLE

Using the V$DB_CACHE_ADVICE View

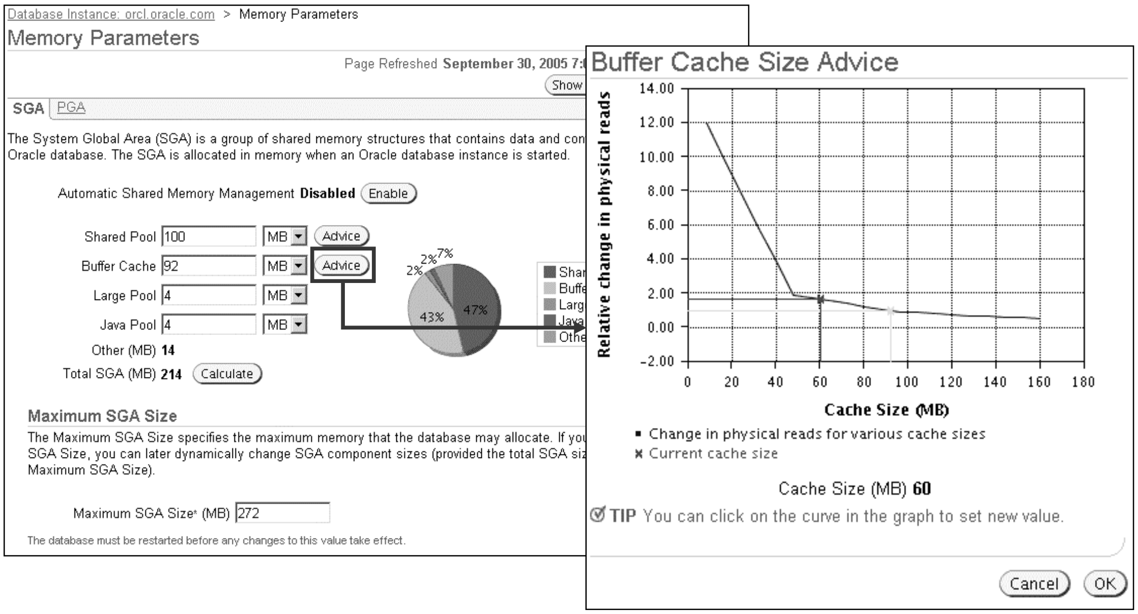
The output in the slide shows that if the cache was 212 MB, rather than the current size of 304 MB, the estimated number of physical reads would increase by a factor of 1.74 or 74%. This means it would not be advisable to decrease the cache size to 212 MB.

However, increasing the cache size to 334 MB would potentially decrease reads by a factor of .93 or 7%. If an additional 30 MB memory is available on the host machine and the SGA_MAX_SIZE setting allows the increment, it would be advisable to increase the default buffer cache pool size to 334 MB.

The process of choosing a cache size is the same, regardless of whether the cache is the default standard block size cache, the KEEP or RECYCLE cache, or a nonstandard block size cache.

Note: With the Oracle database, physical reads do not necessarily indicate disk reads because physical reads may well be satisfied from the file system cache.

Using the Buffer Cache Advisory with EM



Using the Buffer Cache Advisory with EM

Enterprise Manager makes the buffer cache advisory easily available through the Administration > Memory Parameters page. This advisor is available in both Enterprise Edition and Standard Edition. From the Buffer Cache Size Advice graph, it is easy to see that increasing the cache size to 90 MB reduces the physical reads, but beyond that point more memory yields a much smaller benefit. The graph also shows that reducing the cache to about 50 MB will increase the number of physical reads somewhat, but a cache smaller than 50 MB will cause a large increase in physical reads.

Click the graph to set the `DB_CACHE_SIZE` parameter to a new value. The new value is applied immediately if the total size of the SGA is less than the `SGA_MAX_SIZE` parameter.

Caching Tables

- **Enable caching during full table scans by:**
  - **Creating the table with the `CACHE` clause**
  - **Altering the table with the `CACHE` clause**
  - **Using the `CACHE` hint in a query**
- **Caching tables put blocks at the MRU end of the LRU list.**
- **Guideline: Do not overcrowd the buffer cache.**
- **Use a keep pool.**

Caching Tables

When the server retrieves blocks using a full table scan, the buffers go to the least recently used end of the LRU list. These buffers are then used the next time a free buffer is required.

To change this behavior, you must perform one of the following tasks:

- Create a table by using the `CACHE` clause.
- Alter a table by using the `CACHE` clause.
- Code the `CACHE` hint clause into a query.

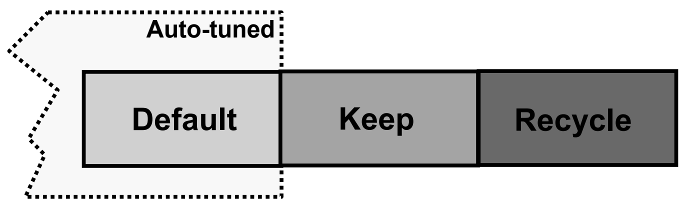
If you use one of these methods, the Oracle server places the table blocks that are higher on the LRU list toward the most recently used (MRU) end. Use the `CACHE` clause when you create small lookup tables used by many users. You may overcrowd the buffer cache if you have too many cached tables. Cached tables should be less than 10% of the buffer cache.

Tables can also be effectively cached by using a keep pool.

Note: The `CACHE` and `NOCACHE` hints or clauses affect system statistics table scans (long tables) and table scans (short tables), as shown in the `V$SYSSTAT` data dictionary view. Short tables as defined internally are automatically cached.

Multiple Buffer Pools

- **Three buffer pools:**
  - **Default: `SYS` and nonflagged table or indexes**
  - **Keep: Hot objects**
  - **Recycle: Infrequent access**
- **Useful for small, simple schemas with well-known access paths**



Multiple Buffer Pools

With the multiple buffer pools feature, you can configure separately managed buffer caches. The benefit of multiple buffer pools is to reduce physical reads without increasing the total SGA. This feature can be very useful for small and simple schemas with well-defined access patterns. Three buffer pools are supported:

- Default: This pool always exists and its size is controlled by the DB_CACHE_SIZE parameter or by Automatic Shared Memory Management. All blocks read by the SYS user use this pool. In addition, if the buffer pool attribute of an object is not set, or is set and the specified pool does not exist, then the object uses this pool. Most objects are cached in this pool.
- Keep: This pool is intended for small or frequently accessed objects that would benefit from being memory-resident. It is sized to the sum of all objects you want to keep plus a small percentage for Consistent Read blocks. Its size is controlled by the DB_KEEP_CACHE_SIZE parameter. The LRU list behavior is the same for each cache. If the keep pool is smaller than the number of blocks to be kept, some blocks will be forced to age out of the cache.

Multiple Buffer Pools (continued)

- Recycle: This pool is used for objects that gain no benefits from caching. These are usually large tables with random access, where the blocks are seldom used beyond a single transaction—for example, an event log table where rows are continuously added, but which is seldom accessed otherwise. Only enough buffers to hold the rows involved in an active transaction are needed, usually a small percentage of the blocks in the object. The recycle pool size is controlled by the `DB_RECYCLE_CACHE_SIZE` parameter.

Note: There is still only one common hash table used for all buffer pools for the purpose of buffer lookups.

Enabling Multiple Buffer Pools

- **Use the `BUFFER_POOL` clause.**
- **This clause is valid for tables, clusters, and indexes.**
- **When altered, buffer pool is used for future reads.**
- **Objects can have more than one buffer pool.**

```
CREATE INDEX cust_idx …
  STORAGE (BUFFER_POOL KEEP …);

ALTER TABLE customer
  STORAGE (BUFFER_POOL RECYCLE);

ALTER INDEX cust_name_idx
  STORAGE (BUFFER_POOL KEEP);
```


The BUFFER_POOL Clause

The BUFFER_POOL clause is used to define the default buffer pool for an object. It is part of the STORAGE clause and is valid for CREATE and ALTER TABLE, CLUSTER, and INDEX statements. The blocks from an object without an explicitly set buffer pool go into the default buffer pool.

The syntax is BUFFER_POOL {KEEP | RECYCLE | DEFAULT}.

When the default buffer pool of an object is changed using the ALTER statement, blocks that are already cached remain in their current buffers until they are flushed out by the normal cache management activity. Blocks read from disk are placed into the newly specified buffer pool for the segment. If the specified pool is not defined, the blocks of the object are placed in the default pool.

Because buffer pools are assigned to a segment, objects with multiple segments can have blocks in multiple buffer pools. For example, an index-organized table can have different pools defined on both the index and the overflow segment.

Note: If a buffer pool is defined for a partitioned table or index, then each partition of the object inherits the buffer pool from the table or index definition, unless you override it with a specific buffer pool.

Calculating the Hit Ratio for Multiple Pools

```
SQL> SELECT name, 1 - (physical_reads /
  2  (db_block_gets + consistent_gets)) "HIT_RATIO"
  3  FROM V$BUFFER_POOL_STATISTICS
  4  WHERE db_block_gets + consistent_gets > 0;

NAME                  HIT_RATIO
------------------   ----------
KEEP                 .983520845
RECYCLE              .503866235
DEFAULT              .790350047
```


Calculating the Hit Ratio for Multiple Pools

The V$BUFFER_POOL_STATISTICS view displays statistics (physical writes, consistent gets, free buffer waits) against the multiple buffer caches (if allocated). When examining the different hit ratios of these pools, you should notice that:

- The keep pool has the highest hit ratio. The hit ratio should be high because you have sized the keep pool larger than the total size of the segments assigned to the pool and, therefore, no buffers (or very few) should need to be aged out. You should confirm that segments assigned to the keep pool are being utilized frequently and warrant the memory space used. You can query the V$BH view to confirm which blocks are in the keep pool. Query V$SEGMENT_STATISTICS for the segments that are assigned to the keep pool to verify that the number of logical reads is much higher than the number of physical reads.

Calculating the Hit Ratio for Multiple Pools (continued)

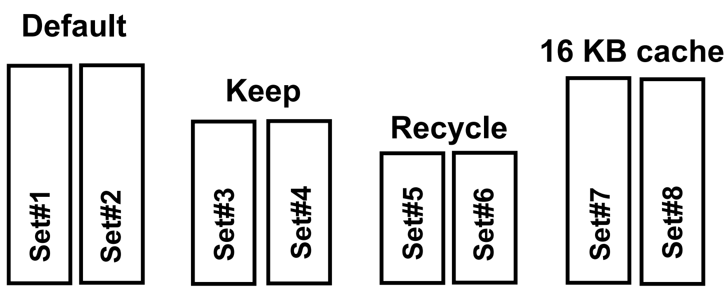
- The recycle pool has the lowest hit ratio. The hit ratio should be low because you have sized the recycle pool much smaller than the total size of the segments assigned to the pool; therefore, buffers have to be aged out due to blocks being read into the pool. You should confirm that segments assigned to the recycle pool are being kept in the pool long enough for the statement to complete processing. Use the DB Buffer Cache Advisor if the recycle pool is not keeping blocks long enough for the statements to finish. If the blocks are aging out too rapidly, the DB Buffer Cache Advisor will show that the number of physical reads are reduced with a larger recycle pool.

The values of the hit ratios are relative. The purpose of multiple buffer pools is not to increase the hit ratios, but to increase the performance by reducing the total number of physical I/Os.

Note: The data in `V$SYSSTAT` reflects aggregation of the logical and physical reads for all buffer pools in one set of statistics.

Multiple Block Sizes

- Allow buffer caches for nonstandard block sizes
- Parameters: `DB_`*n*`K_CACHE_SIZE` {*n* = 2, 4, 8, 16, 32}
- `BLOCKSIZE` attribute of `CREATE TABLESPACE` storage clause
- Intended for transportable tablespaces



Multiple Block Sizes

The standard block size of the database is specified by the DB_BLOCK_SIZE parameter. This is the block size of the SYSTEM tablespace and the default block size for any additional tablespaces. It is also possible to create nonstandard block size tablespaces. The supported set of block sizes are 2 KB, 4 KB, 8 KB, 16 KB, and 32 KB. To accommodate buffers from tablespaces with nonstandard block sizes, additional buffer caches need to be created for those block sizes. The set of five parameters DB_*n*K_CACHE_SIZE is used to specify the size of these buffer caches.

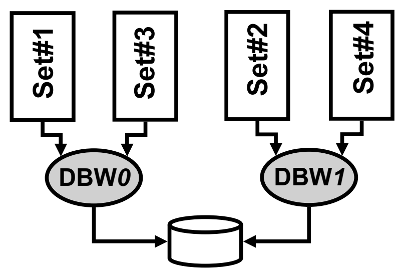
Similar to multiple buffer pools, caches for multiple block sizes are represented internally as ranges of working sets. However, multiple buffer pools are not supported for nonstandard block sizes. Again, buffers of nonstandard block sizes use the common hash table along with the buffers of the standard block size.

The primary intention of the multiple block size feature is to be able to support transportable tablespaces across databases with different block sizes. It is not intended as a performance feature.

Note: The V$BUFFER_POOL view shows the detailed configuration of the buffer cache and its subdivision into buffer pools and multiple block size caches. Statistics corresponding to each cache can be found in V$BUFFER_POOL_STATISTICS.

Multiple Database Writers

- **Multiple database writers are a means to increase write throughput useful for large SMP systems.**
- **Buffer cache is partitioned between database writers by working sets.**
- **Each DBW*n* process scans its own assigned sets.**
- **The number of database writes can be manually controlled by `DB_WRITER_PROCESSES`.**



ORACLE

Multiple Database Writers

For concurrency and throughput, dirty buffers in the cache can be written to disk by more than one database writer process (DBW*n*).

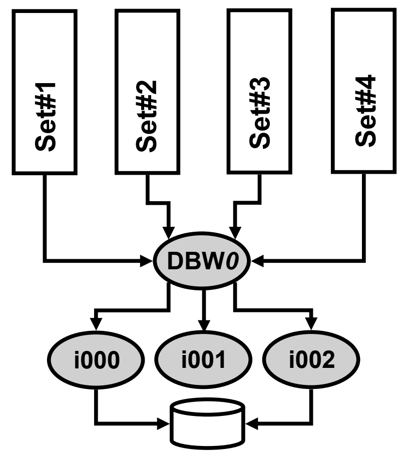
The purpose of multiple database writers is to parallelize the CPU cycles of writing buffers to disk between multiple processes when the total CPU for disk writes exceeds the capacity of one single CPU. This is especially true for large (> 8 CPUs) SMP and NUMA systems for which a single database writer may not be enough to keep up with the rate of buffer dirtying by processes running on other processors. For this reason, the Oracle database server automatically configures multiple database writers: one for every eight CPUs on the system.

These processes wake up and scan their assigned LRU and checkpoint queues for dirty buffers for aging and checkpoint operations. They gather a batch of buffers to write and issue the writes asynchronously using the most efficient available OS mechanism. As shown in the slide, working sets are the units of partitioning between database writers.

The number of database writers can be controlled by the `DB_WRITER_PROCESSES` initialization parameter. A maximum of 20 database writers is supported.

Multiple I/O Slaves

- Allow DBW0 to write in parallel when asynchronous I/O is not well supported
- DBW0 gathers batch of buffers
- Queues batch in round-robin order with I/O slaves
- Waits for all slaves to complete
- Cannot combine multiple DBW*n* with multiple I/O slaves
- Controlled by:
  - `DBWR_IO_SLAVES`
  - `DISK_ASYNCH_IO`



Multiple I/O Slaves

Another mechanism for obtaining increased write throughput is to use I/O slaves (`I`*n*). I/O slaves parallelize the writes from a single batch issued by DBW0. Thus, if DBW0 builds a batch of 1,000 buffers to write, with 10 I/O slaves, each of them will write 100 buffers.

I/O slaves are designed to provide additional write throughput when the system does not support asynchronous I/O, or when the initialization parameter `DISK_ASYNCH_IO` has been set to `FALSE`. In such a situation, it would be prohibitively slow for even multiple database writer processes to issue synchronous writes one by one, and I/O slaves should be used.

The parameter controlling the number of I/O slaves is `DBWR_IO_SLAVES`. The maximum supported number of I/O slaves is 999.

Note: Combining multiple database writer processes and I/O slaves is not supported

Use Multiple Writers or I/O Slaves

To reduce free buffer waits:

- **Implement asynchronous I/O first**
- **Use multiple writers when a database writer is consuming 100% of a CPU**
- **Use I/O slaves when asynchronous I/O is not supported**

Use Multiple Writers or I/O Slaves

When you observe free buffer waits, first eliminate checkpointing and I/O bandwidth as causes.

- Implement asynchronous I/O, if your system supports it.
- If one or more DBW*n* processes are consuming 100% of a CPU, increase the number of database writer processes.
- If asynchronous I/O is not supported on your system, implement I/O slaves.

Writing buffers is not directly in the performance critical path because user processes rarely have to wait for a write of a buffer to complete. Your database writer configuration is adequate if you do not have free buffer waits. If there are no free buffer waits, increasing the number of database writers or I/O slaves does not increase the number of writes.

Symptoms of I/O bandwidth saturation are the following:

- The average time waited for the "db file sequential read" wait event is more than 10–15 ms.
- Large values for free buffer waits, but none of the database writers are consuming anywhere close to a full CPU.

Private Pool for I/O Intensive Operations

- **Use for operations that:**
  - **Do not require caching**
  - **Require efficient I/O**
- **Use for large asynchronous I/O**
- **Requires global checkpoint to ensure consistent data**
- **Use only for direct path I/O:**
  - **Parallel direct inserts**
  - **SQL*Loader direct path**

Private Pool for I/O Intensive Operations

The Oracle database provides a mechanism for bypassing the buffer pool for I/O intensive operations such as bulk loads. This is done for performance reasons and primarily to be able to issue large I/Os without flooding the buffer cache. This mechanism is referred to as direct path I/O.

Consistency must be maintained between the private and the shared buffer pools. By bypassing the buffer cache, the system must make sure that the blocks read from disk by the direct path I/O operation are current. Thus, before launching the direct path I/O operation, the system must issue a global checkpoint across all instances to checkpoint all changes to the object up until to the time stamp needed by the direct path I/O operation.

However, not all operations can use the direct path I/O interface. For examples, updates must go through the buffer cache because they have to update the current version of the blocks. One exception to this rule is for bulk insert operations that insert data above high-water marks of an object. The best candidates for using this mechanism are parallel direct full table scans.

Automatically Tuned Multiblock Reads

- `DB_FILE_MULTIBLOCK_READ_COUNT` is automatically tuned
- Simplifies the determination of the best value
- Optimal I/O size is platform dependent:
  - Cannot exceed 10% of the cache
    Prefetch limited to 64 KB
  - Prevents swamping of the cache
- Automatically enabled if not set or set to zero

ORACLE

Automatically Tuned Multiblock Reads

The `DB_FILE_MULTIBLOCK_READ_COUNT` parameter controls the number of blocks prefetched into the buffer cache during scan operations, such as full table scan and index fast full scan.

Because this parameter has a significant impact on the overall performance, Oracle Database 10*g* Release 2 automatically selects an appropriate value for this parameter depending on the operating system's optimal I/O size and the size of the buffer cache.

The optimal I/O size is the size of the prefetch, as long as the total number of prefetched blocks due to full scans does not exceed 10% of the cache. This limit ensures that prefetches do not swamp the buffer cache and age out more useful data. If the number of prefetched blocks exceeds 10% of the cache, then the prefetch size is limited to 64 KB. When `DB_FILE_MULTIBLOCK_READ_COUNT` parameter is not set or is explicitly set to `0`, this is the default behavior. If you explicitly set a value, then that value is used and is consistent with the behavior of previous releases.

Note: Because this automatic setting can influence the optimizer to use more full table scan plans, the value used by the optimizer remains backward compatible and is set to 8 if you do not set it explicitly.

Faster Instance Startup for Ultralarge Buffer Caches

- **Database available at 10% initialization of buffer cache**
- **Remaining 90% formatted in the background by CKPT**
- **Leverages the dynamic buffer cache infrastructure**
- **Especially useful for very large buffer caches**

Faster Instance Startup for Ultralarge Buffer Caches

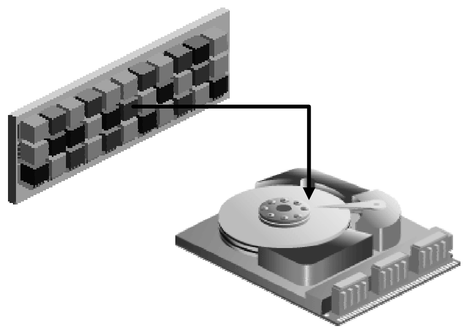
With ultralarge buffer caches (100 GB or greater), the time taken to initialize the buffer caches can be in the order of several minutes. This adds considerably to the time to open the database even when no recovery is required.

In Oracle Database 10g, only a small portion of the buffer cache (10%) is initialized before startup, with the remaining initialization completed after the startup completes. Partial initialization is possible because the entire buffer cache is not required immediately after the database opens; for a very large buffer cache, it takes some time before the workload ramps up to the point of occupying every buffer in the cache.

This functionality leverages the dynamic buffer cache infrastructure. The buffer cache is subsequently formatted in the background by the CKPT process as is done during a normal buffer cache resize operation.

Note: Any automatic cache resizing cannot occur until the buffer pool has been fully initialized.

Flushing the Buffer Cache (for Testing Only)



```
ALTER SYSTEM FLUSH BUFFER_CACHE;
```


Flushing the Buffer Cache

The intent of flushing the database buffer cache is to provide for an identical starting point for comparison of rewritten SQL statements. After you execute this command, the buffer is emptied and the next statement has a 100 percent cache miss. You should execute this command only on a test system; you should also execute it only when everyone using the system is aware of the ramifications.

The benefit of this feature is that it allows you to establish a consistent testing environment. You can flush the buffer cache manually between runs of test queries, thereby making it possible to determine the effects of changes in queries or applications.

Summary

In this lesson, you should have learned how to:

- Describe the buffer cache architecture
- Size the buffer cache
- Resolve common performance issues related to the buffer cache
- Use common diagnostic indicators to suggest a possible solution

Practice 9 Overview: Tuning the Database Buffer Cache

This practice covers the following topics:

- Using the DB buffer cache advisor to size the buffer cache
- Using the keep pool

10

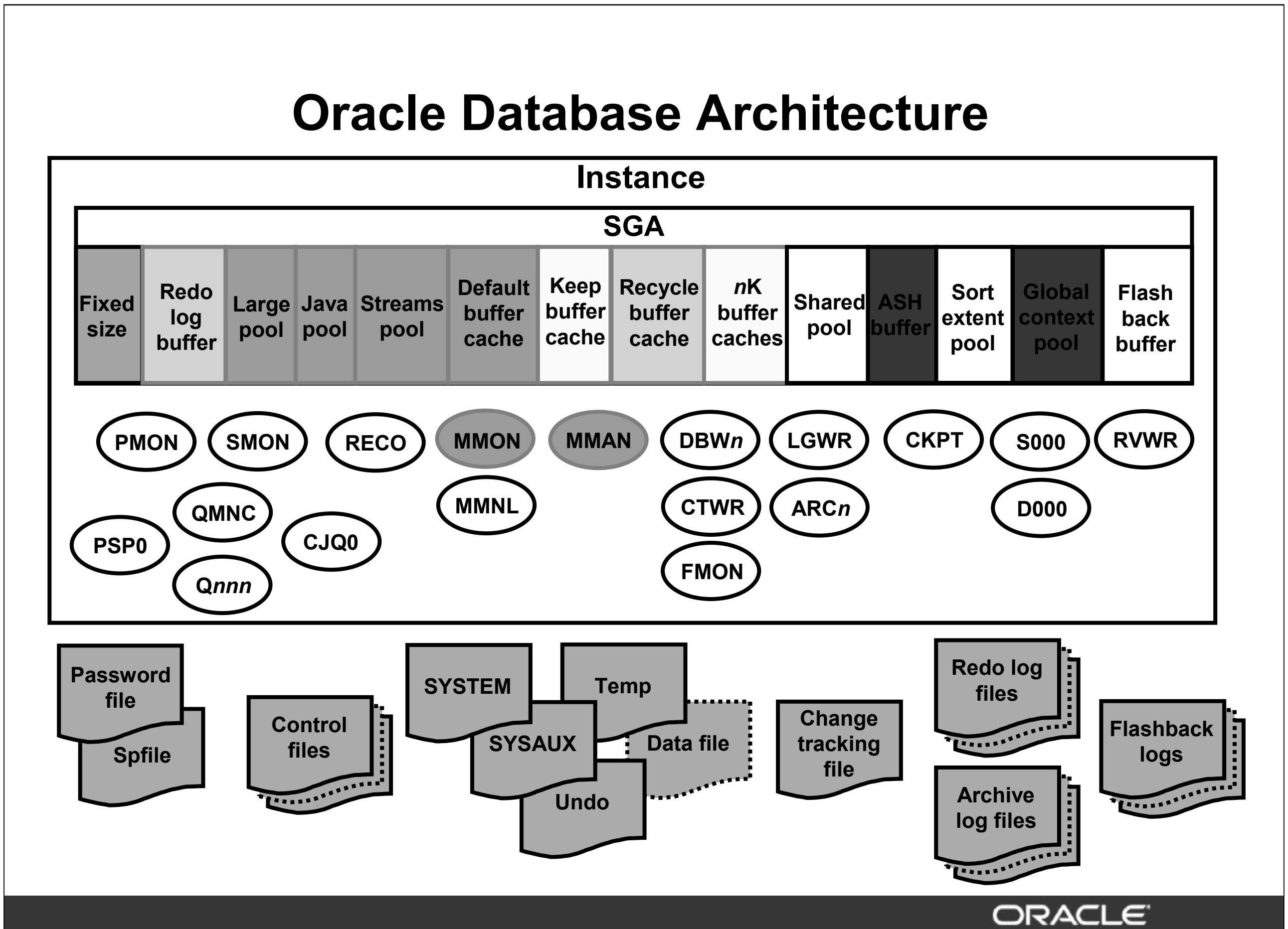
Automatic Shared Memory Management

Objectives

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

- Enable Enterprise Manager (EM) memory parameters
- Set auto-tuned memory parameters
- Override minimum size with the manually tuned SGA parameters
- Use the SGA advisor to set `SGA_TARGET`

ORACLE



Oracle Database Architecture

The slide shows the key SGA components and processes that are involved in Automatic Shared Memory Management (ASMM).

The SGA components shared pool, large pool, Java pool, Streams pool, and the default buffer cache participate in ASMM. The monitoring and dynamic resizing are handled by two background processes: the memory manager (MMAN) and the manageability monitor (MMON).

The ASSM infrastructure is dependent on the statistics collected by the manageability monitor, so the `STATISTICS_LEVEL` parameter must be set to `TYPICAL` or `ALL` to use ASMM.

Note: The ASH buffer, sort extent pool, and the global context pool are all included in the shared pool.

Dynamic SGA Feature

- **Implements an infrastructure to allow the server to change its SGA configuration without shutting down the instance**
- **SGA size is limited by `SGA_MAX_SIZE`:**
  - **Used for reserving virtual memory address space at instance startup**
  - **Cannot be changed dynamically**
- **Allows for certain SGA components to be dynamically resized**

```
SELECT bytes
FROM   V$SGAINFO
WHERE  name = 'Free SGA Memory Available';
```


Dynamic SGA Feature

The dynamic SGA feature was introduced in Oracle9*i* Database. The System Global Area (SGA) is internally divided into memory components. A component represents a pool of memory used to satisfy a particular class of memory allocation requests. The most commonly configured memory components include the database buffer cache, shared pool, large pool, Java pool, and Streams pool.

The dynamic SGA infrastructure allows for the resizing of certain components of the SGA without having to shut down the instance, modify the initialization parameter file, and restart the instance.

In addition, the dynamic SGA infrastructure allows limits to be set at run time on how much memory is used for the entire SGA. The parameter that limits the memory is `SGA_MAX_SIZE`. This is the amount of memory allocated at startup of the instance, regardless of whether the individual components utilize the entire amount of memory.

Note: During startup of the instance, if the sum of all the memory used by the SGA components exceeds the amount allotted by `SGA_MAX_SIZE`, then the value of `SGA_MAX_SIZE` is adjusted to meet the memory requirement.

Granule

- **SGA memory is allocated in units of contiguous memory chunks called granules.**
- **The size of a granule depends on the estimated total SGA. If the estimated SGA size is:**
  - **Less than or equal to 1 GB, the granule size is 4 MB**
  - **Greater than 1 GB, the granule size is 16 MB**

```
SELECT bytes
FROM   V$SGAINFO
WHERE  name = 'Granule Size';
```


Granule

The SGA memory is allocated in granules. The number of granules used by each SGA component is recorded in dynamic performance views.

Granule size is determined by total SGA size. On most platforms, the size of a granule is 4 MB if the total SGA size is less than 1 GB, and granule size is 16 MB for a larger SGA. Some platform dependencies arise. For example, on 32-bit Windows, the granule size is 8 MB for an SGA larger than 1 GB.

The granule size that is currently being used for the SGA for each component can be viewed in the `V$SGAINFO` view.

Memory Advisories

- **Buffer Cache Advice (introduced in 9*i* R1):**
  - **`V$DB_CACHE_ADVICE`**
  - **Predicts physical reads for different cache sizes**
- **Shared Pool Advice (in 9*i* R2):**
  - **`V$SHARED_POOL_ADVICE`**
  - **Predicts parse time savings from having different sizes of the shared pool**
- **Java Pool Advice (in 9*i* R2):**
  - **`V$JAVA_POOL_ADVICE`**
  - **Predicts Java class load time savings from having different sizes of Java pool**
- **Streams Pool Advice (10*g* R2)**
  - **`V$STREAMS_POOL_ADVICE`**
  - **Predicts spill and unspill activity for various sizes**

ORACLE

Memory Advisories

To help you size the most important SGA components, a number of advisories have been introduced as follows:

- `V$DB_CACHE_ADVICE` contains rows that predict the number of physical reads for the cache size corresponding to each row.
- `V$SHARED_POOL_ADVICE` displays information about the estimated parse time in the shared pool for different pool sizes.
- `V$JAVA_POOL_ADVICE` displays information about the estimated parse time in the Java pool for different pool sizes.
- `V$STREAMS_POOL_ADVICE` displays information about the estimated count of spilled or unspilled messages and the associated time spent in the spill or unspill activity for different Streams pool sizes.

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

Manually Adding Granules to Components

- Use the `ALTER SYSTEM` command to dynamically increase memory allocation to a component.
- Increasing the memory use of a component succeeds only if there are enough free granules to satisfy the request.
- Memory granules are not freed automatically from another component to satisfy the increase.
- Decreasing the size of a component is possible, but only if the granules being released are unused by the component.

Adding Granules to Components

You can increase the size of the shared pool or any buffer cache component by issuing an `ALTER SYSTEM` command. The new size is rounded up to the nearest multiple of the granule size.

Increasing the memory use of a component with an `ALTER SYSTEM` command succeeds only if there are enough free granules (`SGA_MAX_SIZE` minus current size of all the SGA components) to satisfy the request. The server does not free another component's granules to enable a SGA component to grow. You must ensure that the instance has enough free granules to satisfy the increase of a component's granule use. If there are free granules available, then the server can allocate more granules until the SGA size reaches `SGA_MAX_SIZE`.

Note: When resizing SGA components, remember that there is a portion of the SGA that is fixed. This fixed memory will have a minimum of one granule.

Increasing the Size of an SGA Component

```
SQL> show parameter

NAME              TYPE        VALUE
-------------     ----------- ----------------------
sga_max_size      big integer 200M
shared_pool_size  big integer 84M
db_cache_size     big integer 92M

SQL> alter system set shared_pool_size=120M;
alter system set shared_pool_size=120M
*
ERROR at line 1:
ORA-02097: parameter cannot be modified …
ORA-04033: Insufficient memory to grow pool

SQL> alter system set db_cache_size=50M;
System altered.

SQL> alter system set shared_pool_size=120M;
System altered.
```


Increasing the Size of an SGA Component

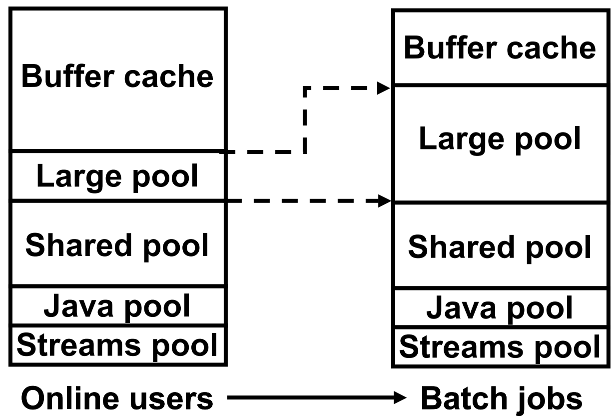
The starting point for the memory allocations is shown in this table:

```
NAME              TYPE        VALUE
-------------     ----------- ----------------------
sga_max_size      big integer 200M
db_cache_size     big integer 92M
java_pool_size    big integer 4M
large_pool_size   big integer 4M
shared_pool_size  big integer 84M
streams_pool_size big integer 8M
```

In the example in the slide, memory is made available for the shared pool by first shrinking the buffer cache.

Automatic Shared Memory Management: Overview

- **Uses dynamic SGA and memory advisors to automatically adapt to workload changes**
- **Maximizes memory utilization**
- **Helps eliminate out-of-memory errors**
- **Avoids relearning when using `SPFILE`**



Automatic Shared Memory Management: Overview

Automatic Shared Memory Management (ASMM) is a key self-management feature. This functionality automates the management of the most important SGA components, and relieves you of having to configure these components manually. ASMM uses memory advisor data to evaluate the best memory configuration, and then resizes components by using the dynamic SGA feature. ASMM makes more effective use of available memory and thereby reduces the cost incurred for acquiring additional hardware memory resources, and significantly simplifies Oracle database administration with a more dynamic, flexible, and adaptive memory management scheme.

For example, in a system that runs high-concurrency OLTP workload during the day, which requires a large buffer cache, you would have to configure both the buffer cache and the large pool to accommodate your peak requirements. With ASMM, when the OLTP workload runs, the buffer cache is given the required memory to optimize buffer cache access. When the decision support system (DSS) batch job starts up later, the memory is automatically migrated to the large pool so that it can be used by parallel query operations without producing memory overflow errors.

Note: With ASMM, component sizes are saved across a shutdown if an `SPFILE` is used. The sizes are resurrected from before the last shutdown to avoid relearning.

SGA Sizing Parameters: Overview

- **With ASMM, five important SGA components can be automatically sized.**
- **Nondefault buffer pools are not auto-tuned.**
- **Log buffer is not a dynamic component but has a good default.**

Auto-tuned parameters	Manual dynamic parameters	Manual static parameters
SHARED_POOL_SIZE DB_CACHE_SIZE LARGE_POOL_SIZE JAVA_POOL_SIZE STREAMS_POOL_SIZE	DB_KEEP_CACHE_SIZE	LOG_BUFFER
	DB_RECYCLE_CACHE_SIZE	SGA_MAX_SIZE
	DB_*n*K_CACHE_SIZE	
	SGA_TARGET	

ORACLE

SGA Sizing Parameters: Overview

As shown in the slide, the five most important pools are automatically tuned when ASMM is activated. The parameters for these pools are referred to as *auto-tuned parameters*.

Manual dynamic parameters are parameters that can be manually resized without having to shut down the instance, but are not automatically tuned by the system.

Manual static parameters are parameters that are fixed in size, and cannot be resized without shutting down the instance first.

Benefits of Automatic Shared Memory Management

Benefits of Automatic Shared Memory Management

Automatic Shared Memory Management (ASMM) simplifies the configuration of the System Global Area (SGA). In past releases of the Oracle database, you needed to manually specify the amount of memory to be allocated for the database buffer cache, shared pool, Java pool, large pool, and Streams pool. It is often a challenge to size these components optimally. Undersizing can lead to poor performance and out-of-memory errors (`ORA-4031`), whereas oversizing can waste memory.

ASMM enables you to specify a total memory amount to be used for all SGA components. The Oracle database server periodically redistributes memory between the components given in the slide according to workload requirements.

In earlier releases, you did not have exact control over the total size of the SGA because memory was allocated for the fixed SGA and for other internal metadata allocations over and above the total size of the user-specified SGA parameters. This additional memory was usually between 10 and 20 MB.

Setting the `SGA_TARGET` parameter to a nonzero value enables ASMM. The `SGA_TARGET` initialization parameter includes all memory in the SGA, including the automatically sized components, the manually sized components, and any internal allocations during startup.

Dynamic SGA Transfer Modes

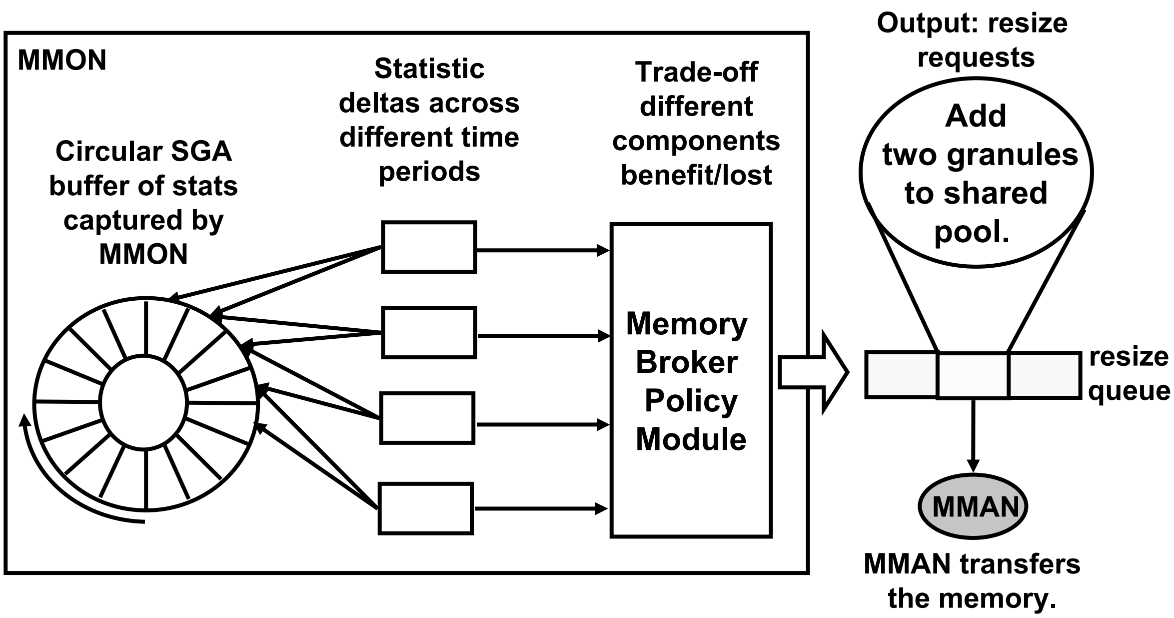
- ASMM `IMMEDIATE` transfer mode:
  - Out-of-memory (`ORA-04031`) errors
  - Partial granules can be used
- ASMM `DEFERRED` transfer mode:
  - Transparently executed in the background
  - Partial granules can be used
- `MANUAL` transfer mode:
  - Used with `ALTER SYSTEM` commands
  - Resize must use full granules

Dynamic SGA Transfer Modes

Currently, there are three types of granule transfer mechanisms between dynamic SGA components:

- `IMMEDIATE`: This mode is used when ASMM is activated and one of the auto-tuned components is about to get an out-of-memory error. To prevent the error from happening, the system tries to transfer a granule from another component. This transfer can be a partial transfer if there are no entirely empty granules available. If that is the case, the system starts cleaning out granules from other components to satisfy the memory request, and partially transfer a granule to the component requesting for memory.
- `DEFERRED`: This mode is used by the system to transfer memory between components when it determines there is a more efficient memory distribution. Advisory-related data is used to determine the optimal memory distribution.
- `MANUAL`: This mode is used when you are asking the system to resize components. This mode can use only empty granules for the resize operation. If there are no empty granules, an `ORA-4033` error is returned for a grow request and an `ORA-4034` error is returned for a shrink request.

Memory Broker Architecture



Memory Broker Architecture

When ASMM is enabled, the memory broker periodically performs the activities in this slide. These operations are in `DEFERRED` mode. Its goal is to automate the sizing of auto-tuned components to adapt to workload changes by distributing the memory to where it is most needed. The transfer is done when there is an overall benefit to do so.

During this background action, statistics and memory advisory data are periodically captured in a circular buffer by MMON.

Deltas between different buffer entries represent statistics for different time periods. These deltas are computed by MMON.

MMON uses the Memory Broker policy to analyze deltas and examines both long-term and short-term trends.

MMON generates resize decisions based on this analysis by posting requests in the resize request system queue. MMAN scans the request system queue periodically to execute the corresponding memory transfers.

Manually Resizing Dynamic SGA Parameters

- **For auto-tuned parameters, manual resizing:**
  – **Results in immediate component resize if the new value is greater than the current size**
  – **Changes the minimum size if the new value is smaller than the current size**
- **Manually tuned parameter resizing affects the tunable portion of the SGA.**

ORACLE

Manually Resizing Dynamic SGA Parameters

When an auto-tuned parameter is resized and SGA_TARGET is set, the resize results in an immediate change to the size of the component only if the new value is larger than the current size of the component. For example, if you set SGA_TARGET to 8 GB and set SHARED_POOL_SIZE to 2 GB, then you ensure that the shared pool has at least 2 GB at all times to accommodate the necessary memory allocations. After this, adjusting the value of SHARED_POOL_SIZE to 1 GB has no immediate effect on the size of the shared pool. It simply allows the automatic memory-tuning algorithm to later reduce the shared pool size to 1 GB if it needs to. However, if the size of the shared pool is 1 GB to begin with, then adjusting the value of SHARED_POOL_SIZE to 2 GB results in the shared pool component growing to a size of 2 GB. The memory used in this resize operation is taken away from one or more auto-tuned components, and the sizes of the manual components are not affected.

Parameters for manually sized components can be dynamically altered as well, but the difference is that the value of the parameter specifies the precise size of that component immediately. Therefore, if the size of a manual component is increased, extra memory is taken away from one or more automatically sized components. If the size of a manual component is decreased, the memory that is released is given to the automatically sized components.

Behavior of Auto-Tuned SGA Parameters

- When `SGA_TARGET` is not set or is set to zero:
  - Auto-tuned parameters are explicitly set
  - Note: `SHARED_POOL_SIZE`

    Internal startup overhead is included
    Value may need to be increased from previous releases
- When `SGA_TARGET` is set:
  - Default value of auto-tuned parameters is zero
  - A nonzero value is a lower bound
  - Current values in megabytes are shown by:

```
SELECT component, current_size/1024/1024
FROM   V$SGA_DYNAMIC_COMPONENTS;
```


Behavior of Auto-Tuned SGA Parameters

When `SGA_TARGET` is not set or is equal to zero, auto-tuned SGA parameters must be explicitly set. `SHARED_POOL_SIZE` is an exception. In previous releases of the Oracle database, internal overhead allocations for metadata (data structures for processes, sessions, and so on) were not part of the `SHARED_POOL_SIZE` parameter, but are now included.

Note: Upgrade issues: To get the same effective shared pool size when upgrading to Oracle Database 10*g*, increase the value of `SHARED_POOL_SIZE` to account for internal allocations. For example, if you were using a `SHARED_POOL_SIZE` value of 256 MB in a prior release, and if the value of the internal allocations was 32 MB, then you need to set `SHARED_POOL_SIZE` to 288 MB with Oracle Database 10*g*. The exact value of internal startup overhead can be found with the following query:

```
select * from V$SGAINFO where name like 'Startup%';
```

When `SGA_TARGET` is set to a nonzero value, the auto-tuned SGA parameters have default values of zero. These components are automatically sized by the ASMM algorithm. However, if they are set to nonzero values, the specified values are used as a lower bound. For example, if `SGA_TARGET` is set to 8 GB and `SHARED_POOL_SIZE` is set to 1 GB, the shared pool should not shrink below 1 GB, but it may grow to larger values. You can use the query to determine the actual size of the auto-tuned components in the SGA.

Behavior of Manually Tuned SGA Parameters

- **Manually tuned components are:**
  - **`KEEP` and `RECYCLE` buffer caches**
  - **Nondefault block size caches**
  - **`LOG_BUFFER`**
- **Manually tuned components are user specified.**
- **Manually tuned components are included in `SGA_TARGET` to precisely control the SGA size.**

Behavior of Manually Tuned SGA Parameters

The manual SGA size parameters are:

- `DB_KEEP_CACHE_SIZE`
- `DB_RECYCLE_CACHE_SIZE`
- `DB_nK_CACHE_SIZE` (*n* = `2`, `4`, `8`, `16`, `32`)
- `LOG_BUFFER`

Manual SGA parameters are specified by the user, and the given sizes precisely control the sizes of their corresponding components.

When `SGA_TARGET` is set, the total size of manual SGA size parameters is subtracted from the `SGA_TARGET` value, and the balance is given to the auto-tuned SGA components.

For example, if `SGA_TARGET` is set to 8 GB and `DB_KEEP_CACHE_SIZE` is set to 1 GB, the total size of the five auto-tuned components (shared pool, Java pool, default buffer cache, large pool, and Streams pool) is limited to 7 GB. The 7 GB size includes the fixed SGA and log buffer. Only after those have been allocated is the rest of the memory divided between the auto-tuned components. The size of the keep cache is 1 GB, as specified by the parameter.

Using the V$PARAMETER View

```
SGA_TARGET = 8G
```

```
SELECT name, value, isdefault
FROM   V$PARAMETER
WHERE  name LIKE '%size';
```

```
DB_CACHE_SIZE = 0
JAVA_POOL_SIZE = 0
LARGE_POOL_SIZE = 0
SHARED_POOL_SIZE = 0
STREAMS_POOL_SIZE = 0
```

ORACLE

Using the V$PARAMETER View

When you specify a nonzero value for SGA_TARGET and do not specify a value for an auto-tuned SGA parameter, the value of the auto-tuned SGA parameters in the V$PARAMETER view is 0, and the value of the ISDEFAULT column is TRUE.

If you have specified a value for any of the auto-tuned SGA parameters, the value displayed when you query V$PARAMETER is the value that you specified for the parameter.

Resizing `SGA_TARGET`

- The `SGA_TARGET` initialization parameter:
  - Is dynamic
  - Can be increased up to `SGA_MAX_SIZE`
  - Can be reduced until all components reach minimum size
  - Changes affect only automatically sized components
- Includes everything in the SGA:
  - Fixed SGA and other internal allocations
  - Automatically sized SGA components
  - Manual SGA components
- Allows precise sizing of the total shared memory allocation by the Oracle server

Resizing `SGA_TARGET`

`SGA_TARGET` is a dynamic parameter and can be changed through Database Control or with the `ALTER SYSTEM` command.

`SGA_TARGET` can be increased up to the value of `SGA_MAX_SIZE`. It can be reduced until any one of the auto-tuned components reaches its minimum size: either a user-specified minimum or an internally determined minimum.

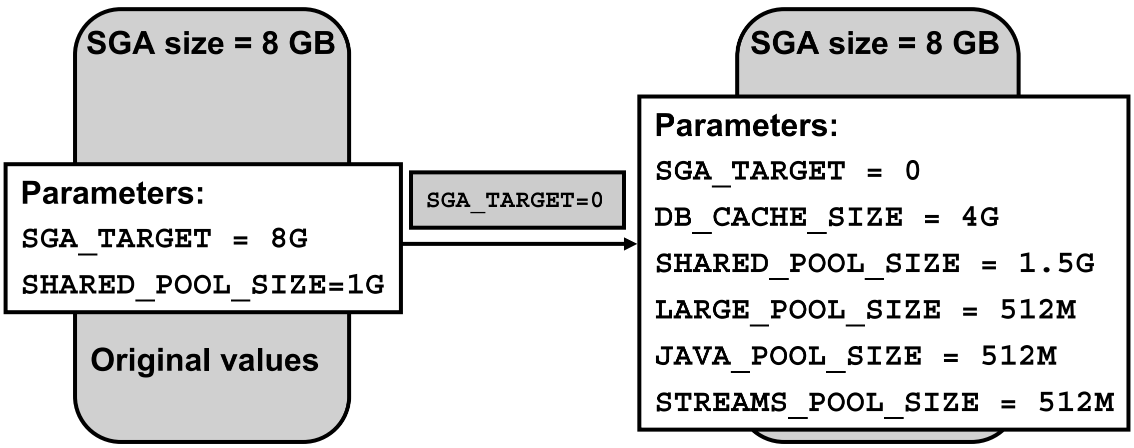
If you increase the value of `SGA_TARGET`, the additional memory is distributed according to the auto-tuning policy across the auto-tuned components.

If you reduce the value of `SGA_TARGET`, the memory is taken away by the auto-tuning policy from one or more of the auto-tuned components. Therefore, any change in the value of `SGA_TARGET` affects only the sizes of the auto-tuned components.

For example, `SGA_MAX_SIZE` is set to 10 GB and `SGA_TARGET` is set to 8 GB. If `DB_KEEP_CACHE_SIZE` is set to 1 GB and you increase `SGA_TARGET` to 9 GB, the additional 1 GB is distributed only among the components controlled by `SGA_TARGET`. The value of `DB_KEEP_CACHE_SIZE` is not affected. Likewise, if `SGA_TARGET` is reduced to 7 GB, the 1 GB is taken from only those components controlled by `SGA_TARGET`. This decrease does not change the minimum settings of parameters you explicitly set such as `DB_KEEP_CACHE_SIZE`.

Disabling Automatic Shared Memory Management

- Setting SGA_TARGET to zero disables auto-tuning.
- Auto-tuned parameters are set to their current sizes.
- SGA size as a whole is unaffected.

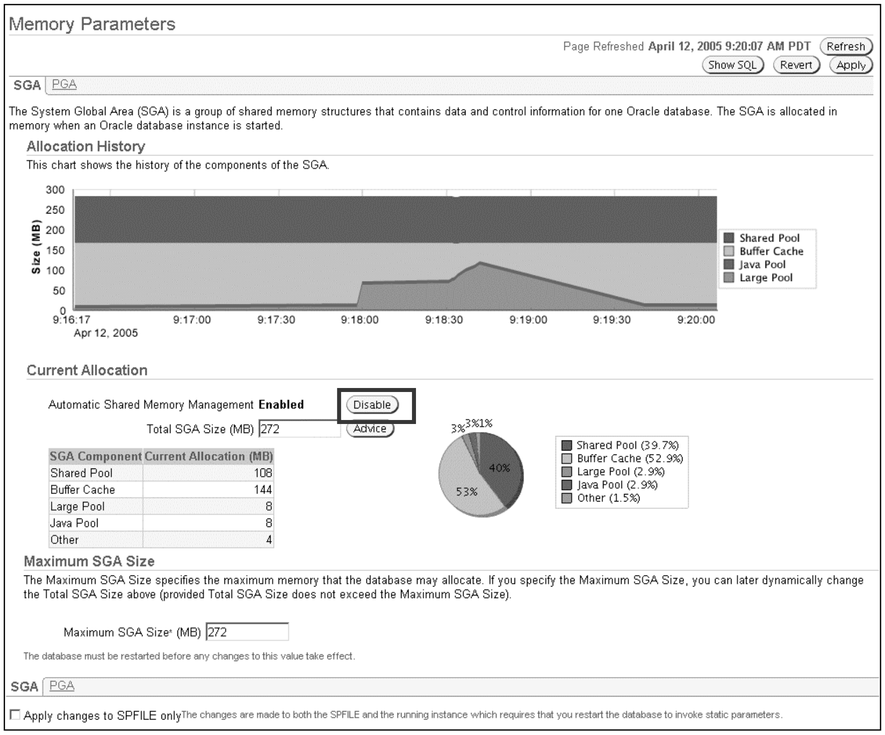


Disabling Automatic Shared Memory Management

You can dynamically choose to disable Automatic Shared Memory Management by setting SGA_TARGET to zero. In this case, the values of all the auto-tuned parameters are set to the current sizes of the corresponding components, even if you had earlier specified a different nonzero value for an auto-tuned parameter.

In the example in the slide, the value of SGA_TARGET is 8 GB and the value of SHARED_POOL_SIZE is 1 GB. If the system has internally adjusted the size of the shared pool component to 1.5 GB, then setting SGA_TARGET to zero results in SHARED_POOL_SIZE being set to 1.5 GB, thereby overriding the original user-specified value.

Configuring ASMM



Configuring Automatic Shared Memory Management

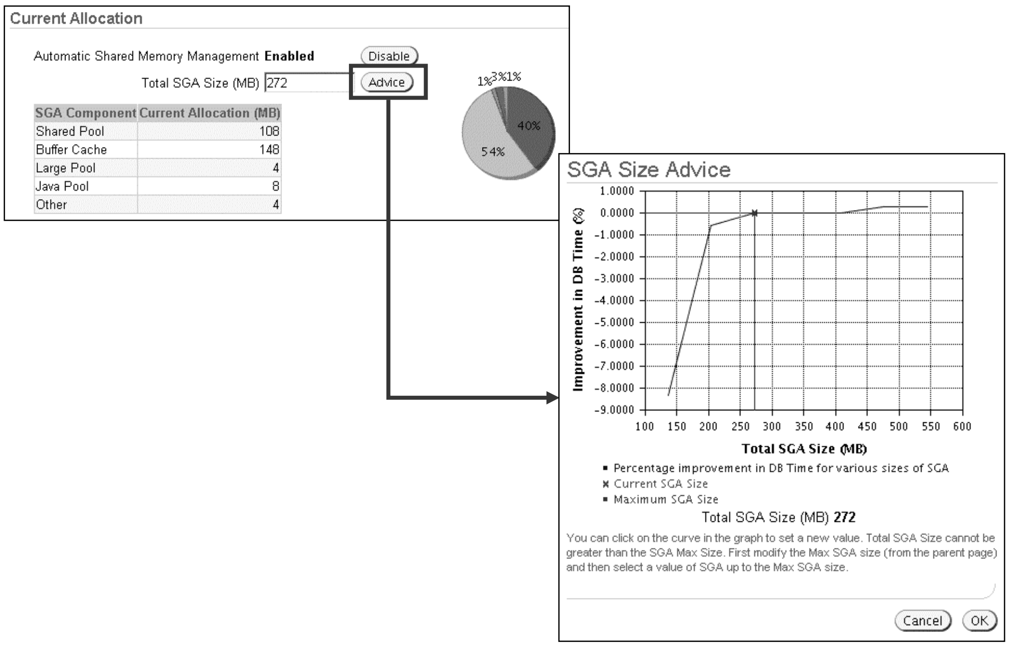
You can use Database Control to configure Automatic Shared Memory Management as follows:

- Click the Administration tab.
- Select Memory Parameters under the Database Configuration heading.
- Click the SGA tab.
- Click the Enable button for Automatic Shared Memory Management, and then enter the total SGA size (in MB). This represents the value given to `SGA_TARGET`.
- To disable Automatic Shared Memory Management, click Disable.

In Oracle Database 10*g* Release 2, the default for Automatic Shared Memory Management is Enabled. On the Memory Parameters page, you see a new chart that displays the history of the SGA allocation. You can put your cursor over the area in the chart you are interested in, which highlights the component in the legend to the right of the chart.

Note: The Streams pool is not exposed through the EM interface.

SGA Advisor



SGA Advisor

The SGA advisor shows the improvement in DB Time that can be achieved if a Total SGA Size is specified. In this example, if the Total SGA Size is increased to approximately 470 MB, you gain approximately 5% improvement in DB Time.

This advisor allows you to reduce trial and error in setting the SGA size.

The advisor data is stored in the `V$SGA_TARGET_ADVICE` table with the following values:

- **`SGA_SIZE`:** Size of the SGA
- **`SGA_SIZE_FACTOR`:** Ratio between the `SGA_SIZE` and the current size of the SGA
- **`ESTD_DB_TIME`:** Estimated `DB_TIME` for this `SGA_SIZE`
- **`ESTD_DB_TIME_FACTOR`:** Ratio between `ESTD_DB_TIME` and `DB_TIME` for the current size of the SGA
- **`ESTD_PHYSICAL_READS`:** Estimated number of physical reads

Note: DB Time is the same as the Database Time discussed in the lesson titled “Using Automatic Workload Repository.” Database Time includes all the waits to perform an operation; in this case, the difference in the reads and writes required with a different `SGA_TARGET` size.

Monitoring ASMM

Monitor Automatic Shared Memory Management and examine the resize decisions it made with the following views:

- `V$SGA_CURRENT_RESIZE_OPS`: **Information about resize SGA operation in progress**
- `V$SGA_RESIZE_OPS`: **Circular history buffer of the last 800 SGA resize requests**
- `V$SGA_DYNAMIC_COMPONENTS`: **Current status of all memory components**
- `V$SGA_DYNAMIC_FREE_MEMORY`: **Information about SGA memory available for future resize operations**

Monitoring ASMM

The following views provide information about dynamic SGA resize operations:

- **V$SGA_CURRENT_RESIZE_OPS:** Information about SGA resize operations that are currently in progress. An operation can be a grow or a shrink of a dynamic SGA component.
- **V$SGA_RESIZE_OPS:** Information about the last 800 completed SGA resize operations. This does not include any operations currently in progress.
- **V$SGA_DYNAMIC_COMPONENTS:** Information about the dynamic components in SGA. This view summarizes information based on all completed SGA resize operations since startup.
- **V$SGA_DYNAMIC_FREE_MEMORY:** Information about the amount of SGA memory available for future dynamic SGA resize operations

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

Summary

In this lesson, you should have learned how to:

- Enable EM memory parameters
- Set auto-tuned memory parameters
- Set the manually tuned SGA parameters
- Use the SGA advisor to set `SGA_TARGET`

Practice 10 Overview: Enabling Automatic Shared Memory

This practice covers the following topics:

- Enabling Automatic Shared Memory Management
- Adjusting memory as workloads change

11

Checkpoint and Redo Tuning

Objectives

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

- Diagnose checkpoint and redo issues
- Implement Fast Start MTTR Target
- Monitor the performance impact of Fast Start MTTR Target
- Tune the redo chain
- Size the redo log file
- Size the redo log buffer

Checkpoint and Redo

- **Checkpoint:**
  - **Transfers changed data to disk**
  - **Makes buffer space available for more data blocks**
  - **Controls mean time to recover (MTTR)**
- **Redo:**
  - **Recovers committed data *not* on disk**
  - **Recovers uncommitted data for rollback operations**
  - **Provides source data for complete recovery**

Checkpoint and Redo

Checkpoint and redo are related tasks. The processes that perform these tasks are coordinated. When there is a performance problem in one area, the entire set of tasks can be affected. The symptoms and diagnostics for these two areas are very similar, even though the purposes of the tasks are different. In general terms, the purposes of each are the following:

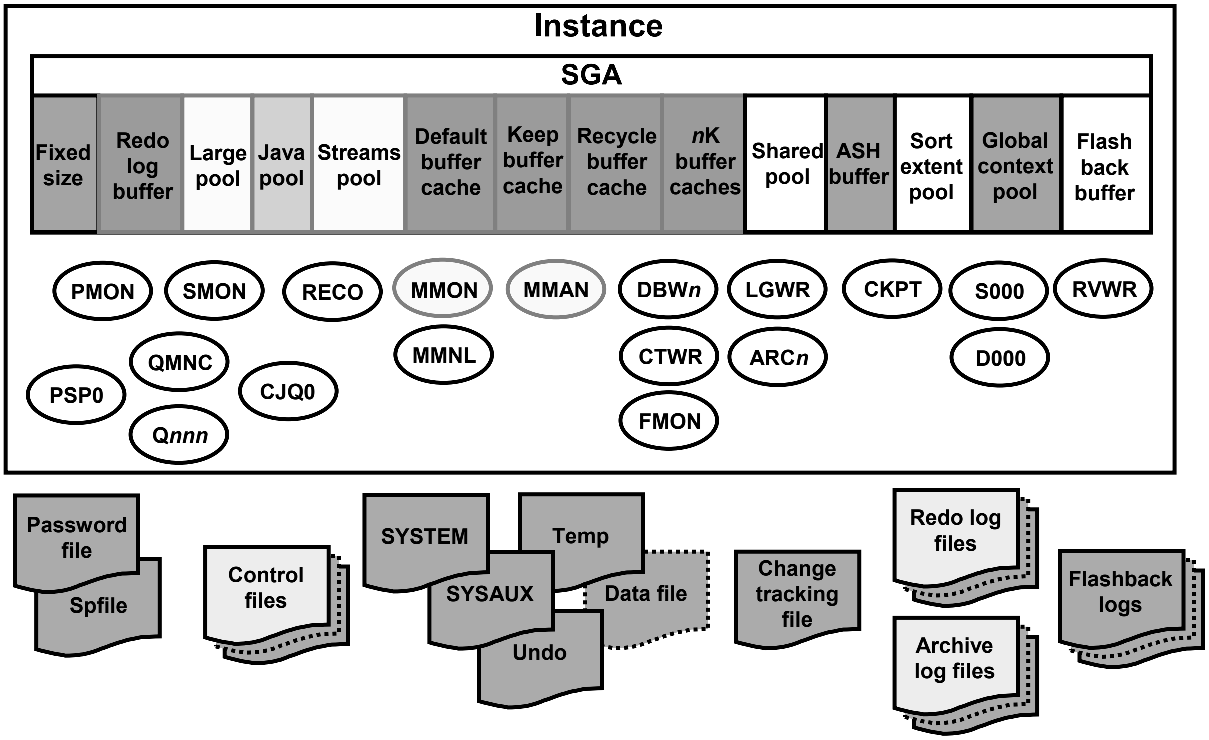
Checkpoint:

- Transfers changed data to disk. The checkpoint records the data blocks that have been transferred to disk and no longer require redo for recovery.
- Makes buffer space available for more data blocks. Buffers are marked free.
- Controls mean time to recover (MTTR). The number of buffers changed but not yet copied to disk determines the instance recovery time.

Redo:

- Recovers committed data *not* on disk
- Recovers uncommitted data for rollback operations
- Provides source data for complete recovery. If being used with `ARCHIVELOG` mode, the redo log files are copied to archive log files, or else the redo logs are overwritten.

Oracle Database Architecture



Oracle Database Architecture

The slide shows the key SGA components and processes that are involved in performing checkpoints and maintaining the Redo Log stream.

SGA Components
The log buffer and all of the buffer cache components participate in the checkpoint and redo operations.

Processes and Files

- The DBWn process is responsible for writing the database blocks to disk and updating the checkpoint position.
- The CKPT process is responsible for writing the checkpoint position information to the control files, and to all data files.
- The LGWR process writes the redo log buffer data to redo log files, and the ARCn processes copy the redo log files to archive log files.

Checkpoint Architecture

The checkpoint architecture provides:

- **Checkpoint position: A starting position in the redo logs to begin recovery**
- **Checkpoint target: A calculated position in the redo logs where the checkpoint position should be**
- **An estimated mean time to recover**
- **A high-performance incremental checkpoint**
- **A full checkpoint when required**

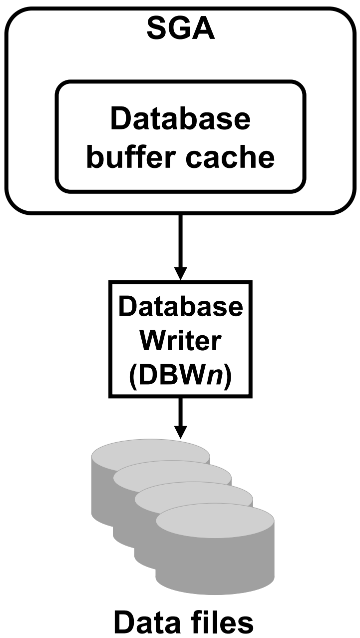
Checkpoint Architecture

The Oracle database checkpoint architecture provides a high-performance method for ensuring that data is correctly written to disk and is recoverable in case of an instance failure. The recoverability aspect is covered in more detail in the *Oracle Database 10g: Database Administrator Workshop II* course.

There are two types of checkpoints: full and incremental. Full checkpoints are performed only by command. A full checkpoint occurs at instance shutdown, except for `SHUTDOWN ABORT` or `STARTUP FORCE`. When a full checkpoint is performed, all the dirty blocks in the buffer cache are written to the data files. The normal checkpoint that occurs is an incremental checkpoint where the blocks that have been dirty the longest time are written to the data files.

After a dirty block has been written to the data file, the redo records of the operations that changed the block are no longer required for instance recovery. The checkpoint position is a Redo Block Address (RBA) in the redo log file that shows that none of the redo records written before this RBA are needed for instance recovery. The checkpoint target is a calculated RBA that is based on an estimated time required to apply the remaining redo entries while performing an instance recovery. Internal algorithms determine how rapidly to advance the checkpoint position to match or exceed the checkpoint target.

Database Writer (DBW*n*) Process



Background Information

DBW*n* writes when one of the following events occurs:

- **Checkpoint**
- **Dirty buffers' threshold**
- **No free buffers**
- **Timeout**
- **RAC ping request**
- **Tablespace `OFFLINE`**
- **Tablespace `READ ONLY`**
- **Table `DROP` or `TRUNCATE`**
- **Tablespace `BEGIN BACKUP`**

Database Writer (DBW*n*)

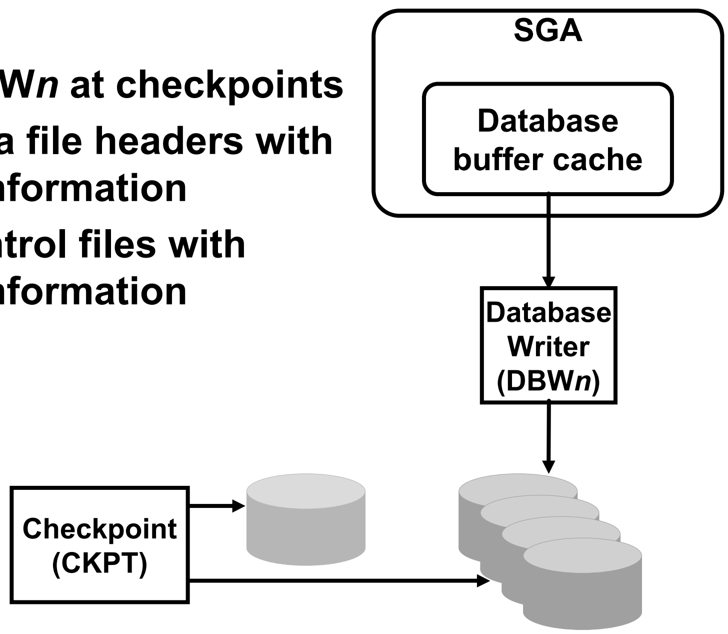
The server process records changes to undo and data blocks in the database buffer cache. DBW*n* writes the dirty buffers from the database buffer cache to data files. It ensures that a sufficient number of free buffers (buffers that can be overwritten when server processes need to read blocks from the data files) are available in the database buffer cache. DBW*n* advances the checkpoint. Server processes make changes only in the database buffer cache. This is a prerequisite for high database performance.

DBW*n* defers writing to the data files until one of the events shown in the slide occurs. If any of these events occur very frequently, DBW*n* must write more and smaller batches. A buffer cache that is too small can cause extra writes for DBW*n*. Excessive checkpoints can also cause DBW*n* to write more frequently and reduce the performance.

Checkpoint (CKPT) Process

Responsible for:

- **Signaling DBW*n* at checkpoints**
- **Updating data file headers with checkpoint information**
- **Updating control files with checkpoint information**



Checkpoint (CKPT) Process

Every three seconds (or more frequently), the CKPT process stores data in the control file to document which modified data blocks DBW*n* has written from the SGA to disk. This is called a "checkpoint." The purpose of a checkpoint is to identify that place in the online redo log file where instance recovery is to begin (called the "checkpoint position").

In the event of a log switch, the CKPT process also writes this checkpoint information to the headers of data files.

Checkpoints exist for the following reasons:

- To ensure that modified data blocks in memory are written to the disk regularly so that data is not lost in case of a system or database failure
- To reduce the time required for instance recovery. Only the online redo log file entries following the last checkpoint need to be processed for recovery.
- To ensure that all committed data has been written to data files during shutdown

The checkpoint information written by the CKPT process includes checkpoint position, system change number, location in the online redo log file where to begin recovery, information about logs, and so on.

Note: The CKPT process does not write data blocks to the disk or redo blocks to the online redo log files.

Redo Architecture

Redo is designed for minimum performance impact.

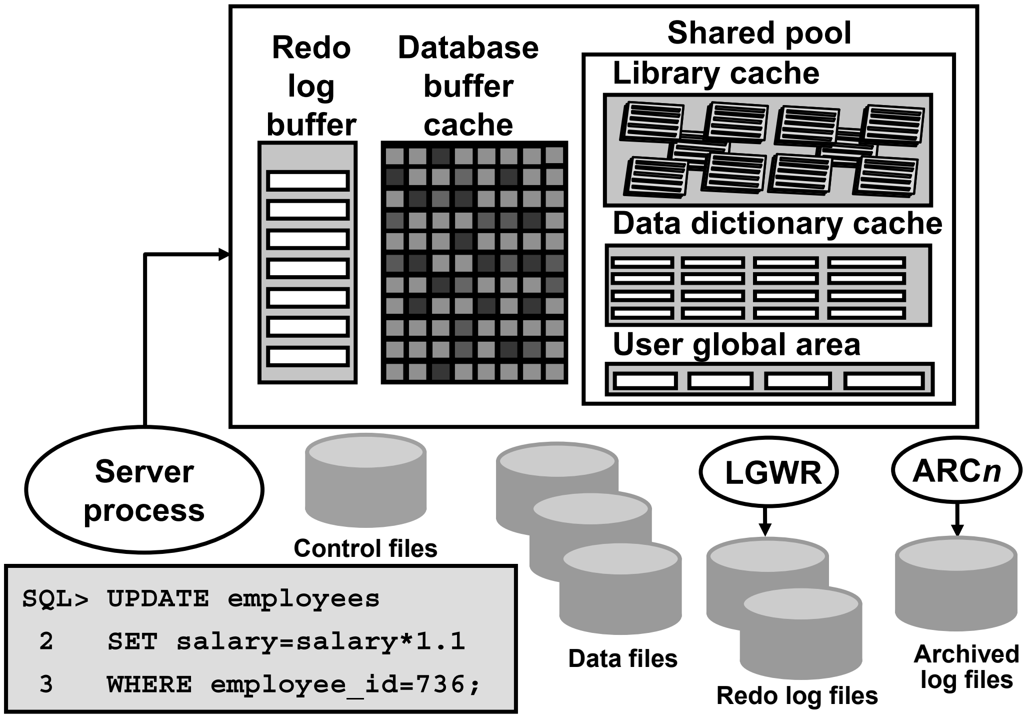
- **Server processes write to the redo log buffer:**
  - **Circular buffer**
  - **Memory-to-memory writes**
  - **Small fast writes**
- **LGWR writes log buffer blocks to log files:**
  - **Circular files**
  - **Memory-to-disk write**
  - **Full blocks if possible**
- **ARC*n* copies log files to archive log files:**
  - **Disk-to-disk writes**
  - **Multiple archiver processes can be started.**

Redo Architecture

Redo logging is a required feature of the Oracle database. Redo is designed in conjunction with checkpoints to provide a high-performance mechanism to protect the database from corruption or data loss due to an instance failure. Checkpoints record the starting point in the redo logs for instance recovery, and the redo records have the changes that need to be applied to the data files to make the data match the last committed transactions before the instance failure. Redo records are small change vectors to the rows that are changed, not the entire data block, except in certain conditions. The server processes perform memory-to-memory writes of these records. LGWR writes full blocks if possible from the log buffer (memory) to log files (disk) to make space for more redo records in the log buffer. If `ARCHIVELOG` mode is enabled, the archiver processes (ARC*n*) copy the redo log file to a unique archive log file at each log switch.

Archiver processes perform disk-to-disk copies and write in the largest batch of the three process types. If archiver processes cannot write fast enough to clear a log file, LGWR waits until the archiver is complete. LGWR does not overwrite a log file until it has been archived. When LGWR cannot clear log buffer space for server processes to write redo records, the user sessions wait. This can happen when LGWR is waiting for archiver, log switches, or checkpoints to complete.

The Redo Log Buffer

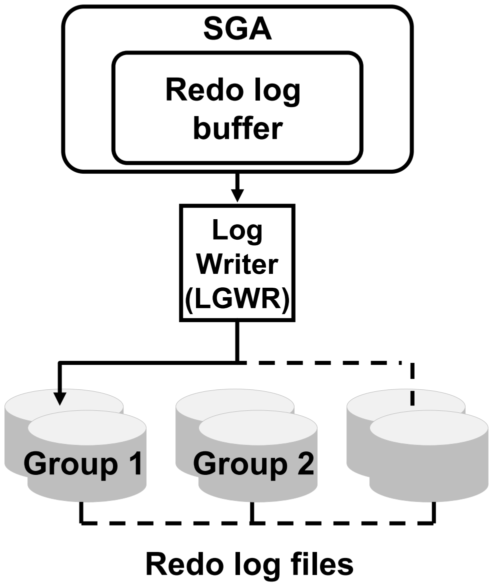


Redo Log Buffer Content

The Oracle server processes copy redo entries from the user's memory space to the redo log buffer for each DML or DDL statement. The redo entries contain the information necessary to reconstruct or redo changes made to the database by DML and DDL operations. These changes include changes to objects, rows, tables, indexes, and undo segments. The redo records are used for database recovery and take up continuous sequential space in the log buffer.

The redo log buffer is a circular buffer. The server processes can copy new entries over the entries in the redo log buffer that have already been written to disk. The LGWR process normally writes fast enough to ensure that space is always available in the buffer for new entries. The LGWR process writes the redo log buffer to the current online redo log file (or members of the current group) on disk. The LGWR process copies to disk all redo entries that have been entered into the buffer since the last time LGWR wrote to disk.

Redo Log Files and LogWriter



Redo log files:

- **Record changes to the database**
- **Should be multiplexed to protect against loss**

LogWriter writes:

- **At commit**
- **When one-third full**
- **Every three seconds**
- **Before DBW*n* writes**

Redo Log Files and LogWriter

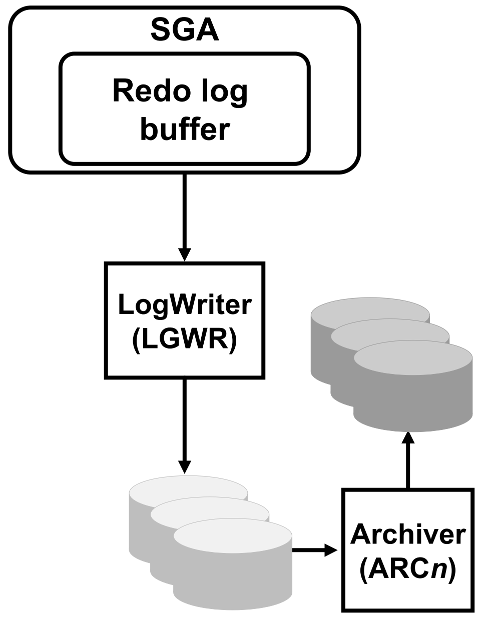
Redo log files record changes to the database as a result of transactions and internal Oracle server actions. (A transaction is a logical unit of work, consisting of one or more SQL statements run by a user.) Redo log files protect the database from the loss of integrity because of system failures caused by power outages, disk failures, and so on. Redo log files should be multiplexed to ensure that the information stored in them is not lost in the event of a disk failure.

The redo log consists of groups of redo log files. A group consists of a redo log file and its multiplexed copies. Each identical copy is said to be a member of that group, and each group is identified by a number. The LogWriter (LGWR) process writes redo records from the redo log buffer to all members of a redo log group until the file is filled or a log switch operation is requested. Then it switches and writes to the files in the next group. Redo log groups are used in a circular fashion.

Best practice tip: If possible, multiplexed redo log files should reside on different disks. This practice spreads I/Os across multiple disks. This is important for archiving as well. The archiver processes read the log file members in a round-robin fashion (one block at a time) while copying, also spreading the I/O across multiple disks.

Archiver (ARC*n*)

- Is an optional background process
- Automatically archives online redo log files when `ARCHIVELOG` mode is set for the database
- Preserves the record of all changes made to the database



Archiver (ARC*n*)

ARC*n* is an optional background process. However, it is crucial to recovering a database after the loss of a disk. As online redo log files get filled, the Oracle instance begins writing to the next online redo log file. The process of switching from one online redo log file to another is called a log switch. The ARC*n* process initiates the copy of the filled log group at every log switch. It automatically archives the online redo log file before the log can be reused, so all the changes made to the database are preserved. This enables recovery of the database to the point of failure even if a disk drive is damaged.

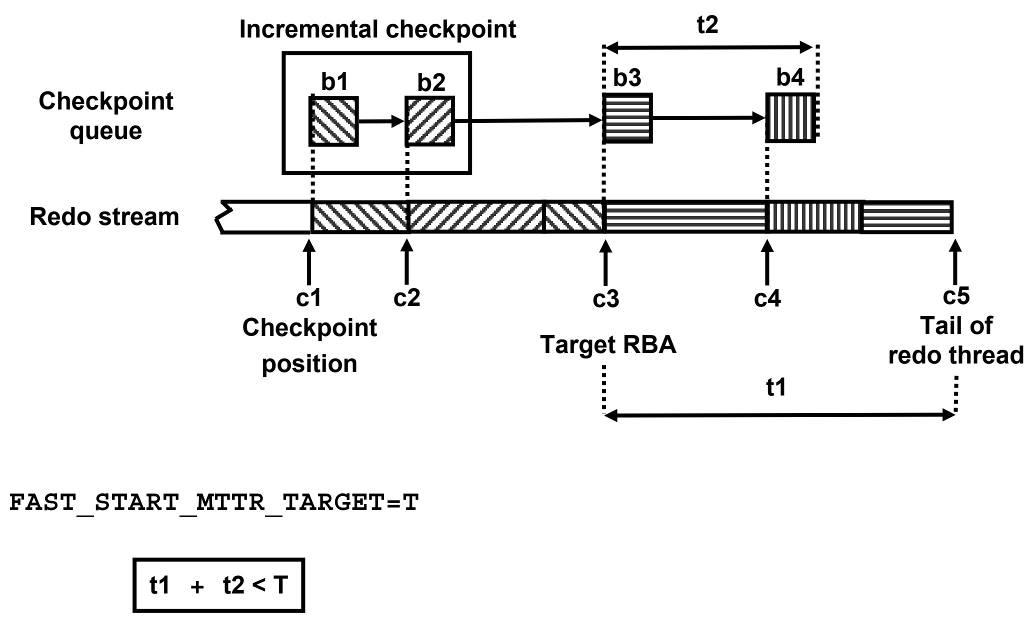
One of the important decisions that a DBA has to make is whether to configure the database to operate in `ARCHIVELOG` mode or in `NOARCHIVELOG` mode.

- In `NOARCHIVELOG` mode, the online redo log files are overwritten each time a log switch occurs.
- In `ARCHIVELOG` mode, inactive groups of filled online redo log files must be archived before they can be used again.

Even though there is a small performance penalty for `ARCHIVELOG` mode, the decision for archiving is generally driven by business needs.

Note: `ARCHIVELOG` mode is essential for most backup strategies (and is very easy to configure).

Incremental Checkpointing



Incremental Checkpointing

Because the Oracle database server uses a write-ahead logging scheme, redo reflecting the intended modification must be generated before a buffer can be modified. This redo is stored in the log buffer, which is periodically drained by the LGWR process to the redo log files. The set of online redo log files for a particular instance is called a redo thread, and acts as a circular buffer. Redo is indexed by Redo Block Address (RBA). The checkpoint RBA is the address at which recovery starts the application of redo in the event of an instance crash. This address is also referred to as the checkpoint position.

If there were one checkpoint queue for the entire instance, each successive write from the beginning of the checkpoint queue could theoretically advance the checkpoint position. This concept is illustrated in the slide. Buffers b1, b2, b3, and b4 are linked in order of their first dirty RBA in the checkpoint queue. Initially, the checkpoint position is at c1, which is the first dirty RBA of b1. Writing b1 to the data file causes the position to advance to c2. Similarly, writing b2 causes the position to advance to c3.

This checkpoint queue structure enables an efficient checkpointing mechanism called incremental checkpointing. You have several parameters that can be set to control incremental checkpointing, but the preferred parameter for configuring incremental checkpointing is `FAST_START_MTTR_TARGET`.

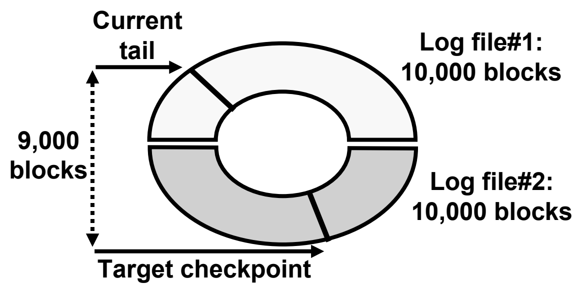
Incremental Checkpointing (continued)

This parameter is an estimate of the mean time to recover your instance in the event of a crash. This time represents the roll-forward phase time and mainly depends on the time it takes to read and process the redo log, as well as the time it takes to read and write data blocks that need recovery. Therefore, if you set `FAST_START_MTTR_TARGET` to T seconds, the database writer process periodically computes a target RBA such that the sum of time to read the redo from that RBA onward (t1) and the time to read and write all the buffers on the checkpoint queue (t2) is less than T seconds. Given this target, the database writer then writes all buffers that have a low RBA (less than the target RBA and greater than the checkpoint position) from the checkpoint queues. This action advances the checkpoint position to the checkpoint target.

Note: With multiple checkpoint queues, the CKPT process updates the control file every three seconds with the new value of the position at which recovery should start. This is the lowest RBA across all the checkpoint queues.

Incremental Checkpoint and Log File Size

- **The maximum checkpoint lag is:**
  $$90\%*(SUM(log\_size_i) - MAX(log\_size_i))$$
- **Checkpoint lag is designed to prevent log switch from blocking.**
- **A few small log files can result in excess checkpoint writes.**



Incremental Checkpoint and Log File Size

The log file size and the number of log file groups also influence checkpointing. Crash recovery must limit itself to online logs. Thus, the checkpoint position must always be within the online log files. When all log files are used up, the system must wrap into one of the used log files. When redo generation is wrapping into a used log file, checkpointing must have advanced ahead of this log file. Otherwise, redo generation will have to wait, and a wait of this type is recorded as a checkpoint complete event.

The incremental checkpointing rules also state that the checkpoint lags by no more than 90% of the size corresponding to the difference between the sum of the sizes of all the redo log files and the largest log file size as shown in the slide. With two log files, this formula corresponds to 90% of the smallest log file size.

This ensures that the incremental checkpoint target passes the beginning of the second log file when the current log tail is at the end of the last log file so that it can perform a log switch without waiting for the checkpoint to complete.

If excessive checkpointing writes are observed and the redo thread consists of multiple small log files, consider using more or larger log files to make sure that log switches do not drive incremental checkpointing.

Adjusting the Checkpoint Rate

The checkpoint rate is determined by the most aggressive of:

- `FAST_START_MTTR_TARGET` **parameter (only on Enterprise Edition)**
- **Size of the smallest redo log file**
- `LOG_CHECKPOINT_TIMEOUT` **parameter (overrides** `FAST_START_MTTR_TARGET` **if set)**
- `LOG_CHECKPOINT_INTERVAL` **parameter (overrides** `FAST_START_MTTR_TARGET` **if set)**

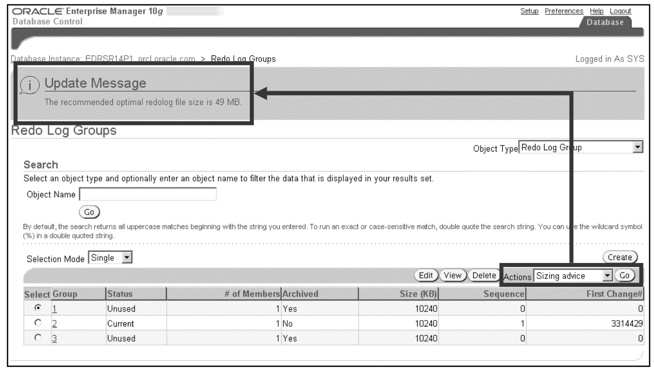
Adjusting the Checkpoint Rate

The checkpoint rate is, by default, the rate at which DBWR must write to maintain free buffers in the buffer pool or meet the checkpoint target. The size of the smallest redo log sets the default checkpoint target.

If you are using Enterprise Edition, `FAST_START_MTTR_TARGET` sets a value that is monitored for the estimated time to complete instance recovery. This value translates into a number of redo entries that are still needed for recovery. The checkpoint position is allowed to lag behind the tail of the redo file by this number of entries. There is a practical minimum that is dominated by the startup time, and a preset maximum of 3,600 seconds (one hour).

In Standard Edition, the log file size and the `LOG_CHECKPOINT_INTERVAL` and `LOG_CHECKPOINT_TIMEOUT` parameters control the checkpoint target position. `LOG_CHECKPOINT_INTERVAL` sets the number of blocks of redo that are allowed between the checkpoint target and the tail of the redo log. `LOG_CHECKPOINT_TIMEOUT` sets the number of seconds between the time a redo record is written and the time the associated database block is written. This has the side effect of setting the number of seconds a block can remain dirty in the buffer cache. Another way of saying this is that the checkpoint target will move to the RBA of the current tail of the redo log file in `LOG_CHECKPOINT_TIMEOUT` seconds.

Redo Logfile Size Advisor



- **This advisor determines the optimal size of your online redo logs:**
  - **No additional checkpoint writes beyond those caused by `FAST_START_MTTR_TARGET`.**
- **`FAST_START_MTTR_TARGET` must be set.**

View name	`V$INSTANCE_RECOVERY`
Column name	`OPTIMAL_LOGFILE_SIZE`
Description	**This column shows the redo log file size (in megabytes) that is considered as minimal.**

ORACLE

Redo Logfile Size Advisor

Oracle Database 10*g* includes the Redo Logfile Size Advisor. This advisor determines the optimal smallest online redo log file size based on the current `FAST_START_MTTR_TARGET` setting and the corresponding statistics. The Redo Logfile Size Advisor is enabled only if `FAST_START_MTTR_TARGET` is set.

The advisor provides a recommended optimal smallest online redo log file size. You can then adjust the online redo log file size to the recommended optimal size.

Using the Database Control home page, you can obtain redo log file sizing advice by accessing the Administration page. Click the Redo Log Groups link in the Storage section. On the Redo Log Groups page, select the redo log group for which you want size advice, and then select the “Sizing advice” option from the Actions drop-down list. Click Go to obtain the advice.

The `OPTIMAL_LOGFILE_SIZE` column in `V$INSTANCE_RECOVERY` shows the redo log file size (in megabytes) that is considered to be optimal on the basis of the current `FAST_START_MTTR_TARGET` setting. It is recommended that you set all online redo log files to at least this value.

Note: An online redo log file size is considered optimal if it does not drive incremental checkpointing more aggressively than needed by `FAST_START_MTTR_TARGET`.

Impact of the Checkpoint Rate

From the V$ views:

```
SELECT c.value-nc.value
FROM V$SYSSTAT c, V$SYSSTAT nc
WHERE  c.name = 'physical writes' AND
  nc.name = 'physical writes non checkpoint';
```

From the Statspack report:

```
Statistic                                   Total
-------------------------- ------------------
physical writes                            47,308
physical writes non checkpoint             44,674
```


Impact of the Checkpoint Rate

When any of the checkpoint parameters is set, the number of writes that DBW*n* makes can increase. The smaller the setting, the greater is the number of writes. However, if the I/O system is not saturated, there will be little impact on performance. If DBW*n* is writing excessively, then LGWR may have to write more frequently as well. The Statspack and AWR reports and the `V$MTTR_TARGET_ADVICE` view show information about the extra writes generated due to the setting.

You can also estimate the number of writes that the system is doing purely for checkpointing reasons by subtracting the `physical writes` statistic from the `physical writes non checkpoint` statistic. `physical writes` is the actual number of writes being done, and `physical writes non checkpoint` is the theoretical number of writes had there been no checkpointing. The difference represents the writes caused by checkpointing.

The following two examples show the difference when the same workload is run. The only difference is that the `FAST_START_MTTR_TARGET` parameter is set to 25 seconds in the second example. Notice that the total number of physical writes per second is lower when the MTTR parameter is not set.

Impact of the Checkpoint Rate (continued)

```
Statistic                                     Total     per Second    per Trans
--------------------------------- ----------------- -------------- ------------
MTTR not set
physical writes                              47,308           26.4         23.0
physical writes non checkpoint               44,674           24.9         21.7

MTTR set
physical writes                              39,969           33.5         29.9
physical writes non checkpoint               27,738           23.2         20.8
```

The Statspack and AWR reports have sections that show the recovery time and the performance impact of the checkpoint parameters. This section of the report is based on the `V$INSTANCE_RECOVERY` view.

```
Instance Recovery Stats  DB/Inst: ORCL/orcl  Snaps: 187-197
-> B: Begin snapshot,  E: End snapshot

  Targt Estd                                 Log File  Log Ckpt    Log Ckpt
  MTTR  MTTR   Recovery   Actual    Target      Size     Timeout    Interval
   (s)   (s)   Estd IOs  Redo Blks Redo Blks Redo Blks  Redo Blks  Redo Blks
- ----- ----- ---------- --------- --------- ---------- --------- ------------
B    25    17        283        27     68876     405450     68876
E    25    20       1561     39418    368640     368640   1084632
```

In this example, `FAST_START_MTTR_TARGET` is set to 25 seconds and the other checkpoint parameters are not set. The checkpoint progress is keeping the estimated MTTR between 17 and 20 seconds. The last three columns show the number of blocks that were written to satisfy the various parameters. Log File Size Redo Blks shows the number of blocks written on the basis of log file size. Log Ckpt Timeout Redo Blks shows the number of blocks written due to the `LOG_CHECKPOINT_TIMEOUT` setting, and Log Ckpt Interval Redo Blks lists the blocks written due to the `LOG_CHECKPOINT_INTERVAL` setting.

Note the differences in the example below; the same workload is run but none of the recovery parameters are set. Checkpoints are driven entirely by the size of the redo log files.

```
Instance Recovery Stats  DB/Inst: ORCL/orcl  Snaps: 247-257
-> B: Begin snapshot,  E: End snapshot

  Targt Estd                                 Log File  Log Ckpt    Log Ckpt
  MTTR  MTTR   Recovery   Actual    Target      Size     Timeout    Interval
   (s)   (s)   Estd IOs  Redo Blks Redo Blks Redo Blks  Redo Blks  Redo Blks
- ----- ----- ---------- --------- --------- ---------- --------- ------------
B     0    17        597      8190    368640     368640
E     0    56       8911    319237    368640     368640   1114844
```

Automatic Checkpoint Tuning

- **There is no longer a continuous manual tuning effort.**
- **Automatic checkpoint tuning is the best-effort checkpointing, without much overhead.**
- **It reduces average recovery time by making use of unused bandwidth.**
- **Automatic checkpoint tuning is enabled when `FAST_START_MTTR_TARGET` is not explicitly set to zero.**

Automatic Checkpoint Tuning

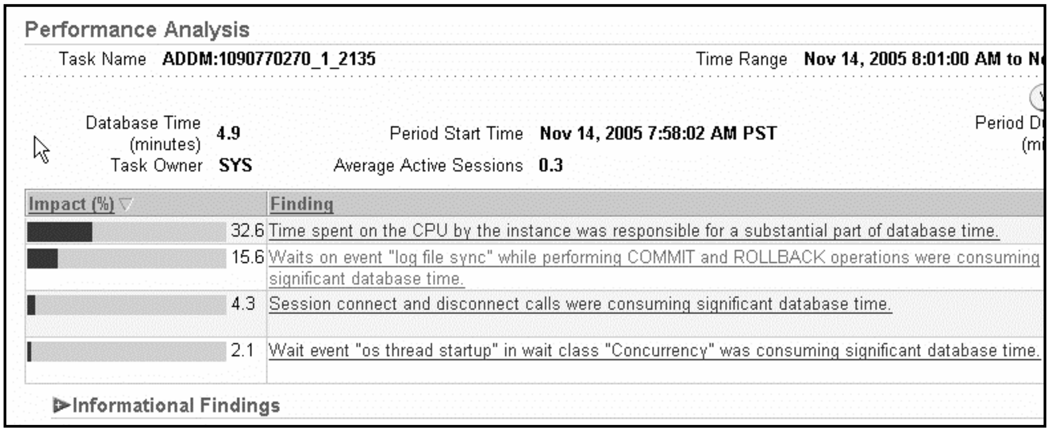
The automatic checkpoint tuning included with Oracle Database 10*g* greatly reduces the need to set any of the checkpoint parameters. These parameters can still be used to set an upper bound on recovery time. In earlier versions, if you had large log files and large buffer caches so that very few buffers were written for aging purposes, then you would set parameters such as `FAST_START_MTTR_TARGET` to reduce crash recovery times.

Although Oracle9*i* Database Release 2 introduced the MTTR (mean-time-to-recover) advisor, it was not easy to set the right target for `FAST_START_MTTR_TARGET`. There was always a trade-off between the small recovery time and run-time physical I/Os.

By default, Oracle Database 10*g* supports automatic checkpoint tuning by making the best effort to write out dirty buffers without an adverse impact on the throughput. Thus, a reasonable crash recovery time can be achieved even if you do not set `FAST_START_MTTR_TARGET` or if you set it to a very large value.

You enable automatic checkpoint tuning by setting the `FAST_START_MTTR_TARGET` parameter to a nonzero value. You can also enable this feature by not setting the `FAST_START_MTTR_TARGET` parameter. However, you disable it by setting the `FAST_START_MTTR_TARGET` parameter to zero.

ADDM Report: Checkpoints

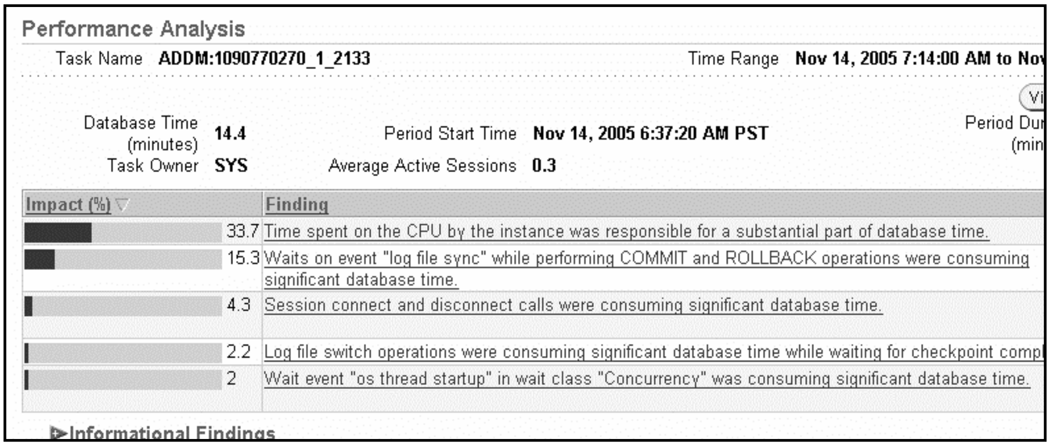


ORACLE

ADDM Report: Checkpoints

The ADDM report shown in the slide is produced from an instance that has only a checkpoint issue. Notice that the waits other than CPU are on `log file sync`. Further investigation is required.

ADDM Report: Redo Logs



ADDM Report: Redo Logs

This ADDM report is created on an instance with only redo log issues. It shows `log file sync` waits just as the report on checkpoint issues did. This report also shows `log file switch(checkpoint complete)` waits. This wait event may lead you in the wrong direction. For both checkpoint and redo log issues, further investigation is required. Use the AWR report, Statspack report, or `V$` views to continue.

Statspack and AWR Reports

Checkpoint and redo show certain symptoms:

- **Alert log shows log switch not complete**
- **I/O symptoms caused by excessive checkpoints**
- **Log switches per hour > 4**
- **`log file` and `latch:redo` in Top Timed Events**

```
Top 5 Timed Events                                      Avg %Total
~~~~~~~~~~~~~~~~~~                                     wait   Call
Event                          Waits    Time (s)   (ms)   Time
----------------------- ------- ----------- ------ ------
CPU time                                     551          56.8
log file parallel write   3,899          201     52   20.7
log file sync               823           58     71    6.0
latch: redo copy            635           44     70    4.6
latch: redo allocation    1,109           42     38    4.4
```


Statspack and AWR Reports

Checkpoint and redo issues show similar symptoms. The Statspack and AWR reports could show any of the following wait events:

- `log file parallel write`: Writing redo records to the redo log files from the log buffer. Even though redo records are written in parallel, the parallel write is not complete until the last I/O is on disk.
- `log file sync`: Waiting on the flush of the redo information of the session to the redo log file on user commit. The user session will post the LGWR to write the log buffer to the redo log file. When the LGWR has finished writing, it will post the user session.
- `log file switch/archive`: Waiting for a log switch because the log that the LGWR will be switching into has not been archived yet. Check the alert file to make sure that archiving has not stopped due to a failed archive write. To speed up archiving, consider adding more archive processes or putting the archive files on striped disks.
- `latch: redo copy`: Waiting on the redo copy latch. The redo copy latch is required by the server processes to copy redo data to the log buffer, indicating that a large number of processes are attempting to write to the log buffer simultaneously.

Statspack and AWR Reports (continued)

- `latch: redo allocation`: Waiting on redo allocation latch. The allocation latch is required to allocate memory in the log buffer to write redo data. This indicates that several processes are attempting to write to the log buffer simultaneously.

The latch waits are more likely due to a redo problem.

Small redo logs will also show a large number of log switches.

```
Instance Activity Stats  DB/Inst: ORCL/orcl  Snaps: 137-138
-> Statistics identified by '(derived)' come from sources other than
SYSSTAT

Statistic                                     Total  per Hour
-------------------------------- ------------------ ---------
log switches (derived)                            7     15.41
```

Check Parameters

Review checkpoint parameters for reasonable values:

```
Parameter Name                 Begin value
---------------------------- -----------
fast_start_mttr_target         25
```

Use `V$MTTR_TARGET_ADVICE` for optimum value:

```
  SQL> SELECT mttr_target_for_estimate,
   2>  estd_total_ios, estd_total_io_factor
   3>  FROM V$MTTR_TARGET_ADVICE ORDER BY 1

MTTR_TARGET_FOR_ESTIMATE ESTD_TOTAL_IOS ESTD_TOTAL_IO_FACTOR
------------------------ -------------- --------------------
                      20        2436690               1.0739
                      22        2330674               1.0272
                      25        2268973                    1
                      37        2204817                .9717
                      62        2181841                .9616
```


Check Parameters

Review the values of the checkpoint parameters:

`FAST_START_MTTR_TARGET` should be null or very large for best performance. This parameter is set to enforce an upper bound for the number of seconds required for instance recovery. When this parameter is set to a very small value as shown in the slide, DBWR must write continuously.

Use the `V$MTTR_TARGET_ADVICE` view. This view is populated whenever the `FAST_START_MTTR_TARGET` parameter is set. The `ESTD_TOTAL_IOS` and `ESTD_TOTAL_IO_FACTOR` columns indicate the differences in I/O at various target settings. In this example, a value of 62 seconds would reduce the I/O but increase the recovery time only by 37 seconds.

Check the Redo Log Size

Review the current size of the redo log files:

Select	Group	Status	# of Members	Archived	Size (KB)	Sequence	First Change#
◉	4	Active	1	No	102400	1406	16028586
○	5	Current	1	No	102400	1407	16034897
○	6	Inactive	1	No	102400	1405	16022132

Check alert log for log switch rate:

```
Thu Nov 17 06:45:38 2005
Thread 1 advanced to log sequence 1403
  Current log# 4 seq# 1403 mem# 0: /u01/app/oracle/oradata/orcl/redo4.log
Thu Nov 17 06:48:36 2005
Thread 1 advanced to log sequence 1404
  Current log# 5 seq# 1404 mem# 0: /u01/app/oracle/oradata/orcl/redo5.log
Thu Nov 17 06:51:37 2005
Thread 1 advanced to log sequence 1405
  Current log# 6 seq# 1405 mem# 0: /u01/app/oracle/oradata/orcl/redo6.log
```


Check the Redo Log File Size

The redo log file size drives the checkpoint if none of the checkpoint parameters are set. The Redo Logfile Size Advisor does not give a recommendation unless the `FAST_START_MTTR_TARGET` parameter is set. An alternative way to check log file size is shown in the slide.

A reasonable size for the redo log file varies with the application and database configuration.

For a single-instance OLTP database, the time between log switches should be 20 minutes or more. In the example, the log switches are occurring every three minutes and the log files have a size of 100 MB. Based on this information, log files should be 600 MB to 700 MB (100 / 3 × 20). If the log switch rate shown in the slide is not frequent or sustained, then the log file size of 100 MB is adequate. A view of the number of log switches per hour is more appropriate.

More frequent log switches are recommended for databases using Data Guard and Streams.

Redo Log Chain Tuning

Redo tuning starts with the slowest part.

- **Reduce the amount of redo generated.**
- **Check archive logging (waits for archiving needed).**
- **Check the redo log file size and log switch rate.**
- **Check the checkpoint parameters.**
- **Look for log space requests.**
- **The Redo Buffer Allocation Retries value should be near 0 and should be less than 1% of redo entries.**

Redo Log Chain Tuning

The redo log records move from the log buffer in memory to the redo log files and then to the archive log files. If any of these transfers are delayed, the issue can be reflected back through the chain to the redo performance. `Log space request` means that the server processes have to wait on LGWR to make space in the log buffer. LGWR can be delayed by: performing frequent log switches, waiting on checkpoints to complete, or waiting on ARC*n* processes to copy the log file to archive log file. While LGWR is waiting, it does not clear space in the log buffer. Very frequent log switches caused by small log files can be the root of `log space request` waits. Slow ARC*n* processes may be caused by disk contention, slow disks, or too few ARC*n* processes.

If the `SECONDS_IN_WAIT` value for the `log buffer space` event in `V$SESSION_WAIT` indicates some time spent waiting for space in the redo log buffer, and the redo chain has no other bottlenecks, then consider:

- Increasing size of the log buffer

Redo Log Chain Tuning (continued)

- Moving the log files to faster disks, such as striped disks, by using RAID level 0:

```
SQL> SELECT sid, event, seconds_in_wait, state
  2  FROM V$SESSION_WAIT
  3  WHERE event = 'log buffer space%';
 SID     EVENT                SECONDS_IN_WAIT    STATE
----     -----------------    ---------------    -------
   5     log buffer space                 110    WAITING
```

Reducing Redo Operations

Ways to avoid logging bulk operations in the redo log:

- Direct Path loading without archiving does not generate redo.
- Direct Path loading with archiving can use `NOLOGGING` mode.
- Direct Load `INSERT` can use `NOLOGGING` mode.
- Some SQL statements can use `NOLOGGING` mode.

ORACLE

Reducing Redo Operations

SQL*Loader and NOLOGGING Mode

Conventional path loading generates redo log entries just as any DML statement. When using direct path, redo log entries are not generated if:

- The database is in `NOARCHIVELOG` mode
- The database is in `ARCHIVELOG` mode, but logging is disabled. Logging can be disabled by setting the `NOLOGGING` attribute for the table or by using the `UNRECOVERABLE` clause in the control file.

Direct Load INSERT and NOLOGGING Mode

The `NOLOGGING` option:

- Applies to tables, tablespaces, and indexes
- Does not record changes to data in the redo log buffer. Some minimal logging is still carried out for operations such as extent allocation.
- Is not specified as an attribute at the `INSERT` statement level, but is instead specified when using the `ALTER` or `CREATE` command for the table, index, or tablespace
- If `NOLOGGING` is set at the tablespace level, it specifies that `NOLOGGING` is the default option for new objects created in the tablespace, but it does not affect the `NOLOGGING` capabilities of objects already created in the tablespace.

Reducing Redo Operations (continued)

Direct Load `INSERT` and `NOLOGGING` Mode (continued)

- Is set before the load and is reset to `LOGGING` after the load completes. If a media failure occurs before a backup is taken, then all tables and indexes that have been modified may be corrupted.

SQL Statements That Can Use `NOLOGGING` Mode

Although you can set the `NOLOGGING` attribute for a table, index, or tablespace, `NOLOGGING` mode applies only to a few operations on the object for which the attribute is set, such as:

- `CREATE TABLE ... AS SELECT`
- `CREATE INDEX`
- `ALTER INDEX ... REBUILD`
- `DIRECT PATH INSERT`

The following statements are unaffected by the `NOLOGGING` attribute: `UPDATE`, `DELETE`, conventional path `INSERT`, and various DDL statements not listed above.

Note: For backward compatibility, `UNRECOVERABLE` is still supported as an alternate keyword with the `CREATE TABLE` statement. This alternate keyword may not be supported in future releases.

Increasing the Performance of Archiving

- **Share the archiving work during a temporary increase in workload:**

```
ALTER SYSTEM ARCHIVE LOG ALL
TO <log_archive_dest>
```

- **Increase the number of archiver processes with `LOG_ARCHIVE_MAX_PROCESSES`.**
- **Multiplex the redo log files, and add more members.**
- **Change the number of archive destinations:**
  - `LOG_ARCHIVE_DEST_n`

Increasing the Performance of Archiving

Share the Archiving Workload Temporarily

If you anticipate a heavy workload for archiving, you can add another process to share the work by executing the `ALTER SYSTEM ARCHIVE LOG ALL TO 'directory_name'` command. This command causes the server process to behave as an archiver until all redo log files have been archived. These processes are not affected by `LOG_ARCHIVE_MAX_PROCESSES`.

Increase the Number of Archiver Processes

Occasionally, on busy databases, a single ARC*n* process cannot keep up with the volume of information written to the redo logs. With the Oracle database, you can define multiple archiver processes by using the `LOG_ARCHIVE_MAX_PROCESSES` parameter. This parameter sets the maximum number of ARC*n* processes that can be started.

The LGWR process starts a new ARC*n* process whenever the current number of ARC*n* processes is insufficient to handle the workload.

Increasing the Performance of Archiving (continued)

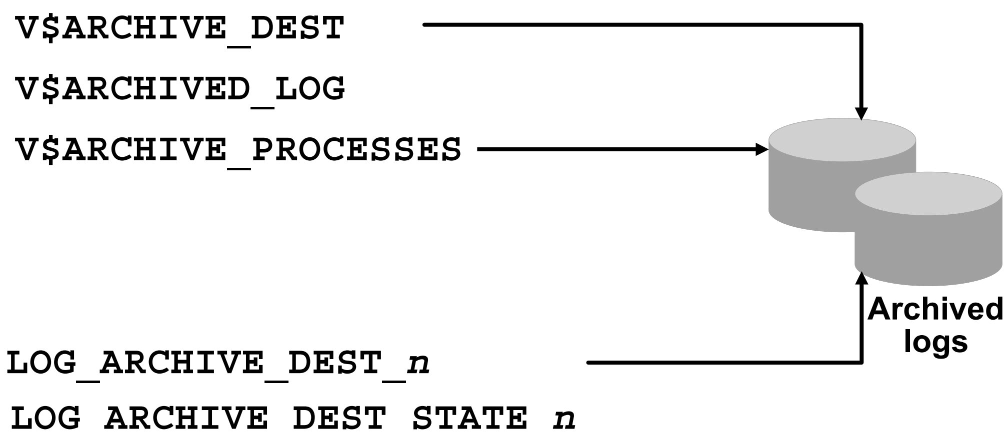
Multiplex Redo Log Files

If you are archiving, it is even more important to have more than two redo log groups with multiple members. When logging switches to another group, the DBW*n* process must perform checkpointing as usual, and one file must be archived. You must allow time for both of these operations before the LGWR process needs to overwrite the file again by having sufficient groups. Take advantage of the ARC*n* behavior of reading a block from each member of the redo log group to spread the I/O across multiple disks by adding more members to the redo log groups.

Decrease the Number of Archive Destinations

This is not often a possible choice. More archive destinations increase the overhead of archiving. An alternative to several archive destinations could be to use a fetch archive log server to make additional archive log file copies on another machine.

Diagnostic Tools



Diagnostic Tools

Obtaining Information About Archived Log Files and Their Locations

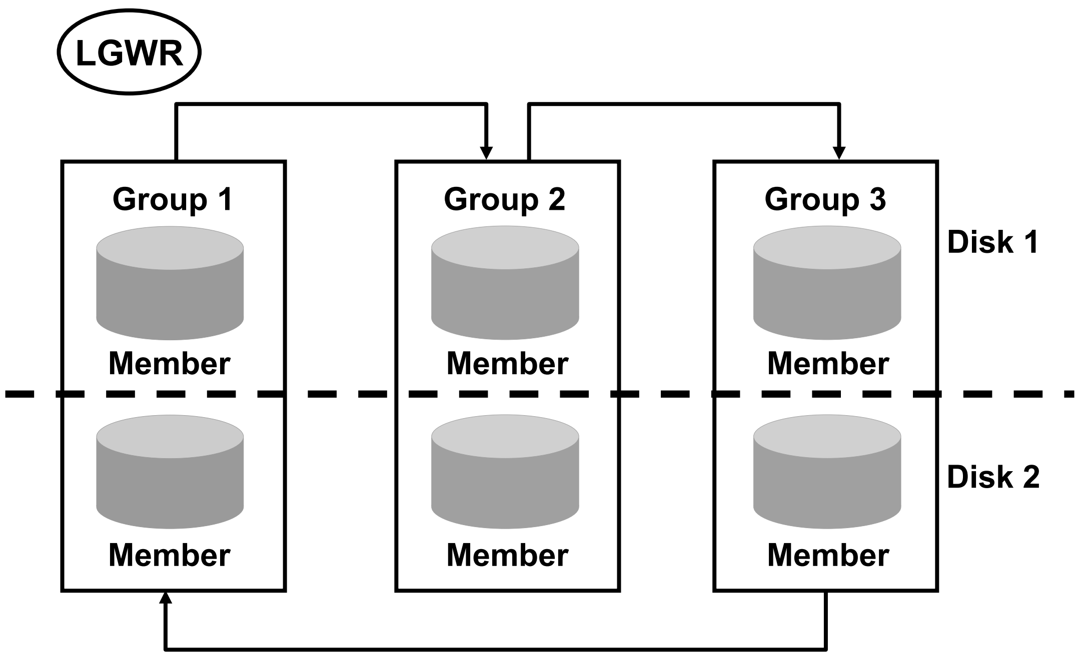
Query the V$ARCHIVED_LOG dynamic performance view to display archived log information from the control file, including archive log names. An archive log record is inserted after the online redo log is successfully archived or cleared (the NAME column is null if the log was cleared).

The V$ARCHIVE_DEST dynamic performance view describes, for the current instance, all the archive log destinations, as well as their current values, modes, and statuses.

The V$ARCHIVE_PROCESSES view has one row for each archive process. The STATUS column shows the state of the archive process (STOPPED, SCHEDULED, STARTING, ACTIVE, STOPPING, or TERMINATED). The STATE column indicates whether the process is currently BUSY or IDLE. The number of archiver processes used by the database is automatically set to 4 when the DBWR_IO_SLAVES parameter is set to a value greater than 0.

Note: The LOG_ARCHIVE_DEST_*n* parameter is valid only if you have installed the Oracle Database Enterprise Edition. However, LOG_ARCHIVE_DEST_*n* and LOG_ARCHIVE_DEST are incompatible and cannot be used together.

Redo Log Groups and Members



Redo Log Groups and Members

Online redo log files are organized in groups. A group must have one or more members. All members of a group have identical contents. You should have two or more members in each group for safety, unless you are mirroring all files at a hardware level.

The diagram in the slide shows one method of assigning redo log members to disk space. This method supports redo logging adequately for recoverability. If archiving is enabled, you may need to have additional members on different disks so that archiving I/O does not contend with the redo log writer.

Online Redo Log File Configuration

- **Size redo log files to minimize contention.**
- **Provide enough groups to prevent waiting.**
- **Store redo log files on separate, fast devices.**
- **Monitor the redo log file configuration with:**
  - **`V$LOGFILE`**
  - **`V$LOG`**
  - **`V$LOG_HISTORY`**

Online Redo Log File Configuration

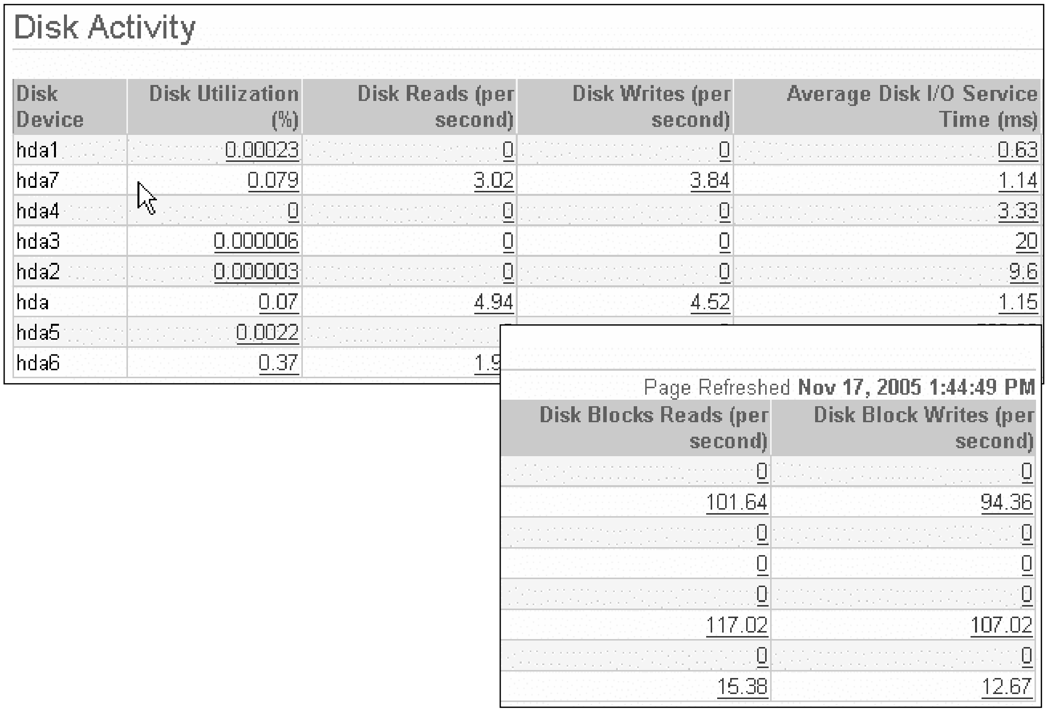
Redo log files in the same group should ideally be on separate, fast devices because LGWR writes to them almost continuously. Size redo log files properly to minimize contention and frequency of log switches. As a guide, a redo log file should be able to contain 20 minutes of redo data. You can use the redo size statistics and log file switch statistics in the Statspack report to help determine the proper size for your redo log files.

Monitoring Redo Log File Information

You can query the `V$LOGFILE` and `V$LOG` dynamic performance views to obtain information about the name, location, size, and status of the online redo log files.

Any waits for Log File Parallel Write in `V$SYSTEM_EVENT` indicate a possible I/O problem with the log files.

Monitoring Online Redo Log File I/O



Disk Activity

Disk Device	Disk Utilization (%)	Disk Reads (per second)	Disk Writes (per second)	Average Disk I/O Service Time (ms)
hda1	0.00023	0	0	0.63
hda7	0.079	3.02	3.84	1.14
hda4	0	0	0	3.33
hda3	0.000006	0	0	20
hda2	0.000003	0	0	9.6
hda	0.07	4.94	4.52	1.15
hda5	0.0022			
hda6	0.37	1.9		

Page Refreshed **Nov 17, 2005 1:44:49 PM**

Disk Blocks Reads (per second)	Disk Block Writes (per second)
0	0
101.64	94.36
0	0
0	0
0	0
117.02	107.02
0	0
15.38	12.67

ORACLE

Monitoring Online Redo Log File I/O

Oracle Enterprise Manager 10*g* provides a means of monitoring disk I/Os, so you do not have to use operating system disk-monitoring commands. The output is similar to `sar` (system activity reporter) or `iostat` on UNIX systems. If the redo logs and archive logs are on separate drives, this tool allows you to monitor the I/O for these files. Disk statistics are stored in the AWR so that you can view data occurring in the recent past.

Sizing the Redo Log Buffer

The size of the redo log buffer is determined by:

- **`LOG_BUFFER` parameter**
- **Remaining space in the fixed area granule**

Default value: Either 512 KB or 128 KB × the value of `CPU_COUNT`, whichever is greater

Sizing the Redo Log Buffer

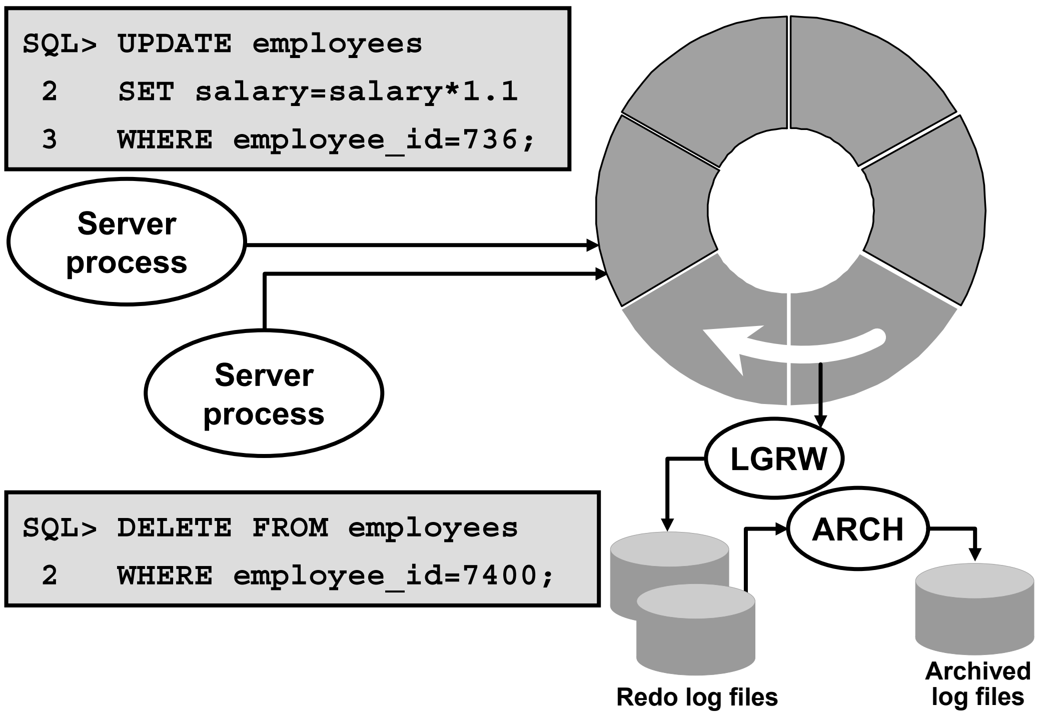
If transactions are long or numerous, then larger values of `LOG_BUFFER` reduce log file I/O. The smaller the buffer is, the more often it will get to be one-third full before a `COMMIT` operation clears the buffer.

The default value of `LOG_BUFFER` is 512 KB or 128 KB * `CPU_COUNT`. But the actual allocation will include the remainder of the granule allocated to the Fixed Area of the SGA. The log buffer size may be different than the `LOG_BUFFER` parameter value. The `V$SGASTAT` view displays the actual log buffer size.

Example:

```
SELECT 'v$parameter' "View name", name,
    to_number (value,'999999999') "Value"
    FROM V$PARAMETER   WHERE name = 'log_buffer'
    UNION
    SELECT 'v$sgastat' "View name", name, bytes
    FROM V$SGASTAT     WHERE name = 'log_buffer';

View name       NAME                Value
------------    -----------      --------
v$parameter     log_buffer       23576576
v$sgastat       log_buffer       23945216
```

Diagnosing Redo Log Buffer Inefficiency



Diagnosing Problems

On machines with fast processors and relatively slow disks, the processors may be filling the rest of the redo log buffer in the time it takes the LGWR process to move a portion of the buffer out to disk.

For this reason, a larger buffer makes it less likely that new entries will collide with the part of the buffer still being written. However, if the amount of redo being generated is greater than the rate at which it is able to be copied out, then no matter how big the redo buffer, it will finally fill up. In such cases, you must either ensure that enough redo log buffer space exists to see the system through to the next “quiet” time or speed up the copying process. The server processes typically write in bursts, and LWGR writes whenever it is called and continues to write until there is sufficient space.

Server processes may request space from the redo log buffer to write new entries and not find any. They must wait for LGWR to flush the buffer to disk.

Tuning Goal

Tuning the redo log buffer means ensuring that the space required by server processes in the redo log buffer is sufficient. However, too much space reduces the amount of memory that can be allocated to other areas. It is also important to note that the DBA can adopt practices that reduce the amount of redo that must be performed.

Diagnosing Log Buffer Problems

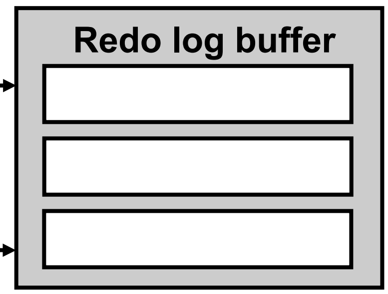
V$SESSION_WAIT

Log Buffer Space event

V$SYSSTAT

Redo Buffer Allocation Retries

Redo Entries



Diagnosing Log Buffer Problems

When the `V$SESSION_WAIT` view indicates Log Buffer Space events, there are waits for space in the log buffer because the session is writing data into the log buffer faster than LGWR can write it out.

```
SQL> SELECT sid, event, seconds_in_wait, state
  2  FROM V$SESSION_WAIT
  3  WHERE event = 'log buffer space%';

SID  EVENT                SECONDS_IN_WAIT   STATE
---  ----------------     ---------------   ---------
15   log buffer space                 110   WAITING
```

Redo Buffer Allocation Retries Statistic Ratio

The value of Redo Buffer Allocation Retries should be near 0. This number should not be greater than 1% of the Redo Entries. If this value increases consistently, then processes have had to wait for space in the buffer.

Diagnosing Log Buffer Problems (continued)

Redo Buffer Allocation Retries Statistic Ratio (continued)

```
SQL> SELECT r.value "Retries", e.value "Entries",
  2  r.value/e.value*100 "Percentage"
  3  FROM V$SYSSTAT r, V$SYSSTAT e
  4  WHERE r.name = 'redo buffer allocation retries'
  5  AND e.name='redo entries';

   Retries    Entries   Percentage
---------- ----------   ----------
         0        189            0
```

The wait may be caused by the log buffer being too small, by checkpointing, or by archiving. In any case, you should:

- Solve the checkpoint, archiving, or log file size issues first
- Increase the size of the redo log buffer, if necessary, by changing the value of the `LOG_BUFFER` initialization parameter

The redo log buffer is normally small and the default values are often adequate. The important events to consider are the following:

- The Log Buffer Space event occurs when processes are waiting for space in the redo log buffer because the log switch does not occur quickly enough. The `SECONDS_IN_WAIT` value of the Log Buffer Space event indicates the time spent. This is an indication that the buffers are being filled up faster than LGWR is writing. Disk I/O contention on the redo log files could cause this wait.
- The Redo Buffer Allocation Retries statistic in the `V$SYSSTAT` view reflects the number of times a user process waits for space in the redo log buffer to copy new entries over the entries that have been written to disk. LGWR normally writes fast enough to ensure that space is always available in the buffer for new entries, even when access to the redo log is heavy.

```
SQL> SELECT name, value
  2  FROM   V$SYSSTAT
  3  WHERE  name = 'redo buffer allocation retries';
```

- The Redo Entries statistic is the number of times a redo record is written to the redo log buffer. This value is compared to Redo Buffer Allocation Retries to determine the significance of the retries.

In Oracle Database 10*g* Release 2, these events are not expected to be common problems. High-volume data changes that produce large volumes of redo possibly cause these problems. When they occur, tune the I/O to the redo log files or the archive log files first.

Note: The `V$SYSSTAT` view displays another statistic, Redo Log Space Requests:

```
SQL> SELECT name, value
  2  FROM V$SYSSTAT
  3  WHERE  name='redo log space requests';
```

This statistic indicates that the active log file is full and that the Oracle server is waiting for archive log space to be allocated.

Log Space Request Waits: Further Investigation

Possible reasons for log space request waits:

- **There is disk I/O contention on redo log files.**
- **LGWR is waiting on DBW*n* to complete the checkpointing of the required redo log file.**
- **LGWR is waiting on ARC*n* to complete archiving of the required redo log file.**
- **Log buffer is too small.**

Log Space Request Waits: Further Investigation

When log buffer space requests are a significant wait event, further investigation is needed to discover the reason LGWR is not freeing the log buffer space quickly enough for the server processes to allocate the needed space. Increasing the size of the log buffer is not effective until all the other causes are eliminated.

Possible Reasons for Log Space Request Waits

- There is disk I/O contention on the redo log files. Check that the redo log files are stored on separate, fast devices.
  - In the `V$SYSTEM_EVENT` view, check the number of occurrences of the Log File Switch Completion event, which identifies the log file switch waits because of log switches.

```
SQL> SELECT event, total_waits, time_waited, average_wait
  2  FROM V$SYSTEM_EVENT
  3  WHERE event like 'log file switch completion%';
```

  - Increase the size of the redo log files.
- DBW*n* has not completed checkpointing the file when the LGWR needs the file again. LGWR must wait.
  - In the alert log file, look for this message: `CHECKPOINT NOT COMPLETE`.

Log Space Request Waits: Further Investigation (continued)

Possible Reasons for Log Space Request Waits (continued)

- In the V$SYSTEM_EVENT view, check the number of occurrences of the Log File Switch (Checkpoint Incomplete) event, which identifies the log file switch waits because of incomplete checkpoints.

```
SQL> SELECT event, total_waits, time_waited, average_wait
  2  FROM V$SYSTEM_EVENT
  3  WHERE event like 'log file switch (check%';
```

- Check the frequency of checkpoints and set an appropriate value for FAST_START_MTTR_TARGET.
- Check the size and number of redo log groups.

• The archiver process cannot write to the archived redo log files or cannot complete the archive operation fast enough. Therefore, it prevents the LGWR from writing.
  - Confirm that the archive device is not full.
  - Add redo log groups.
  - In the V$SYSTEM_EVENT view, check the number of occurrences of the Log File Switch (Archiving Needed) event, which identifies the log file switch waits because of the archiving issue.

```
SQL> SELECT event, total_waits, time_waited, average_wait
  2  FROM V$SYSTEM_EVENT
  3  WHERE event like 'log file switch (arch%';
```

  - Increase the number of archiver processes. The LGWR process starts a new ARC*n* process whenever the current number of ARC*n* processes is insufficient to handle the workload. Increase the maximum number of archiver processes with the LOG_ARCHIVE_MAX_PROCESSES initialization parameter. This parameter is dynamic and can be changed using the ALTER SYSTEM statement.

• Increase the size of the log buffer. Changing this parameter requires the database to be shut down and restarted. Increasing the log buffer size often requires an additional granule of memory to be added to the SGA. A larger log buffer allows LWGR more time to complete file operations before the log buffer fills, and LGWR can make larger, more efficient writes. In systems with high-volume online transaction processing (OLTP) or bulk loading of data, a larger log buffer can significantly improve performance.

Summary

In this lesson, you should have learned how to:

- Diagnose checkpoint and redo issues
- Implement Fast Start MTTR Target
- Monitor the performance impact of Fast Start MTTR Target
- Implement multiple database writers
- Tune the redo chain
- Size the redo log file
- Size the redo log buffer

Practice 11 Overview: Diagnosing Checkpoints and Redo

This practice covers the following topics:

- Diagnosing checkpoint and redo issues
- Resizing log files
- Adjusting the checkpoint parameters

Tuning I/O

Objectives

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

- Diagnose database I/O issues
- Describe the Stripe And Mirror Everything (SAME) concept
- Explain the benefits of asynchronous I/O
- Choose appropriate I/O solutions
- Tune I/O using Automatic Storage Management (ASM)

ORACLE

I/O Architecture

Oracle Database 10*g* includes three standard storage options:

- **File system**
- **Raw partitions**
- **Automatic Storage Management (ASM)**

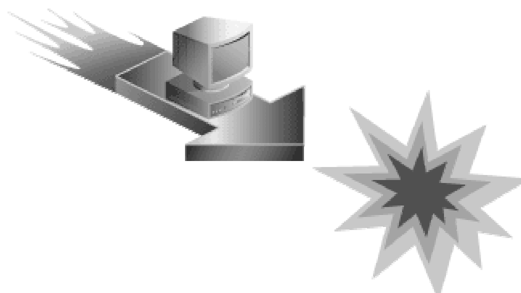
I/O Architecture

The Oracle database uses a logical storage container called a tablespace to store all permanent data. The tablespace is associated with one or more data files. Data files are the physical storage entities. A data file is mapped to an operating system (OS) file, a raw partition, or an ASM file. Each of these methods has certain advantages. OS files provide easy naming and administration, but standard file system access requires the use of the OS buffer cache, which has a longer code path and requires additional copies of the data blocks to be made. Raw partitions have been the measurement standard for performance. Raw partitions do not use the OS buffer cache, are usually accessed by asynchronous I/O, and allow larger reads and writes. However, raw partitions are more difficult to manage. The Automatic Storage Management (ASM) feature allows much of the convenience of OS files and the speed of raw partitions.

File System Characteristics

Certain characteristics are better for database use:

- Write-through-cache ability
- Write acknowledgement
- Security
- Journaling
- High performance



File System Characteristics

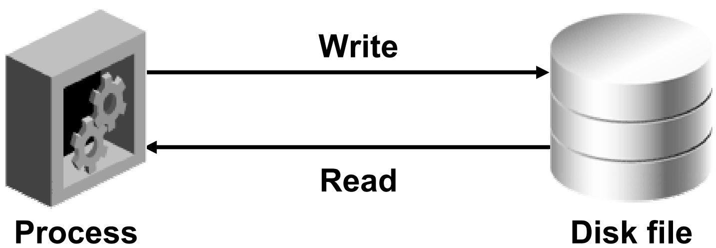
In choosing a file system, performance is important, but not the primary consideration. It does not matter how fast the file system is if the data is corrupted, lost, or compromised. The Oracle database does not support database files on file systems that do not have a write-through-cache capability. Supported file systems must acknowledge writes. For example, the standard Network File System (NFS) uses User Datagram Protocol (UDP). UDP does not include an acknowledgment. One vendor that supplies a supported NFS file system is Network Appliance using a modified NFS protocol. The Oracle database has security requirements as well. Database files require file permissions that are not available on certain file systems (such as VFAT, or Virtual File Allocation Table). When file permissions are not set correctly, the Oracle server does not function properly. Data files should be accessible only to the database owner, and all other access should be controlled by the database itself. Other file systems may have characteristics that reduce performance.

Journaling is a popular characteristic. The major benefit is that the changes to the file system are recorded in a journal file similar to database redo logs. If the server crashes or shuts down without synchronizing the disks, then the journal file can be applied to the file system, thus restoring the file system integrity very quickly. This effect can be seen on bootup. The `fsck` command checks journaled file systems more quickly than non-journaled file systems.

Raw Partitions

Raw partitions have been considered to be the high-performance solution.

- **Raw reads and writes do not use the OS buffer cache.**
- **Raw reads and writes can move larger buffers than file system I/Os.**
- **Using raw partitions complicates administration.**



Raw Partitions

Raw partitions have long been considered as the ultimate in I/O performance. When new file systems are benchmarked, they are frequently compared to raw partitions. The main benefit of raw partitions is the lack of caching. Oracle server caches its own database blocks. For the normal reads in an online transaction processing (OLTP) environment, the simple read ahead that is done by the file system is not effective. By avoiding the OS cache, the raw reads and writes are not copied in memory, and the application performing the reads and writes can choose much larger buffers for the data transfers than are allowed by the file system.

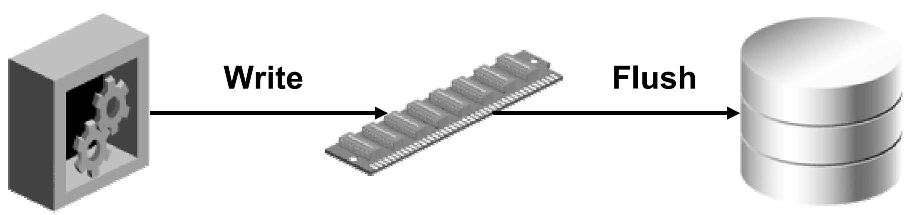
The penalties of using raw partitions are mostly administrative. Raw partitions can have only one data object on a partition. Therefore, each data file, control file, and redo log file needs its own raw partition. Because raw partitions do not have a file system, most OS file manipulation tools, such as `ls`, `cp`, `cpio`, and `mv` on UNIX-like systems, are not available.

Until recently, raw partitions were the only way to share disk storage between nodes of a cluster under most operating systems (VMS and Tru64 were exceptions to this rule). Because of this, you were usually required to use raw partitions for Oracle Real Application Clusters. Oracle Cluster File System (OCFS) and ASM can now be used for Oracle Real Application Clusters instead of raw partitions.

I/O Modes

I/O can be written to disk in several ways by using different system calls:

- **Standard I/O**
- **Synchronous I/O**
- **Asynchronous I/O**



Process **Buffer cache** **Disk file**

I/O Modes

In any OS, there are multiple methods of writing to disk.

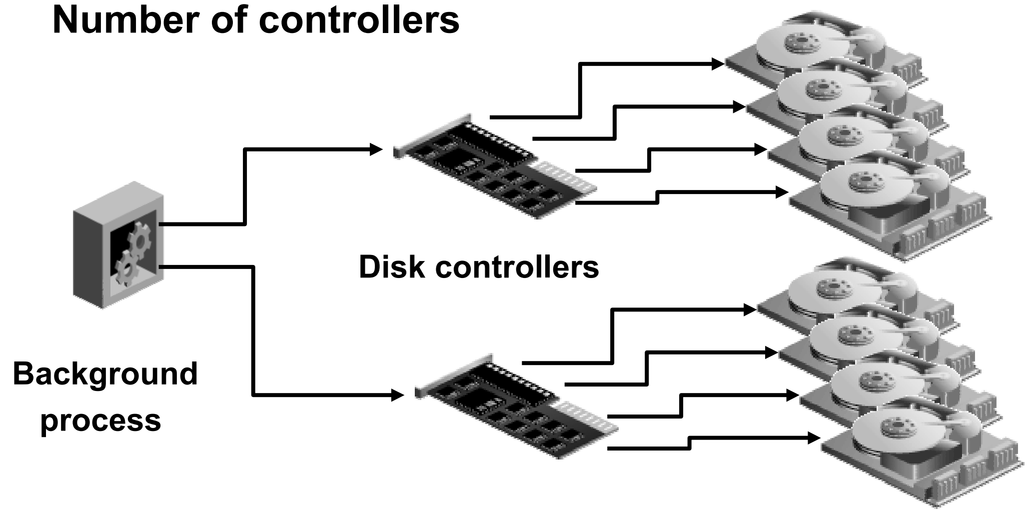
The standard method writes the data to the operating system buffer cache, and then the buffer is written to disk by a kernel process at a later time. If the machine crashes or loses power before the buffer is written to disk, the data is lost. This method is fast, but not acceptable for Oracle database file I/O because of the potential for data loss.

Synchronous I/O, also known as write-thru-cache, writes the data to the buffer but also forces an immediate write to the disk. The process that makes the write request is suspended until the write is completed. The completion is signaled by the disk controller when the buffer is written to disk. On some disk subsystems with large caches, the controller signals a completion when the buffer is written into the subsystem cache. Although synchronous writes are slower, they are the default method for DBWR and LGWR processes to perform their writes because this I/O method is reliable and supported on all operating systems.

Asynchronous writes allow instance background processes to make multiple write requests without being suspended after each request. The completion must still be signaled back to the process. This permits a much larger throughput because the background process can continue work without waiting on the disk I/O to complete. The ability to perform asynchronous writes varies by operating system.

Bandwidth Versus Size

I/O performance depends on bandwidth.
- Number of disks, not size
- Number of controllers



Bandwidth Versus Size

Many times, database disk subsystems are sized only by storage capacity. The main consideration for performance in the I/O subsystem is I/Os per second or bandwidth. More bandwidth becomes available as you increase the number of disks you spread the I/Os across. Each disk may have a transfer rate of 160 MB/sec; therefore, four disks will have a combined bandwidth of 640 MB/sec. The disk controllers play an important role as well because each disk controller has a limited bandwidth. An SCSI disk controller with a transfer rate of 600 MB/sec will become a bottleneck on disk transfers if there are more than 4–5 disks, even though the specifications say that the controller can manage up to 16 disks.

Stripe and Mirror Everything

- **All data files to access all available bandwidth**
- **All database files to be on the same logical devices**
- **Highest performance configuration**

Stripe and Mirror Everything

The problem is how to make use of all the bandwidth all the time. With file systems and raw partitions, data files are placed on particular mount points that map to a single disk or a logical volume. Manual striping by placing active tables and indexes in tablespaces that have multiple data files spread across several disks or volumes is time consuming and requires constant rebalancing. Oracle recommends the Stripe And Mirror Everything (SAME) approach. You form all the disks available for the database files into a logical volume set, and stripe all the data files across this set. This method means that redo log files, archive log files, data files, and control files have equal access to all the bandwidth that is available. If high availability is desired, use two volume sets and mirror them at the OS level.

There are three general methods available to implement the SAME methodology: use a vendor- or OS-supplied logical volume manager (LVM), use hardware-based RAID technology, or use Oracle-supplied Automatic Storage Management (ASM). LVM can be expensive, incompatible with certain Oracle database features such as Real Application Clusters, or limited in functionality.

Using RAID

Redundant Array of Inexpensive Devices (RAID) levels:

- **Level 0**
  - **Striped for performance**
  - **No redundancy**
- **Level 1**
  - **Mirrored for safety**
  - **Little performance benefit**
- **Level 5**
  - **Block level redundancy (rebuild algorithm)**
  - **Improved read performance**
  - **Additional write cost**

RAID Levels

RAID is a Redundant Array of Inexpensive Devices (Disks). The goal of RAID is to group together several inexpensive disks in such a way as to provide fast, reliable, and inexpensive storage. The goal is met in a variety of ways, each having its own associated cost.

RAID 0 is a set of disks with the data striped across all the members of the array. Typically, a RAID 0 array has 2–10 devices. The more devices in the array, the better the performance. The stripe size affects performance and should be relatively small (64 KB–2 MB), so that the data for a single read or write is spread across several disks.

RAID 1 is a very reliable method creating “mirrors” of the data on each disk in the array. The data is duplicated on each member or “mirror.” Typically, these arrays have 2 or 3 members in each array. RAID 1 arrays have better performance in high-read environments than the same array in a mixed read/write environment.

RAID 5 is a striped set of disks with one stripe segment out of every *n* devoted to parity information. The parity segment is distributed across all the disks. This parity information allows “Hot Swap” and continued operation with the loss of any one disk of the array.

RAID Cost Versus Benefits

RAID cost is measured in performance and reliability.

- **RAID 0:**
  - **Fast**
  - **Loss of any device damages the array.**
- **RAID 1:**
  - **Safe and expensive**
  - **Slight benefit in high-read environments**
- **RAID 5**
  - **Fast**
  - **Safe with loss of any one device**
  - **Possible high write cost**

ORACLE

RAID Cost Versus Benefits

RAID 0 is fast with read and write times approximately 80% of a single disk (with a 4-disk array). There is no penalty on the storage. The cost comes in reliability. The mean time between failure (MTBF) is 1/n, where n is the number of disks in the array. The failure of any member of the array destroys the array, and the entire array must be recovered.

RAID 1: The controllers are generally "smart" enough to know that the same data is on all the mirrors, so multiple read requests are sent to different disks parallelizing the requests. The reliability increases with the number of mirrors. A failure of the array means all the mirrors in the array would have to fail at one time. The cost is measured in space, requiring each byte of data to be stored on all the mirrors.

RAID 5: The read cost of a normal operation is similar to RAID 0. The reliability is quite good; the failure of the array can be caused by a failure of two or more disks at the same time. The storage cost translates to the loss of one disk out of the array for storing parity data. This pushes the cost conscious to add more disks to the array, but this also lowers reliability.
A good balance is between three and eight disks in the array. The major cost of RAID 5 is in the write speed. The write algorithm requires four I/Os for each write. Some vendors have optimized the controllers for these operations but the cost of calculating and writing the parity stripe is still there. The write cost can vary between vendors from a little more than writing to a single disk to more than two times the standard disk write.

RAID Cost Versus Benefits (continued)

RAID 1+0, 0+1, 10

These designations are vendor specific and must be looked at carefully because the designation and the configuration may not match expectations. There are two cases: RAID 0 over 1 and RAID 1 over 0. The cost and reliability measures are the same. The difference is in mean time to recover (MTTR). Vendors are not consistent with the nomenclature and RAID 10 could be either of these methods. Each of these configurations has the speed of RAID 0 and the reliability and cost of RAID 1.

The first method, RAID 1 over 0 or 0+1, combines two or more RAID 0 arrays in a RAID 1 configuration. RAID 1 over 0 has a high MTTR. If one disk in the set fails, the entire striped set must be rebuilt. During the rebuild period, the entire array will have a lowered performance while the one striped set is rebuilt. The protection of two sets is lost for the duration of the rebuild.

The other method, RAID 0 over 1 or 1+0, involves creating RAID 1 arrays of individual disks and then striping them together in a RAID 0 array. RAID 0 over 1 has a lower MTTR. If one disk fails, only that disk is lost. The rebuild time is only for that one disk. This configuration has a lowered performance and loss of protection for one disk while the disk is being rebuilt, and the time to rebuild is reduced.

Should I Use RAID 1 or RAID 5?

RAID 1 (Mirroring)
- Recommended by Oracle
- Most demanding applications

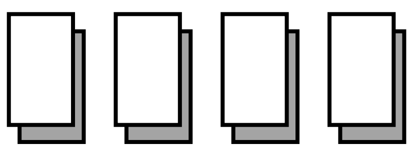
Pros
- Best redundancy
- Best performance
- Low recovery overhead

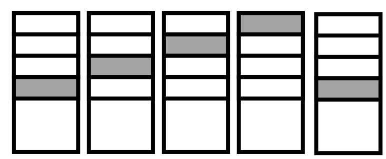
Cons
- Requires higher capacity

RAID 5 (Parity)
- DSS and moderate OLTP

Pros
- Requires less capacity

Cons
- Less redundancy
- Less performance
- High recovery overhead





ORACLE

Should I Use RAID 1 or RAID 5?

The slide lists the pros and cons of using both techniques. Although Oracle recommends using RAID 1, you need to take into account the additional disks required. The general rule is to deploy RAID 5 where the cost of storage is critical, performance is not the primary goal, and applications are primarily read operations such as data warehouse applications. The flash recovery area can be a good use of RAID 5, where the storage capacity requirement is the highest and predominantly sequential I/O.

Note: Although RAID 5 implementations have come a long way, the performance is vastly different from one storage array product to another and caution should be exercised when choosing a storage platform. The ORION tools (http://www.oracle.com/technology/software/index.html#util) can be used to help determine the pros and cons of arrays for your application.

Diagnostics

Indicators of I/O issues:

- **Top waits are reads and writes plus:**
  - **Buffer busy waits**
  - **Write complete waits**
  - **DB file parallel writes**
  - **Enqueue waits**
- **File I/O Statistics section shows high waits and AVG Buffer Wait time higher than average on certain files.**

Note: On a well-performing system, the top events are likely to be CPU time, db file scattered read, and db file sequential read.

ORACLE

Diagnostics

I/O issues can become apparent in several ways. Checkpointing and redo issues often involve I/O issues as well. When the primary issue is I/O, the top wait events show read and write events taking the most time. Even these indicators require more research. If the I/O is saturated, then buffer busy waits, write complete waits, db file parallel writes, and enqueues may all appear in the top wait events.

The File I/O Statistics section of the Statspack or AWR report shows whether the waits are concentrated in one or a few data files. This could indicate a hot spot or a slow device. Hot spots can be caused by poorly designed SQL or tables and index structures. Tuning the SQL to reduce I/O is the first step. Tuning the access methods is the second. Consider moving tables and indexes to different tablespaces on different devices. This last step is unnecessary if the SAME principle has been applied.

Database I/O Tuning

Configuring storage for a database depends on many variables:

- **Which data to put on which disk; complicated by vendor-configured logical units (LUNs)**
- **DB application workloads: OLTP, DSS, batch versus online**
- **Trade-offs between available options**
- **Ongoing tuning: changes in workloads**
- **Expanding or contracting your database**

ORACLE

Database I/O Tuning

When planning your database design storage layout, you need to determine:

- How many volumes and database files are needed?
- Where should each database file reside to reduce contention?
- What is the mapping of physical disks to logical unit (LUN)?
- What are the I/O characteristics of the primary workload?
- How will changes in the workload change the layout?
- Can the layout be changed without shutting down the database?

Consider the types of database files needed for the Oracle database. File types include control files, redo log files, data files, backup files, temp files, and flashback logs.

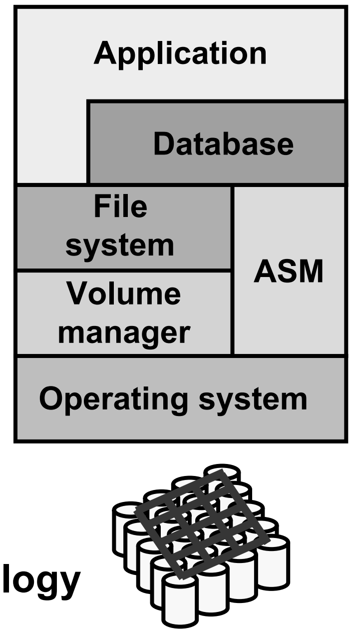
Choose a naming convention for your files that allows clear distinctions for file type and the database it belongs to. Managing these files can be problematic if you manage multiple databases, especially during recovery operations. Is there spare space for recovery operations? Where should it be located and managed?

One solution is to use Automatic Storage Management (ASM). ASM implements the Stripe And Mirror Everything (SAME) technique to optimize storage I/O utilization in terms of bandwidth, sequential and random access, and workload types.

What Is Automatic Storage Management?

ASM:

- **Is a portable and high-performance cluster file system**
- **Manages Oracle database files**
- **Distributes data across disks to balance load**
- **Provides integrated mirroring across disks**
- **Solves many storage management challenges**
- **Encapsulates the SAME methodology**



What Is Automatic Storage Management?

Automatic Storage Management (ASM) is an Oracle Database 10*g* feature.

It provides a file system and volume manager that is specifically built for database files.

ASM can provide management for single SMP machines or across multiple nodes of a cluster for Oracle Real Application Clusters support.

ASM distributes I/O load across all available resources to optimize performance while removing the need for manual I/O tuning.

ASM helps you manage a dynamic database environment by allowing you to increase the size of the database without having to shut down the database to adjust the storage allocation.

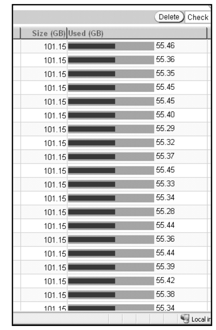
ASM can maintain redundant copies of data to provide fault tolerance, or it can be built on top of vendor-supplied reliable storage mechanisms.

Data management is done by selecting the desired reliability and performance characteristics for classes of data rather than with human interaction on a per-file basis.

ASM capabilities save you time by automating manual storage and thereby increasing your ability to manage larger databases (and more of them) with increased efficiency.

Note: ASM encapsulates the Stripe And Mirror Everything (SAME) methodology.

ASM: Key Features and Benefits



- Stripes files rather than logical volumes
- Online disk reconfiguration and dynamic rebalancing
- Adjustable rebalancing speed
- Provides redundancy on a file basis
- Supports only Oracle database files
- Database cluster file system with performance of raw I/O usable on all storage platforms
- Automatic database file management
- No more hot spots: Eliminates manual I/O tuning

ORACLE

ASM: Key Features and Benefits

ASM divides files into chunks called allocation units (AUs) and spreads the chunks for each file evenly across all the disks. Striping is done by file rather than by logical volume. ASM uses an index technique to track the placement of each chunk. In contrast, traditional striping techniques use mathematical functions to stripe complete logical volumes.

When your storage capacity changes, ASM does not re-stripe all of the data, but moves an amount of data proportional to the amount of storage added or removed to evenly redistribute the files and maintain a balanced I/O load across the disks. This is done while the database is active. You can see the result in the slide: All disks have the same amount of data. This eliminates hot spots on your disks. Manual placement of your database files is no longer required.

You can adjust the speed of a rebalance operation to increase its speed or to lower the impact on the I/O subsystem.

ASM includes mirroring protection without the need to purchase a third-party logical volume manager. One unique advantage of ASM is that the mirroring is applied on a file basis, rather than on a volume basis. Therefore, the same disk group can contain a combination of files protected by mirroring, or not protected at all.

ASM: Key Features and Benefits (continued)

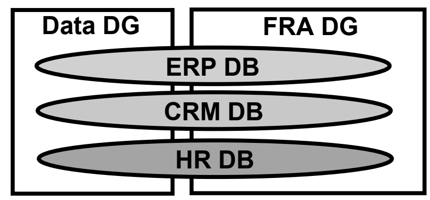
ASM supports data files, log files, control files, archive logs, RMAN backup sets, and other Oracle database file types.

ASM supports Real Application Clusters and eliminates the need for a Cluster Logical Volume Manager or a Cluster File System.

ASM also automatically manages database files by sizing them, naming them, and removing them when appropriate.

How Many Disk Groups per Database

- **Two disk groups are recommended:**
  - **Leverage maximum of LUNs**
  - **Backup for each other**
  - **Lower performance may be used for FRA (or inner tracks)**



- **Exceptions:**
  - **Additional disk groups for different capacity or performance characteristics**
  - **Different ILM storage tiers**

ORACLE

How Many Disk Groups per Database

Most of the time two disk groups are enough in ASM to share the storage between multiple databases.

You can create an ASM disk group of LUNs on the outer edge of your disks, which gives you the best performance for active database files.

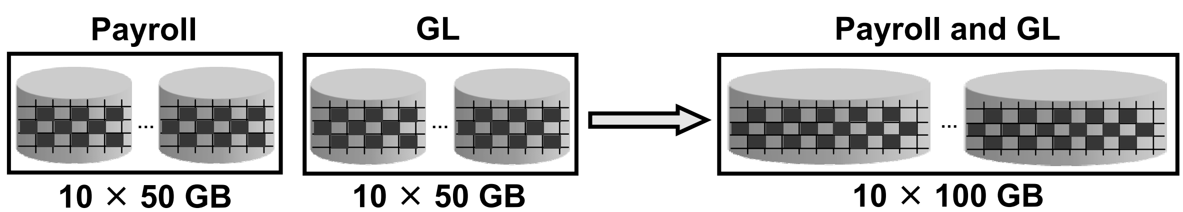
Using a second disk group allows you to have a backup of your data by using it as your common flash recovery area (FRA). You can create a disk group of LUNs on the inner edge of your disks with lower performance for the FRA.

The two noticeable exceptions to this rule are whenever you are using disks with different capacity or performance characteristics, or when you want to archive your data on lower-end disks for Information Lifecycle Management (ILM) purposes.

Note: Disk partitioning utilities start with cylinder 1, which is the outermost cylinder of the disk, and continue to cylinder *n,* which is the innermost. The first half of the disk has the highest performance, the inner half the lowest. This is due to how the sectors are arranged on the disk platter. The outer cylinder can have a maximum transfer rate that is twice the rate of the inner cylinder.

Database Storage Consolidation

- Shared storage across several databases
  - RAC and single-instance can use the same ASM instance
- Benefits:
  - Simplified and centralized management
  - Higher storage utilization
  - Higher performance



Database Storage Consolidation

In Oracle Database 10*g* Release 2, Oracle Clusterware does not require an Oracle Real Application Clusters license. Oracle Clusterware is available with ASM and single-instance Oracle Database 10*g*, allowing support for a shared clustered pool of storage for RAC and single-instance Oracle databases.

This allows you to optimize your storage utilization by eliminating unused spare storage. This is illustrated in the slide. Instead of having various pools of disks used for different databases, you consolidate all that in one single pool shared by all your databases.

By doing this, you can reduce the number of logical units (LUNs) to manage by increasing their sizes, giving you a higher storage utilization as well as a higher performance. Increasing the size of the LUN does not mean increasing the size of the disk. The bandwidth is greatly affected by the total number of physical disks. Combining more physical disks into a single LUN or disk group yields higher bandwidth and fewer logical volumes to manage.

Note: RAC and single-instance databases could not be managed by the same ASM instance in Oracle Database 10*g* Release 1.

Which RAID Configuration for Best Availability?

A. ASM mirroring
B. Hardware RAID 1 (mirroring)
C. Hardware RAID 5 (Parity Protection)
~~D.~~ Both ASM mirroring and hardware RAID

Answer: Depends on business requirement and budget (cost, availability, performance, and utilization)

ASM leverages hardware RAID.

ORACLE

Which RAID Configuration for Best Availability?

To favor availability, you have multiple choices as shown in the slide.

You could just use ASM mirroring capabilities, or hardware RAID 1 (which is a hardware mirroring technique), or hardware RAID 5. Another possibility, although not recommended, is to use both ASM mirroring and hardware mirroring. Oracle recommends that you use external redundancy disk groups when using hardware mirroring techniques to avoid an unnecessary overhead.

Choose between A, B, and C, depending on your business requirements and budget.

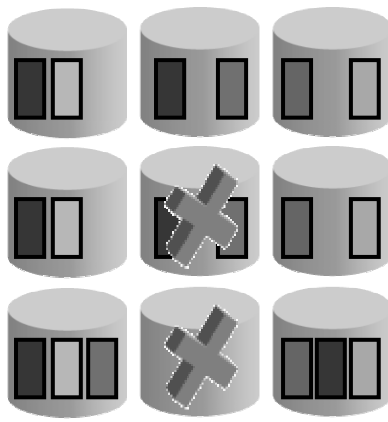
ASM mirroring works with the standard disks without specialized hardware.

RAID 1 has the best performance but requires twice the storage capacity.

RAID 5 is a much more economical solution, but with a performance penalty essentially for write-intensive workloads.

ASM Mirroring Guidelines

- Best choice for low-cost storage
- Enables Extended Clustering solutions
- No hardware mirroring



ASM Mirroring Guidelines

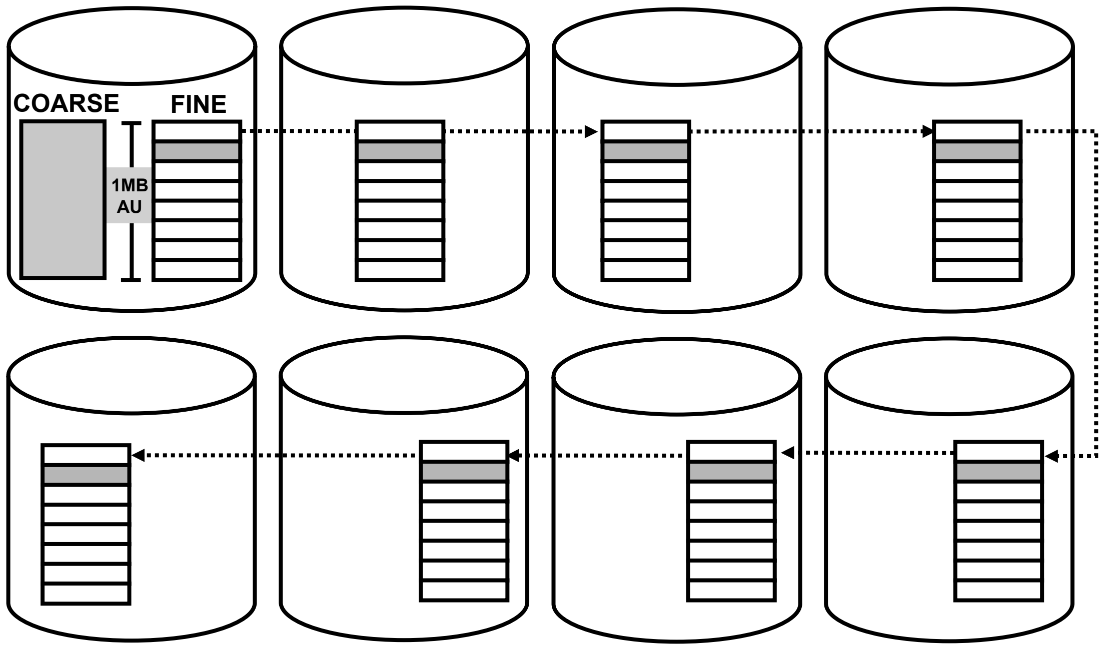
Leverage the storage array hardware RAID 1 mirroring protection when possible to offload the mirroring overhead from the server. Use ASM mirroring in the absence of a hardware RAID capability.

However, hardware RAID 1 in most ATA (or Advanced Technology Attachment) storage technologies is inefficient and degrades the performance of the array. Using ASM redundancy has proven to deliver much better performance in ATA arrays.

Because the storage cost can grow very rapidly whenever you want to achieve extended clustering solutions, ASM mirroring should be used as an alternative to hardware mirroring for low-cost storage solutions.

Note: For more information about the Oracle Resilient Low-cost Storage Initiative, see the Web site at:
http://www.oracle.com/technology/deploy/availability/htdocs/lowcoststorage.html

ASM Striping Granularity



ASM Striping Granularity

The difference between ASM `COARSE` and `FINE` striping is illustrated in this slide.

When a block is allocated for a file with a coarse stripe, a single allocation unit of 1 MB is made. This does not account for any mirroring that would require additional blocks. Data files have the coarse-striping property.

When a block is allocated for a file with a fine stripe, a set of eight allocation units are set aside for this purpose in a round-robin allocation across all the available disks. The write is then made to these eight allocation units in 128 KB chunks with each successive write going to the next allocation unit in the set. Therefore, 8 MB can be written before the next allocation is made. Redo log files have the fine-striping property.

Basically, one redo log write of 1 MB is striped in 128 KB chunks across eight allocation units. This is done automatically by the system when the redo log file is created.

Note: For best performance, redo log files should not be created on a separate disk group from the data files. For test results that show SAME is a better method of utilizing the I/O system where all database files including the redo logs were striped across all the available disks, see the Oracle white paper *Optimal Storage Configuration Made Easy* at http://technet.oracle.com/deploy/availability/pdf/oow2000_sane.pdf.

What Type of Striping Works Best?

A. ASM striping only (no RAID 0)
B. RAID 0 and ASM striping
~~C.~~ Use LVM
~~D.~~ No striping

Answer: A and B

ASM and RAID striping are complementary.

What Type of Striping Works Best?

As shown in the slide, you can use ASM striping only or you can use ASM striping in combination with RAID 0.

With RAID 0, multiple disks are configured together as a set or a bank. Data from any one data file is spread, or striped, across all the disks in the bank.

Combining both ASM striping and RAID striping is called stripe-on-stripe. This combination offers good performance.

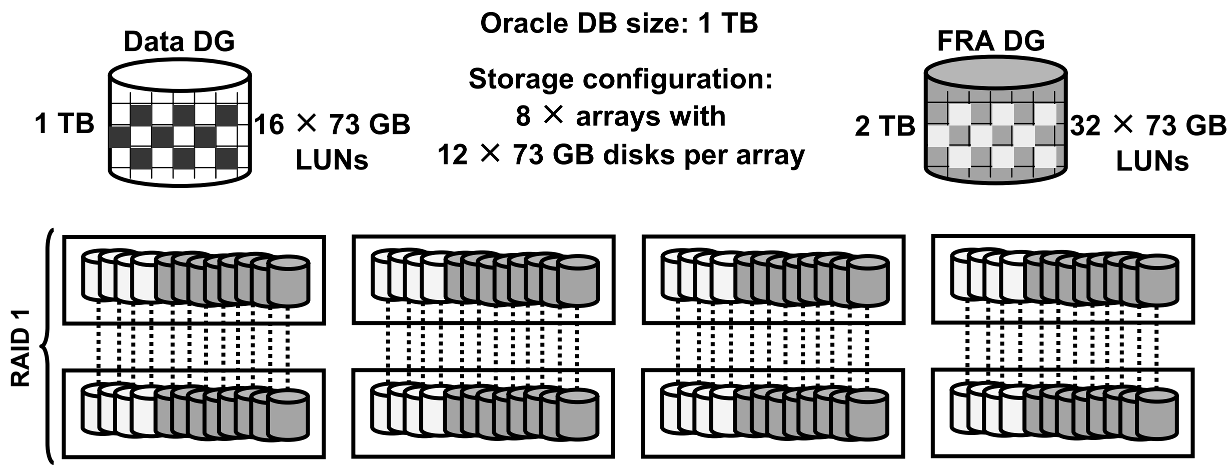
There is no longer a need to use a logical volume manager for your database files.

It is recommended that you use some type of striping: either ASM or RAID.

ASM Striping Only

Pros
- Drives evenly distributed for Data & FRA
- Higher bandwidth
- Allows small incremental growth (73 GB)
- No drive contention

Cons
- Not well balanced across *all* disks
- LUN size limited to disk size



ASM Striping Only

In the case shown in the slide, you want to store a one-terabyte database with a corresponding two-terabytes flash recovery area. You have eight arrays of twelve disks, with each disk being 73 GB. You use RAID 1 to mirror each disk creating 48 logical units (LUNs) of 73 GB each. ASM mirroring and hardware RAID 0 are not used.

The Data disk group is allocated 16 LUNs of 73 GB each. The Flash Recovery Area disk group is assigned 32 LUNs of 73 GB each.

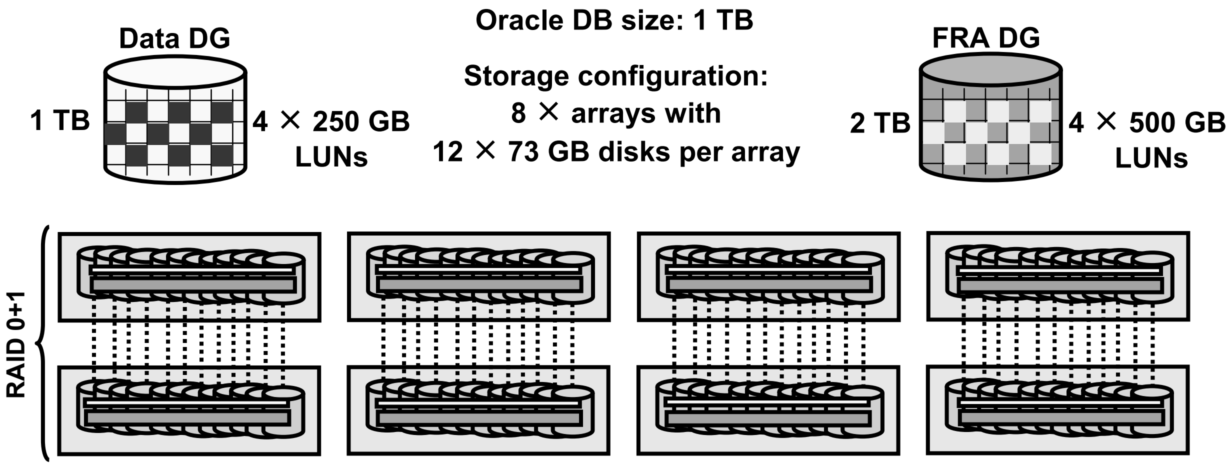
This configuration allows you to evenly distribute disks for your data and backups, achieving good performance and allowing you to manage your storage in small incremental chunks.

However, using a limited number of disks in your Data disk group does not balance your data well across all your disks, and you still have many LUNs to manage at the storage level.

If you want to place all the data on fast regions of the disk, then partition the disks and create LUNs over the fast partitions and slow partitions. This is possible; it just multiplies the number of LUNs to manage by the number of partitions on each disk.

Hardware RAID Striped LUNs

Pros	Cons
• Fastest region for Data DG • Balanced data distribution • Fewer LUNs to manage while max spindles	• Large incremental growth • Data and FRA “contention”



Hardware RAID Striped LUNs

In the case shown in the slide, you have a one-terabyte database with a corresponding two-terabytes flash recovery area. You have eight arrays of twelve disks, with each disk being 73 GB. You use RAID 0+1, a combination of hardware striping and mirroring, to mirror and stripe each partition, creating four logical units of 876 GB each. These can be defined as an ASM single disk group. ASM mirroring is not used.

Here, you define bigger LUNs, not restricted to the size of one of your disks. ASM striping spreads the allocation units across the LUNs. The RAID-level striping further spreads I/Os across the physical disks. By doing this, you achieve a better data distribution across all your disks, and you end up managing significantly fewer LUNs.

However, you must manipulate your storage in much larger chunks than in the previous configuration. To force the use of the faster region for data, an additional LUN would have to be created across a set of partitions of the disks resulting in two LUNs.

Note: The hardware stripe size you choose is very important. You want 1 MB alignment as much as possible, to match the ASM AUs. Selecting “power of two” stripe sizes (64 KB, 128 KB, or 256 KB) is better than selecting other numbers. The choices offered by storage vendors for stripe size and partitioning depend on their storage array technology. These can create unnecessary I/O bottlenecks if not carefully considered.

ASM Guidelines

- **Use external RAID protection when possible.**
- **Create logical units (LUNs) using:**
  – **Outside half of disk drives for highest performance**
  – **Small disk, high rpm (for example, 73 GB/15k rpm)**
- **Use LUNs with the same performance characteristics.**
- **Use LUNs with the same capacity.**
- **Maximize the number of spindles in your disk group.**

ASM Guidelines

Use Oracle Database 10*g* ASM for volume and file management to equalize the workload across disks of logical units and eliminate hot spots.

Follow these simple guidelines and best practices when configuring ASM disk groups:

- Use external RAID protection when possible.
- Create LUNs using:
  - Outside half of disk drives for highest performance
  - Small disk with high rpm (for example, 73 GB with 15k rpm) for best performance. Spindle (platter) speed directly impacts both positioning time and data transfer. Faster spindle speed drives have improved performance regardless of whether they are used for many small, random accesses, or for streaming large contiguous blocks from the disk.
- Maximize the number of spindles in your disk group for greater bandwidth.
- Disks or LUNs assigned to ASM disk groups should have the same storage performance and availability characteristics. Configuring mixed speed drives will default to the lowest common denominator.
- ASM data distribution policy is capacity based. Therefore, LUNs provided to ASM should have the same capacity for each disk group to avoid imbalance and hot spots.

ASM Instance Initialization Parameters

- **ASM instances have static memory needs.**
- **Using default SGA sizing parameters should be enough for most configurations: Add 500 KB to the shared pool per additional disk group after the first five.**

```
INSTANCE_TYPE = ASM
DB_UNIQUE_NAME = +ASM
ASM_POWER_LIMIT = 1
ASM_DISKSTRING = '/dev/rdsk/*s2', '/dev/rdsk/c1*'
ASM_DISKGROUPS = dgroupA, dgroupB
```

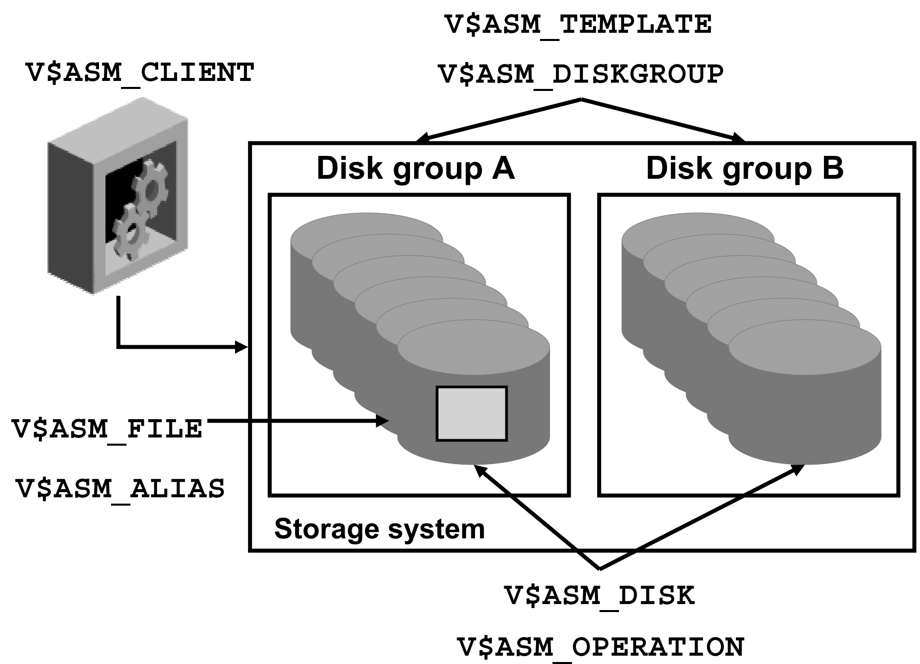
```
PROCESSES = 25 + 15*<#DB inst using ASM for their storage>
```


ASM Instance Initialization Parameters

- `INSTANCE_TYPE` must be set to `ASM` for ASM instances.
- `DB_UNIQUE_NAME` specifies the service name. Use default value of `+ASM`.
- `ASM_POWER_LIMIT` controls the speed of a rebalance operation. Possible values range from the default low value of `1` through a high value of `11`. The rebalance speed may be set per operation by the (`POWER`) setting of the rebalance command.
- `ASM_DISKSTRING` is an OS-dependent value that limits the set of disks considered for discovery. If not specified, it is assumed to be `NULL` and ASM disk discovery finds all disks to which an ASM instance has read and write access.
- `ASM_DISKGROUPS` is the list of names of disk groups to be mounted by an ASM instance at startup. ASM automatically adds a disk group to this parameter when a disk group is successfully mounted, and automatically removes the disk group when it is dismounted except for dismounts at instance shutdown.
- `PROCESSES` is the maximum number of processes started by the instance. A recommended value is shown in the slide.

Note: The memory footprint of an ASM instance is around 100 MB. That should be enough for most sites.

Dynamic Performance Views



Dynamic Performance Views

View	ASM	Database
V$ASM_CLIENT	One row for every database instance using a disk group in the ASM instance	One row for each disk group with the database and ASM instance name
V$ASM_DISKGROUP	One row for every discovered disk group	A row for all disk groups mounted or dismounted
V$ASM_TEMPLATE	One row for every template present in every mounted disk group	Rows for all templates in mounted disk groups
V$ASM_DISK	One row for every discovered disk, including disks that are not part of any disk group	Rows for disks in the disk groups in use by the database instance

Dynamic Performance Views (continued)

View	ASM	Database
V$ASM_OPERATION	One row for every active ASM long-running operation executing in the ASM instance	Contains no rows
V$ASM_FILE	One row for every ASM file in every disk group mounted by the ASM instance	Contains no rows
V$ASM_ALIAS	One row for every alias present in every disk group mounted by the ASM instance	Contains no rows

The `V$ASM_DISK_STAT` and `V$ASM_DISKGROUP_STAT` views return the same information as their counterparts without `_STAT`. They return cached results, rather than doing a new disk discovery. Use `V$ASM_DISK_STAT` instead of `V$ASM_DISK`, and `V$ASM_DISKGROUP_STAT` instead of `V$ASM_DISKGROUP`, in order to query performance statistics. The `V$ASM_DISK` and `V$ASM_DISKGROUP` views require a discovery of disks during every execution, which makes it a relatively expensive operation, and repeats of this query at high frequency are not advisable. For example:

```
SELECT path, reads, writes, read_time, write_time,
   read_time/decode(reads,0,1,reads) "avgrdtime",
   write_time/decode(writes,0,1,writes) "avgwrtime"
FROM V$ASM_DISK_STAT;
```

Note: For more information about ASM data dictionary views, refer to the *Oracle Database Reference* guide.

Monitoring Long-Running Operations by Using V$ASM_OPERATION

Column	Description
GROUP_NUMBER	Disk group
OPERATION	Type of operation: REBAL
STATE	State of operation: WAIT or RUN
POWER	Power requested for this operation
ACTUAL	Power allocated to this operation
SOFAR	Number of allocation units moved so far
EST_WORK	Estimated number of remaining allocation units
EST_RATE	Estimated number of allocation units moved per minute
EST_MINUTES	Estimated amount of time (in minutes) for operation termination

Monitoring Long-Running Operations by Using V$ASM_OPERATION

The ALTER DISKGROUP DROP, RESIZE, and REBALANCE commands return before the operation is complete. To monitor the progress of these long-running operations, you can query the V$ASM_OPERATION fixed view. This view is described in the table in this slide.

Note: In addition to the REBAL operation, you may see other operations relating to recovery and ASM metadata.

ASM Instance Performance Diagnostics

```
SELECT   event, total_waits t_wait,
         total_timeouts t_timeout,
         time_waited t_waittm,
         average_wait a_waittm, wait_class
FROM     V$SYSTEM_EVENT
WHERE    wait_class <> 'Idle' and time_waited > 0
ORDER BY 4 DESC;
```

```
EVENT                                WAIT  TOUT  WAITT     AVG CLASS
------------------------------ ------ ----- ----- ------- -------
ASM mount : wait for heartbeat      1     1   439  438.85 Admin…
kfk: async disk IO                578     0   377     .65 SystI/O
log write(odd)                      7     3   296   42.33 Other
rdbms ipc reply                    37     1   259    7.01 Other
log write(even)                     8     2   197   24.58 Other
SQL*Net message to client      139249     0   103       0 Network
os thread startup                   9     0    79    8.77 Conc…
buffer write wait                   1     0    60   60.31 Other
DBFG waiting for reply             16     0     1     .04 Other
```

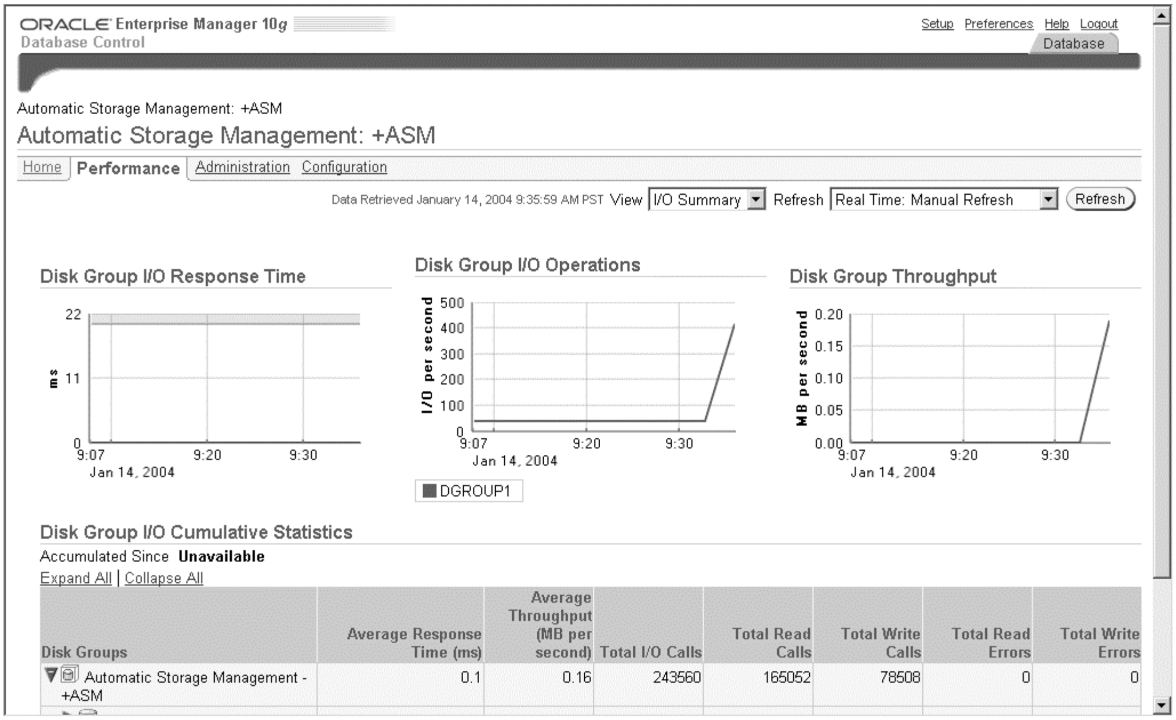

ASM Instance Performance Diagnostics

The ASM instance architecture does not allow the creation of tables, so it is not possible to generate AWR or Statspack reports.

However, ASM instances support querying all the dynamic performance (`V$`) views, so you need to base performance diagnostics on SQL scripts.

One possible example is illustrated in the slide.

ASM Performance Page



ASM Performance Page

The Performance tab of the Automatic Storage Management page shows the I/O response time and throughput for each disk group. You can further drill down to view disk-level performance metrics.

Database Instance Parameter Changes

- **Add the following to `SHARED_POOL_SIZE`:**

```
(DB_SPACE/100+2)*#_External_Red  OR
(DB_SPACE/50+4)*#_Normal_Red     OR
(DB_SPACE/33+6)*#_High_Red
```

```
SELECT d+l+t DB_SPACE
FROM (SELECT SUM(bytes)/(1024*1024*1024) d
       FROM V$DATAFILE),
      (SELECT SUM(bytes)/(1024*1024*1024) l
       FROM V$LOGFILE a, V$LOG b WHERE a.group#=b.group#),
      (SELECT SUM(bytes)/(1024*1024*1024) t
       FROM V$TEMPFILE WHERE status='ONLINE');
```

- **Add at least `16` to `PROCESSES`.**

ORACLE

Database Instance Parameter Changes

If the database instance is not using Automatic Shared Memory Management, then the SGA parameters for a database instance need slight modification to support AU maps and other ASM information. The following are guidelines for SGA sizing on the database instance:

- Additional memory is required to store AU maps in the shared pool. Use the result of the query shown in the slide to obtain the current database storage size (`DB_SPACE`) that either is already on ASM or will be stored in ASM. Then, determine the redundancy type that is used (or will be used), and add to the shared pool size one of the following values:
  - For disk groups using external redundancy: Every 100 GB of space needs 1 MB of extra shared pool plus a fixed amount of 2 MB of shared pool.
  - For disk groups using normal redundancy: Every 50 GB of space needs 1 MB of extra shared pool plus a fixed amount of 4 MB of shared pool.
  - For disk groups using high redundancy: Every 33 GB of space needs 1 MB of extra shared pool plus a fixed amount of 6 MB of shared pool.
- Add at least `16` to the value of the `PROCESSES` initialization parameter.

ASM Disk Metadata Requirements

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

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

  - For external redundancy:

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

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

ASM Disk Metadata Requirements

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

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

As files are created, there is additional metadata overhead:

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

Note: Compared to the space used for storing user data, this is insignificant.

ASM Scalability

ASM imposes the following limits:
- **63 disk groups**
- **10,000 ASM disks**
- **4 petabyte per ASM disk**
- **40 exabyte of storage**
- **1 million files per disk group**
- **Maximum file size:**
  - **External redundancy: 35 TB**
  - **Normal redundancy: 5.8 TB**
  - **High redundancy: 3.9 TB**

ASM Scalability

ASM imposes the following limits:
- 63 disk groups in a storage system
- 10,000 ASM disks in a storage system
- 4 petabyte maximum storage for each ASM disk
- 40 exabyte maximum storage for each storage system
- 1 million files for each disk group
- Maximum file sizes depending on the redundancy type of the disk groups used: 35 TB for external redundancy, 5.8 TB for normal redundancy, and 3.9 TB for high redundancy

Summary

In this lesson, you should have learned how to:

- Diagnose database I/O issues
- Describe the Stripe And Mirror Everything (SAME) concept
- Explain the benefits of asynchronous I/O
- Choose appropriate I/O solutions
- Tune I/O using Automatic Storage Management (ASM)

Practice 12 Overview: Tuning I/O: A Demonstration

This practice covers the following topics:

- Viewing the symptoms and waits on a single-disk system
- Viewing the symptoms and waits on a multidisk system using SAME
- Viewing the symptoms and waits on a multidisk ASM system

ORACLE

Tuning PGA and Temporary Space

Objectives

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

- Diagnose PGA memory issues
- Size the PGA memory
- Diagnose temporary space issues
- Specify temporary tablespace parameters for efficient operation

SQL Memory Usage

- **Memory-intensive SQL operators:**
  - **Sort-based (sort, group-by, rollup, window, ...)**
  - **Hash-join**
  - **Bitmap operators (merge and inversion)**
- **Concept of work area:**
  - **Memory allocated by a memory-intensive operator to process its input data**
- **Performance impact of memory:**
  - **Optimal: Input data fits into the work area (cache).**
  - **One-pass: Perform one extra pass over input data.**
  - **Multi-pass: Perform several extra passes over input data.**

ORACLE

SQL Memory Usage

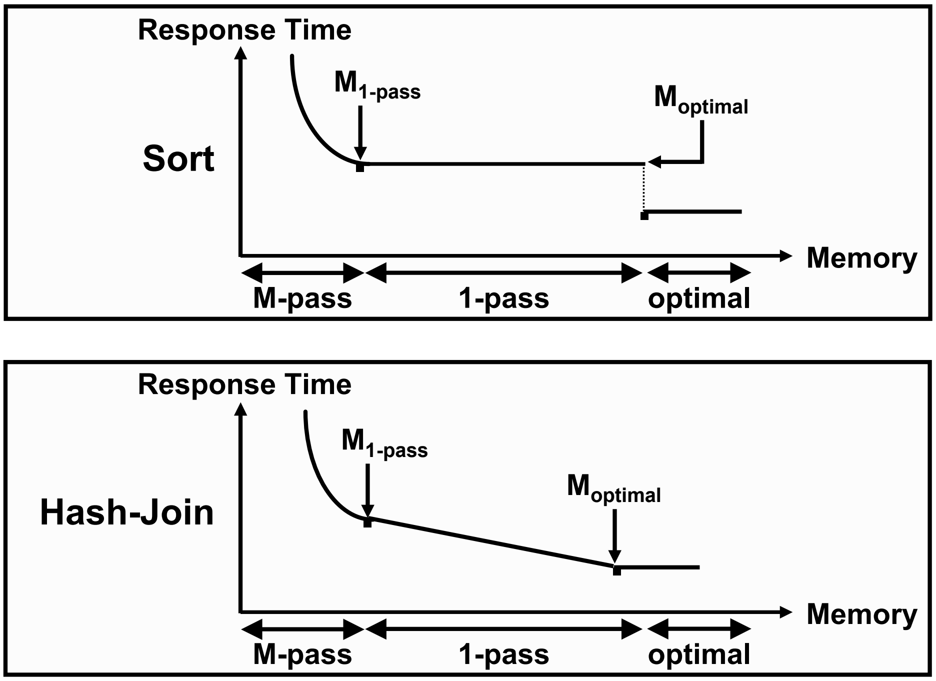
Complex database queries require memory-intensive operators such as sort and hash-join. Those operators need what is called work area memory to process their input data. For example, a sort operator uses a work area to perform the in-memory sort of a set of rows. Similarly a hash-join operator uses a work area to build a hash table on one of the tables in the `FROM` clause.

The amount of memory allocated by these operators greatly affects their performance, and a larger work area can significantly improve the performance of a SQL operator. Ideally, the size of a work area should be big enough to accommodate the input data and auxiliary memory structures. This case is referred to as the optimal or cache size of a work area.

Because there is only a finite amount of memory in the system shared by all concurrent operators, an operator cannot always allocate its optimal size. When the size of the work area is smaller than its ideal cache size, the response time increases because an extra pass is performed on all or some of the input data. This is referred to as the one-pass size of the work area.

When the work area size is less than the one-pass threshold, multiple passes over the input data are needed, causing dramatic increase in response time. This is referred to as the multi-pass size of the work area. For example, a sort operation that needs to sort 10 GB of data needs a little more than 10 GB of memory to run in cache, and 40 MB to run in one-pass mode. It will run in multiple passes with less than 40 MB.

Performance Impact



Performance Impact

The diagram in the slide shows you the performance characteristics of the sort and hash-join operators with regard to memory usage.

The one-pass point on the curve is the start of the area where the operator runs in one-pass mode, and the optimal point corresponds to the case when the work area size is equal to the optimal size.

The sort curve is flat between those two points because a sort does not benefit from additional memory if it cannot use its optimal size. However, the hash-join benefits from additional memory between the one-pass and optimal points.

In online transaction processing (OLTP) systems, the size of the input data to SQL operators is generally small, thus they can run in optimal mode most of the time. This is not the case in decision support systems (DSS), where the input data is very large. Thus it is important to size their work area to obtain good performance.

Automatic PGA Memory

- Dynamically adapts the SQL memory allocation based on:
  - PGA memory available
  - SQL operator needs
  - System workload
- Improves manageability:
  - No need to set `*_AREA_SIZE` parameters
  - DBA sets a memory target: `PGA_AGGREGATE_TARGET`
- Improves performance:
  - PGA memory is really returned to the OS.
  - Memory is allocated to the operation to maximize throughput.
  - Maximize overall memory utilization by dynamically adapting memory with workload variation.
  - Operation adapts its memory usage during the execution.

ORACLE

Automatic PGA Memory Management

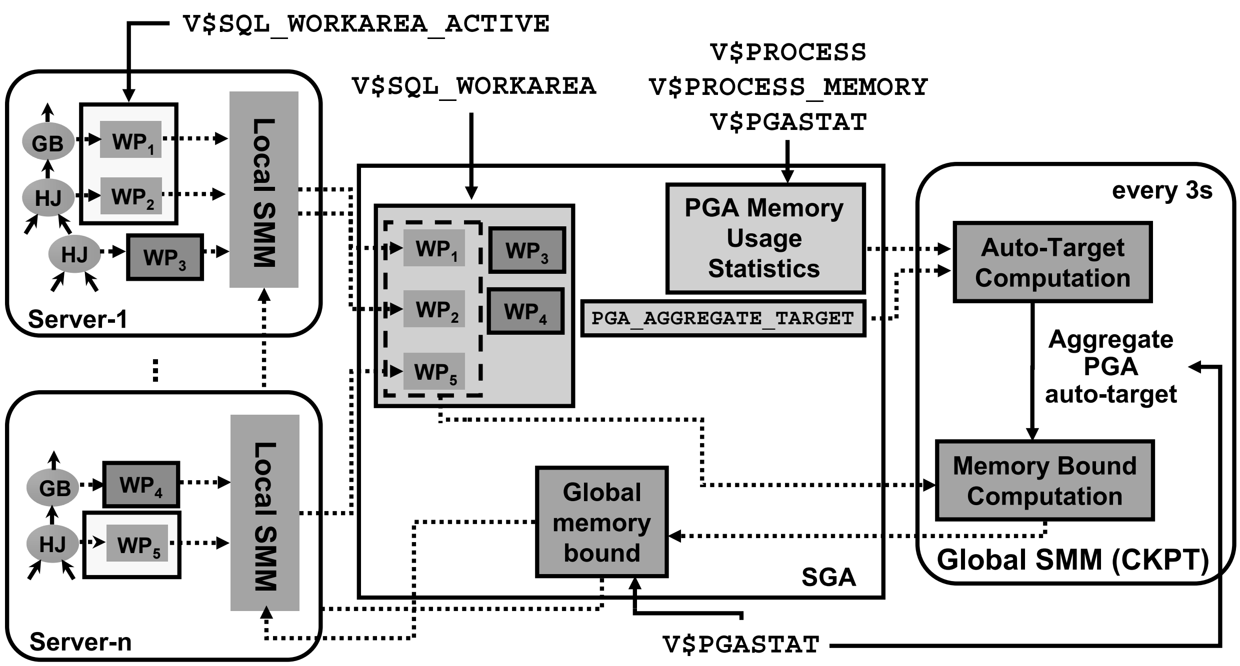
Besides SQL, various components in the database server make use of PGA memory. These other parts are known as the *untunable* parts of the PGA because they require an allocation of memory that cannot be adjusted. This is the case for:

- The context information of each session
- For each open cursor
- PL/SQL, OLAP, or Java memory

The tunable portion of the PGA represents the memory available to SQL work areas. This portion could represent 90% of the overall PGA memory for decision support systems, whereas it could be less than 10% in pure OLTP systems. With automatic PGA memory management, the system attempts to keep the amount of private memory below the target specified by the `PGA_AGGREGATE_TARGET` initialization parameter by adapting the size of the work areas to private memory. When increasing the value of this parameter, you indirectly increase the memory allotted to work areas. Consequently, more memory-intensive operations are able to run fully in memory and less will work their way over to disk.

Note: Oracle Corporation does not recommend the use of static SQL memory management. For more information about `SORT_AREA_SIZE`, `HASH_AREA_SIZE`, `BITMAP_MERGE_AREA_SIZE`, and `CREATE_BITMAP_AREA_SIZE`, refer to the *Oracle Database Reference* guide.

SQL Memory Manager



SQL Memory Manager

SQL PGA memory management (the tunable portion of the PGA) is mainly based on the feedback loop mechanism depicted in the slide.

The left side of the figure represents active SQL statements. When a SQL operator starts, it registers its work area profile (WP_1) using the local SQL memory manager (SMM). A work area profile is a piece of metadata that describes all the characteristics of a work area: its type (sort, hash-join, group-by); its current memory requirement to run with minimum, one-pass, and cache memory; its degree of parallelism; and finally the amount of PGA memory it currently uses. The set of active work area profiles is maintained by the local memory manager in the SGA. These profiles are constantly being updated with the current needs and usage.

The right side of the figure represents the global SQL memory manager implemented by the CKPT background process, which runs every three seconds. It has two main components:

- The first component is responsible for computing the aggregate PGA auto-target, which represents the amount of PGA memory the system can use for the SQL work areas running in automatic mode. This amount is dynamically derived from the value of `PGA_AGGREGATE_TARGET` and also accounts for other PGA memory structures used by components such as PL/SQL, Java, and OLAP.

SQL Memory Manager (continued)

- The second component is responsible for computing a global memory bound. The process takes into account the profiles of active work areas and the aggregate PGA auto-target. This bound is used to constrain the size of each SQL work area; it sets a maximum size of each SQL work area. Therefore, the memory bound is high when the overall memory requirement of all active work areas is low and vice-versa. In simple terms, finding a proper value for the memory bound is finding the maximum value for which the sum of the expected work area size of each operator is less than or equal to the aggregate PGA auto-target.

The feedback loop is closed by the local memory manager. It uses the current value of the memory bound and the current profile of a work area to determine the correct amount of PGA memory, called expected size, which can be allotted to this work area. The expected size is checked periodically by SQL operators, which are then responsible to adapt their work area size to the specified value.

When the local memory manager sees that an operation has finished or reduces the size of its memory allocation, memory is released back to the OS on any operating system that supports this operation.

Configuring Automatic PGA Memory

- `PGA_AGGREGATE_TARGET`:
  - Specifies the target aggregate amount of PGA memory available to the instance
  - Can be dynamically modified at the instance level
  - Examples: 100,000 KB; 2,500 MB; 50 GB
  - Default value: 10 MB or 20% of the size of the SGA, whichever is greater
- `WORKAREA_SIZE_POLICY`:
  - Optional
  - Can be dynamically modified at the instance or session level
  - Allows you to fall back to static SQL memory management for a particular session

Configuring Automatic PGA Memory

PGA_AGGREGATE_TARGET specifies the target of the aggregate of all PGA memory available to all server processes attached to the instance.

The WORKAREA_SIZE_POLICY parameter is automatically set to AUTO when the PGA_AGGREGATE_TARGET parameter is set to a nonzero value. This means that SQL work areas used by memory-intensive SQL operators are automatically sized. The default value for this parameter is 20% of the SGA or 10 MB, whichever is greater. Setting PGA_AGGREGATE_TARGET to 0 automatically sets the WORKAREA_SIZE_POLICY parameter to MANUAL. This means that SQL work areas are sized by using the *_AREA_SIZE parameters. PGA_AGGREGATE_TARGET is not a strict limit. It is used to help the system better manage PGA memory, but the system will exceed this setting if necessary. WORKAREA_SIZE_POLICY can be altered per database session, allowing manual memory management on a per session basis if needed. For example, a session is loading a large import file and a larger sort_area_size is needed. A logon trigger could be used to set the WORKAREA_SIZE_POLICY for the account doing the import.

Note: In Oracle Database 10*g*, PGA_AGGREGATE_TARGET controls work areas allocated by both dedicated and shared connections. In earlier releases, PGA_AGGREGATE_TARGET controlled the sizing of work areas for all dedicated server connections, but it had no effect on shared server connections and the *_AREA_SIZE parameters took precedence.

Setting `PGA_AGGREGATE_TARGET` Initially

- **Leave 20% of the available memory to other applications.**
- **Leave 80% of memory to the Oracle instance.**
- **For OLTP:**

```
PGA_AGGREGATE_TARGET=(total_mem*80%)*20%
```

- **For DSS:**

```
PGA_AGGREGATE_TARGET=(total_mem*80%)*50%
```


Setting `PGA_AGGREGATE_TARGET` Initially

Assume that an Oracle instance is configured to run on a system with 4 GB of physical memory. Part of that memory should be left for the operating system and other non-Oracle applications running on the same hardware system. You might decide to dedicate only 80% (3.2 GB) of the available memory to the Oracle instance.

You must then divide the resulting memory between the SGA and the PGA.

For OLTP systems, the PGA memory typically accounts for a small fraction of the total memory available (for example, 20% of the instance memory), leaving 80% for the SGA.

For DSS systems running large, memory-intensive queries, PGA memory can typically use up to 70% of the instance memory (up to 2.2 GB in this example).

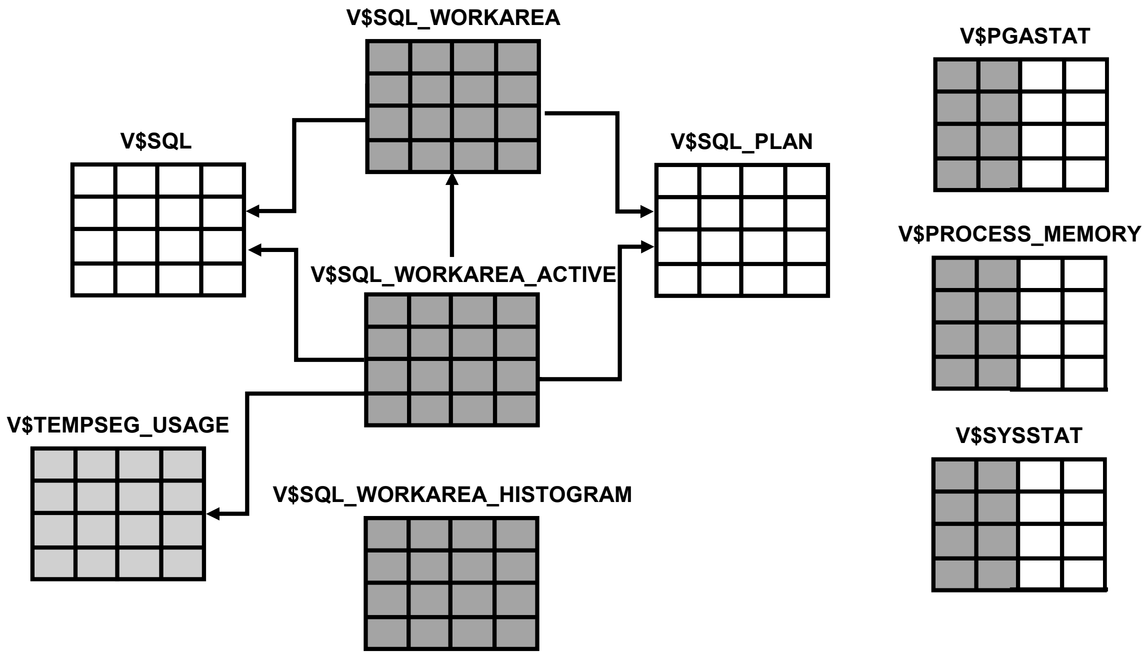
Good initial values for the `PGA_AGGREGATE_TARGET` parameter might be:

- For OLTP: `PGA_AGGREGATE_TARGET=(`*`total_mem`*`*80%)*20%`
- For DSS: `PGA_AGGREGATE_TARGET=(`*`total_mem`*`*80%)*50%`

where *`total_mem`* is the total amount of physical memory available on the system.

In this example, with a value of *`total_mem`* equal to 4 GB, you can initially set `PGA_AGGREGATE_TARGET` to 1600 MB for a DSS system and to 655 MB for an OLTP system.

Monitoring SQL Memory Usage



Monitoring SQL Memory Usage

V$SQL_WORKAREA_HISTOGRAM displays the cumulative statistics for different work area sizes. There are 33 groups of work areas based on their optimal memory requirements increasing in powers of two: work areas whose optimal requirement varies from 0 KB to 1 KB, 1 KB to 2 KB, 2 KB to 4 KB, ... and 2 TB to 4 TB. The V$SQL_WORKAREA_HISTOGRAM view shows for each work area group how many work areas in that group were able to run in optimal mode, how many were able to run in one-pass mode, and finally how many ran in multi-pass mode.

V$PROCESS_MEMORY displays dynamic PGA memory usage by category, for each process. The categories are: SQL, PL/SQL, OLAP, and Java. Special categories are Freeable and Other. The amount of Freeable memory represents the free PGA memory eligible to be released to the operating system.

V$PGASTAT provides cumulative PGA memory usage statistics as well as current statistics about the automatic PGA memory manager when it is enabled.

V$SQL_WORKAREA_ACTIVE can be used to display the work areas that are active (or executing) in the instance. Small active sorts (under 64 KB) are excluded from the view. Use this view to precisely monitor the size of all active work areas and to determine whether these active work areas spill to a temporary segment. If a work area spills to disk, then this view contains information for the temporary segment created on behalf of this work area, and can be joined with V$TEMPSEG_USAGE to retrieve more information.

Monitoring SQL Memory Usage (continued)

V$SQL_WORKAREA maintains cumulative work area statistics for each loaded cursor whose execution plan uses one or more work areas. Every time a work area is deallocated, the V$SQL_WORKAREA table is updated with execution statistics for that work area. V$SQL_WORKAREA can be joined with V$SQL to relate a work area to a cursor. It can even be joined to V$SQL_PLAN to precisely determine which operator in the plan uses a work area.

Statistics in the V$SYSSTAT and V$SESSTAT views show the total number of work areas executed with optimal memory size, one-pass memory size, and multi-pass memory size. These statistics are cumulative since the instance or the session was started.

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

Monitoring SQL Memory Usage: Examples

```
SELECT sql_text,
       sum(onepass_executions) onepass_cnt,
       sum(multipasses_executions) mpass_cnt
FROM V$SQL s, V$SQL_WORKAREA wa
WHERE s.address = wa.address
GROUP BY sql_text
HAVING sum(onepass_executions+multipasses_executions)>0;
```

1

```
SELECT TO_NUMBER(DECODE(sid, 65535, NULL, sid)) sid,
       operation_type                        OPERATION,
       TRUNC(expected_size/1024)                 ESIZE,
       TRUNC(actual_mem_used/1024)                 MEM,
       TRUNC(max_mem_used/1024)                 MAXMEM,
       number_passes                              PASS,
       TRUNC(tempseg_size/1024)                  TSIZE
FROM V$SQL_WORKAREA_ACTIVE
ORDER BY 1,2;
```

2

Monitoring SQL Memory Usage: Examples

The first example shows you how to find the cursors with one or more work areas that have been executed in one or even multiple passes.

The second example shows you how to find information about active work areas. The output of this query might look like the following:

```
SID        OPERATION     ESIZE       MEM     MAXMEM  PASS   TSIZE
--- ----------------- --------- --------- --------- ----- -------
  8   GROUP BY (SORT)       315       280       904     0
  8         HASH-JOIN      2995      2377      2430     1 20000
  9   GROUP BY (SORT)     34300     22688     22688     0
 11         HASH-JOIN     18044     54482     54482     0
 12         HASH-JOIN     18044     11406     21406     1 120000
```

This output shows you that session 12 is running a hash-join having its work area running in one-pass mode. This work area is currently using 11,406 KB of memory and has used, in the past, up to 21,406 KB of PGA memory. It has also spilled to a temporary segment of size 120,000 KB. Finally, the `ESIZE` column indicates the maximum amount of memory that the PGA memory manager expects this hash-join to use. This maximum is dynamically computed by the PGA memory manager according to workload.

Tuning SQL Memory Usage

- Determine the best `PGA_AGGREGATE_TARGET` value by using:
  - `V$PGA_TARGET_ADVICE`
  - `V$PGA_TARGET_ADVICE_HISTOGRAM`
- Monitor AWR reports/Statspack reports for:
  - `direct path read temp`
  - `direct path write temp`

Tuning SQL Memory Usage

The Oracle database server helps you in the task of tuning the value of `PGA_AGGREGATE_TARGET` by providing two advice performance views: `V$PGA_TARGET_ADVICE` and `V$PGA_TARGET_ADVICE_HISTOGRAM`. By examining these two views, you do not need to use an empirical approach to the tuning process.

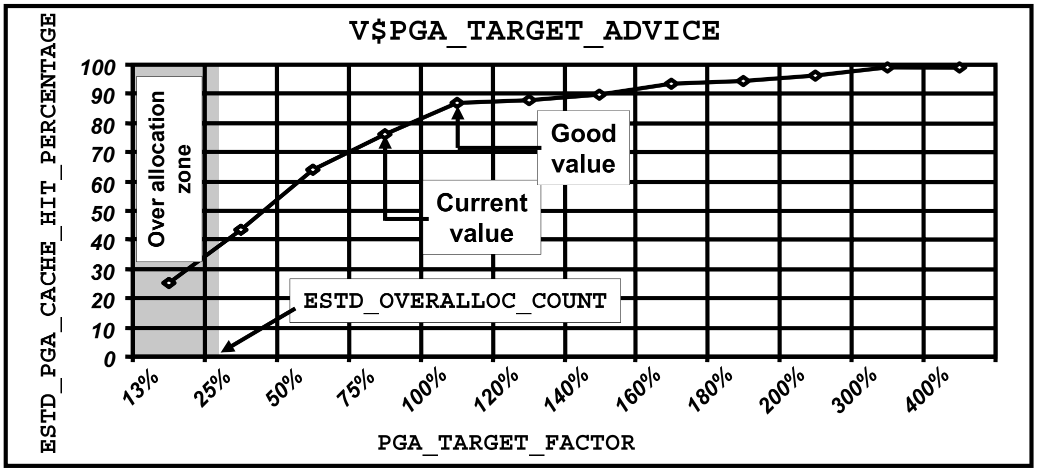
In addition, you can monitor the top wait event sections of your AWR or Statspack reports looking for the following two events: `direct path read temp`, and `direct path write temp`.

These two events are triggered when a session is reading buffers from or writing buffers to disk directly into the PGA because work areas are too large to fit in memory and are written to disk. These are the biggest waits for large data warehouse sites. However, if the workload is not a DSS workload, then examine why this is happening.

You can do so by looking at the SQL statement currently being run by the session experiencing waits to see what is causing the sorts. Query `V$TEMPSEG_USAGE` to find the SQL statement that is generating the sort. Also, query the statistics from `V$SESSTAT` for the session to determine the size of the sort. See whether it is possible to reduce the use of work areas by tuning the SQL statement. In addition, investigate whether to increase `PGA_AGGREGATE_TARGET`.

PGA Target Advice Statistics

- V$PGA_TARGET_ADVICE predicts how cache hit percentages shown in V$PGASTAT evolve.
- STATISTICS_LEVEL must be set to at least TYPICAL.



PGA Advice Statistics

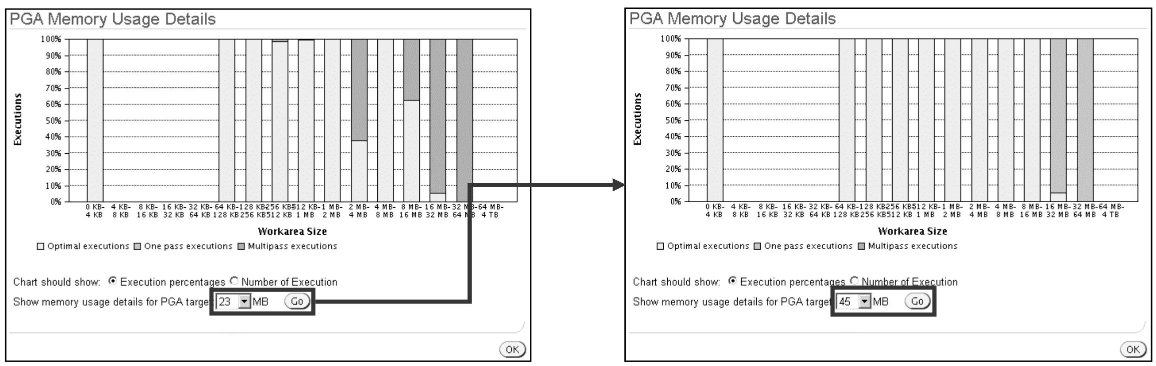
V$PGA_TARGET_ADVICE predicts how the cache hit percentage displayed by the V$PGASTAT performance view would be impacted if the value of the PGA_AGGREGATE_TARGET parameter is changed. The prediction is performed for various values of the PGA_AGGREGATE_TARGET parameter, selected around its current value. The advice statistic is generated by simulating the past workload run by the instance.

The graph in the slide shows how the PGA cache hit percentage metric improves as the value of PGA_AGGREGATE_TARGET increases. This cache hit percentage metric reflects the average percentage of SQL work areas that are able to run cache. The overallocation zone indicates that PGA_AGGREGATE_TARGET is insufficient to meet the minimum PGA memory needs, and forces the memory manager to allocate more PGA memory than the limit that you set. The ESTD_OVERALLOC_COUNT column predicts the number of overallocations. The good value for PGA_AGGREGATE_TARGET is reached when the cache hit percentage improves marginally compared to the extra memory need. Ideally, PGA_AGGREGATE_TARGET should be set to the maximum value possible in the region beyond the overallocation zone.

Note: The content of V$PGA_TARGET_ADVICE is empty if PGA_AGGREGATE_TARGET is not set, or if STATISTICS_LEVEL is set to BASIC. Base statistics for this view are reset at instance startup and when the value of the PGA_AGGREGATE_TARGET initialization parameter is dynamically modified.

PGA Target Advice Histograms

- **`V$PGA_TARGET_ADVICE_HISTOGRAM` predicts how histograms shown in `V$SQL_WORKAREA_HISTOGRAM` evolve.**
- **`STATISTICS_LEVEL` must be set to at least `TYPICAL`.**



ORACLE

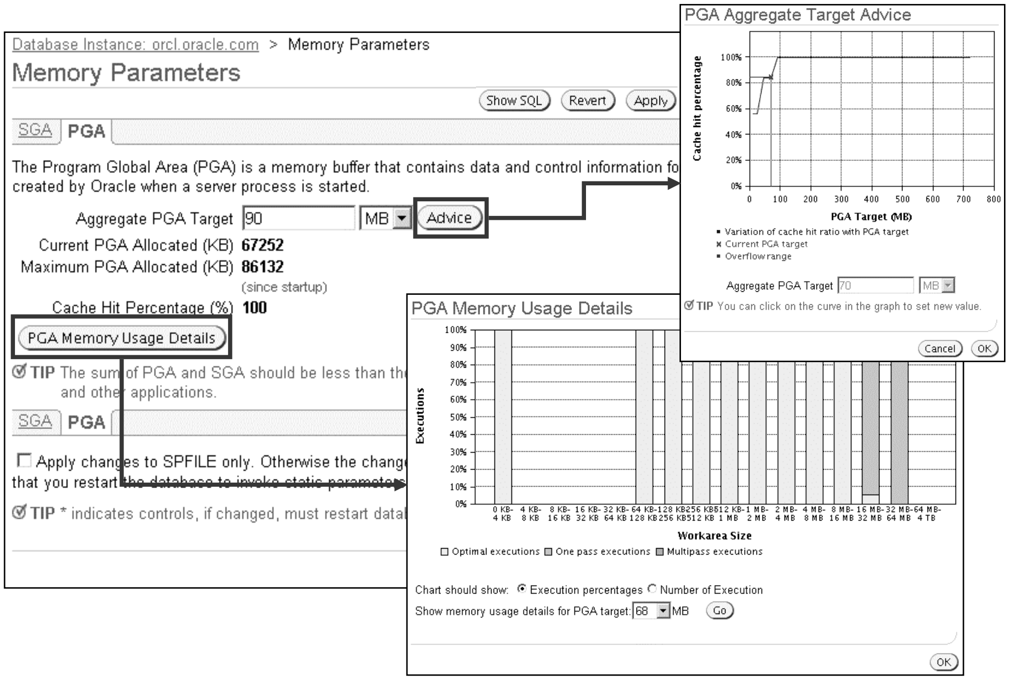
PGA Target Advice Histograms

V$PGA_TARGET_ADVICE_HISTOGRAM predicts how histograms displayed by V$SQL_WORKAREA_HISTOGRAM would be impacted if the value of the PGA_AGGREGATE_TARGET parameter is changed. This prediction is performed for various values of the PGA_AGGREGATE_TARGET parameter, selected around its current value. The advice statistic is generated by simulating the past workload run by the instance to determine the number of work areas executed with optimal memory size, one-pass memory size, and multi-pass memory size.

The slide shows you a graphical interpretation of the content of V$PGA_TARGET_ADVICE_HISTOGRAM for two different values of the PGA_TARGET_FOR_ESTIMATE column used to materialized the corresponding PGA_AGGREGATE_TARGET value used for estimation.

The output shows that increasing PGA_AGGREGATE_TARGET by a factor of 1.55 will allow all work areas under 16 MB to execute in optimal mode.

Automatic PGA and Enterprise Manager



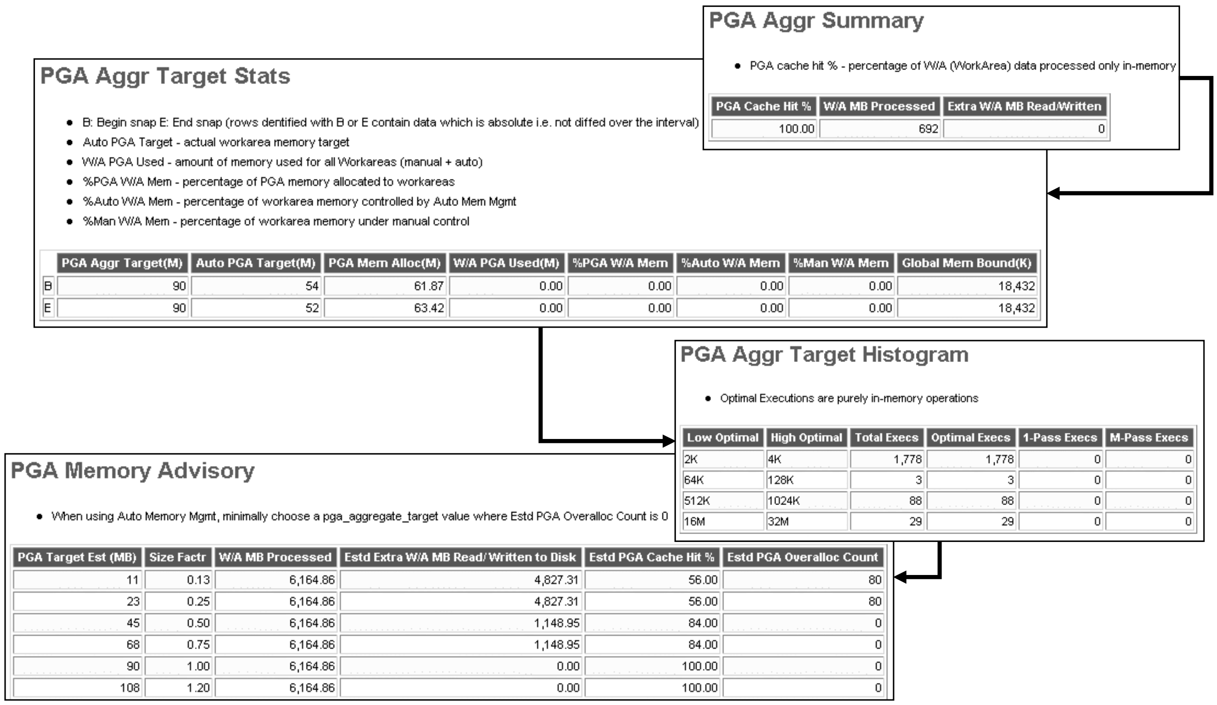
Automatic PGA and Enterprise Manager

Use the PGA tab on the Memory Parameters page to set the Aggregate PGA Target value. You can choose the unit of measure from the drop-down list in bytes, kilobytes, megabytes, or gigabytes. You can also see the values for the Current PGA Allocated, the Maximum PGA Allocated, and the Cache Hit Percentage.

You can also click Advice to display a chart that shows the cache hit percentage and the PGA target in megabytes that provides advice on the value you can set for the Aggregate PGA Target value.

You can click PGA Memory Usage Details to display a chart showing the various types of executions in the available work area size depending on different possible PGA targets.

Automatic PGA and AWR Reports



PGA Aggr Summary

- PGA cache hit % - percentage of W/A (WorkArea) data processed only in-memory

PGA Cache Hit %	W/A MB Processed	Extra W/A MB Read/Written
100.00	692	0

PGA Aggr Target Stats

- B: Begin snap E: End snap (rows dentified with B or E contain data which is absolute i.e. not diffed over the interval)
- Auto PGA Target - actual workarea memory target
- W/A PGA Used - amount of memory used for all Workareas (manual + auto)
- %PGA W/A Mem - percentage of PGA memory allocated to workareas
- %Auto W/A Mem - percentage of workarea memory controlled by Auto Mem Mgmt
- %Man W/A Mem - percentage of workarea memory under manual control

	PGA Aggr Target(M)	Auto PGA Target(M)	PGA Mem Alloc(M)	W/A PGA Used(M)	%PGA W/A Mem	%Auto W/A Mem	%Man W/A Mem	Global Mem Bound(K)
B	90	54	61.87	0.00	0.00	0.00	0.00	18,432
E	90	52	63.42	0.00	0.00	0.00	0.00	18,432

PGA Aggr Target Histogram

- Optimal Executions are purely in-memory operations

Low Optimal	High Optimal	Total Execs	Optimal Execs	1-Pass Execs	M-Pass Execs
2K	4K	1,778	1,778	0	0
64K	128K	3	3	0	0
512K	1024K	88	88	0	0
16M	32M	29	29	0	0

PGA Memory Advisory

- When using Auto Memory Mgmt, minimally choose a pga_aggregate_target value where Estd PGA Overalloc Count is 0

PGA Target Est (MB)	Size Factr	W/A MB Processed	Estd Extra W/A MB Read/ Written to Disk	Estd PGA Cache Hit %	Estd PGA Overalloc Count
11	0.13	6,164.86	4,827.31	56.00	80
23	0.25	6,164.86	4,827.31	56.00	80
45	0.50	6,164.86	1,148.95	84.00	0
68	0.75	6,164.86	1,148.95	84.00	0
90	1.00	6,164.86	0.00	100.00	0
108	1.20	6,164.86	0.00	100.00	0

ORACLE

Automatic PGA and AWR Reports

You can find the main information (shown in the slide) relating to the PGA statistics in the Advisory Statistics section of the AWR report.

In the PGA statistics in the Process Memory Summary section of the AWR report, you can also find the following statistics: maximum PGA allocation size at snapshot time and the historical maximum allocation for still-connected processes.

In the Instance Activity Statistics section of the AWR report, you can also find the following total, per second, and per transaction statistics: session PGA memory, and session PGA maximum memory.

Temporary Tablespace Management: Overview

- Temporary data generated by a database include:
  - Bitmap merges
  - Hash-join
  - Bitmap index creation
  - Sort
  - Temporary LOBs
  - Global temporary tables
- Data persists for the duration of a transaction or session.
- High concurrency of the space management operation is critical.
- Media and instance recovery is not required.

Temporary Tablespace Management: Overview

A temporary tablespace is used to store transient data generated explicitly by the user and implicitly by the system. The data stored in a temporary tablespace are predominantly from hash-join, sort, bitmap merges, bitmap index creation operations as well as temporary LOBs and global temporary tables.

In DSS and OLAP environments, the efficient management of transient data is central to the execution of queries; therefore, the performance of temporary tablespace is critical.

Temporary Tablespace: Best Practice

Using locally managed temporary tablespace:

- Allows high-concurrency space management.
  - At steady state, all space metadata is cached in SGA.
  - Operations are serialized by the SGA latch.
- Allows faster writes to temp files. Redo generated on temporary blocks is not written to disk.
- Makes `READ ONLY` databases possible.

Temporary Tablespace: Best Practice

Tablespaces are classified as temporary and permanent. The space management in temporary tablespaces differs from that in permanent tablespaces in the following ways:

- The temporary tablespaces consist of temp files rather than data files that constitute the permanent tablespace.
- A temporary tablespace allows high-concurrency space management. In a steady state, all the space metadata is cached in the SGA. This allows for completely in-memory space management operations. The space management operations in permanent tablespaces are serialized by databasewide ST enqueue, whereas in a locally managed temporary tablespace, they are serialized by instance-specific SGA latches held for a very short duration.
- Redo logging: The redo generated on temp files is automatically discarded after modifying the block. It is not written to the disk. This allows for faster writes to temp files.
- Read-only database: All the metadata required by the temporary tablespace is stored in the tablespace itself. No files outside the temporary files need to be modified, thereby making read-only databases possible.

Oracle Corporation recommends the use of locally managed temporary tablespaces, due to the inherent performance benefits.

Configuring Temporary Tablespace

- **Locally managed temporary tablespaces are uniform-extent tablespaces.**
- **1 MB to 10 MB extent size:**
  - **For DSS, OLAP applications involving huge work areas**
  - **Large temporary LOBs are predominant**
- **64 KB or multiple less than 1 MB:**
  - **Small global temporary tables are predominant**
  - **OLTP**
- **Use `V$TEMPSEG_USAGE` to monitor space usage and workload distribution.**

Configuring Temporary Tablespace

While creating temporary space for the database, the tablespace extent size is important to consider, because of its impact to the performance. The extent size can vary anywhere from two data blocks to over 2 GB with 1 MB being the default. Choosing the correct extent size involves the usual space versus time trade-off. Very small extent sizes impact read/write performance as well as increase the number of space management operations on large segments. Very large extent sizes can cause space to be wasted in the last extent allocated to the segment, with no improvement in performance. It is important to understand the types of workloads using the temporary tablespace:

- Temporary LOBs: Temporary LOBs are created explicitly by the user as well as implicitly by the database to store transient unstructured data. If the temporary space usage is dominated by large temporary LOBs, then larger extent sizes should be used. Oracle Corporation recommends using 1 MB to 10 MB for the extent size.
- DSS: The DSS workload is typically dominated by complex queries performing intensive sort and hash-join operations. Because the data in the work area is written to the temporary segments in multiples of 64 KB, it is advisable to choose an extent size that is a multiple of 64 KB. The general rule for good performance and efficient space usage is to set an extent size of 1 MB.

Configuring Temporary Tablespace (continued)

- Global temporary tables: These are transient tables created by the user as well as the system in the temporary tablespace. Each global temporary table requires at least one extent to be allocated. If the volume of data loaded into the temporary tables is pretty small, then choosing smaller multiples of 64 KB for the extent size should avoid space wastage. This is also suitable for OLTP.

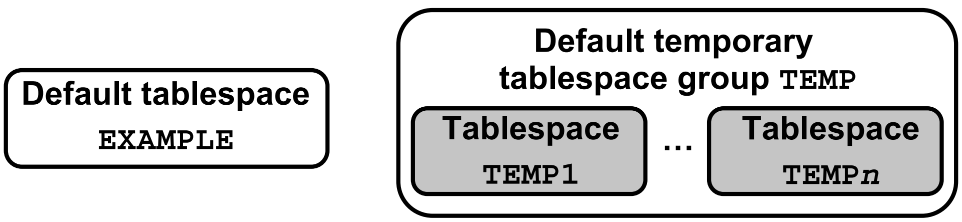
The distribution of workload can be monitored through the `V$TEMPSEG_USAGE` dictionary view:

```
SQL> SELECT session_num, username, segtype, blocks,
tablespace FROM V$TEMPSEG_USAGE;
SESSION_NUM USERNAME SEGTYPE BLOCKS TABLESPACE
----------- -------- --------- ------ ----------
        101 SCOTT    SORT         128 TEMP
        102 SCOTT    LOB_DATA     128 TEMP
        103 SYS      SORT         256 TEMP
        104 BLAKE    LOB_DATA     128 TEMP
```

The `SEGTYPE` column indicates what type of segment is using the space: `SORT`, `HASH`, `INDEX`, `LOB_DATA`, or `DATA`. On the basis of the distribution of segment sizes, an appropriate extent size can be chosen.

Temporary Tablespace Group: Overview

- **Groups multiple temporary tablespaces**
- **Characteristics:**
  - **At least one temporary tablespace**
  - **Same namespace as tablespaces**
  - **Created implicitly on first assignment**
  - **No explicit deletion**



Temporary Tablespace Group: Overview

A temporary tablespace group can be thought of as a shortcut for a list of temporary tablespaces. A temporary tablespace group consists of only temporary tablespaces.

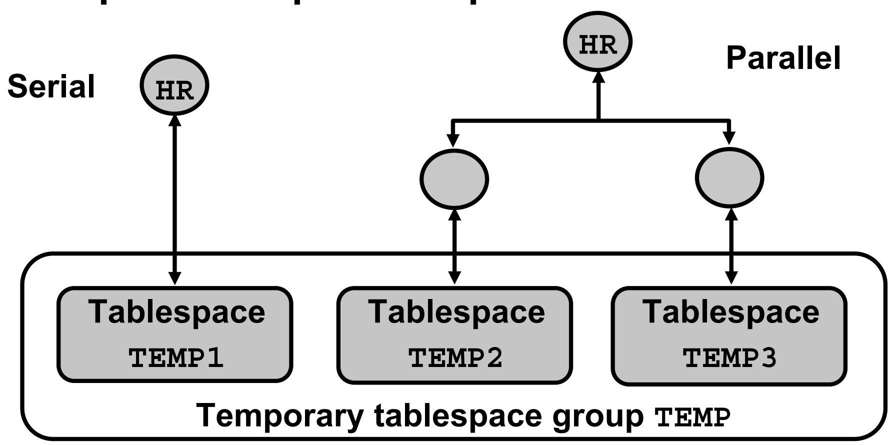
A temporary tablespace group has the following properties:

- It contains at least one temporary tablespace. There is no explicit limit on how many tablespaces are contained in a group.
- It has the same namespace as tablespaces. It is not possible for a tablespace and a temporary tablespace group to have the same name.
- A temporary tablespace group name can appear wherever a temporary tablespace name appears (for example, while assigning a default temporary tablespace for the database, or assigning a temporary tablespace for a user).
- It is not explicitly created. It is created implicitly when the first temporary tablespace is assigned to it, and it is deleted when the last temporary tablespace is removed from the group.

Temporary Tablespace Group: Benefits

Enables a user to use multiple temporary tablespaces:

- **Same user in multiple sessions**
- **One particular parallel operation**



Temporary Tablespace Group: Benefits

This feature has the following benefits:

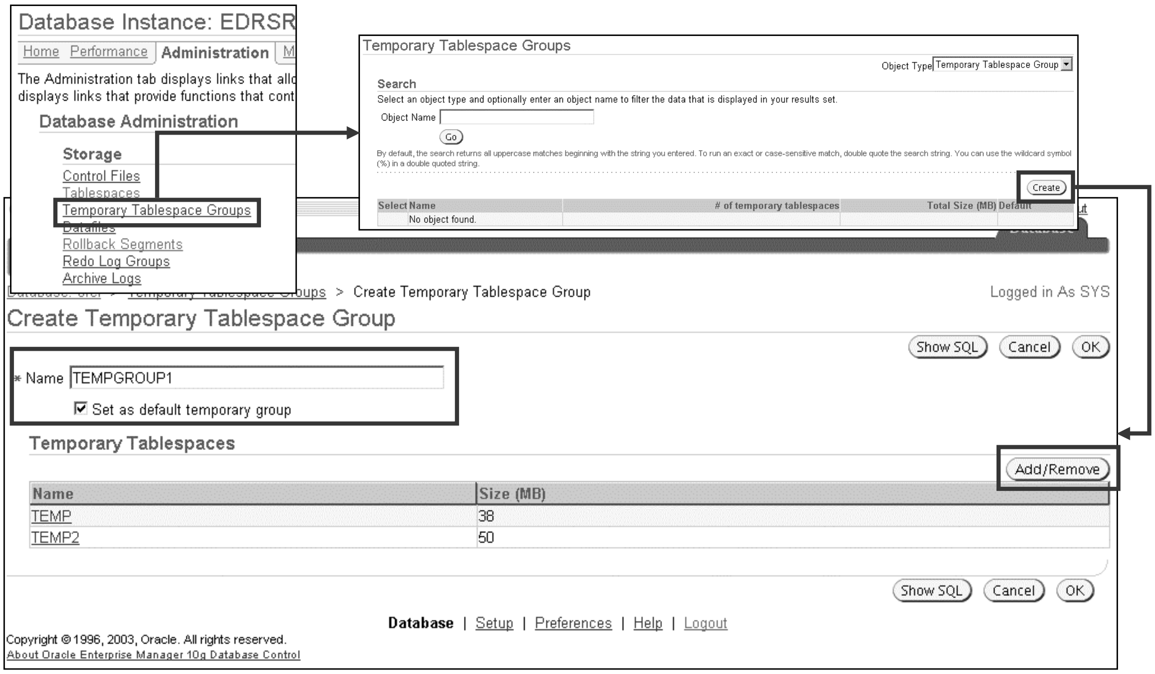
- Enables one particular user to use multiple temporary tablespaces in different sessions at the same time
- Enables the slave processes in a single parallel operation to use multiple temporary tablespaces
- Enables multiple default temporary tablespaces to be specified at the database level. Sort operations are assigned to these tablespaces in a round-robin fashion.

Therefore, you can now define more than one default temporary tablespace, and a single parallel SQL operation can use more than one temporary tablespace for sorting. This prevents large tablespace operations from running out of temporary space. This is especially relevant with the introduction of bigfile tablespaces.

The primary purpose of a temporary tablespace group is to increase the addressability of the temporary space. A single temporary tablespace can have a maximum of four billion blocks. This is 8 TB for a block size of 2 KB and 64 TB for a block size of 16 KB. With temporary tablespace groups, the addressability is increased to several petabytes.

Note: A single, non-parallel SQL statement uses only one temporary tablespace; a single temporary segment does not span multiple tablespaces.

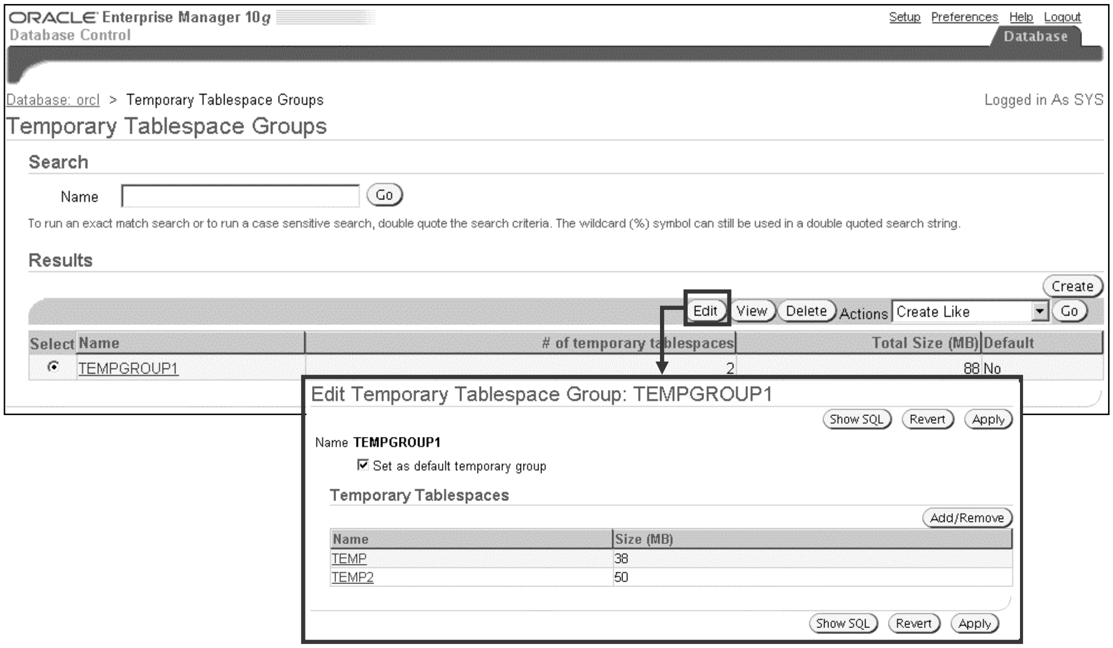
Creating Temporary Tablespace Groups



Creating Temporary Tablespace Groups

You can create and maintain temporary tablespace groups by using Database Control. On the Database Control home page, click the Administration tab to open the Administration tabbed page. On this page, click the Temporary Tablespace Groups link in the Storage section. This displays the Temporary Tablespace Groups page, where you can see a list of existing tablespace groups. On this page, you can view and edit existing tablespace groups. When you click the Create button, the Create Temporary Tablespace Group page is displayed. Enter the name of the new group in the Name field, and specify whether or not you want this new group to be set as the default temporary group. You can do this by selecting the “Set as default temporary group” option. After selecting this option, you need to add existing temporary tablespaces to the group. Click the Add/Remove button, and select the temporary tablespaces that belong to the group. Then, click the OK button to create the new temporary tablespace group.

Maintaining Temporary Tablespace Groups



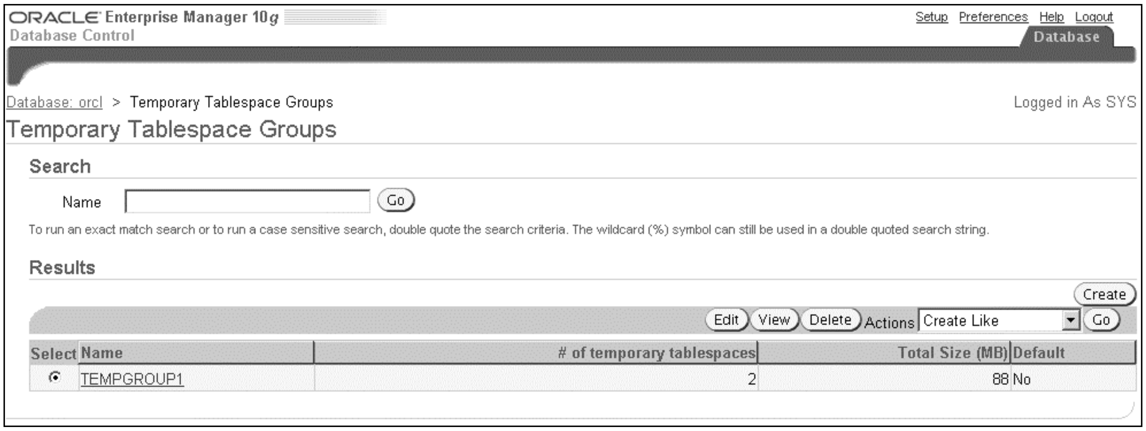
Maintaining Temporary Tablespace Groups

You can maintain temporary tablespace groups by using Database Control. On the Database Control home page, click the Administration tab. On the Administration tabbed page, click the Temporary Tablespace Groups link in the Storage section. This displays the Temporary Tablespace Groups page, where you can see the list of existing tablespace groups.

Select the tablespace group that you want to maintain, and click the Edit button. The Edit Temporary Tablespace Group page opens, where you can add and remove temporary tablespaces by clicking the Add/Remove button. After doing so, click the Apply button for your changes to take effect.

Note: If you remove all temporary tablespaces from a temporary tablespace group, the group is also removed implicitly.

Data Dictionary Changes




```
SELECT group_name, tablespace_name
FROM DBA_TABLESPACE_GROUPS;
```


Data Dictionary Changes

The DBA_TABLESPACE_GROUPS view lists all tablespaces contained inside each temporary tablespace group.

Note: The three views ALL_USERS, USER_USERS, and DBA_USERS have a column named TEMPORARY_TABLESPACE. This column contains either the name of a tablespace or the name of a temporary tablespace group.

Monitoring Temporary Tablespace

- Use `V$TEMPSEG_USAGE` to monitor space usage and workload distribution:

```
SELECT session_num, username, segtype, blocks, tablespace
FROM   V$TEMPSEG_USAGE;
```

- Use `V$SORT_SEGMENT` to determine space usage percentage:

```
SELECT (s.tot_used_blocks/f.total_blocks)*100 as pctused
FROM (SELECT SUM(used_blocks) tot_used_blocks
      FROM   V$SORT_SEGMENT
      WHERE  tablespace_name='TEMP') s,
     (SELECT SUM(blocks) total_blocks
      FROM   DBA_TEMP_FILES
      WHERE  tablespace_name='TEMP') f;
```

ORACLE

Monitoring Temporary Tablespace

The distribution of workload can be monitored through the V$TEMPSEG_USAGE dictionary view. The SEGTYPE column in the first example indicates what type of segment is using the space. It is one of SORT, HASH, INDEX, LOB_DATA, LOB_INDEX, and DATA. The BLOCKS column gives you the number of blocks currently used by the operation on disk. On the basis of the distribution of segment sizes, an appropriate extent size can be chosen.

The query in the second example can be used to show the percentage of space used in a locally managed temporary tablespace.

Note: The temporary segment of a given temporary tablespace is created at the time of the first operation that has to write to disk to free up space in memory. Multiple transactions that need space on disk can share the same segment; however, they cannot share the same extent. The segment expands by allocating new extents. The extents are not deallocated while the instance is running, but are marked as free and can be reused as required. Therefore, the segment grows to a certain steady state or until it fills the tablespace.

Summary

In this lesson, you should have learned how to:

- Diagnose PGA memory issues
- Size the PGA memory
- Diagnose temporary space issues
- Specify temporary tablespace parameters for efficient operation

ORACLE

Practice 13 Overview: Tuning PGA Memory

This practice covers the following topics:

- **Tuning `PGA_AGGREGATE_TARGET`**
- **Tuning temporary tablespace performance**

14

Tuning Block Space Usage

Objectives

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

- Tune segment space management
- Convert from dictionary-managed tablespaces
- Convert to Automatic Segment Space Management
- Tune block space management
- Diagnose and correct row migration
- Diagnose table fragmentation
- Compare characteristics of bigfile and smallfile tablespaces

Space Management

Space is managed at three levels:

- **Files (OS, ASM, raw partitions) assign disk space to tablespaces.**
- **Extents are used to allocate file space to segments in a tablespace.**
- **Blocks are used to organize the space inside data objects.**

Space Management

The Oracle database organizes physical space at three levels. Files (including raw partitions) are used to assign physical disk space to tablespaces with the rule that a file can be assigned to one and only one tablespace. A tablespace may be assigned many files.

Each file is divided into extents. Extents are allocated to segments as needed. Segment types include tables, indexes, LOBs, partitions, and undo segments. As segments grow, more extents are allocated until there are no more free extents in the files. Extents are created in sets of blocks. An extent may be as small as four blocks or as large as the file (under 2 GB). An extent is a subdivision of a file and cannot span file boundaries.

After an extent is allocated to a segment, it is not returned to the tablespace for reuse until the segment is dropped, except in a segment shrink operation. Segments place rows, chunks, or undo records into blocks following certain rules set by parameters on the segment. Blocks can be unformatted, empty, or have some measure of data in them.

Extent Management

Extents are allocated in two ways:

- **Dictionary managed**
  - Only supported for backward compatibility
  - Extents managed in the `UET$` and `FET$` dictionary tables
  - Recursive SQL
- **Locally managed**
  - Extents managed in the file header bitmap
  - No undo created on extent operations
  - Possible contention on file header blocks

ORACLE

Extent Management

The database manages extents in two ways. The first is dictionary managed. This method was used exclusively until local management was introduced in Oracle8*i*. With dictionary management, extents are allocated from tablespace free extents listed in the free extent table (`FET$`) and added to the extent table (`UET$`). Any extent operations, such as `CREATE` or `DROP` segments, generate recursive SQL, undo, redo, and I/O to update these tables. Dictionary management is supported only for backward compatibility. The `DBMS_SPACE_ADMIN` package includes the `TABLESPACE_MIGRATE_TO_LOCAL` procedure to convert dictionary management to local management.

Local management of extents is accomplished through bitmaps located in the file header. Extent operations only switch bits and each bit covers one or more blocks in the file. This method has a much lower overhead than dictionary-managed extents. Local management of extents is the recommended method. Even local extent management can see contention in some cases where there is a high rate of extent operations. For example, in a scenario where there is a high rate of DML operations with many separate transactions, and undo segments are constantly adding and releasing extents, you may see buffer busy waits on file header blocks in the undo tablespace. The solution is to create more files, increasing the number of file header bitmaps that are being used.

Locally Managed Extents

- **Create a locally managed tablespace:**

```
SQL> CREATE TABLESPACE user_data_1
  2  DATAFILE
  3  '/oracle9i/oradata/db1/lm_1.dbf'
  4  SIZE 100M
  5  EXTENT MANAGEMENT LOCAL
  6  UNIFORM SIZE 2M;
```

- **Default extent management is local.**

Locally Managed Extents

Create locally managed tablespaces for the objects that extend continuously.

A locally managed tablespace manages its own extents and maintains a bitmap in each data file to keep track of the free or used status of blocks in that data file. Each bit in the bitmap corresponds to a block or a group of blocks. When an extent is allocated or freed for reuse, the bitmap values change to show the new status of the blocks. These changes do not generate undo information because they do not update tables in the data dictionary.

The Database Creation Assistant (DBCA) creates the system tablespace as locally managed by default. The `CREATE DATABASE` command defaults to a dictionary-managed system tablespace. If the system tablespace is locally managed, then a dictionary-managed tablespace cannot be created in the database. A dictionary-managed tablespace can be added to a database with a locally managed tablespace with transportable tablespaces, but the tablespace cannot be made read/write.

Pros and Cons of Large Extents

- **Pros:**
  - **Are less likely to extend dynamically**
  - **Deliver a small performance benefit**
  - **Enable the server process to read the entire extent map with a single I/O operation**
- **Cons:**
  - **Free space may not be available**
  - **May contain unused space**

Pros and Cons of Large Extents

To ease space management, you should create objects with appropriately sized extents. As a general rule, larger extents are preferred over smaller extents.

Advantages of Large Extents

- Large extents avoid dynamic extent allocation, because segments with larger extents are less likely to need to be extended. This benefit is reduced with locally managed extents.
- Larger extents sometimes provides a small performance benefit for full table scans (including fast full scans) when the server process can read one large extent from disk with fewer reads than would be required to read many small extents. This situation can be avoided by matching extent sizes to the I/O and space allocation sizes, and the performance cost of having many extents in a segment is minimized. As the extent size increases, the benefit is reduced. However, for a table that never has a full table scan operation, it makes no difference in terms of query performance whether the table has few extents or many extents. The performance of searches using an index is not affected by the size of the extents in the index.

Pros and Cons of Large Extents (continued)

Advantages of Large Extents (continued)

- Extent maps list all the extents for a certain segment. When `MAXEXTENTS` is set to `UNLIMITED`, additional extent map blocks are created as needed. For best performance, the extent map is in on block and read with a single I/O. Performance degrades a small amount if multiple I/Os are necessary to get the extent map as in the case of a full table scan. The number of additional I/Os required for the additional extent map blocks are negligible by the time the table reaches the size that requires the additional maps. Also, a large number of extents can degrade data dictionary performance, because each extent uses space in the dictionary cache.

Disadvantages of Large Extents

- Large extents require more contiguous blocks; therefore, the server process may have difficulty finding enough contiguous space to store them. The segment space advisor may be used to shrink tables that do not use all the space that has been allocated to them. The `ALTER TABLESPACE COALESCE` command is used with dictionary-managed extents to combine adjacent free extents. Locally managed extents avoid this issue by automatically combining adjacent free extents.
- The DBA sizes the segment to allow for growth, so some of the space allocated to the segment is not used initially.

To determine whether to allocate a few large extents or many small extents, consider how the benefits and drawbacks of each would affect your plans for the growth and use of your tables.

Migrating the SYSTEM Tablespace to a Locally Managed Tablespace

- Use the DBMS_SPACE_ADMIN package:

```
SQL> EXECUTE DBMS_SPACE_ADMIN. -
  2  TABLESPACE_MIGRATE_TO_LOCAL('SYSTEM');
```

- The locally managed SYSTEM tablespace restrictions are enforced by the migration procedure.
- Migration is possible only when the system is in RESTRICTED mode whereas all tablespaces other than SYSTEM, UNDO, and TEMP are in READ ONLY mode.
- Convert other dictionary-managed tablespaces with the same procedure before converting SYSTEM.

Migrating the SYSTEM Tablespace to a Locally Managed Tablespace

You can use Database Control or the DBMS_SPACE_ADMIN package to migrate any tablespace from dictionary managed to locally managed. The following statement performs the migration for the SYSTEM tablespace:

```
SQL> EXEC DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL('SYSTEM').
```

All the locally managed SYSTEM tablespace restrictions must be met in order for this procedure to be successful. If these restrictions are not met, the procedure will not perform the migration and will abort with an error. Possible error messages are:

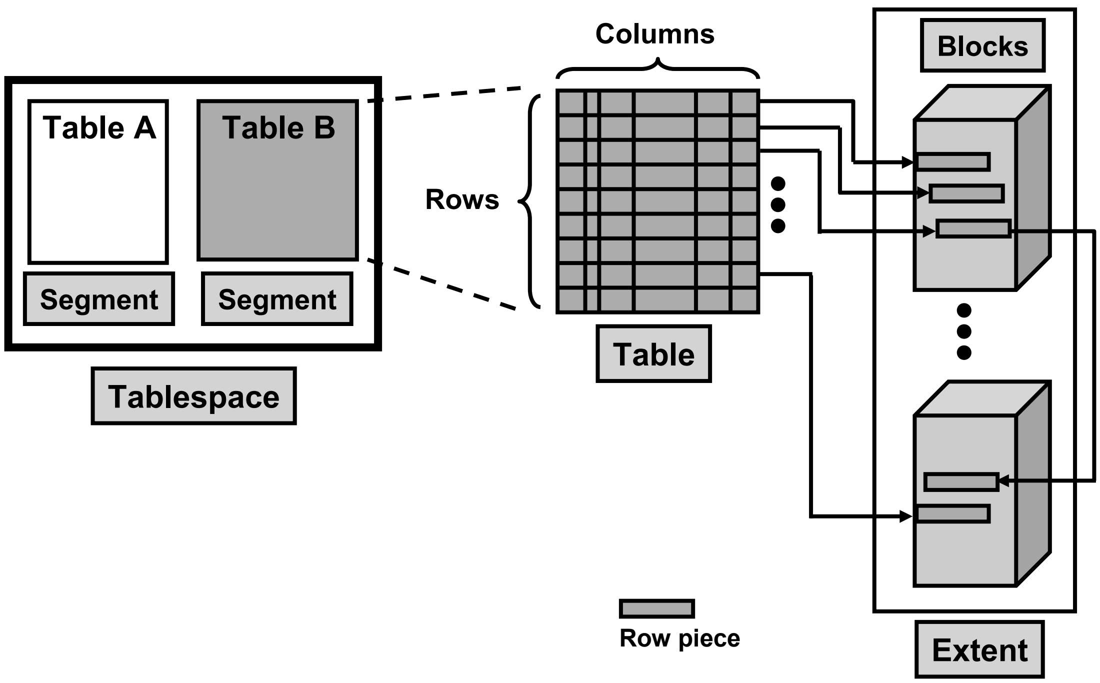
```
ORA-03240 User's temporary tablespace same as tablespace being migrated
ORA-10642 Found rollback segments in dictionary managed tablespaces
ORA-10643 Database should be mounted in restricted mode and Exclusive mode
ORA-10644 SYSTEM tablespace cannot be default temporary tablespace
```

After the SYSTEM tablespace is successfully migrated, you cannot migrate it back into a dictionary-managed tablespace. Also, remember that any remaining dictionary-managed tablespace cannot be altered to READ WRITE mode, nor can it be migrated into a locally managed tablespace. If you want to keep your dictionary-managed tablespaces READ WRITE, you must migrate them before you migrate the SYSTEM tablespace.

Migrating the `SYSTEM` Tablespace to a Locally Managed Tablespace (continued)

For any tablespace being migrated, the operation is performed online and in-place. If the tablespace was fragmented before the migration, then it will still be fragmented afterward. With this procedure, multiple tablespaces may be migrated in parallel.

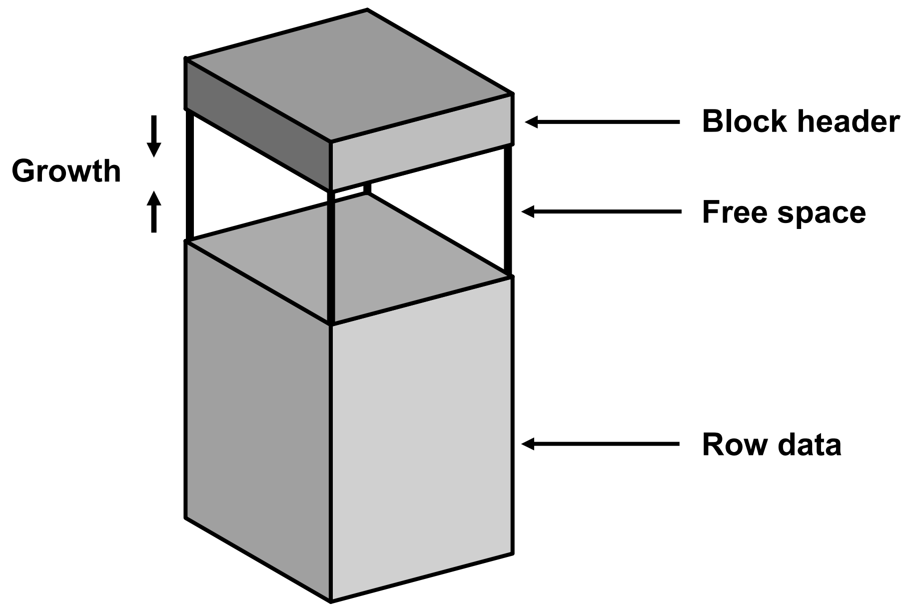
How Table Data Is Stored



How Table Data Is Stored

When a table is created, a segment is created to hold its data. A tablespace contains a collection of segments. Logically, a table contains rows of column values. A row is ultimately stored in a database block in the form of a row piece. It is called a row piece because under some circumstances the entire row may not be stored in one place. This happens when an inserted row is too large to fit into a single block or when an update causes an existing row to outgrow its current space. Rows that are too large to fit into any block are called chained rows, because they are chained across multiple blocks. A row that is moved to a block because it grew too large for the block is called a migrated row.

Anatomy of a Database Block



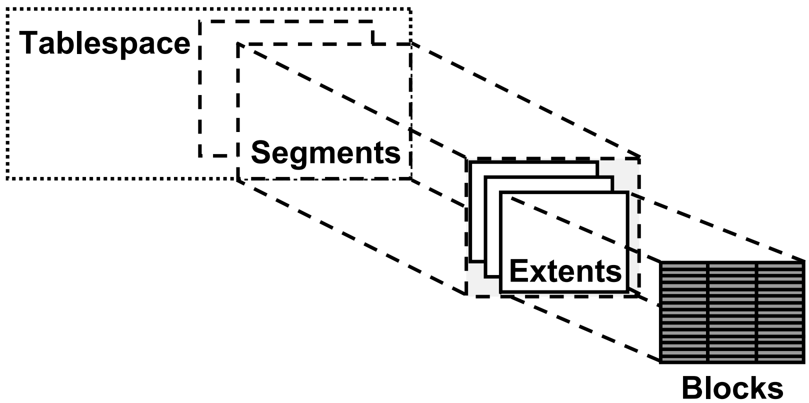
Database Block: Contents

Database data blocks contain the following:

- Block header: The block header contains the segment type (such as table or index), data block address, table directory, row directory, and transaction slots that are used when modifications are made to rows in the block. The block header grows downward from the top.
- Row data: This is the actual data for the rows in the block. Row data space grows upward from the bottom.
- Free space: Free space is in the middle of the block. This enables the header and the row data space to grow when necessary. Row data takes up free space as new rows are inserted or columns of existing rows are updated with larger values. The examples of events that cause header growth are when the row directory needs more row entries or more transaction slots are required than initially configured. Initially, the free space in a block is contiguous. However, deletions and updates may fragment the free space in the block. The free space in the block is coalesced by the server processes when necessary.

Minimize Block Visits

Minimize block visits by:
- Using a larger block size
- Packing rows tightly
- Preventing row migration



Minimize Block Visits

One of the database tuning goals is to minimize the number of blocks visited. The developer contributes to this goal by tuning the application and SQL statements. The DBA reduces block visits by:

- Using a larger block size
- Packing rows as closely as possible into blocks
- Preventing row migration

Row migration occurs when rows are updated. If the updated row grows, and can no longer fit in the block, then the row is moved to another block. A pointer is left in the original row location, that points to the new location.

Unfortunately for the DBA, the last two goals conflict: As more data is packed into a block, the likelihood of migration increases.

The DB_BLOCK_SIZE Parameter

The database block size:

- **Is defined by the DB_BLOCK_SIZE parameter**
- **Is set when the database is created**
- **Becomes the default block size for tablespaces and buffer cache**
- **Is the minimum I/O unit for data file reads**
- **Is 8 KB by default; up to 32 KB allowed on most platforms**
- **Cannot be changed easily**
- **Should be an integer multiple of the operating system (OS) block size**
- **Should be less than or equal to the OS I/O size**

ORACLE

The DB_BLOCK_SIZE Parameter

- When the database is created, the block size is determined by the value of the DB_BLOCK_SIZE parameter.
- It is the minimum I/O unit for data file reads.
- The default block size is 8 KB.
- Most operating systems allow block sizes of up to 32 KB. Check your operating system–specific documentation, specifically the Oracle database installation and configuration guides, to determine the maximum database block size for your platform.
- The size cannot be changed without re-creating or duplicating the database. This makes it difficult to test applications with different block sizes. The database can have multiple block sizes. However, the base block size (that of the SYSTEM tablespace) cannot be changed after database creation.
- The database block size should be an integer multiple of the operating system block size.
- If your operating system reads the next block during sequential reads and your application performs many full table scans, then the database block size should be large, but should not exceed the operating system I/O size.

Small Block Size: Considerations

- Advantages:
  - Reduces block contention
  - Is good for small rows
  - Is good for random access
- Disadvantages:
  - Has a relatively large space overhead
  - Has a small number of rows per block
  - Can cause more index blocks to be read

Small Block Size: Considerations

Advantages

- Small blocks reduce block contention because there are fewer rows per block.
- Small blocks are good for small rows.
- Small blocks are good for random access. If it is unlikely that a block will be reused after it is read into memory, then a smaller block size makes more efficient use of the buffer cache. This is especially important when memory resources are scarce, because the size of the database buffer cache is limited.

Disadvantages

- Small blocks have relatively large space overhead. There is about 100 bytes of overhead required for each block, whether a 2 KB block or a 32 KB block.
- Depending on the size of the row, you may end up storing only a small number of rows per block. This can cause additional I/Os. Large rows may be chained.
- Small blocks can cause more index blocks to be read.

Performance

For random access to a large object, as in an OLTP environment, small blocks are favored.

Large Block Size: Considerations

- **Advantages:**
  - Less space overhead
  - Good for sequential access
  - Good for very large rows
  - Better performance of index reads
- **Disadvantages:**
  - Increases block contention
  - Uses more space in the buffer cache

ORACLE

Large Block Size: Considerations

Advantages

- There is less overhead and thus more room to store useful data.
- Large blocks are good for sequential reads.
- Large blocks are good for very large rows.
- Larger blocks improve the performance of index reads. The larger blocks can hold more index entries in each block, which reduces the number of levels in large indexes. Fewer index levels mean fewer I/Os when traversing the index branches.

Disadvantages

- A large block size is not good for index blocks used in an OLTP environment, because they increase block contention on the index leaf blocks.
- Space in the buffer cache is wasted if you randomly access small rows and have a large block size. For example, with a block size of 8 KB and a row size of 50 bytes, you waste 7,950 bytes in the buffer cache when doing a random access.

Performance

Sequential access to large amounts of data, as in a decision support system (DSS) environment, prefers large blocks.

Block Allocation

- **When an `INSERT` or `UPDATE` operation requires more space, a block must be found with adequate space.**
- **Two methods:**
  - **Free lists**
  - **Automatic Segment Space Management (ASSM)**

Block Allocation

When an insert or update operation requires more space, a block must be found with adequate space. There are two methods for assigning an operation to a block: free lists and Automatic Segment Space Management (ASSM). Both methods are available with locally managed tablespaces.

Free Lists

Free list–managed space characteristics:

- Segment header blocks hold free lists.
- Blocks are added to and removed from the free lists.
- Free lists are searched for available blocks.
- Segment headers are pinned for the search and update of free lists.

Free Lists

A free list is a linked list of blocks that are available in the segment for inserts. As the blocks are filled they are removed from the free list. If rows are deleted and the used space decreases past the PCTUSED threshold, the block returns to the head of the free list. The header block is pinned while the free list is changed. A process that requires a block for INSERT pins the header block while it searches the free list. When multiple processes are inserting into the same segment concurrently, there can be contention for the free list. Free-list contention is shown by buffer busy waits on data header blocks. Free-list contention is common in high-insert environments, when multiple processes attempt to search the free list at the same time.

Block Space Management

Each segment has parameters that control the space usage inside a block.

- **For a table:**
  - `PCTFREE`: **Amount of space reserved for updates**
  - `PCTUSED`: **A minimum level of free space in a block before a block is placed on the free list**
- **For an index:**
  - `PCTFREE`: **Amount of space reserved for new index entries at creation time**
  - `PCTUSED`: **Always `0` for indexes**

Block Space Management

Each segment has parameters that control the space usage inside a block. You use two space management parameters, `PCTFREE` and `PCTUSED`, to control the use of free space within all the data blocks of a table or index. You specify these parameters when creating or altering a table. These parameters may be specified when you create or rebuild an index.

For Tables

`PCTFREE` sets the minimum percentage of a data block to be reserved as free space for possible updates to rows that already exist in that block. When the amount of free space in a block drops below `PCTFREE`, the block is no longer available for inserts.

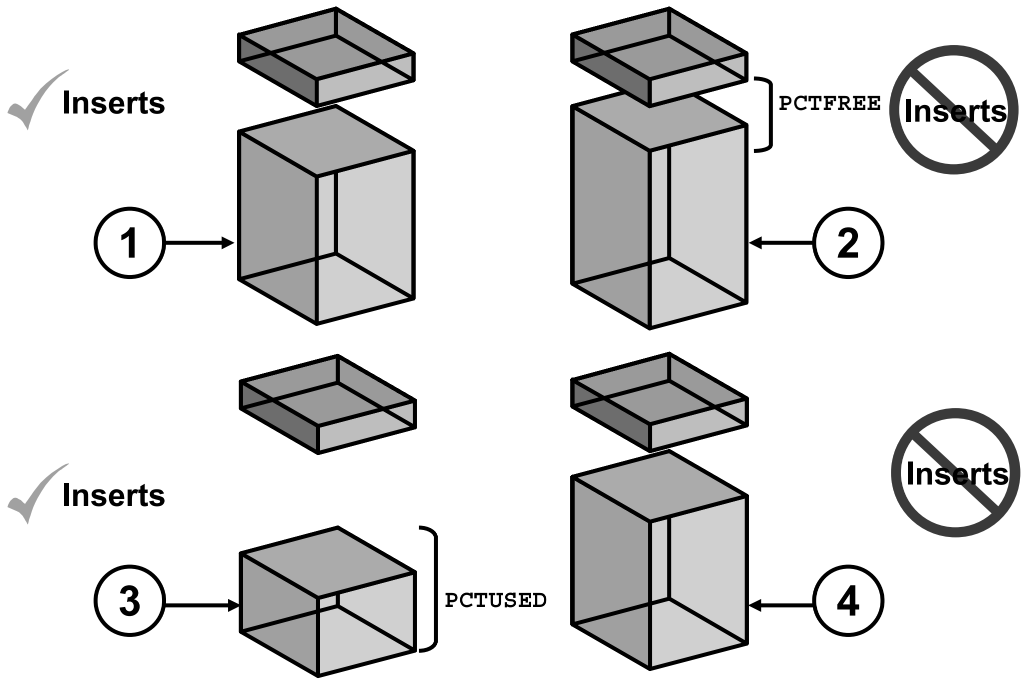
`PCTUSED` sets a lower limit on the percentage of a block that is used for row data plus overhead. When the percentage of space used drops below this limit, the block is available for inserts.

For Indexes

`PCTFREE` sets the amount of space reserved for new index leaf entries. It applies only during index creation if the table is not empty.

`PCTUSED` is always `0` for indexes.

Block Space Management with Free Lists



`PCTFREE` and `PCTUSED`

As an example, if you execute a `CREATE TABLE` statement with `PCTFREE` set to 20, 20% of each data block in the data segment of this table is reserved for updates to the existing rows in each block. The used space in the block can grow (1) until the row data and overhead total 80% of the total block size. Then the block is removed from the free list (2).

With free list–managed space, after you run a `DELETE` or an `UPDATE` statement, the server process checks whether the space being used in the block is now less than `PCTUSED`. If it is, the block is added to the beginning of the free list. When the transaction is committed, free space in the block becomes available for other transactions (3).

After a data block is filled to the `PCTFREE` limit again (4), the server process again considers the block unavailable for the insertion of new rows until the percentage of that block falls below the `PCTUSED` parameter.

DML Statements, `PCTFREE`, and `PCTUSED`

Two types of statements can increase the free space of one or more data blocks:

- `DELETE` statements
- `UPDATE` statements, which update existing values to values that use less space

How `PCTFREE` and `PCTUSED` Work Together

Released space in a block may not be contiguous; for example, when a row in the middle of the block is deleted. The server process coalesces the free space of a data block only when:

- An `INSERT` or an `UPDATE` statement attempts to use a block that contains enough free space to contain a new row piece
- The free space is fragmented so that the row piece cannot be inserted in a contiguous section of the block

The server process performs this compaction only when required, because otherwise the performance of a database system would decrease due to the continuous compaction of the free space in data blocks.

Note: If you change `PCTFREE` and `PCTUSED` for existing tables, there is no immediate impact on blocks. However, future DML activity uses the new rules for tables.

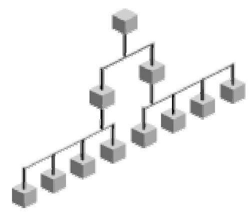
Setting `PCTFREE` for an Index

You can also specify the `PCTFREE` storage parameter when creating or altering an index. Setting `PCTFREE` for an index specifies how much of a block to fill when the index is created. It does not keep space available for updates as is done with data blocks.

Automatic Segment Space Management

Automatic Segment Space Management (ASSM) characteristics:

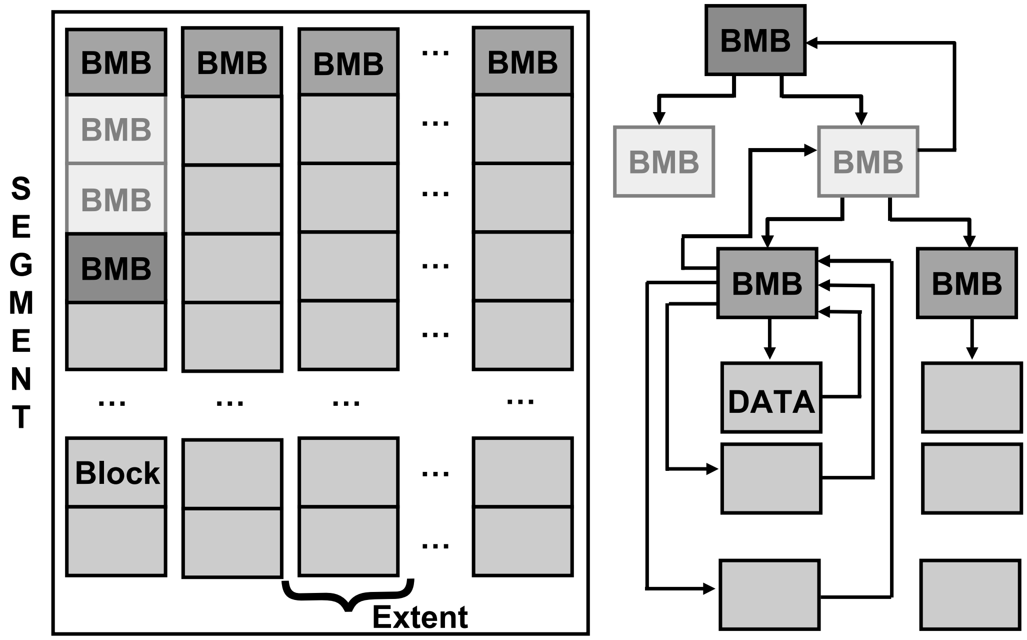
- **Space is managed with bitmap blocks (BMB).**
- **Multiple processes search different BMBs.**
- **Availability of block is shown with a full bit.**
- **The fullness is shown by a percentage full bit for each of 25, 50, 75, and 100 percent used.**



Automatic Segment Space Management

Automatic Segment Space Management uses a set of bitmap blocks (BMB) that are scattered throughout the segment to hold block fill-level information. A few bits for each block indicates the fill level and whether the block is available for inserts.
When multiple processes are attempting to find a block for inserts, they access different BMB based on a hash of the instance ID and process ID.
With ASSM, the free list is eliminated and the contention for finding blocks with available space is greatly reduced or eliminated.
The ASSM structure is recommended. It is especially useful for RAC databases, to avoid free-list contention, and free-list group maintenance.

Automatic Segment Space Management at Work



Automatic Segment Space Management at Work

Segments using Automatic Segment Space Management have a set of bitmap blocks (BMBs) describing the space utilization of the data blocks in that segment. For each data block, there is a set of bits per block indicating the space available in that block.

BMBs are organized in a tree hierarchy. The maximum number of levels inside this hierarchy is three. The leaves of this hierarchy represent the space information for a set of contiguous data blocks that belong to the segment. The BMB leaves are the unit at which space has affinity to an instance in a multi-instance environment.

When segments are growing with larger amounts of data being stored in the database, it becomes prohibitively expensive to search all the leaves of that hierarchy during an insert. Thus, an intermediate level of BMBs is created to contain metadata about the leaves.

During an `INSERT` operation, the server process starts from the root of the hierarchy. It goes down to the leaves to find a useful bitmap block pointing to data blocks containing free space. The BMBs are acquired in shared mode so that multiple processes can search in the same BMB.

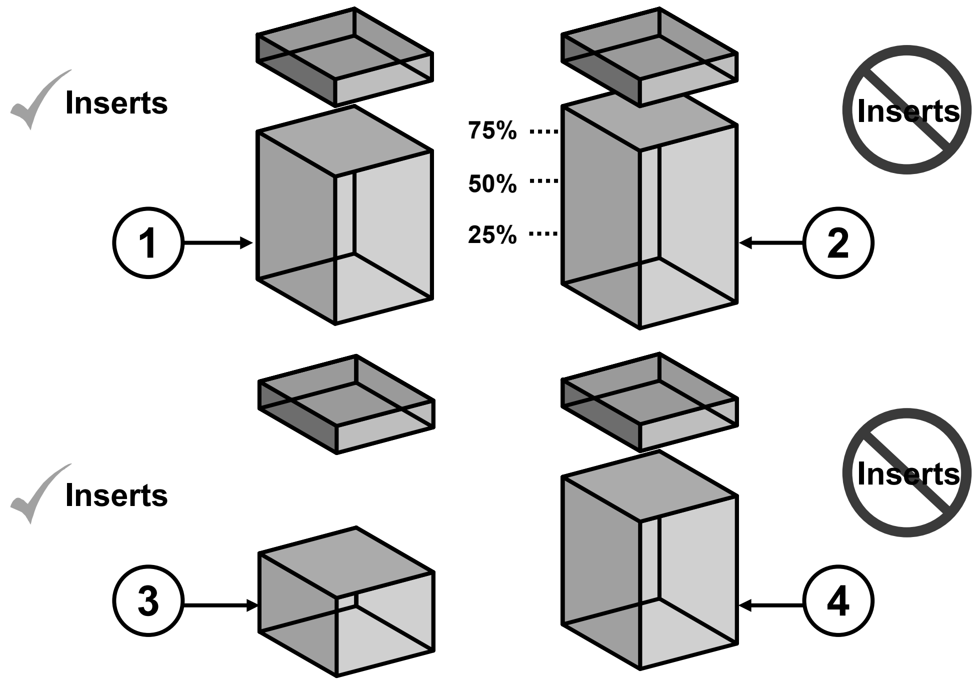
Automatic Segment Space Management at Work (continued)

Within an intermediate BMB level, concurrent processes get distributed by hashing on the instance number and process ID. This operation obtains BMB leaves. The server process hashes the process ID to provide a starting point again inside this BMB leaf.

In this type of BMB hierarchy, the leaves point to a range of data blocks that are potential candidates for the `INSERT` operation.

Note: The exact number of BMBs at each BMB index level depends on the individual extent sizes as well as on the overall segment size.

Block Space Management with ASSM



Block Space Management with ASSM

If you execute a `CREATE TABLE` statement with `PCTFREE` set to 20 in an ASSM tablespace, 20% of each data block in the data segment of this table is reserved for updates to the existing rows in each block. The used space in the block can grow (1) until the row data and overhead total 80% of the total block size. Then the block is marked as full, and the full bit is set (2). After the full bit is set, the block is not available for inserts. There are four other bits that indicate 100%, 75%, 50%, and 25% full. When the block reaches 80%, all the bits except the 100% bit have been set and then the full bit is set.

With ASSM-managed space, after the `DELETE` or `UPDATE` operation, the server process checks whether the space being used in the block is now less than a preset threshold. The threshold levels are 25, 50, 75, and 100 percent. If the used space is less than a threshold that is lower than `PCTFREE`, the block is marked as not full and available for inserts. In this example, the used space must fall below 75 percent to be marked as available for inserts (3).

After a data block is filled to the `PCTFREE` limit again (4), the server process again considers the block unavailable for the insertion of new rows until the percentage of that block falls below the ASSM threshold.

Creating an Automatic Segment Space Management Segment

- SEGMENT SPACE MANAGEMENT is the attribute used for tablespace creation, which cannot be subsequently altered.
- Segment space management is declared at the tablespace level.
- Tablespace must be permanent and locally managed.
- Automatic space management segments are specified through the AUTO keyword.
- For free-list segments, use the default value of MANUAL.
- For ASSM, PCTUSED, FREELIST, and FREELIST GROUPS are ignored at table creation.

Creating an Automatic Segment Space Management Segment

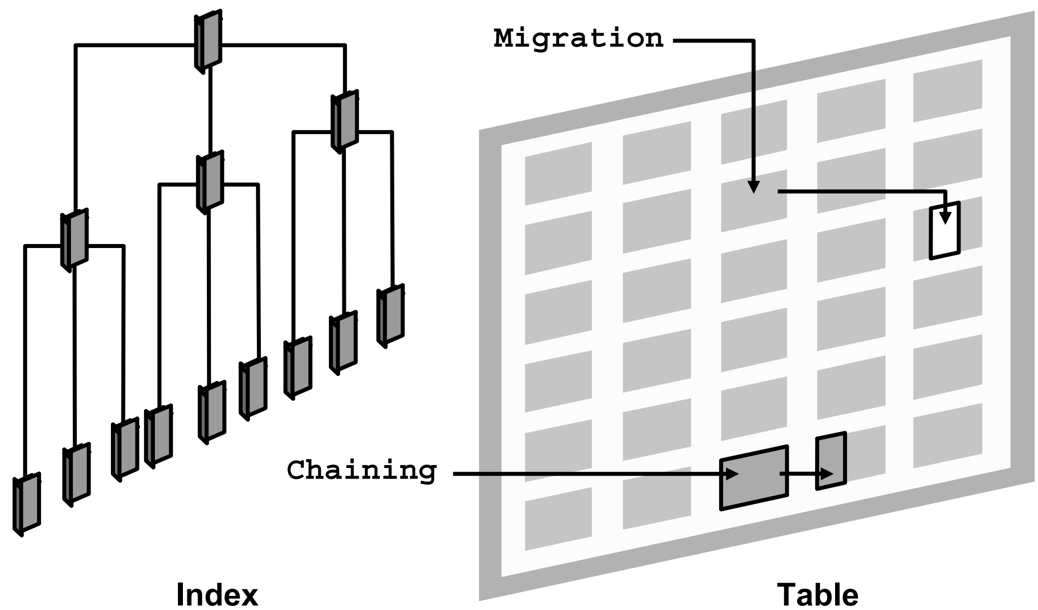
The SEGMENT SPACE MANAGEMENT specification applies only to a tablespace. You cannot specify the SEGMENT SPACE MANAGEMENT clause at the table level.

All segments residing in a tablespace with SEGMENT SPACE MANAGEMENT set to AUTO are automatic space management segments. Segments residing in a tablespace with SEGMENT SPACE MANAGEMENT set to MANUAL are managed by the PCTUSED, FREELIST, and FREELIST GROUPS attributes.

Note: Segments that can be defined as automatic space management segments are heap tables, indexes, index-organized tables (IOTs), and LOBs.

There is no Oracle-supplied procedure to migrate from manual segment space management to Automatic Segment Space Management. To alter a segment to use ASSM, the segment must be moved to a tablespace using ASSM.

Migration and Chaining



Migration and Chaining

In two circumstances, accessing a row requires more than one block to be read. This is caused by:

- Migration: An `UPDATE` statement increases the amount of data in a row so that the row no longer fits in its data block. The server process tries to find another block with enough free space to hold the entire row. If such a block is available, the server process moves the entire row to the new block. The server process keeps the original row piece of a migrated row to point to the new block containing the actual row; the `ROWID` of a migrated row does not change. Indexes are not updated; they continue to point to the original row location. During an index access of the row, the original row piece is found, and then another read to retrieve the row is required.
- Chaining: The row is too large to fit into an empty data block. The server process stores the data for the row in a chain of two or more data blocks. Chaining can occur when the row is inserted or updated. Row chaining usually occurs with large rows, such as rows that contain a large object (LOB). Row chaining in these cases is unavoidable, unless a larger block size is used.

Migration and Chaining (continued)

Migration and chaining have a negative affect on performance:

- INSERT and UPDATE statements that cause migration and chaining perform poorly because they perform additional processing.
- Queries that use an index to select migrated or chained rows must perform additional I/Os.

Migration is caused by PCTFREE being set too low; there is not enough room in the block kept for updates. To avoid migration, all tables that are updated should have their PCTFREE set so that there is enough space within the block for updates.

In a free list–managed segment, changing the PCTFREE value does not affect blocks that are already filled. It applies only to blocks that are added to the free list in subsequent operations.

If you change the PCTFREE value in an ASSM-managed segment, then you must run the DBMS_REPAIR.SEGMENT_FIX_STATUS procedure to implement the new setting on blocks already allocated to the segment.

Changing the PCTFREE value changes the amount of free space reserved for updates in blocks that are not yet full. This helps prevent row migration in future updates, but does not change already migrated rows or blocks that are already full.

Guidelines for PCTFREE and PCTUSED

- PCTFREE
  - Default: 10
  - Zero if no UPDATE activity
  - PCTFREE = 100 × UPD / (average row length)
- PCTUSED
  - Only with free lists
  - Default: 40
  - Set if rows are deleted
  - PCTUSED = 100 – PCTFREE – (100 × rows × average row length / block size)

Guidelines for Setting the Values of PCTFREE and PCTUSED

PCTFREE = 100 × UPD / (average row length)

PCTUSED = 100 – PCTFREE – (100 × rows × average row length / block size)
where:

- UPD = Average amount added by updates, in bytes. This is determined by subtracting the average row length after the insert from the current average row length.
- Average row length: After gathering statistics on the table, get this value from the avg_row_len column of dba_tables.
- Rows = Number of rows to be deleted before free-list maintenance occurs
- The space between PCTFREE and PCTUSED should provide enough space in the block for row updates and for at least one more row.

PCTUSED is relevant only in tables that undergo deletes. In many tables, you may be able to pack rows into blocks more tightly by setting PCTUSED to be higher than the default. Choose a value for PCTUSED that balances data density and free-list contention. For tables with many DELETE and INSERT statements, increasing PCTUSED can improve block storage performance. Fewer blocks are required, tables are smaller, and full table scan operations are faster. However, blocks that are densely populated with a high PCTUSED can cause free-list contention by being placed back on the free list after a very few deletes.

Detecting Migration and Chaining

Use the `ANALYZE` command to detect migration and chaining:

```
SQL> ANALYZE TABLE oe.orders COMPUTE STATISTICS;
Table Analyzed.
SQL> SELECT num_rows, avg_row_len, chain_cnt
  2     FROM DBA_TABLES
  3     WHERE table_name='ORDERS';
  NUM_ROWS AVG_ROW_LEN  CHAIN_CNT
---------- ----------- ----------
      1171          67         83
```

Detect migration and chaining by using Statspack/AWR:

```
Statistic                    Total Per transaction ...
------------------------- ----- -------------- ...
table fetch continued row   495            .02 …
```


Detecting Migration and Chaining

The ANALYZE Command

You can identify the existence of migrated and chained rows in a table or cluster by using the `ANALYZE` command. This command counts the number of migrated and chained rows and places this information into the `CHAIN_CNT` column of `DBA_TABLES`.

The `NUM_ROWS` column provides the number of rows stored in the analyzed table or cluster. Compute the ratio of chained and migrated rows to the number of rows to decide whether migrated rows need to be eliminated.

The `AVG_ROW_LEN` column gives an indication of whether the chain count is due to migrated rows or rows that are too long for any block.

The Table Fetch Continued Row Statistic

You can also detect migrated or chained rows by checking the Table Fetch Continued Row statistic in `V$SYSSTAT` or in the Statspack report under "Instance Activity Stats for DB."

Guidelines

Increase `PCTFREE` to avoid migrated rows. If you leave more free space available in the block for updates, the row has room to grow. You can also reorganize (re-create) tables and indexes with a high deletion rate.

Selecting Migrated Rows

```
SQL> ANALYZE TABLE oe.orders LIST CHAINED ROWS;
Table analyzed.

SQL> SELECT  owner_name, table_name, head_rowid
  2    FROM  chained_rows
  3    WHERE table_name = 'ORDERS';
OWNER_NAME  TABLE_NAME  HEAD_ROWID
----------  ----------  ------------------
SALES       ORDER_HIST  AAAAluAAHAAAAA1AAA
SALES       ORDER_HIST  AAAAluAAHAAAAA1AAB

...
```


Selecting Migrated Rows

You can identify migrated and chained rows in a table or cluster by using the `ANALYZE` command with the `LIST CHAINED ROWS` option. This command collects information about each migrated or chained row and places this information into a specified output table. To create the table that holds the chained rows, execute the `utlchain.sql` script:

```
SQL> CREATE TABLE chained_rows (
  2  owner_name         VARCHAR2(30),
  3  table_name         VARCHAR2(30),
  4  cluster_name       VARCHAR2(30),
  5  partition_name     VARCHAR2(30),
  6  head_rowid         ROWID,
  7  analyze_timestamp  DATE );
```

If you create this table manually, it must have the same column names, data types, and sizes as the `CHAINED_ROWS` table.

Note: Using the ANALYZE TABLE … LIST CHAINED ROWS command has the advantage of *not* overwriting the current statistics.

Eliminating Migrated Rows

- **Export/import:**
  - **Export the table.**
  - **Drop or truncate the table.**
  - **Import the table.**
- **`MOVE` table command:**
  - `ALTER TABLE EMPLOYEES MOVE`
- **Online table redefinition**
- **Copy migrated rows:**
  - **Find migrated rows by using `ANALYZE`.**
  - **Copy migrated rows to a new table.**
  - **Delete migrated rows from the original table.**
  - **Copy rows from the new table to the original table.**

Eliminating Migrated Rows

Any procedure that inserts the rows into a table removes migrated rows.

Export and import eliminate migrated rows by inserting all the rows into a re-created table, but the table is unavailable for the duration of the operation.

The `Move` table command is faster than export and import. It requires two times the space during the operation, and the table is unavailable. All indexes will be unusable and must be rebuilt after the move operation.

Online redefinition allows the table to be available for DML and select for the duration of the operation, and indexes are maintained. This method requires the use of the `DBMS_REDEFINITION` package. This method requires the most space, because there are two copies of the table, a journal table, and multiple materialized views.

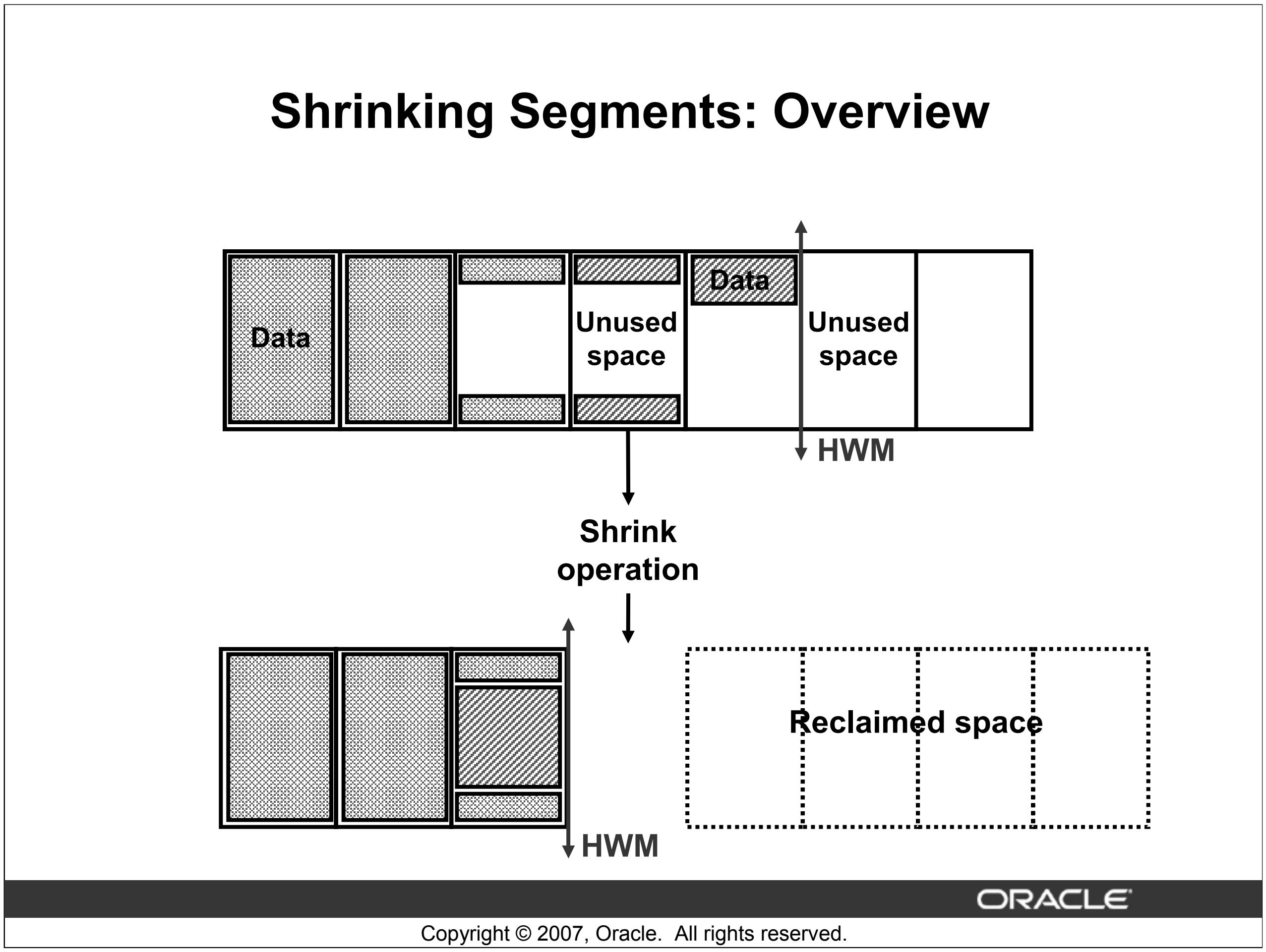
The copy method only locks the rows being moved and uses additional space only for the chained rows. You can eliminate migrated rows by using this SQL*Plus script:

```
/* Get the name of the table with migrated rows */
ACCEPT table_name PROMPT 'Enter the name of the table with
migrated rows: '
```

Eliminating Migrated Rows (continued)

```
/* Clean up from last execution */
SET ECHO OFF
DROP TABLE migrated_rows;
DROP TABLE chained_rows;

/* Create the CHAINED_ROWS table */
@?/rdbms/admin/utlchain
SET ECHO ON
SPOOL fix_mig
/* List the chained & migrated rows */
ANALYZE TABLE &table_name LIST CHAINED ROWS;

/* Copy the chained/migrated rows to another table */
CREATE TABLE migrated_rows AS
  SELECT orig.*
  FROM &table_name orig, chained_rows cr
  WHERE orig.rowid = cr.head_rowid
  AND cr.table_name = upper('&table_name');

/* Delete the chained/migrated rows from the original
table */
DELETE FROM &table_name
WHERE rowid IN (
  SELECT head_rowid
  FROM chained_rows);

/* Copy the chained/migrated rows back into the original
table */
INSERT INTO &table_name
  SELECT *
  FROM migrated_rows;
SPOOL OFF
```

When using this script, you must disable any foreign key constraints that would be violated when the rows are deleted. Additional considerations are the implications of insert and delete triggers, row-level security, and auditing.



Shrinking Segments: Overview

The upper part of the diagram in the slide shows a sparsely populated segment. There is some unused space both above and below the segment's high-water mark (HWM). Shrinking a sparsely populated segment improves the performance of scan and DML operations on that segment. This is because there are fewer blocks to look at after the segment has been shrunk. This is especially true for:

- Full table scans (fewer and denser blocks)
- Better index access (fewer I/Os on range `ROWID` scans due to a more compact tree)

You can make more free space available in tablespaces by shrinking sparsely populated segments. When a segment is shrunk, its data is compacted, its HWM is pushed down, and unused space is released back to the tablespace containing the segment.

Row IDs are not preserved. When a segment is shrunk, the rows move inside the segment to different blocks, causing the row ID to change.

Note: The number of migrated rows in a segment may be reduced as a result of a segment shrink operation. Because a shrink operation does not touch all the blocks in the segment, you cannot depend on reducing the number of migrated rows after a segment has been shrunk.

Shrinking Segments: Considerations

- A shrink operation is an online and in-place operation.
- It is applicable only to segments residing in ASSM tablespaces.
- Candidate segment types:
  - Heap-organized tables and index-organized tables
  - Indexes
  - Partitions and subpartitions
  - Materialized views and materialized view logs
- Indexes are maintained.
- Triggers are not fired.

ORACLE

Shrinking Segments: Considerations

A shrink operation is an online and in-place operation.

You cannot execute a shrink operation on segments managed by free lists. Segments in automatic segment space–managed tablespaces can be shrunk. However, the following objects cannot be shrunk:

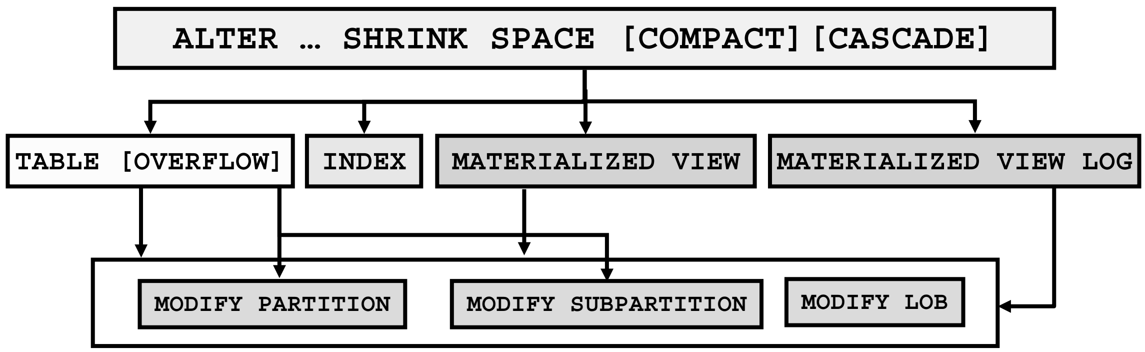
- Tables in clusters
- Tables with `LONG` columns
- Tables with on-commit materialized views
- Tables with `ROWID`-based materialized views
- LOB segments
- IOT mapping tables
- IOT overflow segments
- Tables with function-based indexes

`ROW MOVEMENT` must be enabled for heap-organized segments.

Indexes are in a usable state after shrinking the corresponding table.

The actual shrink operation is handled internally as an `INSERT/DELETE` operation. However, DML triggers are not fired because the data itself is not changed.

Shrinking Segments by Using SQL




```
ALTER TABLE employees ENABLE ROW MOVEMENT;                    1
ALTER TABLE employees SHRINK SPACE CASCADE;                   2
ALTER TABLE employees MODIFY LOB(resume) (SHRINK SPACE)       3
ALTER TABLE employees OVERFLOW SHRINK SPACE;                  4
```


Shrinking Segments by Using SQL

Because a shrink operation may cause ROWIDs to change in heap-organized segments, you must enable row movement on the corresponding segment before executing a shrink operation on that segment. Row movement by default is disabled at the segment level. To enable row movement, the ENABLE ROW MOVEMENT clause of the CREATE TABLE or ALTER TABLE command is used. This is illustrated in the first example in the slide.

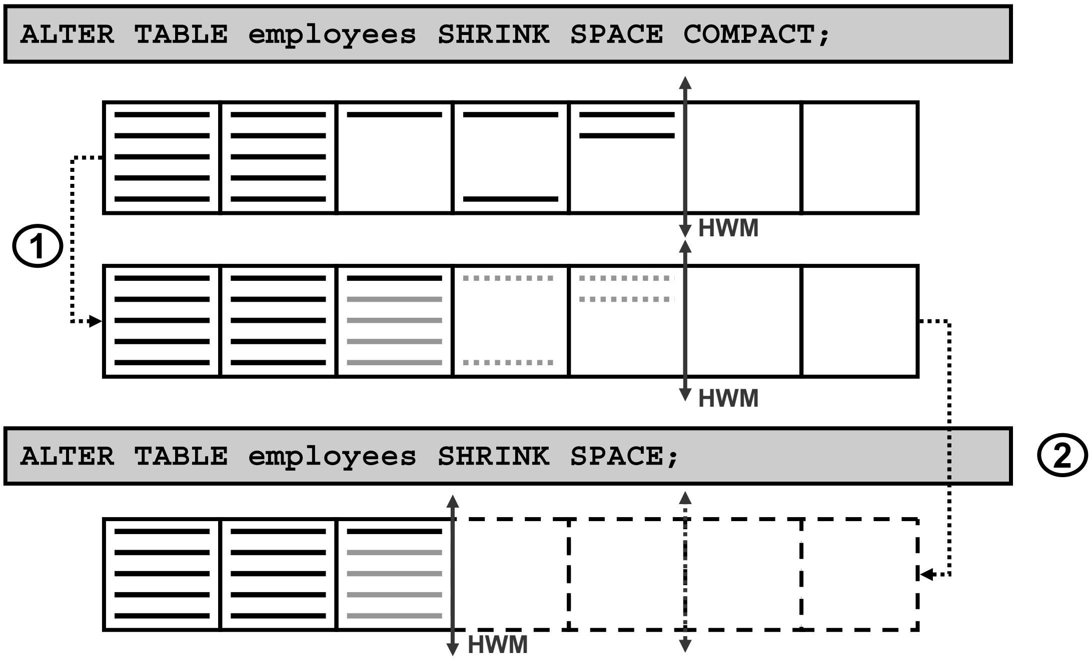
Use the ALTER command to invoke segment shrink on an object. The object's type can be one of the following: table (heap- or index-organized), partition, subpartition, LOB (data and index segment), index, materialized view, or materialized view log. Use the SHRINK SPACE clause to shrink space in a segment. If CASCADE is specified, the shrink behavior is cascaded to all the dependent segments that support a shrink operation, except materialized views, LOB indexes, and IOT (index-organized tables) mapping tables. The SHRINK SPACE clause is illustrated in the second example.

In an index segment, the shrink operation coalesces the index before compacting the data.

Example 3 shows a command that shrinks a LOB segment, given that the RESUME column is a CLOB. Example 4 shows a command that shrinks an IOT overflow segment belonging to the EMPLOYEES table.

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

Segment Shrink: Basic Execution



Segment Shrink: Basic Execution

The diagram in the slide describes the two phases of a table shrink operation. Compaction is performed in the first phase. During this phase, rows are moved to the left part of the segment as much as possible. After the rows have been moved, the second phase of the shrink operation is started. During this phase, the HWM is adjusted, and the unused space is released.

During a shrink operation, you can execute only the compaction phase by specifying the `SHRINK SPACE COMPACT` clause. This is illustrated by the first example in the slide.

As shown by the second example in the slide, if `COMPACT` is not specified, the segment space is compacted, and at the end of the compaction phase, the HWM is adjusted and the unused space is released.

Segment Shrink: Execution Considerations

- **Use compaction only:**
  - **To avoid unnecessary cursor invalidation**
  - **During peak hours**
- **DML operations and queries can be issued during compaction.**
- **DML operations are blocked when HWM is adjusted.**

ORACLE

Segment Shrink: Execution Considerations

The compaction phase of a segment shrink operation is performed online. Conventional DML operations as well as queries can coexist with a segment shrink operation.

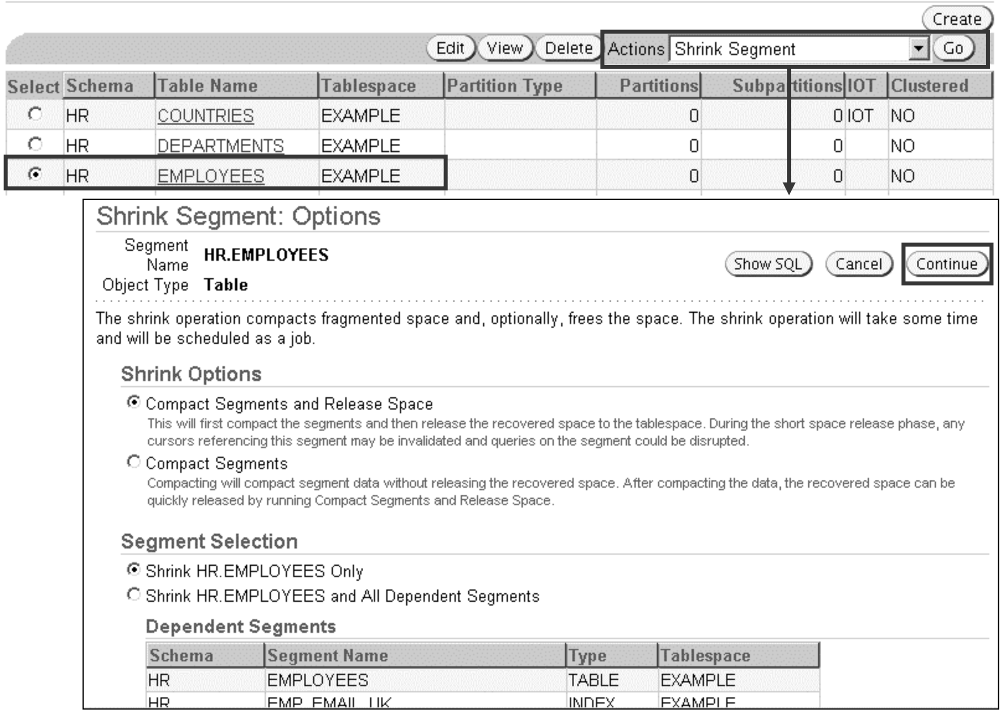
During the compaction phase, locks are held on individual rows containing the data. Concurrent DML operations such as updates and deletes serialize on these locks. However, only packets of rows are locked at one time to avoid the locking of the entire segment.

Similarly, conventional DML operations can block the progress of the compaction phase until they commit.

The `COMPACT` clause is useful if you have long-running queries that might span the shrink operation and attempt to read from blocks that have been reclaimed. When you specify the `SHRINK SPACE COMPACT` clause, the progress of the shrink operation is saved in the bitmap blocks of the corresponding segment. This means that the next time a shrink operation is executed on the same segment, the Oracle database remembers what has been done already. You can then reissue the `SHRINK SPACE` clause without the `COMPACT` clause during off-peak hours to complete the second phase.

During the second phase of the segment shrink operation, when the HWM is adjusted, the object is locked in exclusive mode. This occurs for a very short duration and does not affect the availability of the segment significantly. Dependent cursors are invalidated.

Using EM to Shrink Segments



ORACLE

Using EM to Shrink Segments

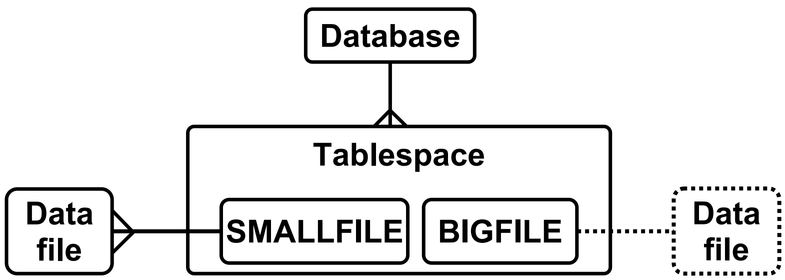
Using Database Control, you can shrink individual segments as follows:

1. On the Database Control home page, click the Tables link in the Storage section.
2. On the Tables page, select your table, and then select Shrink Segment in the Actions drop-down list. Click Go.
3. On the Shrink Segment page, choose the dependent segments to shrink. You can either compact only, or compact and release the space. You can also choose the CASCADE option in the Segment Selection section by selecting “Shrink HR.EMPLOYEES and All Dependent Segments.”
4. Click the Continue button. This allows you to submit the shrink statements as a scheduled job.

Note: Before shrinking a heap-organized table, you must enable row movement on that table. You can do that using the Options tab of the Edit Table page of Database Control.

Bigfile Tablespaces: Overview

- **A bigfile tablespace contains a single file.**
- **Maximum file size ranges from 8 TB to 128 TB.**
- **Tablespaces are logically equivalent to data files.**



Bigfile Tablespaces: Overview

A bigfile tablespace is a tablespace containing a single file that can be very large in size. The new addressing scheme allows you to have up to four billon blocks in a single data file, and the maximum file size can be in the range 8 TB–128 TB, depending on the database block size.

To distinguish bigfile tablespaces from the traditional tablespace that can contain multiple relatively small files, the traditional tablespace is now referred to as a smallfile tablespace. An Oracle database can now contain both bigfile and smallfile tablespaces.

Bigfile tablespaces eliminate the need for adding new data files to a tablespace. This feature simplifies manual or automatic management of the disk space (using Oracle Managed Files or ASM) and provides data file transparency. The graphic in the slide shows the entity-relationship diagram of the Oracle database space management architecture. The one-to-many relationship between tablespace and data file complicates the architecture and can create difficulties in managing disk space utilization whenever one tablespace is associated with hundreds of data files.

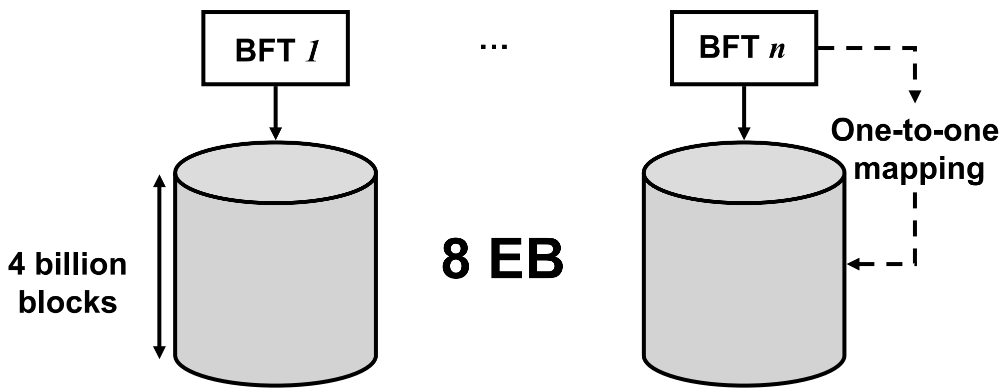
Bigfile tablespaces make the notion of a tablespace logically equivalent to a data file. All operations that were previously performed on data files can now be performed on tablespaces.

Bigfile Tablespaces: Overview (continued)

When bigfile tablespaces are used with Oracle Managed Files (OMF), data files become completely transparent for users because they never need to refer to them. In this case, the data file node can be removed from the diagram, making the architecture more intuitive.

Bigfile Tablespaces: Benefits

- **Significantly increases the storage capacity**
- **Simplifies data file management for large databases by making tablespaces the main units of disk space administration**



Bigfile Tablespaces: Benefits

The purpose of bigfile tablespaces is to significantly increase the storage capacity of Oracle databases and, at the same time, simplify management of data files in large databases. This feature enables an Oracle database to contain up to 8 exabytes (8,000,000 TB) of data. This is rendered possible by allowing users to store data in much larger files and, by doing so, to decrease the number of files in large databases. It also simplifies database management by providing data file transparency and making tablespaces the main units of the disk space administration, and backup and recovery. The maximum amount of data M that can be stored in the Oracle database can be calculated using the formula: $M = D \times F \times B$, where D is the maximum number of data files in the database, F is the maximum number of blocks per data file, and B is the maximum block size.

Bigfile tablespaces are performance neutral. A possible exception is file header contention when there is a high rate of allocation and deallocation of extents, but this is not a common condition.

Bigfile Tablespace: Benefits (continued)

With the older limits (D = 64 KB, F = 4 M, and B = 32 KB), the value of M is 8 PB, or 2 PB for the most typical 8 KB block size. Using bigfile tablespaces, you can increase M by three orders of magnitude to 8 EB for a 32KB block, or to 2 EB for an 8 KB block size. The availability of 64-bit operating systems that can handle much larger files, and technologies that result in 500 GB hard drives that are already in the market, are good reasons to use bigfile tablespaces. Additionally, this approach also reduces the number of files that you need to manage. The use of bigfile tablespaces eliminates the need to add new data files to a tablespace. This simplifies management of the disk space and provides data file transparency.

Note: 1 PB = 1,024 TB; 1 EB = 1,024 PB = 1,048,576 TB

Using Bigfile Tablespaces

- Supported only for locally managed tablespaces using Automatic Segment Space Management
- Use with logical volume managers or Automatic Storage Management (ASM)
- OMF used for complete data file transparency

Database Block Size	Recommended Maximum Number of Extents
2 KB	100,000
4 KB	200,000
8 KB	400,000
16 KB	800,000

ORACLE

Using Bigfile Tablespaces

Bigfile tablespaces are supported only for locally managed tablespaces with bitmapped segments (ASSM), as well as locally managed `UNDO` and `TEMPORARY` tablespaces. Use bigfile tablespaces with a logical volume manager or Automated Storage Management (ASM) that supports striping, mirroring, and dynamically extensible logical volumes. You should avoid creating a bigfile tablespace on a system that does not support striping, because of negative implications for parallel executions.

Using bigfile tablespaces with OMF provides users with more benefits because of the higher degree of data file transparency. When using bigfile tablespaces, you should think about extent size before creating such a tablespace. Although the default allocation policy is `AUTOALLOCATE`, you might want to change that default to `UNIFORM` with a large extent size when the file size is in terabytes. Otherwise, `AUTOALLOCATE` is probably the best choice. The table in the slide gives you recommendations regarding the maximum number of extents per tablespace depending on the block size. These figures are not a hard limit, but if more extents are created, there may be a performance penalty under high concurrency and during DDL operations involving high space management activity.

Note: Using bigfile tablespaces on platforms that do not support large files can dramatically limit the capacity of a tablespace.

Summary

In this lesson, you should have learned how to:
- Tune segment space management
- Convert from dictionary-managed tablespaces
- Convert to Automatic Segment Space Management
- Tune block space management
- Diagnose and correct row migration
- Diagnose table fragmentation
- Compare characteristics of bigfile and smallfile tablespaces

ORACLE

Practice 14 Overview: Tuning Database Space Usage

This practice covers the following topics:

- Diagnosing space management issues
- Converting from dictionary-managed to locally managed tablespaces

15

Performance Tuning: Summary

Objectives

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

- List best practices identified throughout the course
- Summarize the performance tuning methodology

Necessary Initialization Parameters with Little Performance Impact

Parameter	Description
`DB_NAME`	**Name of the database. This should match the `ORACLE_SID` environment variable.**
`DB_DOMAIN`	**Location of the database in Internet dot notation**
`OPEN_CURSORS`	**Limit on the maximum number of cursors for each session. The setting is application-dependent; 500 is recommended.**
`CONTROL_FILES`	**Set to contain at least two files on different disk drives to prevent failures from control file loss**
`DB_FILES`	**Set to the maximum number of files that can be assigned to the database**
`STATISTICS_LEVEL`	**Set to `TYPICAL`**

Necessary Initialization Parameters Without Performance Impact

A running Oracle instance is configured using initialization parameters, which are set in the initialization parameter file. These parameters influence the behavior of the running instance, including influencing performance. In general, a very simple initialization file with a few relevant settings covers most situations. The initialization file should not be the first place you expect to do performance tuning.

The table in the slide describes the parameters necessary in a minimal initialization parameter file. Although these parameters are necessary, they have little performance impact.

`STATISTICS_LEVEL` can have a large impact on the performance of your database. Timed statistics are turned off when `STATISTICS_LEVEL` is set to `BASIC`, removing the effective information from the Statspack and AWR reports. `STATISTICS_LEVEL` set to `TYPICAL` has a small performance overhead. The entire AWR suite of tools is supposed to have no more than a 5% performance overhead.

Important Initialization Parameters with Performance Impact

Parameter	Description
COMPATIBLE	To take advantage of the latest improvements of a new release
DB_BLOCK_SIZE	8192 for OLTP and higher for DSS
SGA_TARGET	Automatically sized SGA components
PGA_AGGREGATE_TARGET	Automatic PGA management
PROCESSES	Maximum number of processes that can be started by that instance
SESSIONS	To be used essentially with shared server
UNDO_MANAGEMENT	AUTO mode recommended
UNDO_TABLESPACE	Undo tablespace to be used by instance

Important Initialization Parameters with Performance Impact

The table in the slide includes the most important parameters to set with performance implications:

- COMPATIBLE: Specifies the release with which the Oracle server must maintain compatibility. It lets you take advantage of the maintenance improvements of a new release immediately in your production systems without testing the new functionality in your environment. If your application is designed for a specific release, and you are actually installing a later release, then you might want to set this parameter to the version of the previous release. This parameter does not guarantee that the behavior of the new release will be identical to the release specified by COMPATIBLE. COMPATIBLE allows you to use a new release, while at the same time guaranteeing backward compatibility with an earlier release. This is helpful if it becomes necessary to revert to the earlier release. Some features of the release may be restricted.
- DB_BLOCK_SIZE: Sets the size of the database blocks stored in the database files and cached in the SGA. The range of values depends on the operating system, but it is typically 8192 for transaction processing systems and higher values for database warehouse systems.

Important Initialization Parameters with Performance Impact (continued)

- SGA_TARGET: Specifies the total size of all automatically tuned SGA components
- PGA_AGGREGATE_TARGET: Specifies the target aggregate PGA memory available to all server processes attached to the instance
- PROCESSES: Sets the maximum number of processes that can be started by the instance. This is the most important primary parameter to set, because many other parameter values are derived from this.
- SESSIONS: This is set by default from the value of PROCESSES. However, if you are using the shared server, the derived value is likely to be insufficient.

Sizing Memory Initially

As an initial guess for memory allocation:

- **Leave 20% of available memory to other applications.**
- **Leave 80% of memory to the Oracle instance.**
- **For OLTP:**

```
SGA_TARGET=(total_mem*80%)*80%
```

```
PGA_AGGREGATE_TARGET=(total_mem*80%)*20%
```

- **For DSS:**

```
SGA_TARGET=(total_mem*80%)*50%
```

```
PGA_AGGREGATE_TARGET=(total_mem*80%)*50%
```

ORACLE

Sizing Memory Initially

You may use this method to make an estimate that you will later refine based on workload. Example: Your Oracle instance will run on a system with 4 GB of physical memory. Part of that memory will be used for the operating system and other applications running on the same hardware system. You decide to dedicate only 80% (3.2 GB) of the available memory to the Oracle instance. You then allocate that memory to SGA and PGA.

For OLTP systems, the PGA memory typically accounts for a small fraction of the total memory available (for example, 20% of the instance memory), leaving 80% for the SGA.

For DSS systems running large, memory-intensive queries, PGA memory can typically use up to 70% of the instance memory (up to 2.2 GB in this example).

Reasonable initial values for the `PGA_AGGREGATE_TARGET` parameter might be:

- For OLTP: `PGA_AGGREGATE_TARGET=(total_mem*80%)*20%`
- For DSS: `PGA_AGGREGATE_TARGET=(total_mem*80%)*50%`

where *`total_mem`* is the total amount of physical memory available on the system (4 GB).

In this example, where *`total_mem`* is equal to 4 GB, set `PGA_AGGREGATE_TARGET` to 1600 MB for a DSS system or to 655 MB for an OLTP system and the remainder to SGA.

Database High Availability: Best Practices

- Use `SPFILE`.
- Multiplex redo logs.
- Use resumable space allocation.
- Create at least two control files.
- Enable Flashback Database.
- Enable block checking.
- Use auto-tune checkpointing.
- Log checkpoints to the alert log.
- Use database resource manager.
- Use Automatic Undo Management.
- Use Automatic Segment Space Management.
- Use locally managed tablespaces.
- Use locally managed temporary tablespaces.
- Enable `ARCHIVELOG` mode and use a flash recovery area.
- Set time long enough for `CONTROL_FILE_RECORD_KEEP_TIME`.
- Designate a default permanent tablespace other than `SYSTEM` and `SYSAUX`.

ORACLE

Database High Availability: Best Practices

The list in the slide gives you a short summary of the recommended practices that apply to single-instance databases, RAC databases, and Data Guard standby databases.

These practices affect the performance, availability, and mean time to recover (MTTR) of your system. Some of these practices may reduce performance, but they are necessary to reduce or avoid outages. The minimal performance impact is outweighed by the reduced risk of corruption or the performance improvement for recovery.

Note: For more information about how to configure the features listed above, refer to the following documents:

- *Oracle Database Administrator's Guide*
- *Oracle Data Guard Concepts and Administration*
- *Oracle Database Net Services Administrator's Guide*

Undo Tablespace: Best Practices

- Use Automatic Undo Management.
- UNDO_RETENTION:
  - Oracle Database 10*g* Release 1: Set it to your flashback requirement.
  - Oracle Database 10*g* Release 2: Do not set it.
- Undo tablespace size:
  - Initial size: Small with AUTOEXTEND enabled
  - Steady state: Fix size using the Undo Advisor and add a 20% safe margin.

Undo Tablespace: Best Practices

Automatic Undo Management (AUM) is the preferred option because it greatly simplifies undo management and also provides an easy way to avoid errors such as ORA-1555.

The UNDO_RETENTION parameter's behavior has changed in the last few releases.
In Oracle Database 10*g* Release 1, undo retention was made auto-tuning. The system automatically learns how long to maintain the committed undo data for read consistency purposes. It represents the lower limit for which committed undo is kept. The main purpose of this parameter in Oracle Database 10*g* Release 1 is to support flashback features because the read-consistency aspect of undo retention no longer needs manual configuration.

In Oracle Database 10*g* Release 2, auto-tuning of undo retention has been further enhanced. In this release, undo retention is tuned to the maximum allowable by the tablespace, even if it is greater than the longest running query in the system. This gives the best possible retention and in effect makes the setting of the UNDO_RETENTION parameter unnecessary. Therefore, the only consideration in undo management is now the proper sizing of the undo tablespace.

Undo Tablespace: Best Practices (continued)

The most accurate and recommended way of sizing the undo tablespace is by using the Undo Advisor. However, a newly created database does not have the kind of historical workload information that is needed by the advisor to make sound recommendations. For such new systems, the best way to size the undo tablespace is to start with a small size, say 100 MB, but to set the `AUTOEXTEND` data file attribute to `ON`. This allows the tablespace to grow automatically as needed. After the database has reached a steady state, it is recommended to adjust the size of the tablespace. To determine the appropriate size of the undo tablespace, you should use the Undo Advisor. Add 20 percent to the size provided by the Undo Advisor for safety purposes and disable the `AUTOEXTEND` attribute.

Temporary Tablespace: Best Practices

Locally managed temporary tablespaces use a uniform extent. Extent size should be:

- **1 MB to 10 MB extent size:**
  - **For DSS, OLAP applications involving huge work areas**
  - **Large temporary LOBs are predominant**
- **64 KB or multiple less than 1 MB:**
  - **Small global temporary tables are predominant**
  - **OLTP**

Temporary tablespace group increases addressability from TB to PB.

Temporary Tablespace: Best Practices

It is important to understand the types of workloads on the temporary tablespace:

- Temporary LOBs: Temporary LOBs are created explicitly by the user as well as implicitly by the database to store transient unstructured data. If the temporary space usage is dominated by large temporary LOBs, then larger extent sizes should be used. Oracle Corporation recommends you to use 1 MB to 10 MB as the extent size.
- DSS: The DSS workload is typically dominated by complex queries performing intensive sort and hash join operations. The general rule for good performance and efficient space usage is to set an extent size of 1 MB.
- OLTP operations needing temporary space tend to be small. Extent sizes that are small multiples 64 KB up to 1 MB give better performance and use less space.
- Global temporary tables: These are transient tables created by the user as well as the system in the temporary tablespace. Each global temporary table requires at least one extent to be allocated. If the volume of data loaded into the temporary tables is pretty small, choosing smaller multiples of 64 KB for the extent size should avoid wasting space.

Temporary Tablespace: Best Practices (continued)

- General: 1 MB extent size is a good initial choice until workload patterns are established

Temporary tablespace groups: These are several temporary tablespaces assigned to a temporary tablespace group. Different sort operations can be assigned to different temporary tablespaces in the same group. Each parallel query server assigned to a parallel operation can use a different temporary tablespace in the same temporary tablespace group. This increases the available temporary space and reduces contention on a single temporary tablespace. A single sort operation (non-parallel) uses only one temporary tablespace in a temporary tablespace group.

Note: The primary purpose of a temporary tablespace group is to increase the addressability of the temporary space. A single temporary tablespace can have a maximum of four billion blocks. This is 8 TB for a 2 KB block size and 64 TB for a 16 KB block size. With temporary tablespace groups, the addressability is increased to several petabytes.

General Tablespace: Best Practices

- Use locally managed tablespaces with auto-allocate extents policy.
- Use Automatic Segment Space Management (ASSM).
- Use online segment shrink to eliminate internal fragmentation.
- Periodically review the results of the Automatic Segment Advisor.
- Monitor tablespace space usage using server-generated alerts.
- Size of extents matter more than the number of extents in the segments.

ORACLE

General Tablespace: Best Practices

- Use locally managed tablespaces with auto-allocate to avoid fragmentation of tablespace. Note: the largest auto-allocate extent size possible is 64 MB. This should not impact segments that are less than a terabyte in size.
- Use ASSM for better segment space management performance.
- The Automatic Segment Advisor examines segments residing in tablespaces with outstanding tablespace-full alerts and those with heavy user access for signs of fragmentation. If the segments are fragmented, a recommendation is made to rectify.
- Use server-generated alerts to monitor tablespaces for impending out-of-space conditions.
- Use properly sized extents, when assigning the extent value manually. If the extents are too small, then multiblock reads are prevented. If the extents are too large, space is wasted in the last extent. There are only a few cases where the number of extents should be a cause for concern:
  - Tablespace is dictionary-managed and DDL commands such `TRUNCATE` and `DROP` are common.
  - Parallel queries are common and extents are noncontiguous in file.

Internal Fragmentation: Considerations

- **Watch for:**
  - **Bad choices of `PCTFREE` and `PCTUSED` for heap segments**
  - **Bad choices of `PCTVERSION` and `RETENTION` for LOB segments**
  - **Low density of data in segment**
  - **Direct loads followed by deletes (no inserts)**
  - **Indexes on tables with random updates and deletes with no further inserts**
- **Remedy:**
  - **Online segment shrink**
  - **Online redefinition**
  - **`MOVE` operations**

ORACLE

Internal Fragmentation: Considerations

Segment fragmentation is a serious issue for two main reasons:

- Poor space utilization, when the free space under the HWM of one segment cannot be reused by other segments without performing shrink first
- Poor performance, where certain data access patterns are impacted due to having to do more I/O than is necessary to read the data. For example, a full table scan reads all the blocks below the HWM, whether or not there are rows in those blocks. In the case of LOBs, data in such segments is always accessed through an index. Therefore, there is no impact in access performance.

Internal fragmentation can impact heap, index, as well as LOB segments. The slide shows you the main reasons leading to internal fragmentation.

Fragmentation is an issue that database administrators (DBAs) have been battling for a long time. Use of locally managed tablespaces in conjunction with Automatic Segment Space Management (ASSM) offers the best way to minimize internal fragmentation.

Internal Fragmentation: Considerations (continued)

However, after internal fragmentation has occurred, the traditional methods you can use to solve the problem includes the following:

- Alter table `MOVE`, CTAS (`create table as select`), Import/Export.
- Reorganize the object using online redefinition set of procedures.

These techniques are effective in removing fragmentation, but require additional disk space and involve considerable manual labor. In Oracle Database 10*g*, it is thus recommended to use the online segment shrink feature.

Block Size: Advantages and Disadvantages

Block Size	Advantages	Disadvantages
Smaller	Small rows with lots of random access	Relatively high overhead (block header)
	Reduce block contention (OLTP)	For large rows (chaining)
Larger	Lower overhead: more room for data	Waste cache space with random access of small rows
	Good for sequential access	Index leaf block contention (OLTP)

Block Size: Advantages and Disadvantages

A block size of 8 KB is optimal for most systems. However, OLTP systems occasionally use smaller block sizes and DSS systems occasionally use larger block sizes. For read-intensive operations, and regardless of the size of the data, the goal is to minimize the number of reads required to retrieve the desired data.

- If the rows are small and access is predominantly random, then choose a smaller block size.
- If the rows are small and access is predominantly sequential, then choose a larger block size.
- If the rows are small and access is both random and sequential, then it might be effective to choose a larger block size.
- If the rows are large, such as rows containing large object (LOB) data, then choose a larger block size.

Note: The use of multiple block sizes in a single database instance is not encouraged because of manageability issues.

Sizing Redo Log Files

- Size of redo log files can influence performance.
- Larger redo log files provide better performance.
- Generally, redo log files should range between 100 MB and a few gigabytes.
- Switch redo log file at most once every twenty minutes.
- Use the Redo Logfile Size Advisor to correctly size your redo logs.

ORACLE

Sizing Redo Log Files

The size of the redo log files can influence performance, because the behavior of the database writer and archiver processes depend on the redo log sizes. Generally, larger redo log files provide better performance. Undersized log files increase checkpoint activity and reduce performance.

If the `FAST_START_MTTR_TARGET` parameter is set to limit the instance recovery time, the system automatically tries to checkpoint as frequently as necessary. Under this condition, the size of the log files should be large enough to avoid additional checkpointing. The optimal size can be obtained by querying the `OPTIMAL_LOGFILE_SIZE` column from the `V$INSTANCE_RECOVERY` view. You can also obtain sizing advice on the Redo Log Groups page of Oracle Enterprise Manager Database Control.

It is not possible to provide a specific size recommendation for redo log files, but redo log files in the range of a hundred megabytes to a few gigabytes are considered reasonable. Size your online redo log files according to the amount of redo your system generates. A rough guide is to switch logs at most once every twenty minutes.

Automatic Statistics Gathering

- `STATISTICS_LEVEL = TYPICAL | ALL`
- **Statistics are gathered by the predefined `GATHER_STATS_JOB` job.**
- **This job implicitly determines the following:**
  - **Database objects with missing or stale statistics**
  - **Appropriate sampling percentage necessary to gather good statistics on those objects**
  - **Appropriate columns that require histograms and the size of those histograms**
  - **Degree of parallelism for statistics gathering**
  - **Prioritization of objects on which to collect statistics**

Automatic Statistics Gathering

Optimizer statistics are automatically gathered by GATHER_STATS_JOB. This job gathers statistics on all objects in the database that have missing or stale statistics.

This job is created automatically at database creation time and is managed by the Scheduler. GATHER_STATS_JOB gathers optimizer statistics by calling the DBMS_STATS.GATHER_DATABASE_STATS_JOB_PROC procedure. This procedure operates in a similar fashion to the DBMS_STATS.GATHER_DATABASE_STATS procedure by using the GATHER AUTO option. The primary difference is that the GATHER_DATABASE_STATS_JOB_PROC procedure prioritizes the database objects that require statistics, so that those objects that need updated statistics the most are processed first. This ensures that the most needed statistics are gathered before the maintenance window closes.

You should make sure that GATHER_STATS_JOB is enabled, and that STATISTICS_LEVEL is set to at least TYPICAL.

Automatic Statistics Collection: Considerations

- You should still manually gather statistics in the following cases:
  - After bulk operations
  - When using external tables
  - To collect system statistics
  - To collect statistics on fixed objects
- Prevent automatic gathering for volatile tables:
  - Lock with statistics for representative values
  - Lock without statistics implies dynamic sampling.

Automatic Statistics Collection: Considerations

For tables that are being bulk-loaded, the statistics-gathering procedures should be run immediately following the load process.

Automatic statistics collection is not supported for external tables, system statistics, and statistics on fixed objects (such as the dynamic performance tables). These statistics should be gathered. External tables should have statistics gathered after a change in size. Statistics should be gathered on fixed objects after a change in workload.

With Oracle Database 10*g*, you can lock statistics on a specified table with the new `LOCK_TABLE_STATS` procedure of the `DBMS_STATS` package.

You can lock statistics on a volatile table at a point when it is fully populated so that the table statistics are representative of the table population. This is good for interim tables that are volatile, but have a reasonable consistent size.

You can lock statistics on a table without statistics to prevent automatic statistics collection so that you can use dynamic sampling on a volatile table with no statistics. This option works best on tables that have a widely ranging size.

Commonly Observed Wait Events

Wait Event	Area	Possible cause	Examine
buffer busy waits	Buffer cache, DBWR	Depends on buffer type. PK index and seq.	V$SESSION_WAIT (block) while issue is occurring
free buffer waits	Buffer cache, DBWR, I/O	Slow DBWR	Write time using OS stats. Buffer cache stats
db file scattered read, db file sequential read	I/O, SQL Tuning	Poorly tuned SQL, Slow I/O system	V$SQLAREA disk reads. V$FILESTAT read time
Enqueue waits (enq:)	Locks	Depends on enq type	V$ENQUEUE_STAT
Library cache waits	Latches	SQL parsing/sharing	V$SQLAREA parse calls, child cursors
log buffer space	Log buffer I/O	Small buffer, slow I/O	V$SYSSTAT redo buffer allocation retries, disk
Log file sync	Over-commit, I/O	Slow I/O, un-batched commits	commits + rollbacks from V$SYSSTAT, Disks.

Commonly Observed Wait Events

This slide links wait events to possible causes and gives an overview of the diagnostics that could be most useful to review next.

You may also want to review the following Oracle MetaLink notices on `buffer busy waits` (34405.1) and `free buffer waits` (62172.1):

- http://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_database_id=NOT&p_id=34405.1
- http://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_database_id=NOT&p_id=62172.1

Additional Statistics

Statistic name	Description	Recommended action
Redo Log Space Requests	How many times a server process had to wait for space in the online redo log	Checkpoints, DBWR, or archiver activity should be tuned, larger log files
Consistent changes	How many rollbacks done for consistent read purposes	Use automatic undo, tune the workload
Consistent gets	How many blocks are read in Consistent Read mode	Use automatic undo, tune the workload
Table Fetch by Continued Row	Migrated or chained rows	Reorganize

Additional Statistics

There are several statistics that can indicate performance problems that do not have corresponding wait events. The most important are listed in the slide.

The `redo log space requests` statistic indicates how many times a server process had to wait for space in the online redo log, not for space in the redo log buffer.
A significant value for this statistic and the wait events should be used as an indication that checkpoints, DBWR, or archiver activity should be tuned, not LGWR. Increasing the size of the redo log buffer does not help. Increase the log file size, reducing the amount of time spent switching log files.

A large number of `consistent gets` and `consistent changes` can indicate that the workload has long-running queries and many short transactions against the same tables. Other scenarios that produce these statistics are the result of rollback segment contention.

Top 10 Mistakes Found in Oracle Systems

1. Bad connection management
2. Bad use of cursors and shared pool
3. Bad SQL
4. Use of nonstandard initialization parameters
5. Getting database I/O wrong
6. Redo log setup problems
7. Serialization of data blocks in the buffer cache
8. Long full-table scans
9. High amount of recursive SQL
10. Deployment and migration errors

ORACLE

Top 10 Mistakes Found in Oracle Systems

This slide lists the most common mistakes found in Oracle systems:

1. The application connects and disconnects for each database interaction. This problem is common with stateless middleware in application servers. It has more than two orders of magnitude impact on performance, and is not scalable.
2. Not using shared cursors results in repeated parses. If bind variables are not used, then there is hard parsing of all SQL statements. This has an order of magnitude impact in performance, and it is not scalable. Use cursors with bind variables that open the cursor and execute it many times. Be suspicious of applications generating dynamic SQL.
3. Bad SQL is SQL that uses more resources than appropriate for the application requirement. This can be a decision support system (DSS) query that runs for more than 24 hours or a query from an online application that takes more than a minute. SQL that consumes significant system resources should be investigated for potential improvement. ADDM identifies high-load SQL and the SQL Tuning Advisor can be used to provide recommendations for improvement.

Top 10 Mistakes Found in Oracle Systems (continued)

4. These might have been implemented on the basis of poor advice or incorrect assumptions. Most systems give acceptable performance using only the set of basic parameters. In particular, parameters associated with `SPIN_COUNT` on latches and undocumented optimizer features can cause a great deal of problems that can require considerable investigation. Likewise, optimizer parameters set in the initialization parameter file can override proven optimal execution plans. For these reasons, schemas, schema statistics, and optimizer settings should be managed together as a group to ensure consistency of performance.
5. Many sites lay out their databases poorly over the available disks. Other sites specify the number of disks incorrectly, because they configure the I/O subsystem by disk space and not by I/O bandwidth.
6. Many sites run with too few redo logs that are too small. Small redo logs cause system checkpoints to continuously put a high load on the buffer cache and I/O system. If there are too few redo logs, then the archive cannot keep up, and the database waits for the archive process to catch up.
7. Serialization of data blocks in the buffer cache due to lack of free lists, free list groups, transaction slots (`INITRANS`), or shortage of rollback segments. This is particularly common on insert-heavy applications, in applications that have raised the block size above 8 KB, or in applications with large numbers of active users and few rollback segments. Use Automatic Segment Space Management (ASSM) and Automatic Undo Management to solve this problem.
8. Long full-table scans for high-volume or interactive online operations could indicate poor transaction design, missing indexes, or poor SQL optimization. Long table scans, by nature, are I/O intensive and not scalable.
9. Large amounts of recursive SQL executed by `SYS` could indicate space management activities, such as extent allocations, taking place. This is not scalable and impacts user response time. Use locally managed tablespaces to reduce recursive SQL due to extent allocation. Recursive SQL executed under another user ID is probably SQL and PL/SQL, and this is not a problem.
10. In many cases, an application uses too many resources because the schema owning the tables has not been successfully migrated from the development environment or from an older implementation. Examples of this are missing indexes or incorrect statistics. These errors can lead to suboptimal execution plans and poor interactive user performance. When migrating applications of known performance, export the schema statistics to maintain plan stability by using the `DBMS_STATS` package.

Summary

In this lesson, you should have learned how to:

- Create your initial database following the best practices identified throughout the course
- Summarize the performance tuning methodology

Appendix A
Practices

Table of Contents

Practice for Lesson 1

No practices for this lesson.

Practice for Lesson 2

Practice 2-1: Writing Tuning Goals

You have an Order Entry database system that has a load of hundreds of users around the clock. The average response time of 12 seconds has been acceptable for several months, but recently users have started to complain about the response time. Management has noticed a drop in the number of orders entered per day. The peak order entries were 35,000 per day, and they have dropped to 28,000 per day. The latest Statspack report shows the Host CPU utilization of 57% of total CPU for Instance and 71% of busy CPU for Instance. The following table shows the top wait events:

```
Top 5 Timed Events                                                  Avg %Total
~~~~~~~~~~~~~~~~~~                                                 wait   Call
Event                                         Waits    Time (s)    (ms)   Time
-------------------------------------- ------------ ----------- ------ ------
CPU time                                                     652           33.5
buffer busy waits                             1,202          268    223    13.8
log file parallel write                      94,598          182      2     9.4
latch: cache buffers chains                     776          160    206     8.2
latch: library cache                            534          133    249     6.8
          -------------------------------------------------------------
```

1) On the basis of the scenario above, write three goals:
   a) A user-based goal: What can the user expect?
   b) A management goal: What can management expect?
   c) A database statistics–based goal: What statistics will change?
2) Which of the goals in step 1) are achievable?
3) Which of the goals in step 1) are measurable?
4) Rewrite one of the goals to make it achievable and measurable.

Practice for Lesson 3

In this practice, you find the relevant tables and files for manual performance tuning.

Practice 3-1: Using the Alert Log Information for Tuning

Review the alert log for information that is relevant to performance tuning.

1) Set up the `orcl` database for a workload. Execute the `setup` script:

```
./setup 3 1
```

2) Run a workload using the workload generator: `./workgen 3 1`
Check that the workload has started by using the `ps` command in the same terminal window.

3) View the alert log through Enterprise Manager (EM) Database Control. Click Alert Log Contents in the Related Links section at the bottom of either the Database Home page or the Performance page.

4) Find the alert log file for the `orcl` database. Use the operating system login and locate the alert log for the `orcl` database.

5) Find the settings for the `DB_BLOCK_SIZE`, `DB_MULTIBLOCK_READ_COUNT`, `FAST_START_MTTR_TARGET`, and `SGA_TARGET` parameters. **Hint:** All nondefault parameter values are listed in the alert log on every instance startup. Scroll through the alert log entries.

6) Determine how often the log files have switched in the last 10 minutes, while the workload has been running.

Practice 3-2: Viewing System Statistics and Wait Events

Use dynamic performance views to view tuning information.

1) If the workload generator is not running, review steps 1 and 2 from section 3-1. Do not run the `setup` script if it has already been run.

2) View the system statistics. Use the `V$SYSSTAT` view and select `NAME`, `CLASS`, and `VALUE`. Order the output by `CLASS`. These statistics are cumulative since the instance was started. What can you deduce about the instance workload and performance from a listing of the system statistics?

3) View system wait events. Use the `V$SYSTEM_EVENT` view to see wait event statistics. Find the total number of waits, the time waited and the average time waited for buffer busy waits, and the time waited relative to the other wait events.

4) Check the session wait events. Find the sessions that are currently waiting on `buffer busy waits`. Use the `V$SESSION_WAIT` view. Why are there very few or no sessions listed?

5) Stop the workload by executing the `rm runload` command. And clean up the practice environment with `./cleanup 3 2`.

Practice for Lesson 4

The first exercise takes you through setting server-generated alerts for some important performance metrics. You then see how you are alerted when thresholds are crossed. The second exercise explains how to set up and use metric baselines.

Practice 4-1: Working with Metrics

This lab shows you how to set up server-generated alerts as well as retrieve alert information from Enterprise Manager. During this practice, you are going to see how alerts are triggered so that you can react. However, this practice does not involve analysis of the cause of the generated alerts. This analysis will be performed during subsequent practices using Statspack, AWR snapshots, and ADDM.

1) Change to the `/home/oracle/labs` directory. Connect to the `orcl` instance as the `sysdba` user. Execute the `lab_04_01_01.sql` script to set up the necessary objects before starting the workload.

2) Using EM Database Control for the `orcl` instance, determine the current threshold values set for the `DB_TIME_WAITING` metric. Pay special attention to the Concurrency wait class. Use both Enterprise Manager and SQL*Plus to find those values.

3) Using a PL/SQL command or the `lab_04_01_03.sql` script, change the thresholds for the Concurrency class to 60% for the warning level and 90% for the critical level. Using both SQL and Enterprise Manager, verify that your change took effect.

4) Execute the `start_04_01_04.sh` script to start the workload for this practice. While the script is executing, observe the variation of the curve on the Wait Class Concurrency page. After the workload is finished, find the alert history from a SQL*Plus session. Check both the outstanding alerts and the alert history. The workload script takes up to 6 minutes to complete on some machines.

5) Execute the `lab_04_01_05.sql` script to clean up your environment.

Practice 4-2: Working with Baselines

The goal of this practice is to manipulate metric baselines. You are going to create a static metric baseline based on a past workload. Then, various workloads are executed to show how you are alerted.

1) Connect as `SYSDBA` using Database Control and enable Metric Baselines. After this is done, log out from your Database Control Console.

2) From your terminal emulator session, connected as the `oracle` user, execute the `lab_04_02_02.sh` script. This script takes as long as 15 minutes to run and loads many statistics that are needed for this practice.

3) Using Database Control, create a new static metric baseline called MB1. Make sure that you retrieve the statistics from Automatic Workload Repository (AWR) between March 12, 2005, and March 21, 2005. Also, make sure that your baseline is using only the By Hour of Day Time Grouping. After the baseline is created, retrieve and determine the statistics cycle for the Number of Transactions (per second) metric. Then, set adaptive thresholds for your baseline as follows: Select "Preserve prior threshold" and then set Occurrences to 1. Select High (.95) in the Warning Level field and Very High (.99) in the Critical Level field.

4) Look at the Baseline Normalized Metrics page, and then generate a workload of 40 transactions per second (tps). Execute the `lab_04_02_04.sh` script with a parameter of 40. Example: `./lab_04_02_04.sh 40`. Verify that the script is running with the `ps` command. You will see a process named `wkld_sess.sh`. Observe the Number of Transactions (per second) metric on the Baseline Normalized Metrics page. After a while, stop the execution of the script by deleting the `/home/oracle/workshops/runload` file. What do you observe, and why?

5) Create a workload that matches the baseline workload. Depending on the time of the day (the hour) when you do this practice, generate a transaction rate of 8 transactions per second (tps), or generate a transaction rate of 4 tps. Use the `lab_04_02_04.sh` script with a parameter to set the transaction rate. Choose the tps rate to execute on the basis of the cycle you determined in step 3 on the Baseline Statistics Visualization page. Match the hour on the visualization graph to the current time. After you make your observations, end the workload by deleting the `runload` file with `rm /home/oracle/workshops/runload`. What do you observe, and why?

6) Create a workload that exceeds the baseline workload. Depending on the time of the day (the hour) when you do this practice, generate a transaction rate of 14 transactions per second (tps), or generate a transaction rate of 9 tps. Use the `lab_04_02_04.sh` script with a parameter to set the transaction rate. Choose the tps rate to execute on the basis of the cycle you determined in step 3 on the Baseline Statistics Visualization page. Match the hour on the visualization graph to the current time. After you make your observations, end the workload by deleting the `runload` file with `rm /home/oracle/workshops/runload`. What do you observe, and why?

7) From your Database Control Console, deactivate your metric baseline, and then disable metric baselines. Then log out of Enterprise Manager.

Practice 4-2: Working with Baselines (continued)

8) To clean up your environment, execute the `lab_04_02_08.sh` script. This script can take up to 15 minutes to complete.

Practice for Lesson 5

During this practice, you are going to install Statspack and capture Statspack snapshots to analyze a workload running on your `orcl` database.

Practice 5-1: Installing Statspack

The goal of this practice is to install the Statspack repository and packages in the `orcl` database. You need to install the Statspack repository in the `SYSAUX` tablespace. Make sure you use the `oracle` password for the `PERFSTAT` user.

1) Connected as `SYSDBA` using SQL*Plus, execute the `spcreate.sql` script to create the Statspack repository and packages.
2) Make sure that you did not have errors while installing Statspack. Check the log files that were generated during the installation.

Practice 5-2: Creating Statspack Snapshots

The goal of this practice is to run a workload on your `orcl` database and capture Statspack snapshots for that workload.

1) Before you capture your first Statspack snapshot, execute the `lab_05_02_01.sh` script located in your `$HOME/labs` directory. This script creates a tablespace and a user as the setup for the rest of the practice.
2) Create a level 7 Statspack snapshot. Make sure that you retrieve the corresponding snapshot ID.
3) Execute the `start_05_02_03.sh` script from the labs directory. This script starts the workload on your `orcl` database. The script takes about 1 minute to execute.
4) After the workload finishes running, capture a new level 7 Statspack snapshot. Make sure that you retrieve the corresponding snapshot ID. The number is system assigned and is not controllable. The snap numbers are not necessarily consecutive.

Practice 5-3: Generating Statspack Reports

The goal of this lab is to generate a Statspack report, examine it, and fix any issues that you discover while interpreting the Statspack report.

1) Generate a Statspack report between the two previously captured snapshots.
2) Examine the report. Use the `less` utility, which uses `vi` navigation commands, or use the `gedit` graphic text viewer. What are your conclusions?
3) Fix the problem by implementing the recommendations from the previous step. Note that you can re-create the table for this purpose.
4) Regenerate the same workload, and verify that the problem disappears.

Practice for Lesson 6

During this practice, you are going to use AWR, ADDM, and ASH reports to identify and fix the same issues as the ones you discovered in the previous practice.

Practice 6-1: Using AWR-Based Tools

You are going to follow a similar sequence of steps as with the previous practice, but this time you will use AWR-based tools instead of Statspack. Note that during this practice, you explicitly capture AWR snapshots. However, by default, the system automatically captures those every hour.

1) Execute the `lab_06_01_01.sql` script to set up this practice.
2) Take your first AWR snapshot. You can use either Enterprise Manager or the `DBMS_WORKLOAD_REPOSITORY` package.
3) Execute the `start_06_01_03.sh` script to generate the workload for this lab.
4) Take your second AWR snapshot.
5) Look at the AWR and ADDM reports. What are your conclusions?
6) To implement the suggested recommendation, execute the `lab_06_01_06.sql` script.
7) Execute the `start_06_01_03.sh` script to start generating the workload for this lab again.
8) Take a third AWR snapshot.
9) Generate and review a Compare Periods report, also known as a diff-diff report, comparing the periods between the three snapshots.
10) Generate the ASH Report from the Top Activity page for the hottest period.

Practice for Lesson 7

The goal of this practice is to identify the sources of performance issues.

Practice 7-1: Using Enterprise Manager to Identify OS Issues

The workload generator in this practice starts an application load on the `prod` database and an OS load.

1) In a terminal window, change directory to `/home/oracle/workshops`. Start the workload generator with the command: `./workgen 7 1`. Verify that the workload generator has started the workload processes by executing the `ps` command.
2) Use Enterprise Manager to determine the source of the loads. In the browser, enter the URL http://localhost:5500/em to view the `prod` database information.
3) Find the OS process that is consuming the most CPU.
4) Stop the workload by removing the `runload` file from the `/home/oracle/workshops` directory.

Practice 7-2: Using Enterprise Manager to Identify Database Performance Issues

The goal of this practice is to use the EM performance pages to identify performance issues associated with an application workload. You will generate the same workload as in Practice 7-1, without the OS workload.

1) Start the workload generator with the command: `./workgen 7 2`.
   Navigate to the Database Home page in EM for the `prod` database. Use the URL http://localhost:5500/em. Allow the workload generator to run for at least 5 minutes.
2) After about 5 minutes, take an ADDM snapshot.
3) Find the top application performance issues by using ADDM.
4) Take a Statspack snapshot.
5) Stop the workload by removing the `runload` file from the `/home/oracle/workshops` directory.
6) Find the top database issues by using Statspack.

Practice for Lesson 8

The goal of this practice is to observe and correct various shared pool issues.

Practice 8-1: Tuning a Hard Parse Workload

In this section, you will run a hard parse workload, view the characteristic report events, and fix the problem. Set up the scenario by running the `./setup 8 1` script. Run the workload against the `prod` database. The workload is in the `/home/oracle/workshops` directory.

1) Set the environment variables for the `prod` database. Run the setup script.
2) Run the workload generator: `./workgen 8 1`.
   The workload generator runs runs a Statspack snapshot at the beginning and end. It also runs an ADDM snapshot at the beginning.
3) View the ADDM report.
4) Drill down to the recommendations. Click the finding: "SQL Statements were not shared due to the use of literals. This resulted in additional hard parses which were consuming significant database time."
5) Based on the ADDM report, what is the recommended action?
6) Create a Statspack or AWR report from the last two snapshots.
7) Review the AWR report for the characteristics of a hard parse problem.
8) Implement the ADDM recommendation: either click the ADDM report Implement button or use the `ALTER SYSTEM SET CURSOR_SHARING = FORCE SCOPE=BOTH` command.
9) Run the workload generator again and observe the differences in the tool of your choice.

Practice 8-2: Sizing the Shared Pool

In this practice, you use the Statspack and ADDM reports to diagnose and fix the shared pool size.

1) In a terminal window, run the following command: `./setup 8 2`
2) In a terminal window, determine the machine time, run the workload generator, and check that the `wkld_one_sess.sh` processes are running by using the `ps` command. The `workgen` script creates an AWR snapshot and a Statspack snapshot, and then runs the workload. Record the time: ______________

   ```
   date
   ./workgen 8 2
   ps
   ```
3) Wait about 5 minutes. In a terminal window, stop the workload with the `rm runload` command.
4) Navigate to the Performance page of the `prod` database. Create an ADDM report. Notice any findings referring to "Hard parses" (for example, "Hard parsing of SQL statement are consuming significant database time," or "Hard Parses due to an inadequately sized shared pool….").
5) Create an AWR report between the last two snapshots.
6) View the AWR report and note the values for the Instance Efficiencies section, the Top 5 Timed Events, and the Library Cache Activity Statistics for the SQL AREA namespace.
7) Find an optimum shared pool size for this workload.
8) Record the time: __________________
   Run the workload generator again: `./workgen 8 2` for 5 minutes. Then stop the workload with the `rm runload` command.
9) Navigate to the Performance page, wait until you see a drop in the Average Active Sessions graph, and then take another ADDM snapshot. What is the difference? What are the primary issues now?
10) Navigate to the Snapshots page and create an AWR report.
11) View the AWR report and compare the values from the previous report.
    Note: The actual values you see will vary from those shown in the solutions.
12) Clean up the database by running: `./cleanup 8 2`

Practice 8-3: Keeping Objects in the Shared Pool

The performance of some applications is hindered by having to reload frequently used objects into the shared pool. If an object (such as a procedure, package, or sequence) is used often but not frequently enough to prevent aging out of the shared pool, keeping it will reduce the number of reloads. Keeping objects that are used very frequently is not necessary, but it does not hurt performance, because those objects will stay in the shared pool without being kept.

1) Start the workload generator: `./workgen 8 3`

2) After the workload has run for a few minutes, choose a candidate object to `KEEP` in the shared pool. Write and save a query to find a candidate object. Write the query to test whether the object has been `KEPT`. A suggested query is:

   ```
   SELECT name, type, sharable_mem, kept
       FROM v$db_object_cache
       WHERE sharable_mem > 4000
       AND EXECUTIONS > 5
       AND (type='FUNCTION' OR type='PROCEDURE')
   /
   ```

   This query is available in the `labs` directory as `sp_objects.sql`.

3) Use the `KEEP` procedure to keep the `DEPLETE_INV PROCEDURE` object in the shared pool.

4) Flush the shared pool and observe the state of `KEPT` objects.

5) Stop the workload generator by executing the `rm runload` command.

Practice for Lesson 9

The goal of this practice is to observe the symptoms of, diagnose, and apply solutions for buffer cache problems.

Practice 9-1: Using the DB Cache Advisor

1) Complete the setup for this scenario by running: `./setup 9 1`
2) Start the workload generator with `./workgen 9 1`.
3) Wait about 5 minutes, and then in a terminal window, stop the workload with the `rm runload` command.
4) Create an ADDM report. View the Performance Analysis; check the performance findings. Find the top finding that you can address.
5) Navigate to the Snapshots page; create an AWR report. Save the report as `awr_91_1.html`.
6) Review the AWR report and identify the primary performance issue.
7) Using the Buffer Pool Advisory in the AWR report, what is an appropriate size for the database buffer cache?
8) Navigate to the Memory Parameters page. Use the Buffer Cache Advisor. What is the appropriate buffer cache size?
9) Change the buffer cache size. To reduce the number of trials, increase `DB_CACHE_SIZE` to 60 MB.
10) Start the workload generator again: `./workgen 9 1`
11) After about 5 minutes, stop the workload with the `rm runload` command.
12) Create an ADDM report. View the Performance Analysis; check the performance findings. Did the primary issues change?
13) Navigate to the Snapshots page, and create an AWR report.
14) Review the AWR report and identify the primary performance changes.
15) Run the cleanup script: `./cleanup 9 1`

Practice 9-2: Using the Keep Pool

In this scenario, assume the workload as in scenario 9-1 but you are constrained on memory resources. You are allowed only 112 MB of memory. The shared pool is already set as small as possible at 80 MB. Can you find a way to achieve a reasonable performance for both shared pool and database buffer cache?

1) Run the setup script for scenario 9-2: `./setup 9 2`
2) Run the workload generator for Practice 9-1: `./workgen 9 1`
3) Wait about 5 minutes, and then end the workload by deleting the `runload` file. While you are waiting, examine the performance graph. Notice the peak values and the I/O per second values.
4) Create an ADDM report.
5) Create an AWR report and save it as `awr_92_1.html`.
6) Examine the AWR report. Are there tables that should be cached in a keep pool?
7) The developers have provided a script named `w92_keep_pool.sql` that will create a small keep pool of 4 MB, assign some small but very active tables to the keep pool, and load those tables into the pool. Review this script, and then run it.
8) Test this new configuration with the workload from scenario 9-1.
9) Wait about 5 minutes, and then end the workload by deleting the `runload` file. While you are waiting, examine the performance graph. Notice the peak values and the I/O per second values.
10) Create an ADDM report.
11) Create an AWR report.
12) Examine the AWR report and compare it with the previous report.
    Answer the following questions:
    a) What is the difference in the Top 5 Timed Events?
    b) Is there a difference in physical reads and transactions?
    c) Is there a difference in Buffer Hit %?
    d) Is the keep pool being used?
    e) What is the difference in the table scan statistics?
13) Based on the reports, what are some possible next steps?
14) Run the cleanup script for Practice 9-2. From the `/home/oracle/workshops` directory, execute: `./cleanup 9 2`

Practice for Lesson 10

The goal of this practice is to use the Automatic SGA Tuning capability of Oracle Database 10*g*.

Practice 10-1: Enabling Automatic Shared Memory

1) Prepare the instance for this practice.
   a) Run the setup script `./setup 10 1`.
   b) Enable the Automatic Shared Memory Management feature in Enterprise Manager. Assume the same situation as in Practice 9-2 (that SGA memory is constrained to 112 MB). Ensure that Maximum SGA Size is set to 112 MB.
2) Run the workload generator for Practice 9-1 again: `./workgen 9 1`
3) Observe the Performance graph. Is there any difference from the previous run of this workload?
4) After about 5 minutes, run an ADDM report. For any findings related to memory sizing, drill down to the details.
5) What are the memory parameter settings?
6) Check the Total SGA Size Advisory for an optimal SGA size.
7) Stop the workload with the `rm runload` command.
8) Based on the advisor, the memory constraint was raised to 140 MB. Make this change. (This will require a database shutdown to change the `SGA_MAX_SIZE` parameter.)
9) Run the `wksh_10_01_09.sql` script from SQL*Plus.

   ```
   $sqlplus / as sysdba @wksh_10_01_09.sql
   ```
10) Observe the initial settings for the SGA on the Memory Parameters page.
11) Start the workload for Practice 9-1 again: `./workgen 9 1`
12) Observe the Performance graph. What are the major differences between this and the previous graph?
13) After about 5 minutes, stop the workload.
14) Run an ADDM report. View the results and drill down to the detail pages
15) What would be a reasonable next tuning step based on the ADDM report?

Practice 10-2: Adjusting Memory As Workloads Change

On the basis of the recommendation of the ADDM report in the previous scenario, the memory constraint was raised to 140 MB. The current application has seen an increase in activity and now another application is being added to the database. With it comes an additional allocation of SGA memory. How do you allocate this additional memory to give the best performance? There is the additional constraint being added that the database must be more available. You are not allowed to shut down during your tuning sessions. You are allowed one shutdown to adjust the `SGA_MAX_SIZE`.

1) You are given an absolute maximum of 300 MB of SGA and are instructed to use as little as required to get reasonable performance. (This is not a well-formed goal.) Set the maximum SGA size to 300 MB. Use a reasonable guess, and set the SGA size to 200 MB for the starting point for the new workload.
2) Navigate to the Memory Parameters page and observe the initial memory settings.
3) Start the workload generator. The new workload is simulated by:
   `./workgen 10 2`
4) Observe the Performance graphs for about 5 minutes and then stop the workload.
5) What did you observe in the Performance graphs?
6) View the ADDM findings. Is there a recommendation for SGA size?
7) Check the changes to the memory parameters during the time the workload was running. What were the final buffer cache size and shared pool size? How much tuning was required to find an adequately sized SGA?
8) How would you find a possible smaller SGA size that would still be adequate?
9) Exceute a cleanup script before the next practice; run `./cleanup 10 2`.

Practice for Lesson 11

In this practice, you observe the symptoms of checkpoint and redo performance issues. You also diagnose and correct those issues.

Practice 11-1: Diagnosing Checkpoint and Redo

Use a workload generator to create a scenario that demonstrates checkpoint and redo tuning. This scenario uses the Resource Manager to limit the active session pool. The Resource Manager prevents the workload scripts from overloading the CPU and reduces contention in the database for buffer cache blocks, so that the same scripts run with similar results on different type of machines. You will see waits for the Resource Manager as shown in the Scheduler wait class.

1) Set up the database by running the `./setup 11 1` script from the `/home/oracle/workshops` directory.
2) Start a workload by running: `./workgen 11 1`
3) Observe the graphs on the Performance page. High I/O rates are always a pointer to a possible performance issue. What processes are contributing to the high rate of writes?
4) After about 5 minutes, stop the workload.
5) Create an ADDM report.
6) Create an AWR report.
7) Review the AWR report.
    a) Review the Load profile. What kind of load is being performed?
    b) Review the Top 5 Timed Events. What are the top events? Are these events expected for this type of workload?
    c) Review the Time Model Statistics section. Does this section give more insight?
    d) What sections would you check next?
8) Based on the finding of many log switches and the recommendation that the log should switch about three to four times per hour, what is the next tuning step?
9) The service level agreement requires a recovery time of 20 minutes. Find the current recovery time and adjust the `FAST_START_MTTR_TARGET` to 1200 seconds.
10) Start the workload generator again: `./workgen 11 1`
11) Observe the graphs on the Performance page.
12) After about 5 minutes, stop the workload.
13) Create an AWR report.
14) Review the AWR report.
    a) Review the Load Profile. Have the load characteristics changed?
    b) Review the Top 5 Timed Events section. What has changed?
    c) Does the Time Model give any additional information?
15) Clean up by running: `./cleanup 11 1`

Practice for Lesson 12

This lesson has a demonstration of diagnosing I/O performance problems and possible changes to mitigate those problems.

Practice 12-1: Tuning I/O: A Demonstration

This practice consists of a series of viewlets. These viewlets display some relatively severe I/O performance issues. The system has been tuned to eliminate as many other issues as possible. Automatic shared memory management is enabled with as `SGA_MAX_SIZE` and `SGA_TARGET` set to 600 MB. The demonstration system has a slower CPU and disk drives than are available in most systems, but the relative improvements in I/O performance are valid for any system. The demonstration system has the following specifications:

- RH AS 4
- 2 CPU INTEL 386 – 666 MHz
- 1 GB RAM
- 2 SCSI-3 controllers with a 40 MB/sec Transfer Rate
- 4 SCSI-3 disk drives 18 GB each, with a 40 MB/sec transfer rate
- The disks are assigned 2 per controller
- Each disk is partitioned into four partitions of 4 GB each.
- The first partition on each disk (the fastest), located on the outside edge of the disk, is not used.
- The second partition of each disk is assigned to the ASM disk group +DG2.
- The third partition of each disk is mounted as /a3, /b3, /c3, and /d3. These are the partitions used for the single disk and manual striping demonstrations
- The fourth partitions on each disk are only used for incidental activity, such as the flash recovery area. These partitions are the slowest.
- The insert activity is against the `LOADTEST*` tablespaces There are 10 tablespaces. The setup script for each test drops and recreates the `LOADTEST*` tablespaces.
- There will be significant undo and redo generated during each test.
- The source of data for the inserts is the example tablespace. There will be significant reads against the `EXAMPLE` tablespace.

The workload consists of 10 sessions each simulating a different user each user is assigned to a different tablespace. Each user creates tables from existing tables in the `EXAMPLE` tablespace, and then inserts additional rows, updates those rows, and deletes a fraction of the rows. Each user runs the workload once and exits. The start time and end time for each user session is recorded in the table `SYSTEM.CYCLES`. The elapsed time for each session, and the average time for all session provide a clear indicator of the performance differences.

In a terminal window issue the command:

```
mozilla file:///home/oracle/labs/lab_12_01.html
```

Click the links in this file to play the viewlets.

Practice 12-1: Tuning I/O: A Demonstration (continued)

1) View the I/O Performance Test with Single Disk (`SINGLE_DISK_PERF_TEST`). The configuration for this test has all control files and data files on one disk (/a3) and the redo log files on another disk (/b3). Notice the ADDM findings record the redo average write speed _______________ and the data file write throughput speed_________________. Record the average session time _______________.

2) View the I/O Performance Test with Manual Striping (`MANUAL_STRIPE_PERF_TEST`). The configuration for this test has the `SYSTEM`, `UNDO`, `SYSAUX`, and `EXAMPLE` tablespaces each on a different disk. The four redo log files are each on a different disk. The `LOADTEST*` tablespaces each have four data files; they are arranged so that the first data file of each is on a different disk. When there are multiple data files, each new extent that is allocated will be allocated on the next data file. The behavior of extent allocation will give a striping across all the data files for the `LOADTEST*` tablespaces. Notice the ADDM findings record the redo average write speed _______________ and the data file write throughput speed_________________. Record the average session time _______________.

3) View the I/O Performance Test with ASM. (`ASM_PERF_TEST`). Notice the ADDM findings record the redo average write speed _______________ and the data file write throughput speed_________________. Record the average session time _______________.

4) What conclusions can you draw from these demonstrations?

Practice for Lesson 13

In this practice, you use the appropriate tools to adjust the PGA memory usage and temporary tablespace properties for improved performance.

Practice 13-1: Tuning `PGA_AGGREGATE_TARGET`

You have a DSS system that has a variety of queries being performed. Users are complaining that the queries are slow. `PGA_AGGREGATE_TARGET` is set at 20 MB. Determine the smallest setting that eliminates all multipass work areas.

1) Be sure that the environment variables are set to access the `prod` database and the working directory is `/home/oracle/workshops`. Execute the setup script for this practice: `./setup 13 1`.
2) Start the workload generator with `./workgen 13 1`. This workload consists of several sessions doing a variety of queries that require different sizes of work areas.
3) Wait a minute or two and then, while the workload is running, use the PGA Advisor and PGA Memory Usage Details to select a new `PGA_AGGREGATE_TARGET`.
4) After the workload has been running about 5 minutes, stop the workload with `rm runload.`
5) Create an AWR snapshot and an AWR report for the last two snapshots.
6) Review the AWR report for symptoms and recommendations concerning PGA area sizing. Save the report.
   a) Check Load Profile for physical reads and writes.
   b) Determine the SQL with the greatest number of reads. View the SQL statements to determine the cause of the high number of reads.
   c) Determine the extra writes that are caused by the current PGA sizing.
   d) Determine the recommended PGA sizing from the AWR report.
   e) Observe the number of multipass and one-pass work areas that are being used.
7) Make a new setting for `PGA_AGGREGATE_TARGET` on the basis of the value you determined.
8) Execute the workload again.
9) After a few minutes, check the PGA Advisor and the PGA Memory Usage Details pages.
   a) View the PGA Advice page; have the recommended values changed?
   b) View the PGA Memory Usage Details page. Are there any multipass work areas?
10) After the workload has been running about 5 minutes, stop the workload with `rm runload`.
11) Create an AWR snapshot and an AWR report for the last two snapshots.
12) Review the AWR report for symptoms, recommendations concerning PGA area sizing, and changes that show improved performance.
    a) Save this report to file for later viewing. Name the file `awr13_1_12.html`.
    b) Check Load Profile for physical reads and writes per second.
    c) Have the SQL statements that perform the most reads changed?

Practice 13-1: Tuning PGA_AGGREGATE_TARGET (continued)

d) Determine the improvement in the PGA cache hit percentage. How could the extra write statistic be decieving?

e) Determine the recommended PGA sizing for the AWR report.

f) Were there any multipass work areas in the reported period?

Practice 13-2: Tuning Temporary Tablespace Performance

The current temporary tablespace is a locally managed temporary tablespace. To compare the performance with that of a dictionary-managed temporary tablespace, create a dictionary-managed temporary tablespace and make it the database default tablespace. Execute the workload for Practice 13-1 again. Compare the AWR reports.

1) The `setup` script for this practice creates a dictionary-managed temporary tablespace named `TEMP_D` with a 150 MB data file named `temp_D01`. Set the data file properties to autoextend on, maxsize 300 MB, and next 1 MB. Then make it the default temporary tablespace for the database. Run `./setup 13 2`.
2) Generate the Practice 13-1 workload with `./workgen 13 1`.
3) After the workload has been running about 5 minutes, stop the workload with `rm runload`.
4) Create an AWR snapshot and create an AWR report.
5) Compare the AWR report for this set of snapshots to the previous AWR report in Practice 13-1, step 12. You saved this report to `awr13_1_12.html`.
    a) Compare the Load Profile.
    b) Compare the Top 5 Timed Events.
    c) Has there been a change in the PGA work area statistics?
    d) Is there a change in the elaped time per execution of the longest-running SQL statements?
6) Save this AWR report as `awr13_2_5.html`.
7) The primary difference between the locally managed tablespace and the dictionary-managed temporary tablespace is in space allocation. The dictionary-managed tablespace uses locks and enqueues to manage space allocation in extent tables in the system tablespace, while the locally managed temporary tablespace uses bitmaps for extent management. Check the difference in Enqueue Activity in the Wait Statistics section of each report.
8) Run the cleanup script for 13-2: `./cleanup 13 2`

Practice for Lesson 14

In this practice, you change tablespace properties that are impacting performance. These changes do not apply to every database; they are specific to certain types of applications.

Practice 14-1: Tuning Database Space Usage

In this practice, you convert a dictionary-managed tablespace to a locally managed tablespace. You will also observe performance differences between dictionary-managed and locally managed tablespaces.

The workload for this scenario is a high-volume insert with multiple processes. This load requires many new extents. This scenario uses the `prod` database; therefore, set the environment variables appropriately.

1) Set the environment variables. Run the setup script for Practice 14-1 found in the `/home/oracle/workshops` directory.

```
$ cd /home/oracle/workshops
$ ./setup 14 1
```

   The setup script moves the OE schema to the `USERS` tablespace. The `USERS` tablespace is dictionary managed.

2) Run the workload. Use the `workgen` script:

```
$ ./workgen 14 1
```

   Run the ADDM report as soon as the `take ADDM snapshot now` prompt appears. Do *not* convert the tablespace to a locally managed tablespace until the script is finished.

3) The workload will end after 3 minutes. Then run an ADDM report and view the findings. Drill down on findings related to database blocks.

4) Convert the `USERS` tablespace to locally managed. Use the Enterprise Manager Database Control to convert the tablespace to locally managed.

5) Run the workload again.

6) Navigate to the Performance page, run an ADDM report, and view the findings.

7) What conclusions can you draw from this?

   There is a small but measurable improvement in performance with the simulated workload. More improvements are expected for a high-volume insert load using locally managed tablespaces. The performance difference will be magnified with workloads that perform more extent allocations.

8) Reset your environment; run the `cleanup 14 1` script.

Practice for Lesson 15

No practices for this lesson.

Practice for Lesson 16

Practice 16 is the optional practice for Appendix C.

Practice 16-1: Using Services with a Single-Instance Oracle Database (Optional)

The first step in creating a service configuration is to plan services for each application or application function that uses your database.

In this practice, you create the following configuration in the `orcl` database:

Service Name	**Usage**	**Response Time (sec)– Warning/Critical**
`SERV1`	Client service	0.4, 1.0

1) Use the `DBMS_SERVICE` package to create a service called `SERV1`. Then, make sure that you add your service name to your `tnsnames.ora` file.
2) After you have created your services, try connecting to your database by using your service name. What happens and why?
3) How would you make sure that you could connect using your service? Do it, and connect to your instance using your service.
4) Execute the `lab_c_01_04.sql` script `as sysdba`. This script creates a new user; connect to your instance using this user and the `SERV1` service. The script then executes the following query:

   ```
   SELECT COUNT(*) FROM DBA_OBJECTS,DBA_OBJECTS,DBA_OBJECTS
   ```

   Do not wait for the script to complete continue to next step.
5) After the execution starts, access the EM Top Consumers page from the Performance tabbed page, and check that `SERV1` is using more and more resources. Also, check the statistics on your service with `V$SERVICE_STATS` from a SQL*Plus session connected as `SYSDBA`.
6) Set alert thresholds for your `SERV1` service by using Database Control. Specify the values defined at the beginning of this practice.
7) From your terminal emulator session, execute the `lab_c_01_07.sql` script. This script executes a workload under the `SERV1` service. Observe the Metric Value graph on the Service Response Time (per user call) (microseconds): Service Name SERV1 page. What is your conclusion?
8) Use Database Control to remove the thresholds that you specified during this practice.

Practice 16-2: Trace Services in a Single-Instance Oracle Environment (Optional)

Unless specified differently, you should log in as `SYSDBA` under your Database Control Console.

1) Execute the `lab_c_02_01.sh` script to remove all the trace files already generated in your `$ORACLE_BASE/admin/orcl/udump` directory. Then, execute the `start_servwork.sh` script to start four sessions using the `SERV1` service.
2) Using Database Control, determine the list of services, modules, and actions that the workload is using.
3) Using Database Control, enable statistics aggregation for both `SERV1/MODULE1/ACTION1` and `SERV1/MODULE2/ACTION1`.
4) Using Database Control, enable tracing for both `SERV1/MODULE1/ACTION1` and `SERV1/MODULE2/ACTION1`. Let the system generate the trace files for one minute, and then disable tracing for both `SERV1/MODULE1/ACTION1` and `SERV1/MODULE2/ACTION1`. After this is done, determine the list of generated trace files.
5) Stop your workload by executing the `stop_servwork.sh` script, and then start it up again by executing the `start_servwork.sh` script. When done, check that the statistics for your enabled aggregation actions are increasing.
6) Look again at your Top Actions page. You should see statistics refreshing for your two actions.
7) Using Database Control, disable statistic aggregations for both the actions.
8) Stop the workload by using the `stop_servwork.sh` script. Use the trace session utility and `tkprof` to interpret the generated trace files for the `SERV1` service and the `MODULE1` module.

Appendix B
Practice Solutions

Table of Contents

Practice Solutions for Lesson 1

No practices for this lesson.

Practice Solutions for Lesson 2

Solutions for Practice 2-1: *Writing Tuning Goals*

You have an Order Entry database system that has a load of hundreds of users around the clock. The average response time of 12 seconds has been acceptable for several months, but recently users have started to complain about the response time. Management has noticed a drop in the number of orders entered per day. The peak order entries were 35,000 per day, and they have dropped to 28,000 per day. The latest Statspack report shows the Host CPU utilization of 57% of total CPU for Instance and 71% of busy CPU for Instance. The following table shows the top wait events:

```
Top 5 Timed Events                                          Avg %Total
~~~~~~~~~~~~~~~~~~                                         wait   Call
Event                                      Waits    Time (s)   (ms)   Time
-------------------------------------- ------------ ----------- ------ ------
CPU time                                                   652          33.5
buffer busy waits                             1,202         268    223   13.8
log file parallel write                      94,598         182      2    9.4
latch: cache buffers chains                     776         160    206    8.2
latch: library cache                            534         133    249    6.8
          -------------------------------------------------------------
```

1) On the basis of the scenario above, write three goals:

   a) A user-based goal: What can the user expect?

   b) A management goal: What can management expect?

   c) A database statistics–based goal: What statistics will change?

   Answer:

   Sample answers are the following (your answers may vary):

   a. Average user response time will be 12 seconds or less.

   b. Order entry rate will be supported to 35,000 per day.

   c. Reduced time for waits

2) Which of the goals in step 1) are achievable?

   Answer:

   Average user response time will be 12 seconds or less. This may be achievable, but how can you know for sure?

   Order entry rate will be supported to 35,000 per day. This may be achievable, but until the order entry rate reaches a level that the database does not support, or 35,000, how can you know for sure?

   Reduced time in waits is achievable, but is very vague. Which waits will be reduced?

Solutions for Practice 2-1: Writing Tuning Goals (continued)

3) Which of the goals in step 1) are measurable?

   Answer:

   Average user response time will be 12 seconds or less. Measuring a user response time at a certain load rate for a particular operation is possible. Measuring an average response time over all possible load rates and all possible operations is impractical.

   Order entry rate will be supported to 35,000 per day. This goal is measurable if there is a way to simulate this load rate. The point of the scenario is that management is concerned that the database is limiting the orders. The cause could be something unrelated to the database that is reducing the number of orders, such as an economic downturn.

4) Rewrite one of the goals to make it achievable and measurable.

   Answer:

   Reduce buffer busy waits to less than 10% of the Total Call Time.
   Note: These are examples. The scenario given was deliberately vague to correspond to the information that you as a DBA may be given. To write more meaningful goals, further investigation would be needed in almost all cases.

Practice Solutions for Lesson 3

In this practice, you find the relevant tables and files for manual performance tuning.

Solutions for Practice 3-1: *Using the Alert Log Information for Tuning*

Review the alert log for information that is relevant to performance tuning.

1) Set up the `orcl` database for a workload. Execute the `setup` script:

```
./setup 3 1
```

Answer:

```
$ cd /home/oracle/workshops
$ ./setup 3 1

Sequence dropped.


PL/SQL procedure successfully completed.


Table altered.


Table altered.


Procedure created.


System altered.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


System altered.

$
```

2) Run a workload using the workload generator: `./workgen 3 1`
Check that the workload has started by using the `ps` command in the same terminal window.

Answer:

You will see a set of processes as illustrated below. The order and number of each will vary. In most cases you will see the `wkls_sess.sh`, `sqlplus`, and `sleep` processes. Not all of these processes may be active at the time the `ps` command is issued.

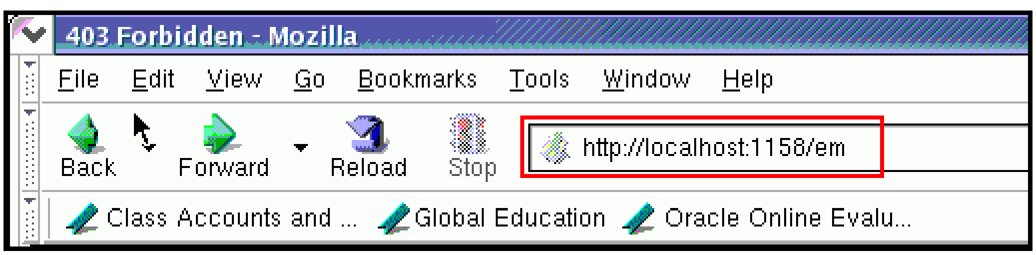
Solutions for Practice 3-1: Using the Alert Log Information for Tuning (continued)

```
 $ cd /home/oracle/workshops  $ ./workgen 3 1
 $ ps
   PID TTY          TIME CMD
28276 pts/2     00:00:00 bash
  673 pts/2     00:00:00 wkld_sess.sh
  677 pts/2     00:00:00 wkld_sess.sh
  679 pts/2     00:00:00 wkld_sess.sh
  … Lines deleted …
  706 pts/2     00:00:00 wkld_sess.sh
  708 pts/2     00:00:00 wkld_sess.sh
  709 pts/2     00:00:00 wkld_sess.sh
  719 pts/2     00:00:00 sleep
  720 pts/2     00:00:00 sleep
  721 pts/2     00:00:00 sleep
  … Lines deleted …
  833 pts/2     00:00:00 sqlplus
  836 pts/2     00:00:00 wkld_sess.sh
  838 pts/2     00:00:00 sleep
  … Lines deleted …
  857 pts/2     00:00:00 sleep
  858 pts/2     00:00:00 ps
```

3) View the alert log through Enterprise Manager (EM) Database Control. Click Alert Log Contents in the Related Links section at the bottom of either the Database Home page or the Performance page.

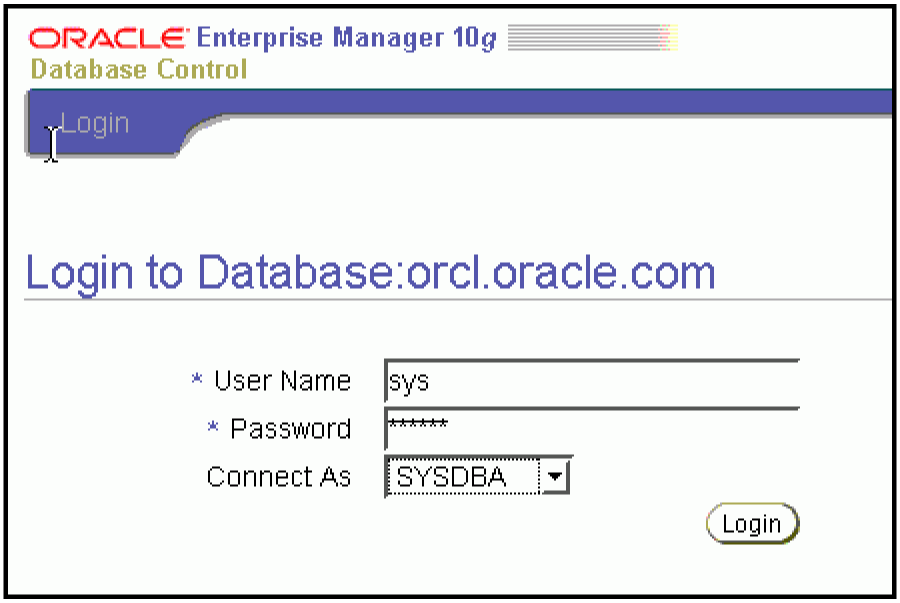
Answer:

1. Invoke the browser (Mozilla) and enter the URL for Enterprise Manager Database Control for the `orcl` database. The URL is shown here:

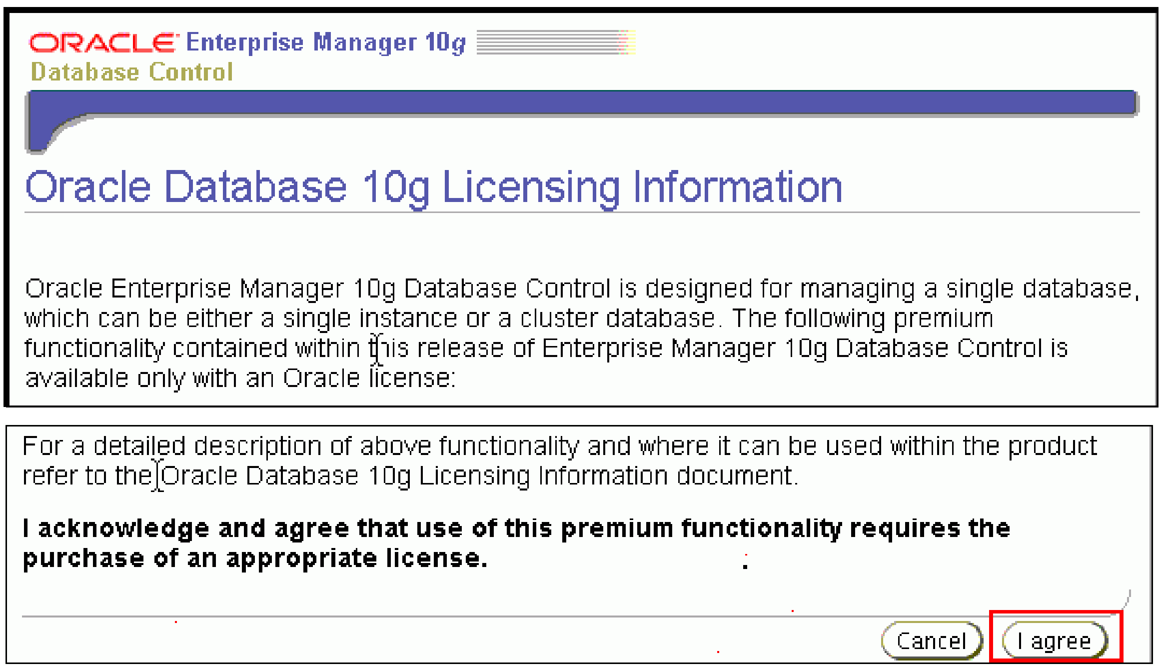


2. Log in to EM for the `orcl` database.

Solutions for Practice 3-1: Using the Alert Log Information for Tuning (continued)

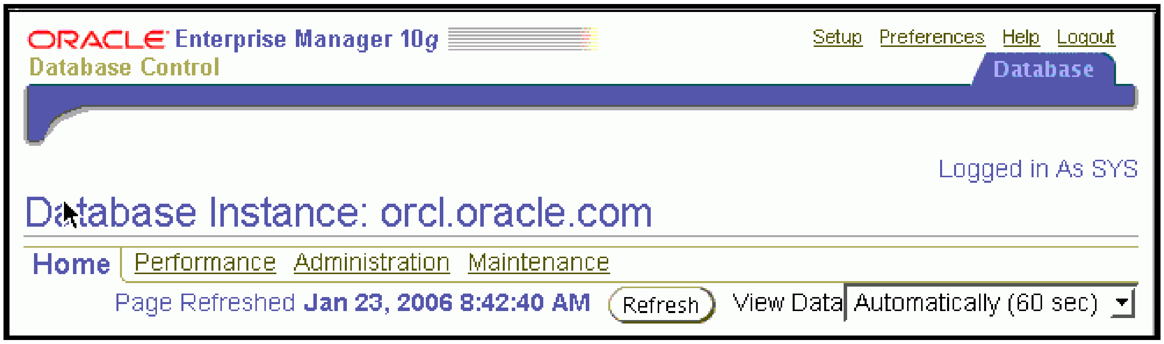


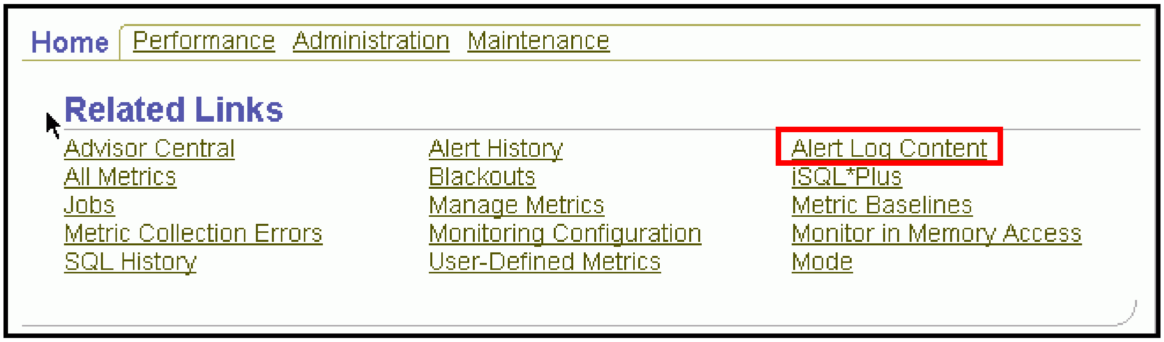
3. If this is the first login to EM Database Control for this database and user, then the Licensing Information page appears. Scroll to the bottom of the page and click "I agree."



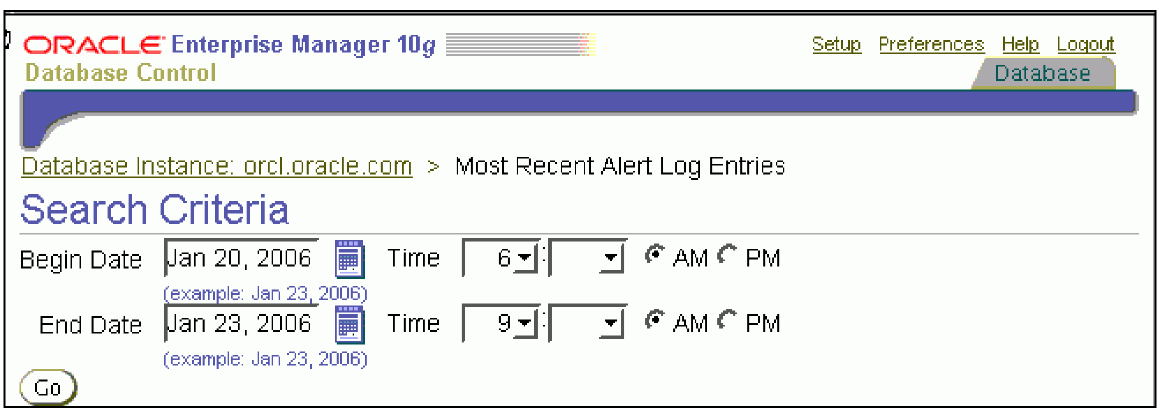
4. From the Database Home page, navigate to the Alert Log Content page. Scroll to the bottom of the Database Home page and click the Alert Log Content link in the Related Links section.

Solutions for Practice 3-1: Using the Alert Log Information for Tuning (continued)





5. Enter a begin date and an end date that include the current time (the date of the Friday preceding today and an hour before the current time to a time later than now).



Solutions for Practice 3-1: Using the Alert Log Information for Tuning (continued)

Most Recent Alert Log Entries

Page Refreshed Jan 23, 2006 8:55:32 AM

Number of Lines Displayed **427**

```
Fri Jan 20 11:28:11 2006
Starting ORACLE instance (normal)
Cannot determine all dependent dynamic libraries for /proc/self/exe
Unable to find dynamic library libocr10.so in search paths
RPATH = /ade/aime1_build2101/oracle/has/lib/:/ade/aime1_build2101/oracle/lib/:/ade/aime1_build2101/oracle/has/lib/:
LD_LIBRARY_PATH is not set!
The default library directories are /lib and /usr/lib
Unable to find dynamic library libocrb10.so in search paths
Unable to find dynamic library libocrutl10.so in search paths
Unable to find dynamic library libocrutl10.so in search paths
LICENSE_MAX_SESSION = 0
LICENSE_SESSIONS_WARNING = 0
Picked laUsing LOG_ARCHIVE_DEST_10 parameter default value as USE_DB_RECOVERY_FILE_DEST
Autotune of undo retention is turned on.
IMODE=BR
ILAT =18
LICENSE_MAX_USERS = 0
```

4) Find the alert log file for the `orcl` database. Use the operating system login and locate the alert log for the `orcl` database.

Answer:

1. In a terminal window, use SQL*Plus to find the `BACKGROUND_DUMP_DEST` directory.

```
[oracle@edrsr3p1 bdump]$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Mon Dec 5
17:43:08 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> show parameter dump

NAME                   TYPE        VALUE
---------------------- ----------- ------------------------------
background_core_dump   string      partial
background_dump_dest   string      /u01/app/oracle/admin/orcl/bdu
                                   mp
core_dump_dest         string      /u01/app/oracle/admin/orcl/cdu
                                   mp
max_dump_file_size     string      UNLIMITED
shadow_core_dump       string      partial
user_dump_dest         string      /u01/app/oracle/admin/orcl/udu
                                   mp


SQL> exit
```

2. In the terminal window, view the alert log.

```
$ cd /u01/app/oracle/admin/orcl/bdump
$ less alert_orcl.log

Thu Jul  7 01:57:05 2005
Starting ORACLE instance (normal)
LICENSE_MAX_SESSION = 0
LICENSE_SESSIONS_WARNING = 0
Shared memory segment for instance monitoring created
Picked latch-free SCN scheme 2
Using LOG_ARCHIVE_DEST_10 parameter default value as
USE_DB_RECOVERY_FILE_DEST
Autotune of undo retention is turned on.
IMODE=BR
ILAT =18
LICENSE_MAX_USERS = 0
SYS auditing is disabled
ksdpec: called for event 13740 prior to event group
initialization
Starting up ORACLE RDBMS Version: 10.2.0.1.0.
System parameters with non-default values:
  processes                = 150
  sga_target               = 285212672
  control_files            =
/u01/app/oracle/oradata/orcl/control01.ctl,
/u01/app/oracle/oradata/orcl/control02.ctl,
/u01/app/oracle/oradata/orcl/control03.ctl
  db_block_size            = 8192
  compatible               = 10.2.0.1.0
  db_file_multiblock_read_count= 16
  db_recovery_file_dest    = /u01/app/oracle/flash_recovery_area
  db_recovery_file_dest_size= 2147483648
  undo_management          = AUTO
  undo_tablespace          = UNDOTBS1
  remote_login_passwordfile= EXCLUSIVE
  db_domain                = oracle.com
  dispatchers              = (PROTOCOL=TCP) (SERVICE=orclXDB)
  job_queue_processes      = 10
  background_dump_dest     = /u01/app/oracle/admin/orcl/bdump
  user_dump_dest           = /u01/app/oracle/admin/orcl/udump

Lines deleted …
```

5) Find the settings for the `DB_BLOCK_SIZE`, `DB_MULTIBLOCK_READ_COUNT`, `FAST_START_MTTR_TARGET`, and `SGA_TARGET` parameters. **Hint:** All nondefault parameter values are listed in the alert log on every instance startup. Scroll through the alert log entries.

Answer:

1. Using SQL*Plus, use the `SHOW PARAMETER` *<parameter name>* command.

Solutions for Practice 3-1: Using the Alert Log Information for Tuning (continued)

```
$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Mon Jan 23
09:10:17 2006

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> show parameter DB_BLOCK_SIZE

NAME                                 TYPE        VALUE
----------------------------- ----------- ----------------
db_block_size                        integer     8192
SQL> show parameter DB_FILE_MULTIBLOCK_READ_COUNT

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------
db_file_multiblock_read_count        integer     16
SQL> show parameter MTTR

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------
fast_start_mttr_target               integer     0

 Note:FAST_START_MTTR_TARGET is not set for this instance

SQL> show parameter SGA_TARGET

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------
sga_target                           big integer 272M
```

2. Alternatively, scroll through the alert log entries. Start from the most recent entries in the file to find the current parameter setting. Note that with this method, you will see only those parameters that have been set to a nondefault value.

```
Starting up ORACLE RDBMS Version: 10.2.0.1.0.
System parameters with non-default values:
  processes                = 150
  __shared_pool_size       = 88080384
  __large_pool_size        = 4194304
  __java_pool_size         = 4194304
  __streams_pool_size      = 0
  sga_target               = 285212672
  control_files            =
/u01/app/oracle/oradata/orcl/control01.ctl,
```

```
/u01/app/oracle/oradata/orcl/control02.ctl,
/u01/app/oracle/oradata/orcl/control03.ctl
  db_block_size            = 8192
  __db_cache_size          = 184549376
  compatible               = 10.2.0.1.0
  db_file_multiblock_read_count= 16
  db_recovery_file_dest    = /u01/app/oracle/flash_recovery_area
  db_recovery_file_dest_size= 2147483648
  undo_management          =
  undo_tablespace          = UNDOTBS1
  remote_login_passwordfile= EXCLUSIVE
  db_domain                = oracle.com
  dispatchers              = (PROTOCOL=TCP) (SERVICE=orclXDB)
  job_queue_processes      = 10
  background_dump_dest     = /u01/app/oracle/admin/orcl/bdump
  user_dump_dest           = /u01/app/oracle/admin/orcl/udump
  core_dump_dest           = /u01/app/oracle/admin/orcl/cdump
  audit_file_dest          =
/u01/app/oracle/admin/orcl/adump
  db_name                  = orcl
  open_cursors             = 300
  pga_aggregate_target     = 94371840
```

6) Determine how often the log files have switched in the last 10 minutes, while the workload has been running.

   Answer:

   The log file may have switched several times. In the following example, the log file switch is taking place about every 6 minutes. The tail command lists the last 10 lines of a file. The –n 20 option causes tail to display 20 lines. **Note:** On slower machines the log file may not have switched.

```
$ cd $ORACLE_BASE/admin/orcl/bdump
$ tail -n 20 alert_orcl.log

Mon Jan 23 08:52:19 2006
Thread 1 advanced to log sequence 22
  Current log# 3 seq# 22 mem# 0:
/u01/app/oracle/oradata/orcl/redo03.log
Mon Jan 23 08:58:43 2006
Thread 1 advanced to log sequence 23
  Current log# 1 seq# 23 mem# 0:
/u01/app/oracle/oradata/orcl/redo01.log
Mon Jan 23 09:04:48 2006
Thread 1 advanced to log sequence 24
  Current log# 2 seq# 24 mem# 0:
/u01/app/oracle/oradata/orcl/redo02.log
Mon Jan 23 09:10:55 2006
Thread 1 advanced to log sequence 25
  Current log# 3 seq# 25 mem# 0:
/u01/app/oracle/oradata/orcl/redo03.log
```

Solutions for Practice 3-1: Using the Alert Log Information for Tuning (continued)

```
Mon Jan 23 09:17:04 2006
Thread 1 advanced to log sequence 26
  Current log# 1 seq# 26 mem# 0:
/u01/app/oracle/oradata/orcl/redo01.log
```

Solutions for Practice 3-2: *Viewing System Statistics and Wait Events*

Use dynamic performance views to view tuning information.

1) If the workload generator is not running, review steps 1 and 2 from section 3-1. Do not run the `setup` script if it has already been run.

2) View the system statistics. Use the `V$SYSSTAT` view and select `NAME`, `CLASS`, and `VALUE`. Order the output by `CLASS`. These statistics are cumulative since the instance was started. What can you deduce about the instance workload and performance from a listing of the system statistics?

   Answer:

   There is very little that you can determine from just a listing of the system statistics by itself. Two listings that are used to create a difference in values over a known period of time can yield rate and load information.

```
SQL> COLUMN name FORMAT A40

SQL> SELECT name, class, value
  2  FROM V$SYSSTAT
  3  ORDER BY class;

NAME                                        CLASS      VALUE
---------------------------------------- -------- ----------
SQL*Net roundtrips to/from dblink               1          0
logons cumulative                               1      13515
logons current                                  1         24
opened cursors cumulative                       1     570581
opened cursors current                          1        277
user commits                                    1     867714
user rollbacks                                  1        503
user calls                                      1     491374
recursive calls                                 1   11201127
recursive cpu usage                             1     482756
session logical reads                           1   49300922
session stored procedure space                  1          0
CPU used by this session                        1     593469
DB time                                         1   17606174
cluster wait time                               1          0
concurrency wait time                           1      97321
application wait time                           1       4367
user I/O wait time                              1      26816
session connect time                            1 1133831863
session uga memory                              1 3.8916E+12
session uga memory max                          1 9063650444
session pga memory                              1 1013164768
session pga memory max                          1 1236904672

Lines deleted …

NAME                                        CLASS      VALUE
```

```
---------------------------------- --------- ----------
redo log space requests                    2        120
redo log space wait time                   2        503
redo log switch interrupts                 2          0
redo ordering marks                        2          5
redo subscn max counts                     2          0
flashback log writes                       2          0
enqueue timeouts                           4          0
enqueue waits                              4        297
enqueue deadlocks                          4          0
enqueue requests                           4    2864146
enqueue conversions                        4     125829
enqueue releases                           4    2864130
physical read total IO requests            8      36343
physical read total multi block requests   8       1834
physical read total bytes                  8  548881920
physical write total IO requests           8     760036
physical write total multi block requests  8     634639
physical write total bytes                 8 5303801344
db block gets                              8   16403583
db block gets from cache                   8   16403209
db block gets direct                       8        374
consistent gets                            8   32897339
consistent gets from cache                 8   32897337

Lines deleted …

NAME                                    CLASS      VALUE
---------------------------------- ---------- ----------
OS User time used                         16          0
OS Involuntary context switches           16          0
OS Maximum resident set size              16          0
OS Integral shared text size              16          0
OS Integral unshared data size            16          0
OS Integral unshared stack size           16          0
OS Page reclaims                          16          0
OS Page faults                            16          0
OS Swaps                                  16          0
OS Block input operations                 16          0
OS Block output operations                16          0
OS Socket messages sent                   16          0
OS Socket messages received               16          0
OS Signals received                       16          0
OS Voluntary context switches             16          0
OS System time used                       16          0
global enqueue gets sync                  32          0
PX remote messages recv'd                 32          0
global enqueue get time                   32          0
global enqueue releases                   32          0

Lines deleted …
```

```
PX local messages sent                                  32          0
PX local messages recv'd                                32          0
PX remote messages sent                                 32          0
global enqueue gets async                               32          0
NAME                                                 CLASS      VALUE
---------------------------------- ---------- ----------
DBWR fusion writes                                      40          0
gc CPU used by this session                             40          0
gc cr block build time                                  40          0
gc cr block flush time                                  40          0
gc cr block send time                                   40          0
gc current blocks served                                40          0
gc current block pin time                               40          0
gc current block flush time                             40          0
gc current block send time                              40          0
gc cr blocks received                                   40          0

Lines deleted …

workarea memory allocated                               64          0
session cursor cache count                              64     194285
workarea executions - onepass                           64          0
workarea executions - multipass                         64          0
parse time cpu                                          64      47646
parse time elapsed                                      64      84205
parse count (total)                                     64     513645
parse count (hard)                                      64      24627
parse count (failures)                                  64          8
frame signature mismatch                                64          0
execute count                                           64    4696003
sorts (memory)                                          64    1324226
sorts (disk)                                            64          0
sorts (rows)                                            64   13165990

Lines deleted …

Misses for writing mapping                             128          0
queue update without cp update                         128          0
leaf node splits                                       128       7182
leaf node 90-10 splits                                 128       3289
branch node splits                                     128         10

Lines deleted …

363 rows selected.

SQL>
```

Solutions for Practice 3-2: Viewing System Statistics and Wait Events (continued)

3) View system wait events. Use the `V$SYSTEM_EVENT` view to see wait event statistics. Find the total number of waits, the time waited and the average time waited for buffer busy waits, and the time waited relative to the other wait events.

Answer:

The `buffer busy wait` event has collected some waits and an average wait time of 6.54 hundredths of a second. There are several wait events that are not important, such as `rdbms ipc message` and `smon timer`. These are internal waits for a command to be issued, or a timer to expire to initiate a task. They do not affect performance. Your output from this query will vary.

```
SQL> SELECT event, total_waits, time_waited, average_wait
  2  FROM V$SYSTEM_EVENT
  3  ORDER BY time_waited DESC;


EVENT                      TOTAL_WAITS TIME_WAITED AVERAGE_WAIT
-------------------------- ----------- ----------- ------------
rdbms ipc message               570405    40524793        71.05
SQL*Net message from client     297609    20085962        67.49
pmon timer                       18332     4648397       253.57
Streams AQ: qmn coordinator       3444     4633755      1345.46
idle wait

virtual circuit status            1587     4633437      2919.62
Streams AQ: qmn slave idle wait   1696     4633282      2731.89
dispatcher timer                   793     4612043      5815.94
wait for unread message on       47405     4591649        96.86
broadcast channel

Streams AQ: waiting for messages  9416     4587145       487.16
in the queue

Streams AQ: waiting for time       820     4562070       5563.5
management or cleanup tasks

smon timer                         322     4488912     13940.72
jobq slave wait                  14797     4321737       292.07
log file parallel write         423745      106721          .25
latch: library cache              2501       31050        12.42
db file sequential read          13879       19901         1.43
log file sync                     7091       18103         2.55
buffer busy waits                 2050       13403         6.54
latch: cache buffers chains       1622       11020         6.79
latch: redo allocation            2364        8587         3.63
os thread startup                  707        7648        10.82
control file parallel write      17971        7620          .42
latch free                        1021        7095         6.95
latch: library cache pin           340        6526        19.19
db file scattered read            1521        5930          3.9
latch: shared pool                 584        5438         9.31
latch: In memory undo latch        473        4530         9.58
latch: redo copy                   508        4522          8.9
```

```
Lines deleted …

79 rows selected.
```

4) Check the session wait events. Find the sessions that are currently waiting on `buffer busy waits`. Use the `V$SESSION_WAIT` view. Why are there very few or no sessions listed?

   Answer:

   The `V$SESSION_WAIT` view shows waits for currently connected sessions only. The waits for all sessions, past and current, are summed in the `V$SYSTEM_EVENTS` view. Your output from this query will vary.

```
SQL> select sid, wait_time, seconds_in_wait, state
  2  from v$session_wait
  3  where event = 'buffer busy wait';

no rows selected

SQL> SELECT sid, event, total_waits,
  2  time_waited, average_wait
  3  FROM V$SESSION_EVENT
  4  WHERE event = 'buffer busy waits';


 SID EVENT             TOTAL_WAITS TIME_WAITED AVERAGE_WAIT
---- ----------------- ---------- ----------- -----------
 146 buffer busy waits          1           3        2.63
 164 buffer busy waits         76         117        1.55

SQL> exit
```

5) Stop the workload by executing the `rm runload` command. And clean up the practice environment with `./cleanup 3 2`.

   Answer:

```
$ cd /home/oracle/workshops
$ rm runload
rm: remove regular empty file `runload'? y
$ ./cleanup 3 2

System altered.


PL/SQL procedure successfully completed.

$
```

Practice Solutions for Lesson 4

The first exercise takes you through setting server-generated alerts for some important performance metrics. You then see how you are alerted when thresholds are crossed. The second exercise explains how to set up and use metric baselines.

Solutions for Practice 4-1: *Working with Metrics*

This lab shows you how to set up server-generated alerts as well as retrieve alert information from Enterprise Manager. During this practice, you are going to see how alerts are triggered so that you can react. However, this practice does not involve analysis of the cause of the generated alerts. This analysis will be performed during subsequent practices using Statspack, AWR snapshots, and ADDM.

1) Change to the `/home/oracle/labs` directory. Connect to the `orcl` instance as the `sysdba` user. Execute the `lab_04_01_01.sql` script to set up the necessary objects before starting the workload.

   Answer:

   You can find the script in your `$HOME/labs` directory.

```
$ cd /home/oracle/labs
$ . oraenv
ORACLE_SID = [orcl] ? orcl

$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Thu Dec 1
02:21:57 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> @lab_04_01_01.sql
SQL>
SQL> drop tablespace tbsjfv including contents and
datafiles;
drop tablespace tbsjfv including contents and datafiles
*
ERROR at line 1:
ORA-00959: tablespace 'TBSJFV' does not exist


SQL>
SQL> CREATE SMALLFILE TABLESPACE "TBSJFV"
  2  DATAFILE 'tbsjfv1.dbf' SIZE 50M
  3  AUTOEXTEND ON NEXT 10M MAXSIZE 200M
  4  LOGGING
  5  EXTENT MANAGEMENT LOCAL
  6  SEGMENT SPACE MANAGEMENT MANUAL;

Tablespace created.
```

Solutions for Practice 4-1: Working with Metrics (continued)

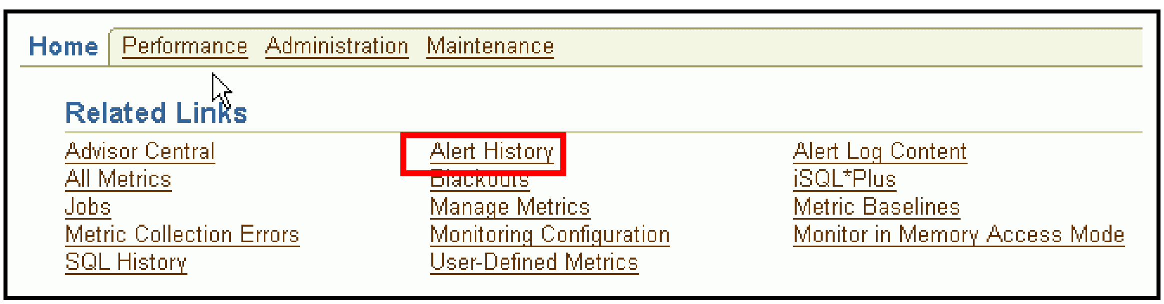
```
SQL>
SQL> execute
dbms_workload_repository.modify_snapshot_settings(interval
=> 0);

PL/SQL procedure successfully completed.

SQL>
SQL> drop user jfv cascade;
drop user jfv cascade
          *
ERROR at line 1:
ORA-01918: user 'JFV' does not exist


SQL>
SQL> create user jfv identified by jfv
  2  default tablespace tbsjfv
  3  temporary tablespace temp;

User created.

SQL>
SQL> grant connect, resource, dba to jfv;

Grant succeeded.

SQL>
SQL> connect jfv/jfv
Connected.
SQL>
SQL> drop table jfvt purge;
drop table jfvt purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> create table jfvt(id number, name varchar2(2000));

Table created.

SQL>
SQL> exec DBMS_STATS.GATHER_TABLE_STATS(-
> ownname=>'JFV', tabname=>'JFVT',-
> estimate_percent=>DBMS_STATS.AUTO_SAMPLE_SIZE);

PL/SQL procedure successfully completed.
SQL>
SQL> SQL> BEGIN
  2  DBMS_SERVER_ALERT.set_threshold(
  3     metrics_id => dbms_server_alert.DB_TIME_WAITING,
```

```
        warning_operator => dbms_server_alert.OPERATOR_GE,
        warning_value => 30,
        critical_operator => NULL,
        critical_value => NULL,
        observation_period => 1,
        consecutive_occurrences => 1,
        instance_name => 'orcl',
        object_type =>
            dbms_server_alert.OBJECT_TYPE_EVENT_CLASS,
        object_name => 'Concurrency');
   END;
   /

PL/SQL procedure successfully completed.

SQL>
SQL> exit;
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
$
```

2) Using EM Database Control for the `orcl` instance, determine the current threshold values set for the `DB_TIME_WAITING` metric. Pay special attention to the Concurrency wait class. Use both Enterprise Manager and SQL*Plus to find those values.

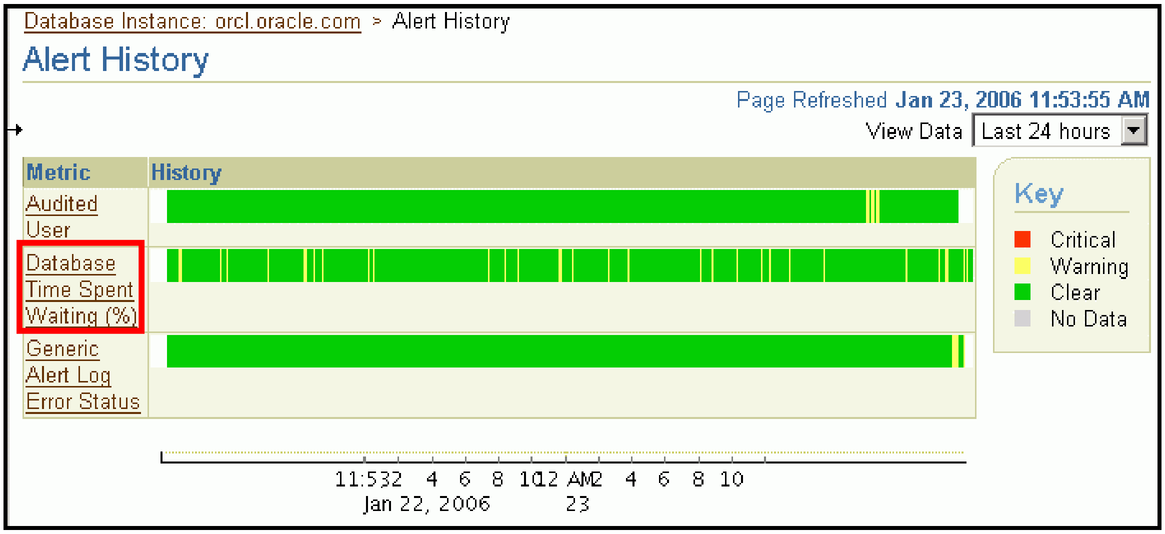
Answer:

1. On the Database Home page, scroll down to the Related Links section, and click Alert History.

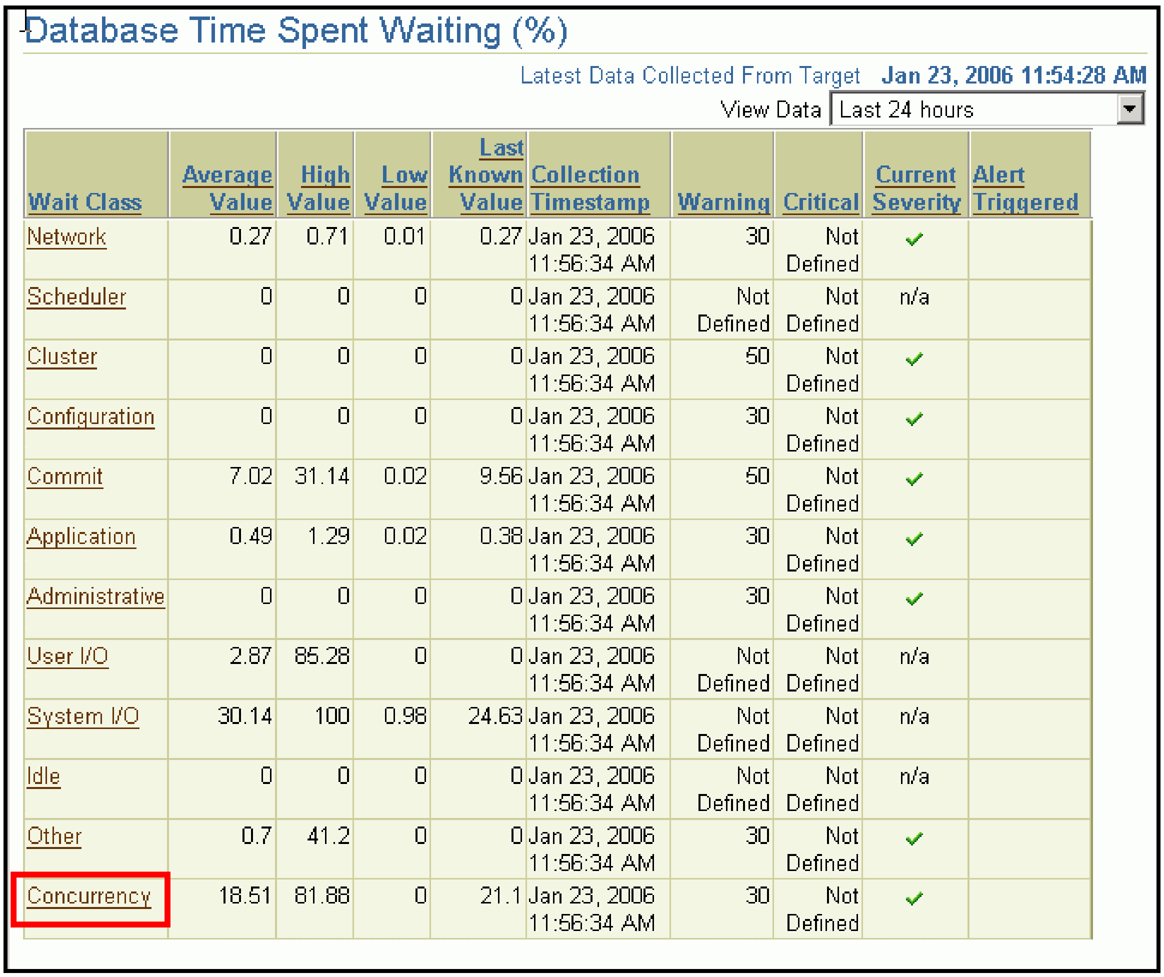


2. On the Alert History page, click Database Time Spent Waiting (%). The number of monitored alerts shown may vary.

Solutions for Practice 4-1: Working with Metrics (continued)



3. On the Database Time Spent Waiting (%) page, you can see all the thresholds set for each class. Click Concurrency.



Database Time Spent Waiting (%)

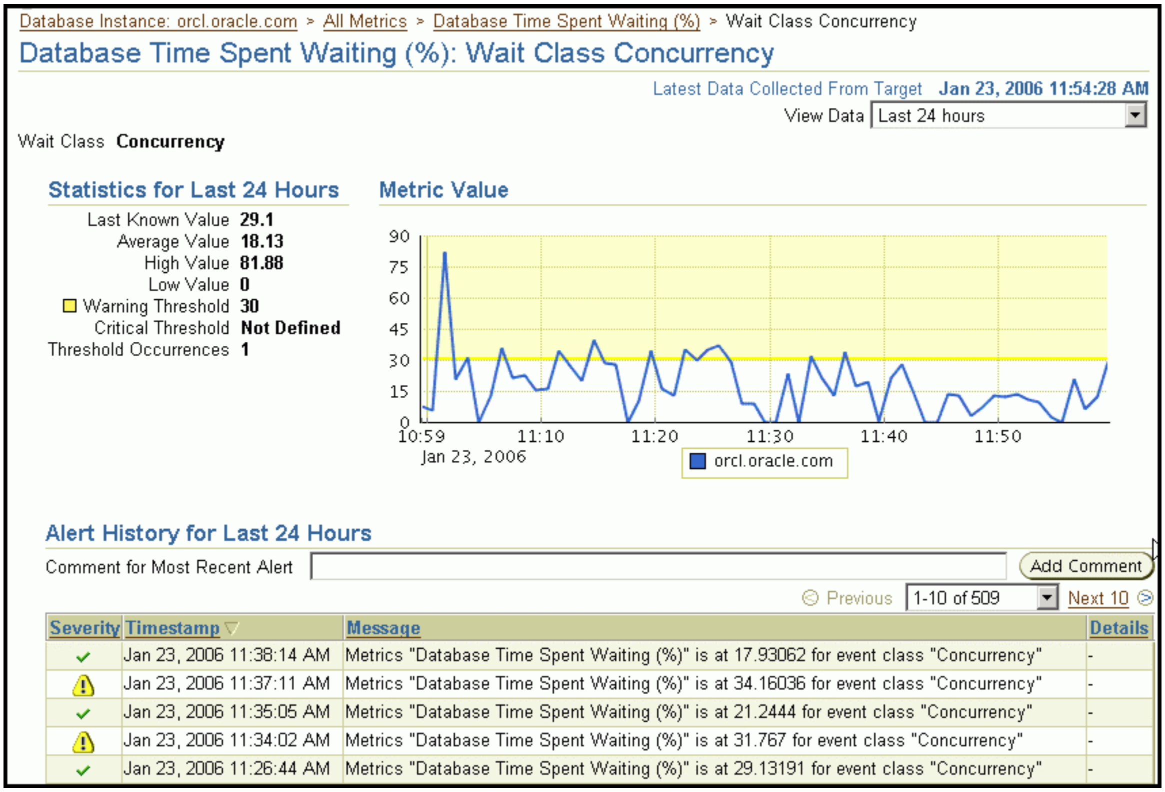
Latest Data Collected From Target Jan 23, 2006 11:54:28 AM

View Data Last 24 hours

Wait Class	Average Value	High Value	Low Value	Last Known Value	Collection Timestamp	Warning	Critical	Current Severity	Alert Triggered
Network	0.27	0.71	0.01	0.27	Jan 23, 2006 11:56:34 AM	30	Not Defined	✓	
Scheduler	0	0	0	0	Jan 23, 2006 11:56:34 AM	Not Defined	Not Defined	n/a	
Cluster	0	0	0	0	Jan 23, 2006 11:56:34 AM	50	Not Defined	✓	
Configuration	0	0	0	0	Jan 23, 2006 11:56:34 AM	30	Not Defined	✓	
Commit	7.02	31.14	0.02	9.56	Jan 23, 2006 11:56:34 AM	50	Not Defined	✓	
Application	0.49	1.29	0.02	0.38	Jan 23, 2006 11:56:34 AM	30	Not Defined	✓	
Administrative	0	0	0	0	Jan 23, 2006 11:56:34 AM	30	Not Defined	✓	
User I/O	2.87	85.28	0	0	Jan 23, 2006 11:56:34 AM	Not Defined	Not Defined	n/a	
System I/O	30.14	100	0.98	24.63	Jan 23, 2006 11:56:34 AM	Not Defined	Not Defined	n/a	
Idle	0	0	0	0	Jan 23, 2006 11:56:34 AM	Not Defined	Not Defined	n/a	
Other	0.7	41.2	0	0	Jan 23, 2006 11:56:34 AM	30	Not Defined	✓	
Concurrency	18.51	81.88	0	21.1	Jan 23, 2006 11:56:34 AM	30	Not Defined	✓	

4. On the Wait Class Concurrency page, you can see that having a default threshold value of 30% might not be a good default for your database, because many alerts were already generated despite the fact that you have not done much so far.

Solutions for Practice 4-1: Working with Metrics (continued)



5. From a SQL*Plus session, determine the threshold values for the classes above. Connect as a `sysdba` user, and then execute the `lab_04_01_02.sql` script.

```
$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Mon Jan 23
12:31:37 2006

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> @lab_04_01_02.sql
SQL>
SQL> variable wo number
SQL> variable wv varchar2(10)
SQL> variable co number
SQL> variable cv varchar2(10)
SQL> variable op number
SQL> variable ct number
SQL>
SQL> BEGIN
  2  DBMS_SERVER_ALERT.get_threshold(
  3       metrics_id => dbms_server_alert.DB_TIME_WAITING,
```

```
  4       warning_operator        => :wo,
  5       warning_value           => :wv,
  6       critical_operator       => :co,
  7       critical_value          => :cv,
  8       observation_period      => :op,
  9       consecutive_occurrences => :ct,
 10       instance_name           => 'orcl',
 11       object_type =>
 12             dbms_server_alert.OBJECT_TYPE_EVENT_CLASS,
 13       object_name             => 'Concurrency');
 14  END;
 15  /

PL/SQL procedure successfully completed.

SQL>
SQL> print  wv

WV
--------------------------------
30

SQL>
SQL> col object_name format a20
SQL> col metrics_name format a25
SQL> col warning_value format a10
SQL> col critical_value format a10
SQL>
SQL> select object_name,warning_value,critical_value
  2  from dba_thresholds
  3  where metrics_name='Database Time Spent Waiting (%)';

OBJECT_NAME          WARNING_VA CRITICAL_V
-------------------- ---------- ----------
Concurrency          30
Administrative       30
Application          30
Configuration        30
Network              30
Other                30
Cluster              50
Commit               50

8 rows selected.

SQL>
SQL> exit;
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
 $
```

Solutions for Practice 4-1: Working with Metrics (continued)

3) Using a PL/SQL command or the `lab_04_01_03.sql` script, change the thresholds for the Concurrency class to 60% for the warning level and 90% for the critical level. Using both SQL and Enterprise Manager, verify that your change took effect.

Answer:

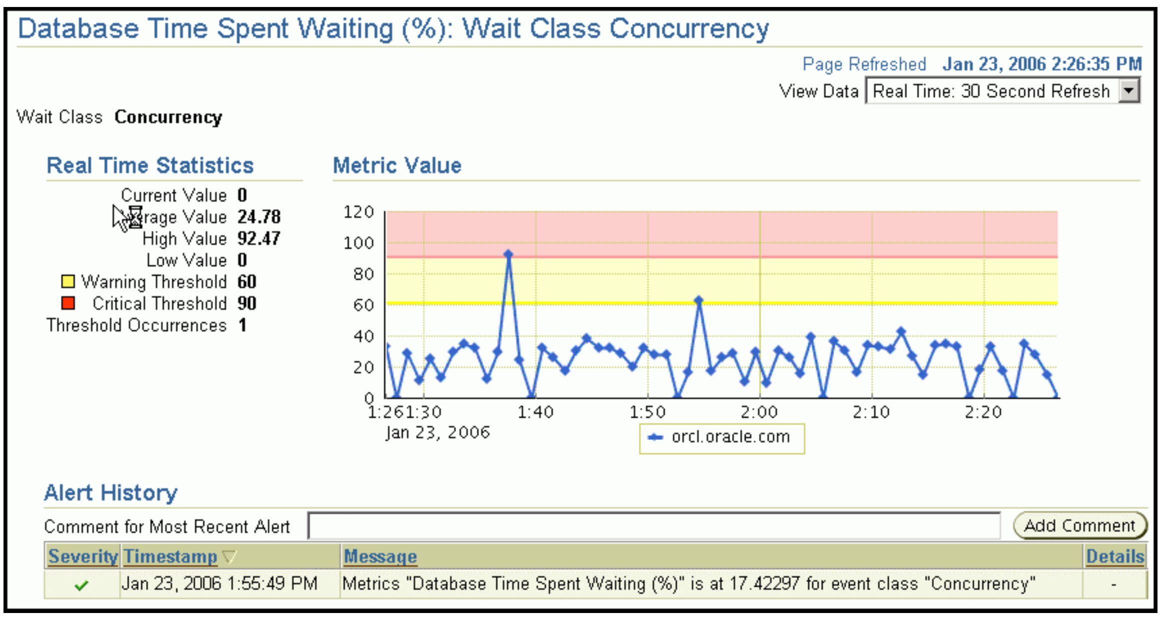
1. Use the `DBMS_SERVER_ALERT` package to set the new threshold values.

```
$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Mon Jan 23
12:31:37 2006

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> @lab_04_01_03.sql
SQL>
SQL> BEGIN
  2     DBMS_SERVER_ALERT.set_threshold(
  3     metrics_id => dbms_server_alert.DB_TIME_WAITING,
  4     warning_operator => dbms_server_alert.OPERATOR_GE,
  5     warning_value    => 60,
  6     critical_operator=> dbms_server_alert.OPERATOR_GE,
  7     critical_value   => 90,
  8     observation_period=> 1,
  9     consecutive_occurrences => 1,
 10     instance_name    => 'orcl',
 11     object_type      =>
 12          dbms_server_alert.OBJECT_TYPE_EVENT_CLASS,
 13     object_name      => 'Concurrency');
 14     END;
 15  /

PL/SQL procedure successfully completed.

SQL>SQL> col object_name format a20
col metrics_name format a25
col warning_value format a10
col critical_value format a10
SQL> SQL> SQL> SQL>
SQL> select object_name,warning_value,critical_value
  2    from dba_thresholds
  3    where metrics_name=
  4          'Database Time Spent Waiting (%)';

OBJECT_NAME          WARNING_VA CRITICAL_V
```

Solutions for Practice 4-1: Working with Metrics (continued)

```
-------------------- ---------- ----------
Concurrency          60         90
Administrative       30
Application          30
Configuration        30
Network              30
Other                30
Cluster              50
Commit               50

8 rows selected.

SQL> exit;
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
[oracle@edrsr14p1 labs]$
```

2. Still on the Wait Class Concurrency page, select Real Time: 30 Second Refresh from the View Data drop-down list.



4) Execute the `start_04_01_04.sh` script to start the workload for this practice. While the script is executing, observe the variation of the curve on the Wait Class Concurrency page. After the workload is finished, find the alert history from a SQL*Plus session. Check both the outstanding alerts and the alert history. The workload script takes up to 6 minutes to complete on some machines.

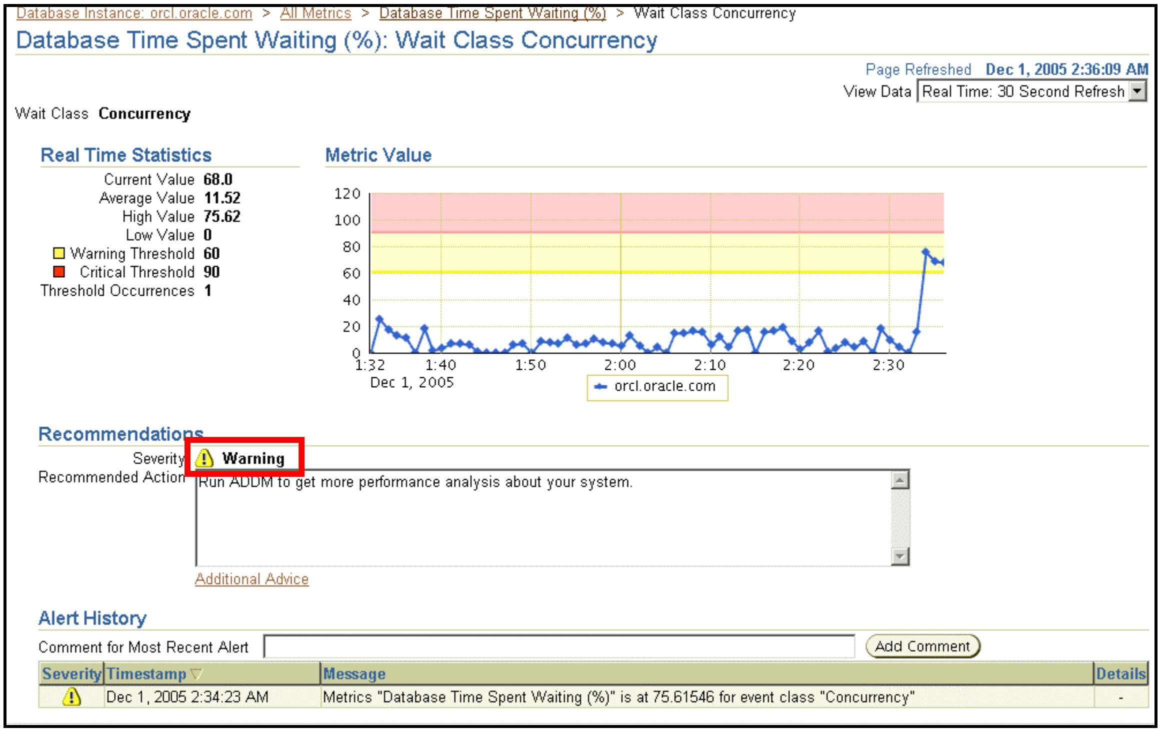
Solutions for Practice 4-1: Working with Metrics (continued)

Answer:

1. You can find the script in your `$HOME/labs` directory.

```
[oracle@edrsr14p1 labs]$ ./start_04_01_04.sh
[oracle@edrsr14p1 labs]$
… The following messages will appear for 1 to 6 minutes …
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

[oracle@edrsr14p1 labs]$
```

2. While the script executes, look at the evolution of the curve on the Wait Class Concurrency graph. After a while, you should see a warning or critical alert.



3. After the script finishes, you can retrieve the history of your alerts by using SQL. Depending on your timing, the alert may appear in the `DBA_OUTSTANDING_ALERTS` view or in the `DBA_ALERT_HISTORY` view. If the alert has not yet been cleared, it will show in outstanding alerts views. After it has been cleared, it will be visible in the alert history view. The following example shows the output when the alert has been cleared. (**Hint:** `sol_04_01_04.sql` in the `/home/oracle/solutions` directory)

```
$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Tue Jan 24
11:51:08 2006

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options
SQL>
SQL> col reason format a75
SQL>
SQL> SELECT reason FROM DBA_OUTSTANDING_ALERTS;

no rows selected
```

```
SQL>
SQL> SELECT TO_CHAR(begin_time,'hh24:mi:ss'),
  2     dbtime_in_wait,average_waiter_count
  3  FROM V$WAITCLASSMETRIC_HISTORY
  4  WHERE wait_class#=4
  5   AND  wait_class_id=3875070507
  6   AND  begin_time > SYSDATE-(10/1440);

TO_CHAR( DBTIME_IN_WAIT AVERAGE_WAITER_COUNT
-------- -------------- --------------------
15:23:38     58.5625786           2.12069855
15:22:39     91.5446177            33.513199
15:21:38     92.0310253           33.7787911
15:20:39            100           15.9836338
15:19:39     5.80381931           .000395055
15:18:36     5.24422643            .00039374
15:17:36     4.79691341           .000407325
15:16:36     9.19376056           .000393323

9 rows selected.

SQL>
SQL>
SQL> -- Cleared!
SQL>
SQL> SELECT reason,resolution
  2  FROM DBA_ALERT_HISTORY
  3  WHERE reason like
  4       '%Database Time Spent Waiting (%)%Concurrency%'
  5   AND TO_DATE(SUBSTR(TO_CHAR(creation_time),1,18)||
  6           SUBSTR(TO_CHAR(creation_time),26,3),
  7           'DD-MON-YY HH:MI:SS AM') > SYSDATE-(10/1440)
  8  ORDER BY creation_time DESC;

REASON
-----------------------------------------------------------
-----------------
RESOLUT
-------
Metrics "Database Time Spent Waiting (%)" is at 58.56258
for event class "Concurrency" cleared

SQL>
SQL> EXIT;
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
```

Solutions for Practice 4-1: Working with Metrics (continued)

5) Execute the `lab_04_01_05.sql` script to clean up your environment.

   Answer:

   You can find the script in your `$HOME/labs` directory.

```
$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Thu Dec 1
02:49:42 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> @lab_04_01_05.sql
SQL>
SQL> DROP USER jfv CASCADE;

User dropped.

SQL>
SQL> DROP TABLESPACE tbsjfv INCLUDING CONTENTS AND
DATAFILES;

Tablespace dropped.

SQL>
SQL> BEGIN
  2  DBMS_SERVER_ALERT.set_threshold(
  3     metrics_id => dbms_server_alert.DB_TIME_WAITING,
  4     warning_operator => dbms_server_alert.OPERATOR_GE,
  5     warning_value => 30,
  6     critical_operator => NULL,
  7     critical_value => NULL,
  8     observation_period => 1,
  9     consecutive_occurrences => 1,
 10     instance_name => 'orcl',
 11     object_type =>
 12         dbms_server_alert.OBJECT_TYPE_EVENT_CLASS,
 13     object_name => 'Concurrency');
 14  END;
 15  /

PL/SQL procedure successfully completed.

SQL>
SQL> EXIT;
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
$
```

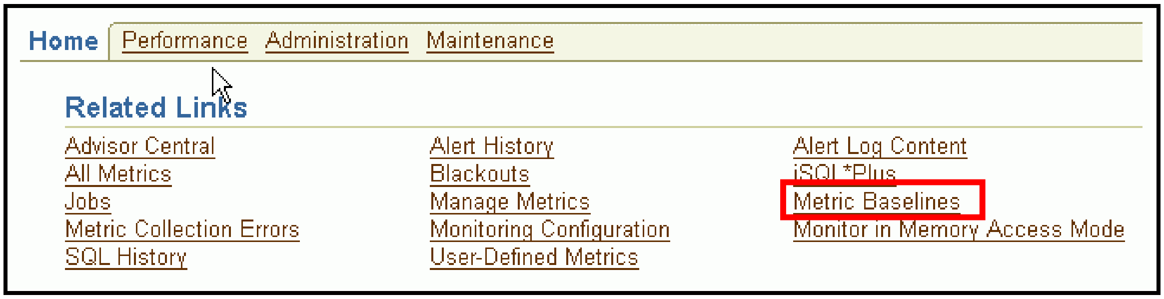
Solutions for Practice 4-2: *Working with Baselines*

The goal of this practice is to manipulate metric baselines. You are going to create a static metric baseline based on a past workload. Then, various workloads are executed to show how you are alerted.

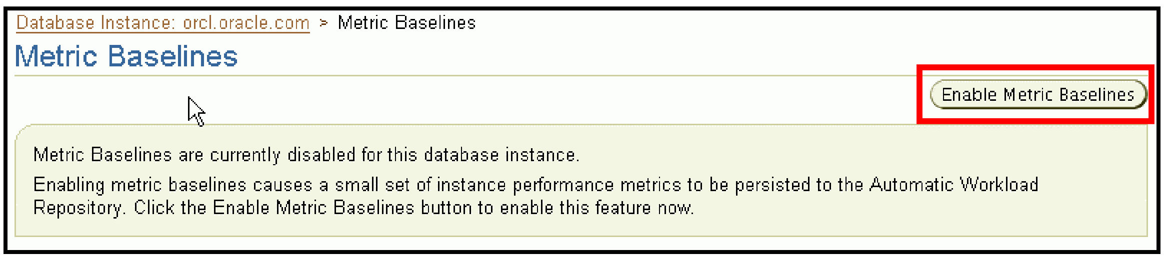
1) Connect as `SYSDBA` using Database Control and enable Metric Baselines. After this is done, log out from your Database Control Console.

   Answer:

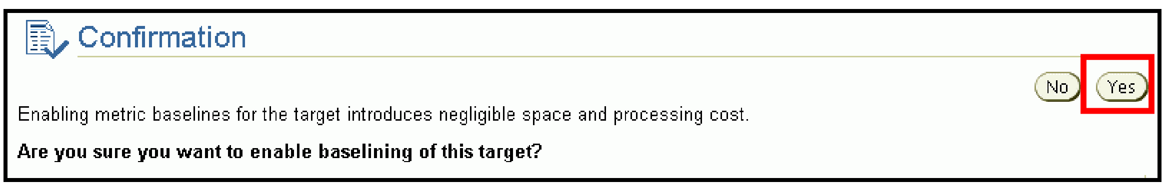
   1. In the Related Links section on the Database Control home page, click the Metric Baselines link.



   2. On the Metric Baselines page, click the Enable Metric Baselines button.



   3. On the Confirmation page, click Yes.



   4. Log out from your Database Control Console.

2) From your terminal emulator session, connected as the `oracle` user, execute the `lab_04_02_02.sh` script. This script takes as long as 15 minutes to run and loads many statistics that are needed for this practice.

   Answer:

   You can find the script in your `$HOME/labs` directory.

```
$ ./lab_04_02_02.sh
TZ set to US/Pacific
```

```
Oracle Enterprise Manager 10g Database Control Release
10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation.  All rights
reserved.
http://edrsr3p1.us.oracle.com:5500/em/console/aboutApplica
tion
Stopping Oracle Enterprise Manager 10g Database Control
...
 ...  Stopped.
Database closed.
Database dismounted.
ORACLE instance shut down.
TZ set to US/Pacific
Oracle Enterprise Manager 10g Database Control Release
10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation.  All rights
reserved.
http://edrsr3p1.us.oracle.com:1158/em/console/aboutApplica
tion
Stopping Oracle Enterprise Manager 10g Database Control
...
 ...  Stopped.
drop user jfv cascade
          *
ERROR at line 1:
ORA-01918: user 'JFV' does not exist



User created.


Grant succeeded.

drop table work purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist



Table created.


System altered.

ORACLE instance shut down.

ORACLE instance started.

Total System Global Area  629145600 bytes
Fixed Size                  1220988 bytes
```

```
Variable Size              306187908 bytes
Database Buffers           318767104 bytes
Redo Buffers                 2969600 bytes
Database mounted.
Database opened.

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

~~~~~~~~~~
AWR LOAD
~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~  This script will load the AWR data from a dump file. ~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|  Loading the AWR data from the following
|  directory/file:
|   /u01/app/oracle/product/10.2.0/db_1/rdbms/log/
|   awrdat_1_141.dmp
| ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|
|  *** AWR Load Started ...
|
|  This operation will take a few moments. The
|  progress of the AWR load operation can be
|  monitored in the following directory/file:
|   /u01/app/oracle/product/10.2.0/db_1/rdbms/log/
|   awrdat_1_141.log
|
Register the DBID: 1080838556
Append Data for JFV.WRM$_DATABASE_INSTANCE.
Append Data for JFV.WRM$_SNAPSHOT.
Append Data for JFV.WRH$_STAT_NAME.
Append Data for JFV.WRH$_PARAMETER_NAME.
Append Data for JFV.WRH$_EVENT_NAME.
Append Data for JFV.WRH$_LATCH_NAME.
Append Data for JFV.WRH$_FILESTATXS.
Append Data for JFV.WRH$_FILESTATXS_BL.
Append Data for JFV.WRH$_TEMPSTATXS.
Append Data for JFV.WRH$_DATAFILE.
Append Data for JFV.WRH$_TEMPFILE.
Append Data for JFV.WRH$_SQLSTAT.
Failed append data for JFV.WRH$_SQLSTAT
ORA-00904: "FLAG": invalid identifier
Append Data for JFV.WRH$_SQLSTAT_BL.
Failed append data for JFV.WRH$_SQLSTAT
ORA-00904: "FLAG": invalid identifier
Failed Insert Snap Error: WRH$_SQLSTAT
```

```
ORA-00001: unique constraint (SYS.WRM$_SNAP_ERROR_PK)
violated
ORA-00904: "FLAG": invalid identifier
Append Data for JFV.WRH$_SQLTEXT.
Append Data for JFV.WRH$_SQL_SUMMARY.
Append Data for JFV.WRH$_SQL_PLAN.
Append Data for JFV.WRH$_SYSTEM_EVENT.
Append Data for JFV.WRH$_SYSTEM_EVENT_BL.
Append Data for JFV.WRH$_BG_EVENT_SUMMARY.
Append Data for JFV.WRH$_WAITSTAT.
Append Data for JFV.WRH$_WAITSTAT_BL.
Append Data for JFV.WRH$_ENQUEUE_STAT.
Append Data for JFV.WRH$_LATCH.
Append Data for JFV.WRH$_LATCH_BL.
Append Data for JFV.WRH$_LATCH_CHILDREN.
Append Data for JFV.WRH$_LATCH_CHILDREN_BL.
Append Data for JFV.WRH$_LATCH_PARENT.
Append Data for JFV.WRH$_LATCH_PARENT_BL.
Append Data for JFV.WRH$_LATCH_MISSES_SUMMARY.
Append Data for JFV.WRH$_LATCH_MISSES_SUMMARY_BL.
Append Data for JFV.WRH$_LIBRARYCACHE.
Append Data for JFV.WRH$_DB_CACHE_ADVICE.
Append Data for JFV.WRH$_DB_CACHE_ADVICE_BL.
Append Data for JFV.WRH$_BUFFER_POOL_STATISTICS.
Append Data for JFV.WRH$_ROWCACHE_SUMMARY.
Append Data for JFV.WRH$_ROWCACHE_SUMMARY_BL.
Append Data for JFV.WRH$_SGA.
Append Data for JFV.WRH$_SGASTAT.
Append Data for JFV.WRH$_SGASTAT_BL.
Append Data for JFV.WRH$_PGASTAT.
Append Data for JFV.WRH$_PROCESS_MEMORY_SUMMARY.
Append Data for JFV.WRH$_RESOURCE_LIMIT.
Append Data for JFV.WRH$_SHARED_POOL_ADVICE.
Append Data for JFV.WRH$_SQL_WORKAREA_HISTOGRAM.
Append Data for JFV.WRH$_PGA_TARGET_ADVICE.
Append Data for JFV.WRH$_INSTANCE_RECOVERY.
Append Data for JFV.WRH$_SYSSTAT.
Append Data for JFV.WRH$_SYSSTAT_BL.
Append Data for JFV.WRH$_PARAMETER.
Append Data for JFV.WRH$_PARAMETER_BL.
Append Data for JFV.WRH$_UNDOSTAT.
Failed append data for JFV.WRH$_UNDOSTAT
ORA-00904: "RUNAWAYQUERYSQLID": invalid identifier
Append Data for JFV.WRH$_SEG_STAT.
Append Data for JFV.WRH$_SEG_STAT_BL.
Append Data for JFV.WRH$_SEG_STAT_OBJ.
Append Data for JFV.WRH$_DLM_MISC.
Append Data for JFV.WRH$_DLM_MISC_BL.
Append Data for JFV.WRH$_SERVICE_NAME.
Append Data for JFV.WRH$_SERVICE_STAT.
Append Data for JFV.WRH$_SERVICE_STAT_BL.
Append Data for JFV.WRH$_SYSMETRIC_HISTORY.
```

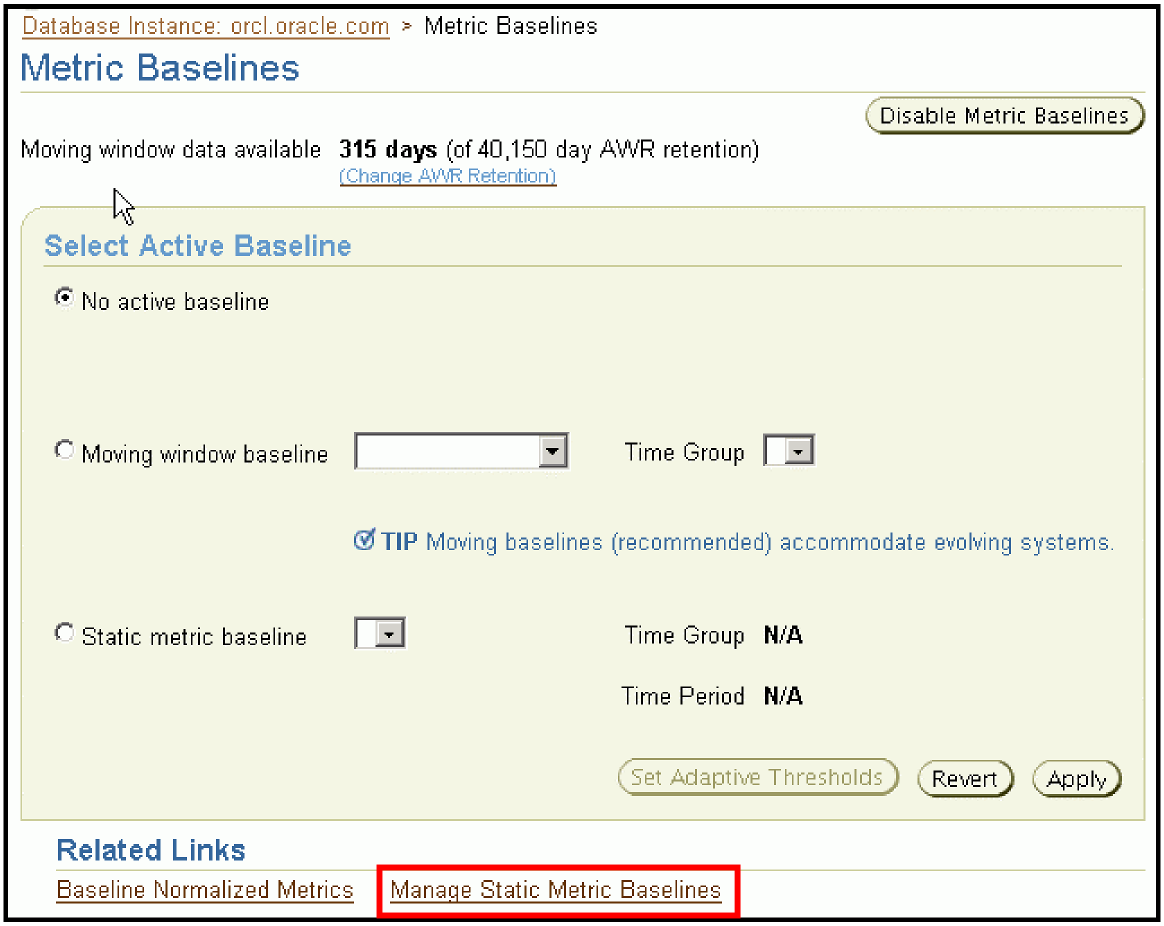
```
Append Data for JFV.WRH$_FILEMETRIC_HISTORY.
Append Data for JFV.WRH$_WAITCLASSMETRIC_HISTORY.
Append Data for JFV.WRH$_TABLESPACE_STAT.
Append Data for JFV.WRH$_TABLESPACE_STAT_BL.
Append Data for JFV.WRH$_LOG.
Append Data for JFV.WRH$_MTTR_TARGET_ADVICE.
Append Data for JFV.WRH$_TABLESPACE_SPACE_USAGE.
Append Data for JFV.WRH$_METRIC_NAME.
Append Data for JFV.WRH$_SYSMETRIC_SUMMARY.
Append Data for JFV.WRH$_SQL_BIND_METADATA.
Append Data for JFV.WRH$_JAVA_POOL_ADVICE.
Append Data for JFV.WRH$_THREAD.
Append Data for JFV.WRH$_SESSMETRIC_HISTORY.
Append Data for JFV.WRH$_OSSTAT.
Append Data for JFV.WRH$_OSSTAT_BL.
Append Data for JFV.WRH$_OSSTAT_NAME.
Append Data for JFV.WRH$_SYS_TIME_MODEL.
Append Data for JFV.WRH$_SYS_TIME_MODEL_BL.
Append Data for JFV.WRH$_OPTIMIZER_ENV.
Append Data for JFV.WRH$_SERVICE_WAIT_CLASS.
Append Data for JFV.WRH$_SERVICE_WAIT_CLASS_BL.
Append Data for JFV.WRH$_CR_BLOCK_SERVER.
Append Data for JFV.WRH$_CURRENT_BLOCK_SERVER.
Append Data for JFV.WRH$_INST_CACHE_TRANSFER.
Append Data for JFV.WRH$_INST_CACHE_TRANSFER_BL.
Append Data for JFV.WRH$_STREAMS_POOL_ADVICE.
Append Data for JFV.WRH$_COMP_IOSTAT.
Append Data for JFV.WRH$_SGA_TARGET_ADVICE.
Append Data for JFV.WRH$_SESS_TIME_STATS.
Append Data for JFV.WRH$_STREAMS_CAPTURE.
Append Data for JFV.WRH$_STREAMS_APPLY_SUM.
Append Data for JFV.WRH$_BUFFERED_QUEUES.
Failed append data for JFV.WRH$_BUFFERED_QUEUES
ORA-00904: "QUEUE_ID": invalid identifier
Append Data for JFV.WRH$_BUFFERED_SUBSCRIBERS.
Append Data for JFV.WRH$_RULE_SET.
Append Data for JFV.WRH$_ACTIVE_SESSION_HISTORY.
Failed append data for JFV.WRH$_ACTIVE_SESSION_HISTORY
ORA-00904: "BLOCKING_SESSION_SERIAL#": invalid identifier
Append Data for JFV.WRH$_ACTIVE_SESSION_HISTORY_BL.
Failed append data for JFV.WRH$_ACTIVE_SESSION_HISTORY
ORA-00904: "BLOCKING_SESSION_SERIAL#": invalid identifier
Failed Insert Snap Error: WRH$_ACTIVE_SESSION_HISTORY
ORA-00001: unique constraint (SYS.WRM$_SNAP_ERROR_PK)
violated
ORA-00904: "BLOCKING_SESSION_SERIAL#": invalid
identifier
Append Data for JFV.WRM$_SNAP_ERROR.
Finished MOVE_TO_AWR procedure
TZ set to US/Pacific
Oracle Enterprise Manager 10g Database Control Release
10.2.0.1.0
```

```
Copyright (c) 1996, 2005 Oracle Corporation.  All rights
reserved.
http://edrsr3p1.us.oracle.com:1158/em/console/aboutApplica
tion
Starting Oracle Enterprise Manager 10g Database Control
.......... started.
----------------------------------------------------------
Logs are generated in directory
/u01/app/oracle/product/10.2.0/db_1/edrsr3p1.us.oracle.com
_orcl/sysman/log
 $
```

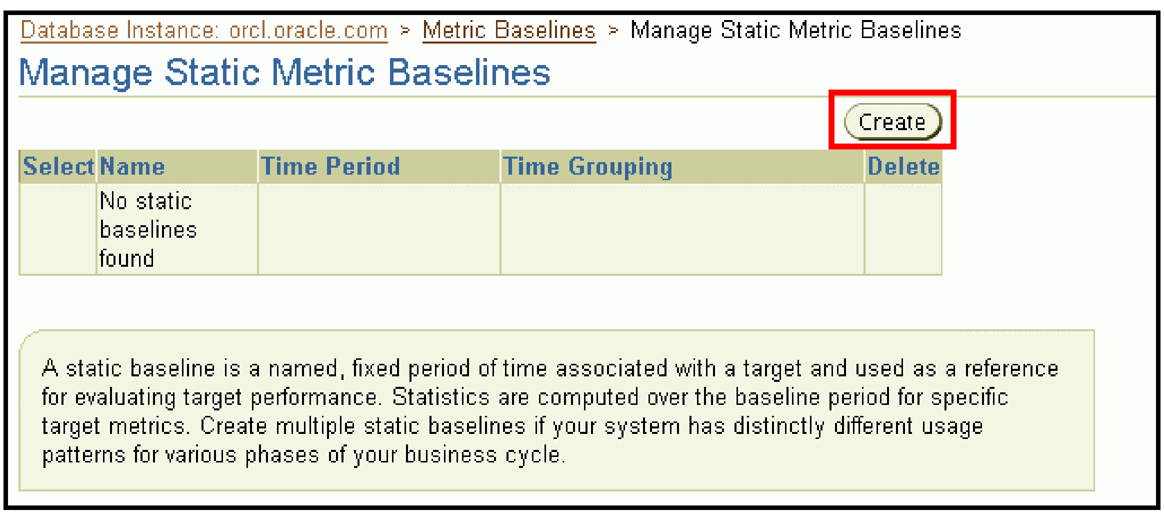
3) Using Database Control, create a new static metric baseline called MB1. Make sure that you retrieve the statistics from Automatic Workload Repository (AWR) between March 12, 2005, and March 21, 2005. Also, make sure that your baseline is using only the By Hour of Day Time Grouping. After the baseline is created, retrieve and determine the statistics cycle for the Number of Transactions (per second) metric. Then, set adaptive thresholds for your baseline as follows: Select “Preserve prior threshold” and then set Occurrences to 1. Select High (.95) in the Warning Level field and Very High (.99) in the Critical Level field.

Answer:

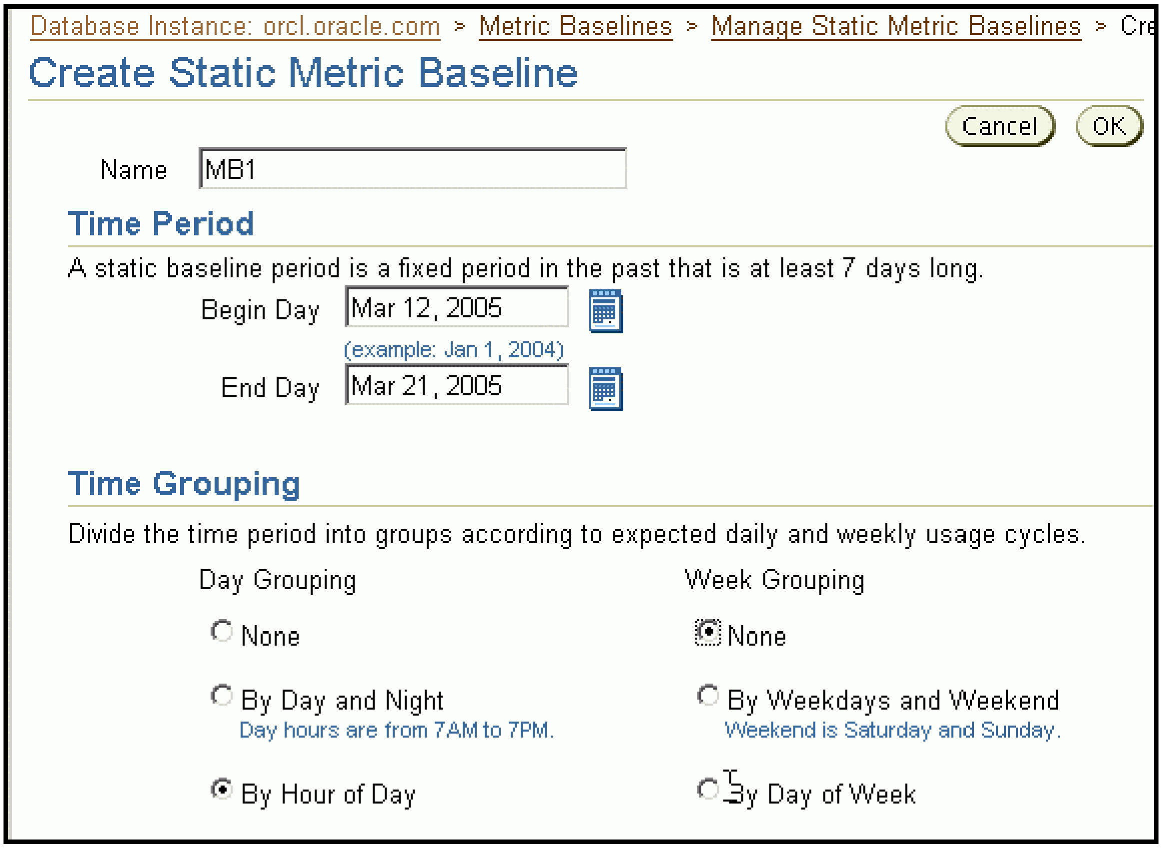
1. Log in to EM Database Control.
2. On the Database Control home page, click the Metric Baselines link.
3. On the Metric Baselines page, click Manage Static Metric Baselines.



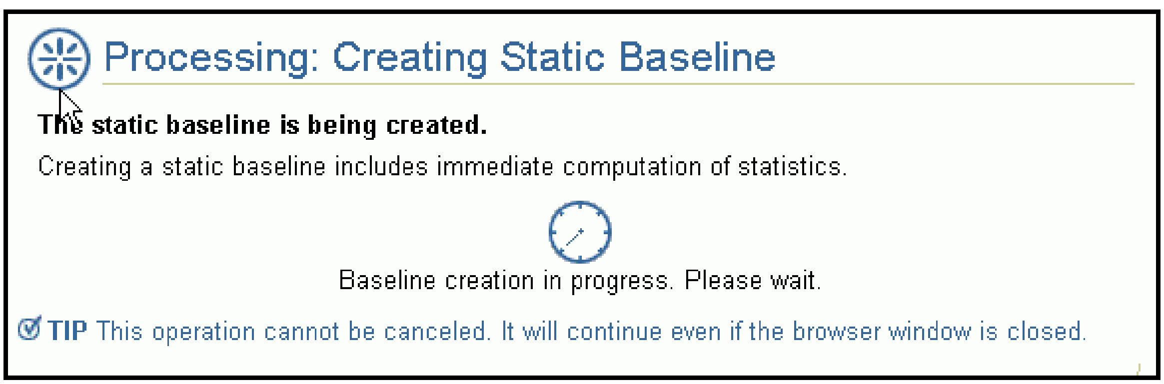
4. On the Manage Static Metric Baselines page, click Create.



5. On the Create Static Metric Baseline page, set Name to MB1, Begin Day to Mar 12, 2005, and End Day to Mar 21, 2005. In the Time Grouping section of the page, select By Hour of the Day, and None in the Week Grouping area. After this is done, click OK.

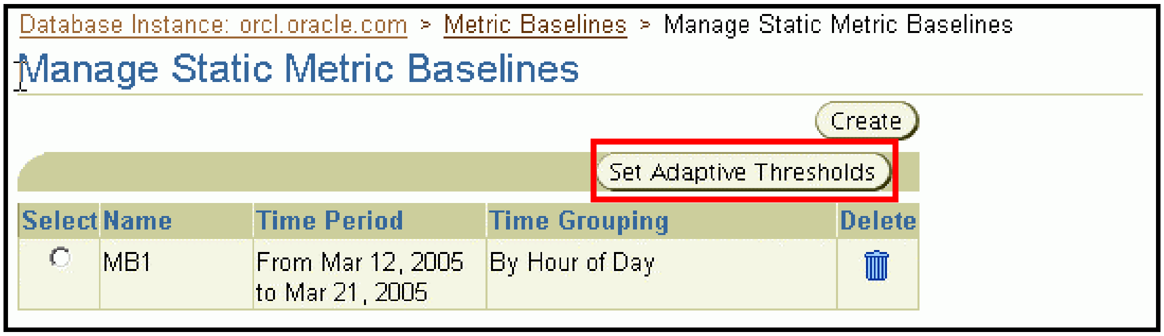


6. This opens the Processing: Creating Static Baseline page, and then the Manage Static Metric Baselines page.

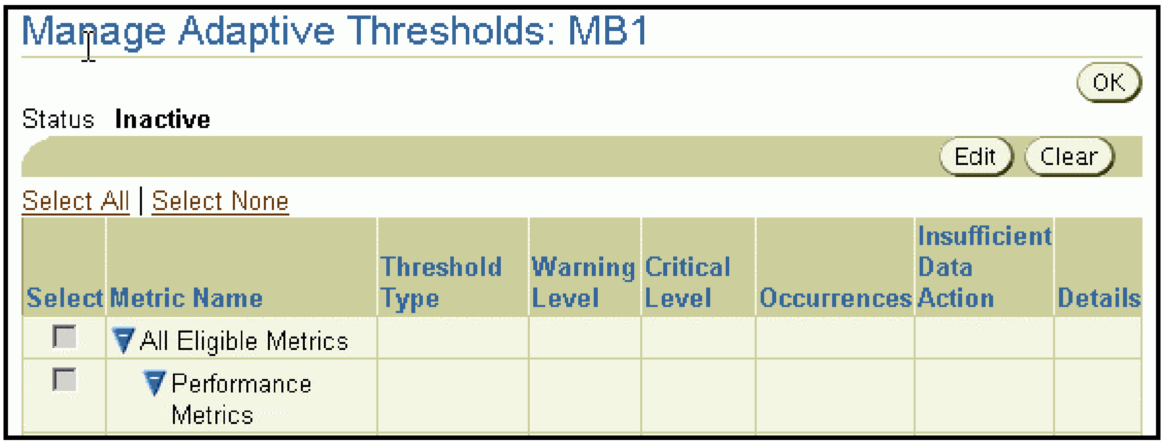


7. On the Manage Static Metric Baselines page, click the Set Adaptive Thresholds button.

Solutions for Practice 4-2: Working with Baselines (continued)

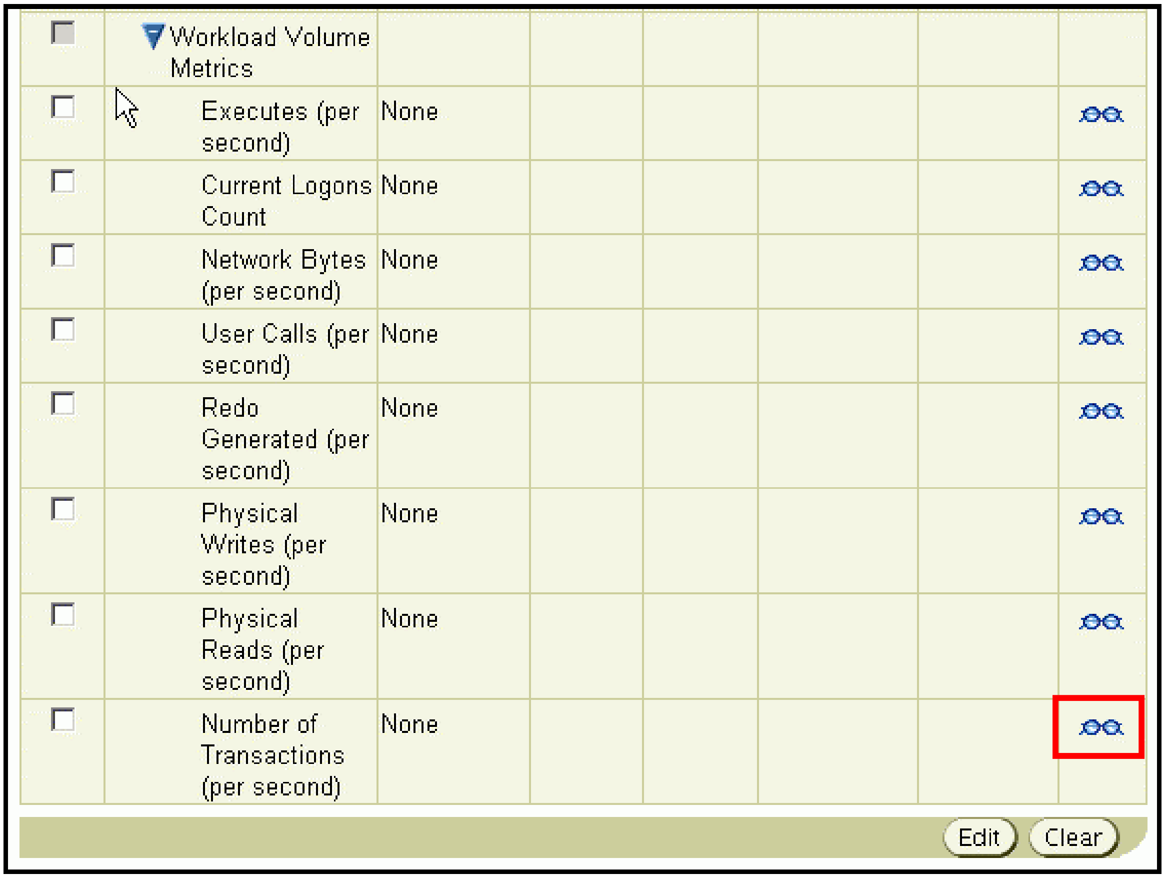


8. On the Manage Adaptive Thresholds: MB1 page, expand the Workload Volume Metrics category.



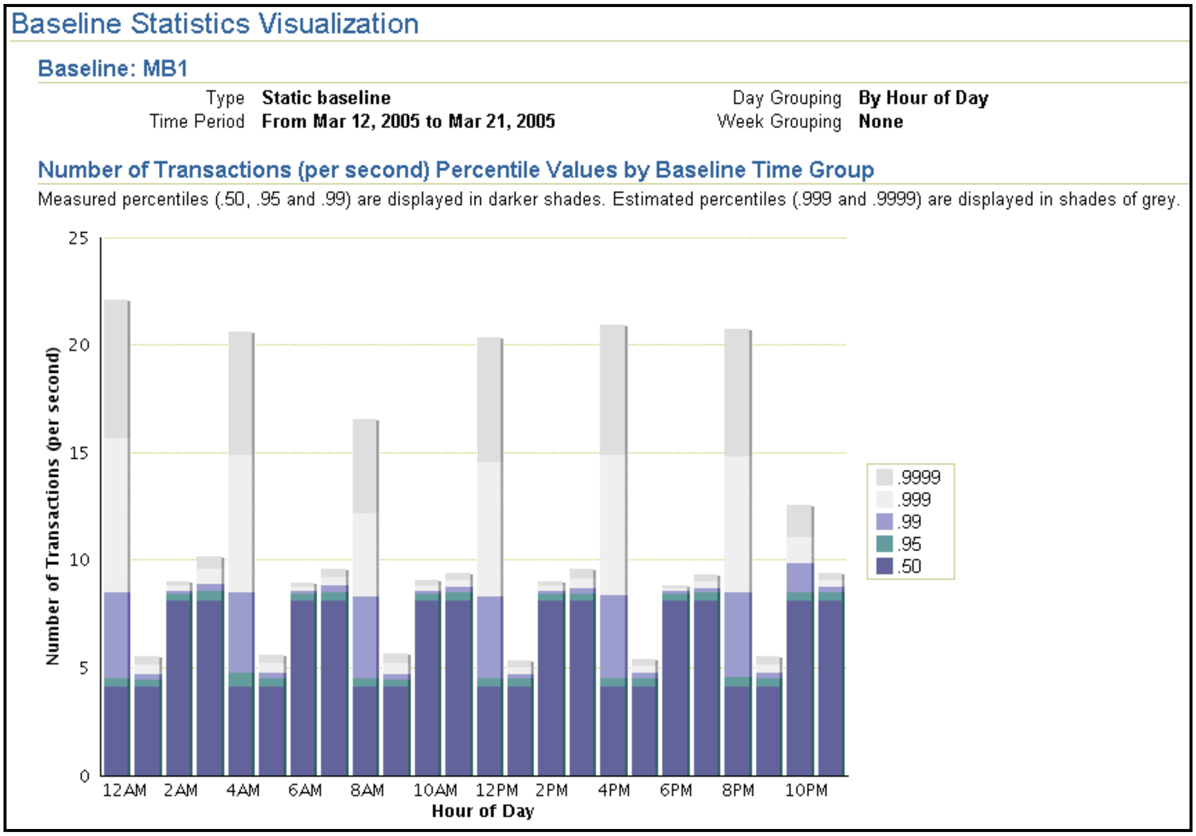
9. Still on the Manage Adaptive Thresholds: MB1 page, select the Number of Transactions (per second) check box, and click the corresponding eyeglasses icon on the right of the page.

Solutions for Practice 4-2: Working with Baselines (continued)



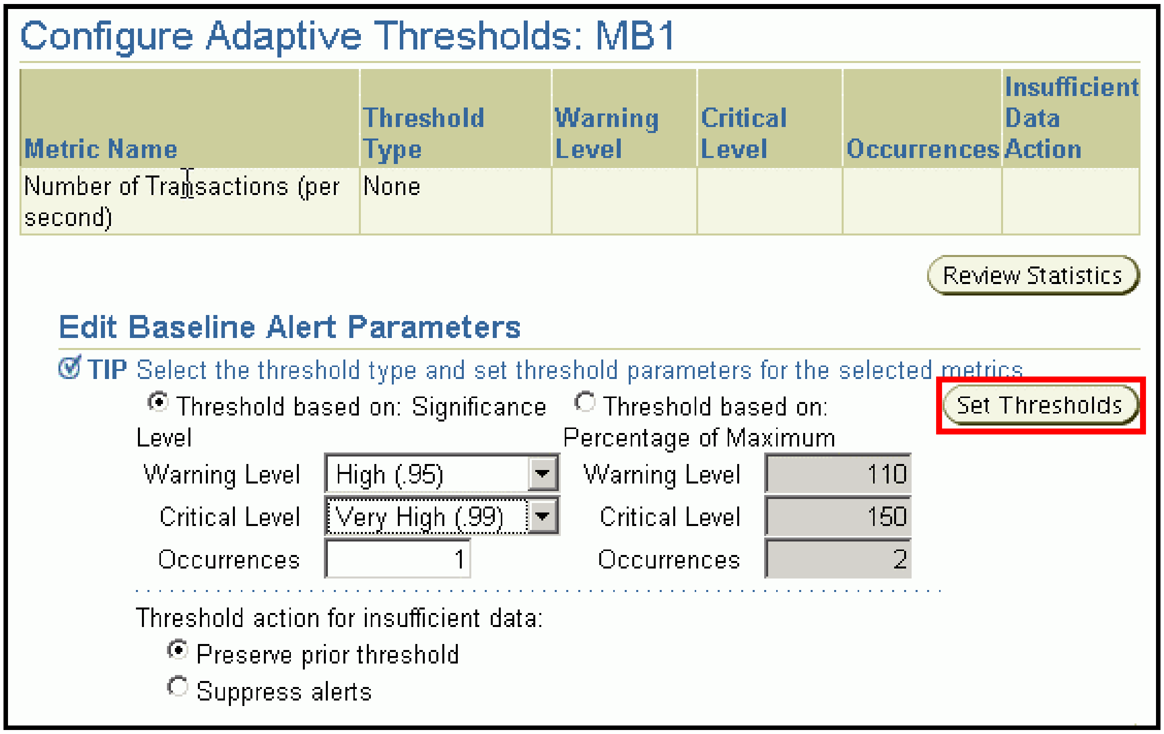
10. On the Baseline Statistics Visualization page, you can view the computed statistics for this metric for that period of time. You can clearly see that you have cycles every two hours. For the first two hours, the number of transactions per second (tps) is around 5; in the next two hours, it goes up to 10, and this continues the whole day. **Note:** This is the rate at the current time. Example: It is 1:15 p.m. now. The rate at .95% is about 4 tps.

Solutions for Practice 4-2: Working with Baselines (continued)

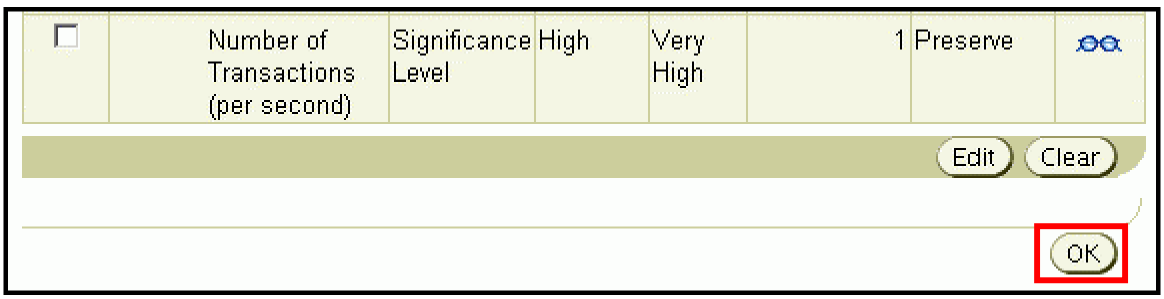


11. Close the browser window corresponding to the Baseline Statistics Visualization page.
12. Back on the Manage Adaptive Thresholds: MB1 page, click the Edit button.
13. On the Configure Adaptive Thresholds: MB1 page, select “Preserve prior threshold,” and then set Occurrences to 1. Then, select High (.95) in the Warning Level field, and Very High (.99) in the Critical Level field. After this is done, click the Set Thresholds button.

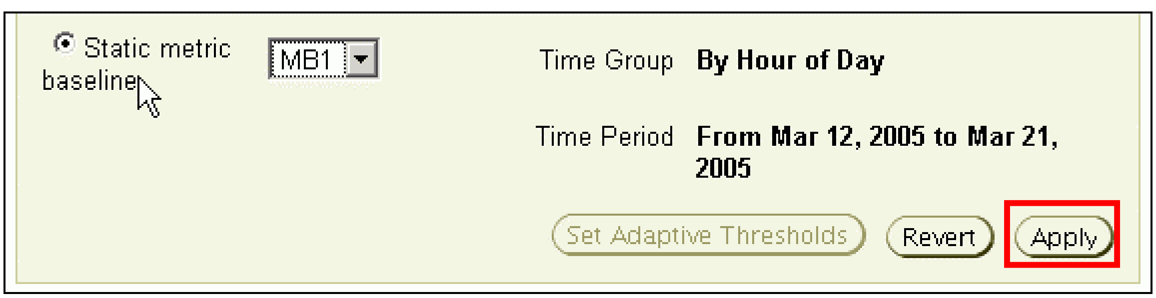
Solutions for Practice 4-2: Working with Baselines (continued)



14. Back on the Manage Adaptive Thresholds: MB1 page, verify that your baseline was successfully updated, and click the OK button.

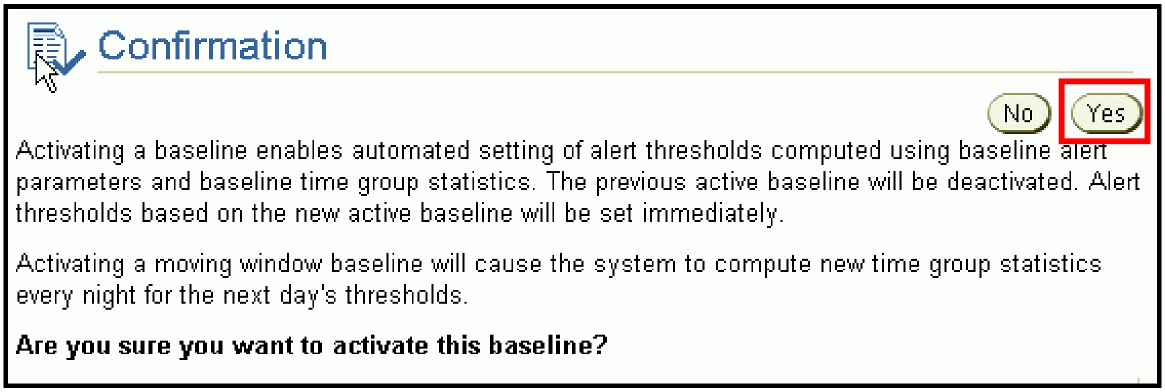


15. Back on the Metric Baselines page, make sure you click the "Static metric baseline" option button, and that MB1 is selected from the corresponding drop-down list. Then, click the Apply button.



16. On the Confirmation page, click Yes to enable your metric baseline.

Solutions for Practice 4-2: Working with Baselines (continued)



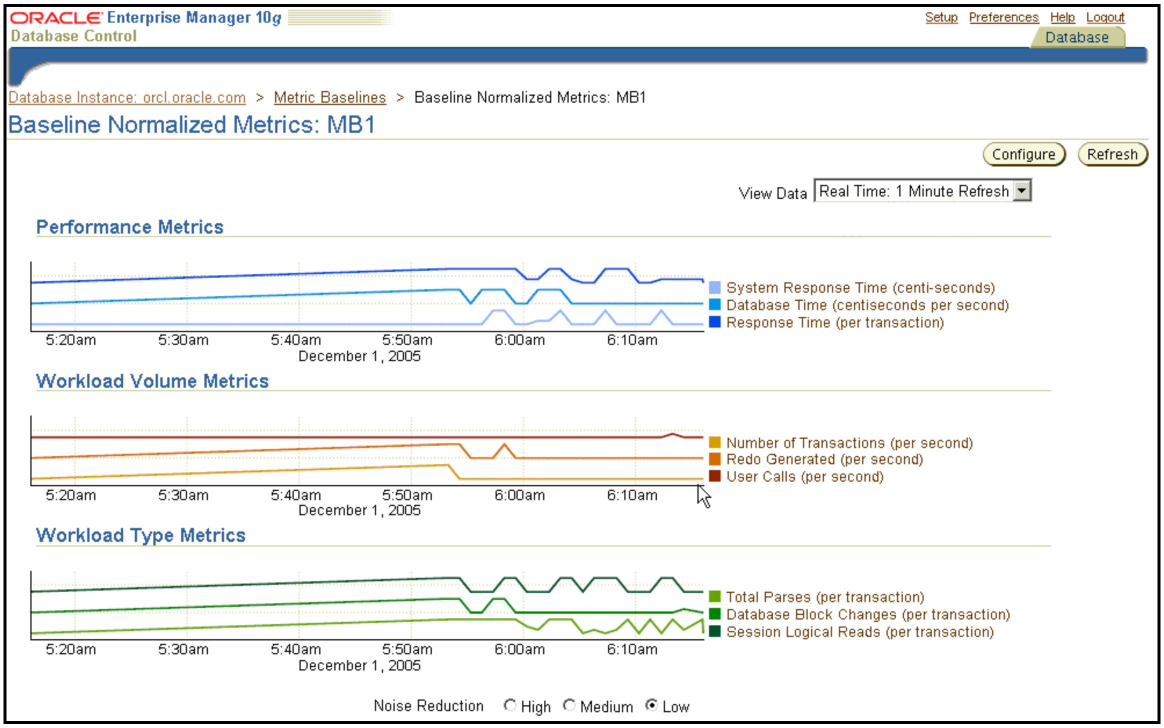
4) Look at the Baseline Normalized Metrics page, and then generate a workload of 40 transactions per second (tps). Execute the `lab_04_02_04.sh` script with a parameter of 40. Example: `./lab_04_02_04.sh 40`. Verify that the script is running with the `ps` command. You will see a process named `wkld_sess.sh`. Observe the Number of Transactions (per second) metric on the Baseline Normalized Metrics page. After a while, stop the execution of the script by deleting the `/home/oracle/workshops/runload` file. What do you observe, and why?

Answer:

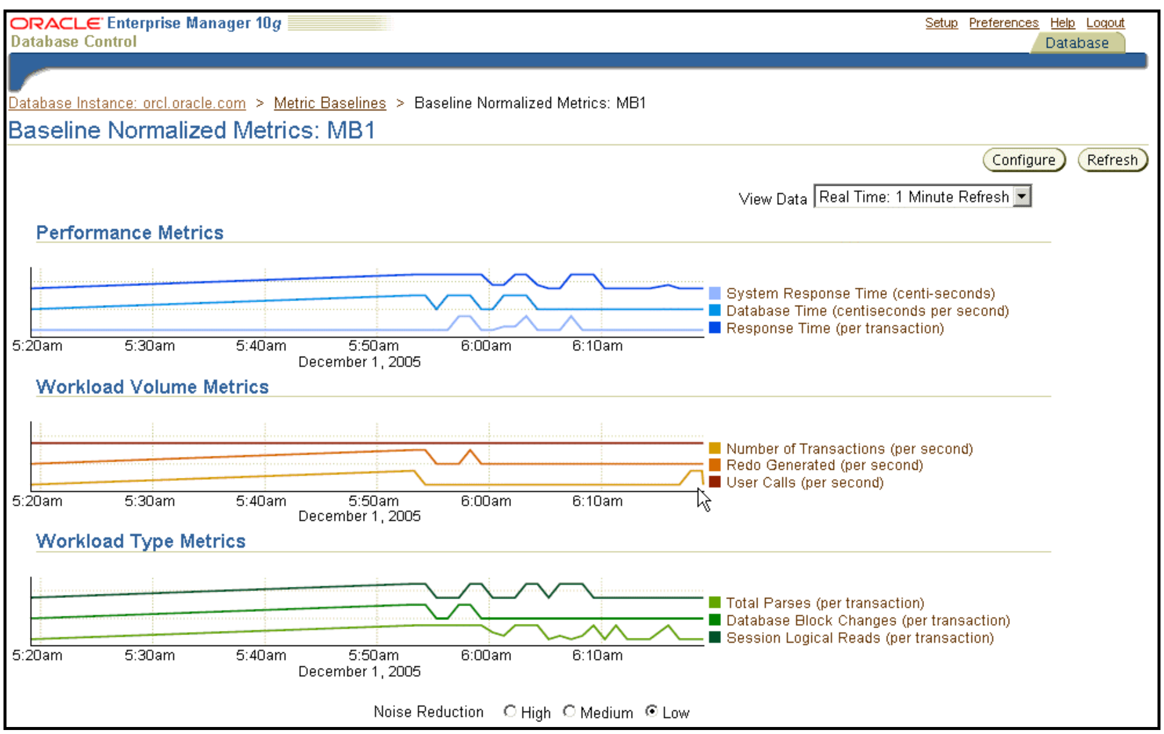
1. Because the script generates a transaction rate of 40 transactions per second, you should observe a deviation on the Baseline Normalized Metrics page for the Number of Transactions (per second) metric. As soon as you stop the script's execution, you should no longer observe any deviation on the graph.

```
$ ./lab_04_02_04.sh 40
 $ ps
  PID TTY          TIME CMD
32080 pts/0    00:00:00 bash
 3900 pts/0    00:00:00 wkld_sess.sh
 3908 pts/0    00:00:00 sleep
 3909 pts/0    00:00:00 ps
```

2. On the Database Control home page, click the Metric Baselines link.
3. On the Metric Baselines page, click the Baseline Normalized Metrics link. **Note:** If the Adobe SVG Viewer's end-user license agreement screen appears, then press [A] on the keyboard to proceed.

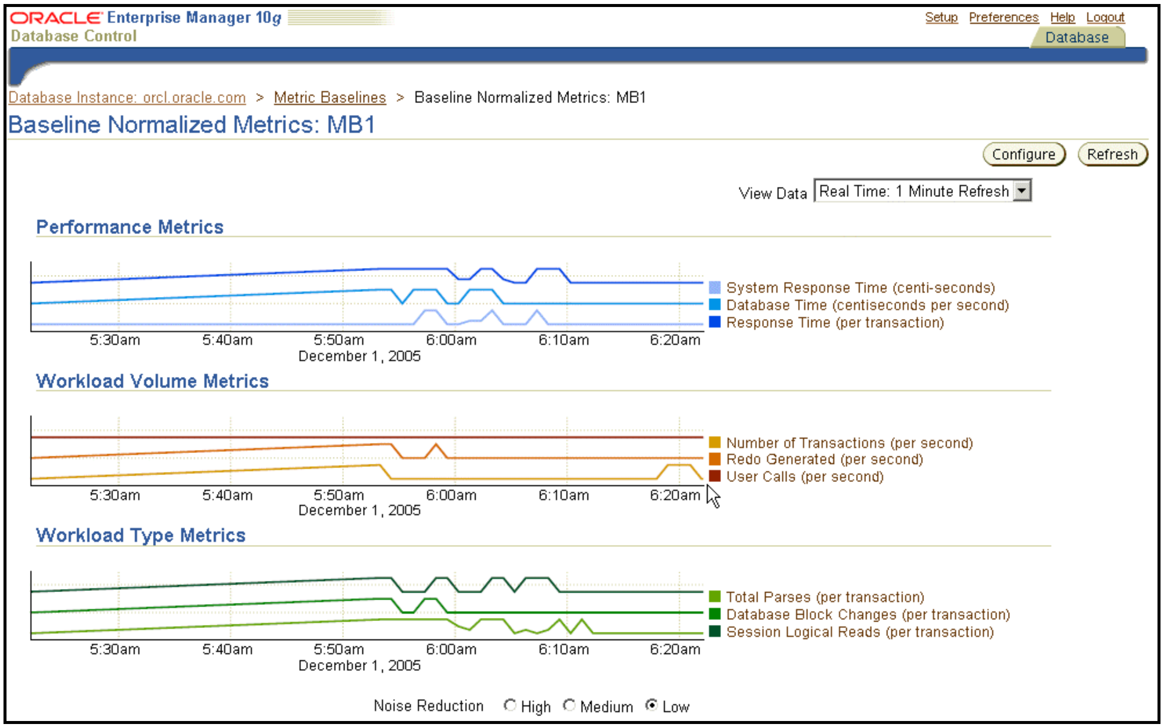


4. After a while, you should see a spike.



5. After you stop the execution of the script, the spike should go away.

```
$ rm /home/oracle/workshops/runload
rm: remove regular empty file
`/home/oracle/workshops/runload'? y
```



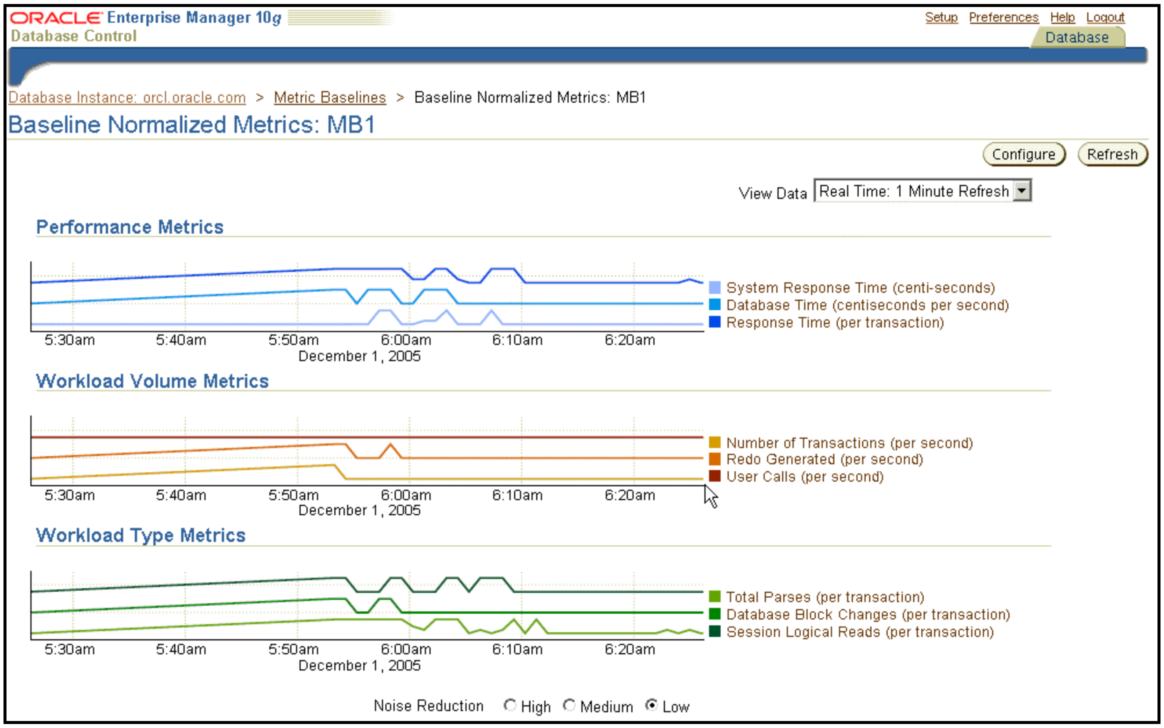
5) Create a workload that matches the baseline workload. Depending on the time of the day (the hour) when you do this practice, generate a transaction rate of 8 transactions per second (tps), or generate a transaction rate of 4 tps. Use the `lab_04_02_04.sh` script with a parameter to set the transaction rate. Choose the tps rate to execute on the basis of the cycle you determined in step 3 on the Baseline Statistics Visualization page. Match the hour on the visualization graph to the current time. After you make your observations, end the workload by deleting the `runload` file with `rm /home/oracle/workshops/runload`. What do you observe, and why?

Answer:

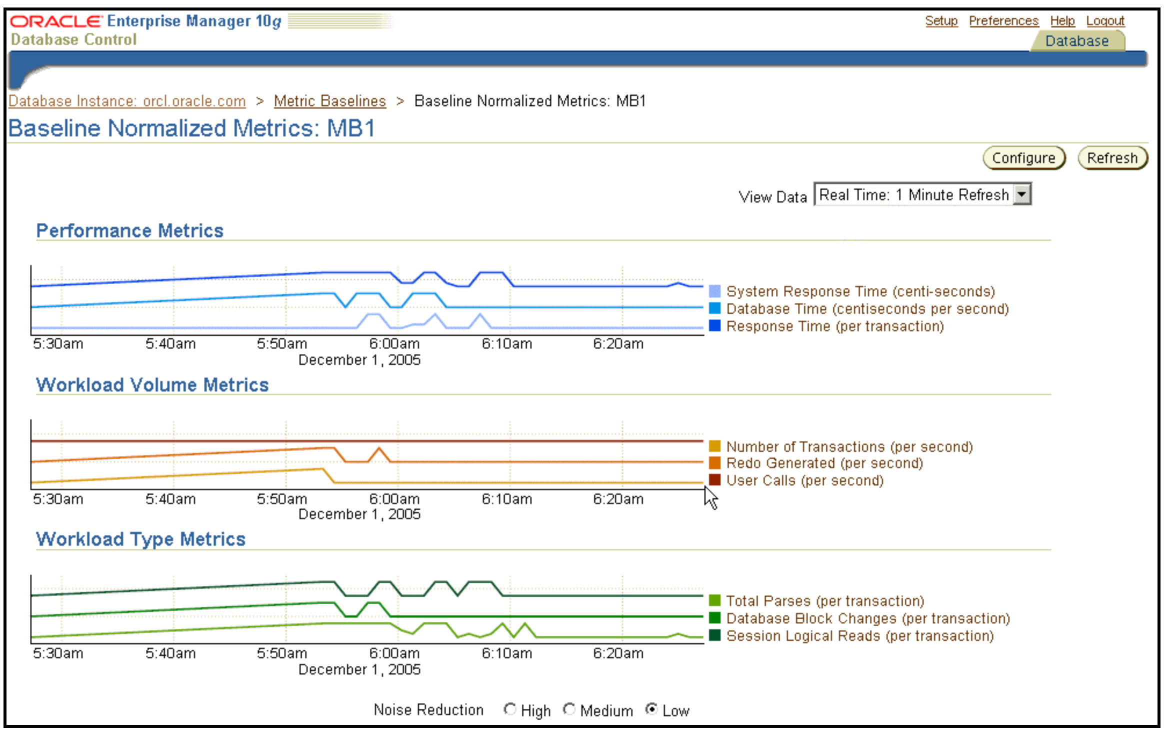
1. If you choose the correct script, you should not see any deviation on the Baseline Normalized Metrics page for the Number of Transactions (per second) metric. For the 6:00 a.m. time period, the visualization graph shows a level of 8 tps; therefore, run `lab_04_02_04.sh` with a parameter of 8.

```
$ ./lab_04_02_04.sh 8
```

2. On the Database Control home page, click the Metric Baselines link.
3. On the Metric Baselines page, click the Baseline Normalized Metrics link.

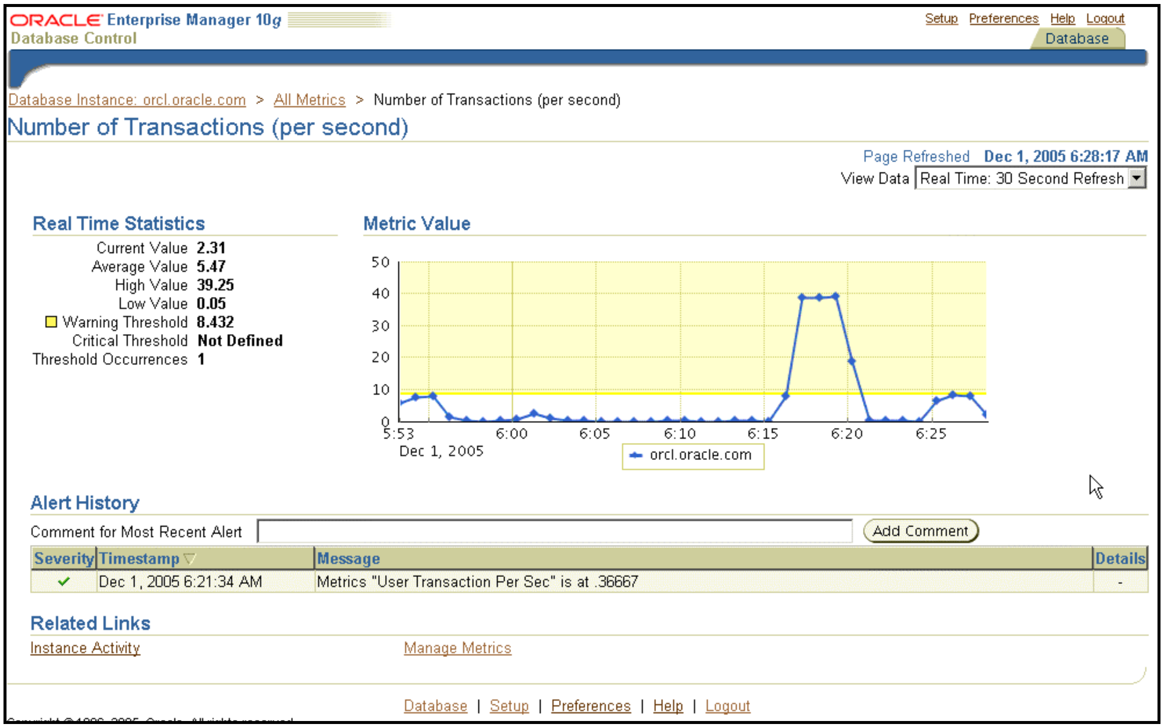


4. After a while, there is still no spike.



5. You can verify the transaction rate by looking at the Number of Transactions (per second) page, which is accessible from the All Metrics page.

Solutions for Practice 4-2: Working with Baselines (continued)



6. Remove the `runload` file with `rm /home/oracle/workshops/runload`.

```
$ rm /home/oracle/workshops/runload
rm: remove regular empty file
`/home/oracle/workshops/runload'? y
```

6) Create a workload that exceeds the baseline workload. Depending on the time of the day (the hour) when you do this practice, generate a transaction rate of 14 transactions per second (tps), or generate a transaction rate of 9 tps. Use the `lab_04_02_04.sh` script with a parameter to set the transaction rate. Choose the tps rate to execute on the basis of the cycle you determined in step 3 on the Baseline Statistics Visualization page. Match the hour on the visualization graph to the current time. After you make your observations, end the workload by deleting the `runload` file with `rm /home/oracle/workshops/runload`. What do you observe, and why?

Answer:

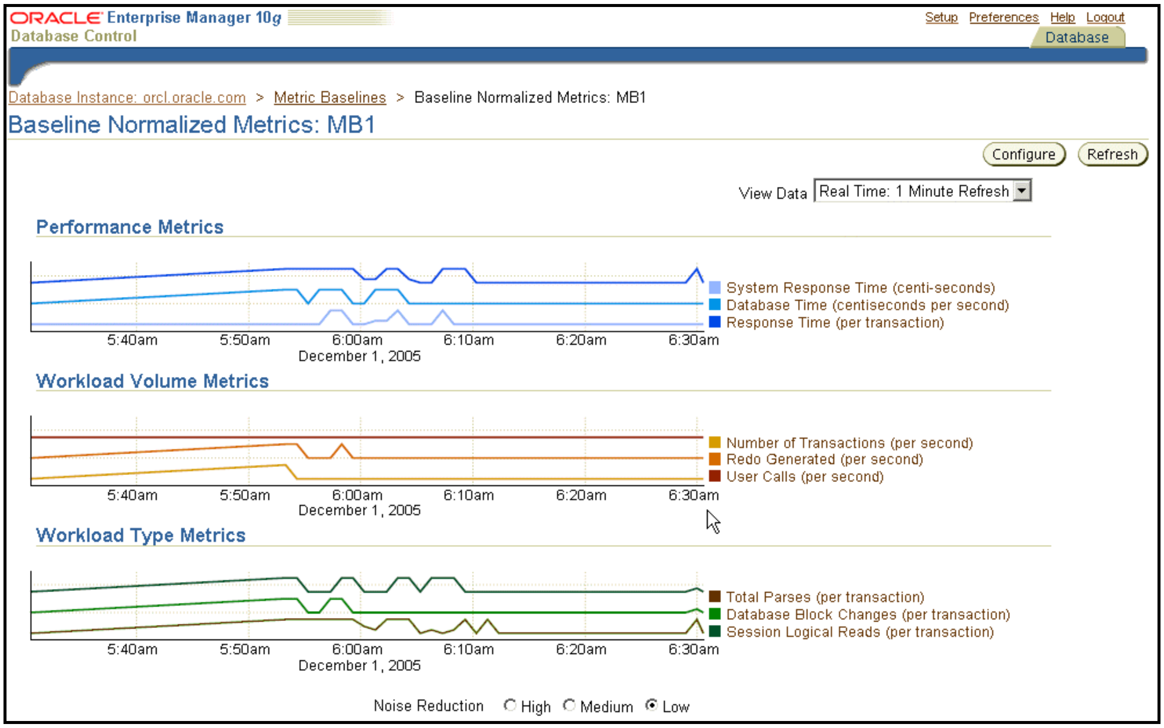
1. If you choose the correct parameter, you should see a deviation on the Baseline Normalized Metrics page for the Number of Transactions (per second) metric. For the 6:00 a.m. time period, the visualization graph shows a level of 8 tps; therefore, run `lab_04_02_04.sh` with a parameter of 14.

```
$ ./lab_04_02_04.sh 14
```

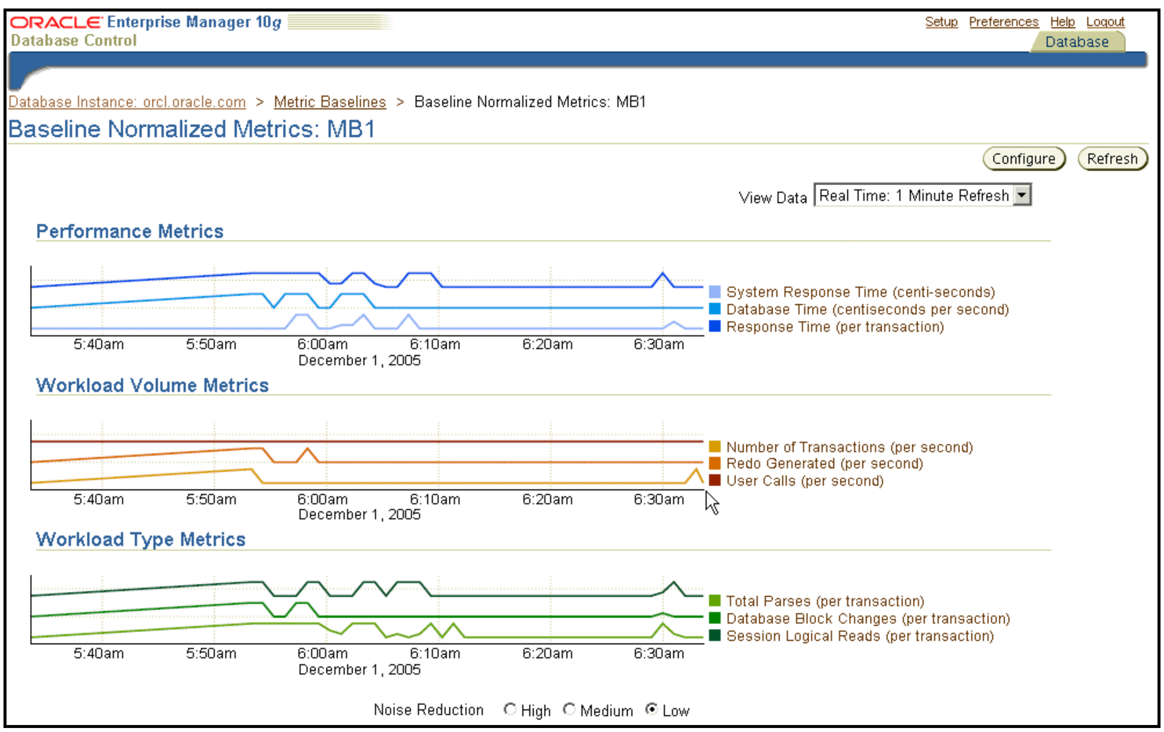
2. On the Database Control home page, click the Metric Baselines link.

Solutions for Practice 4-2: Working with Baselines (continued)

3. On the Metric Baselines page, click the Baseline Normalized Metrics link.

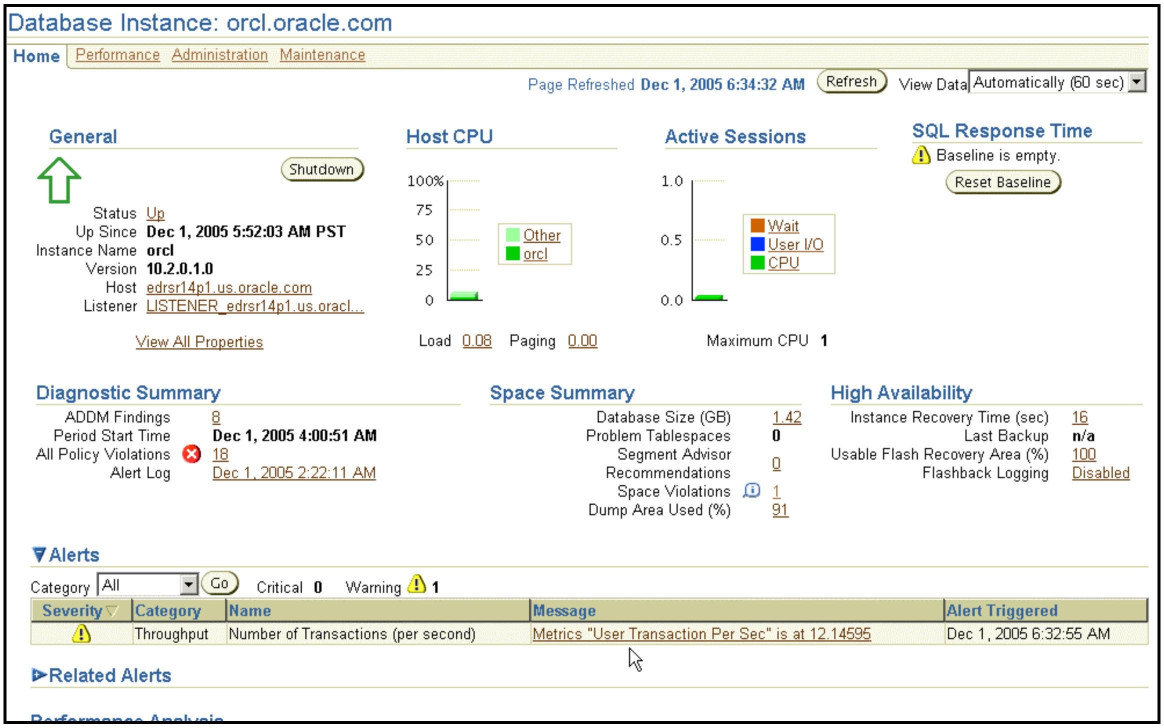


4. After a while, you should see a spike.

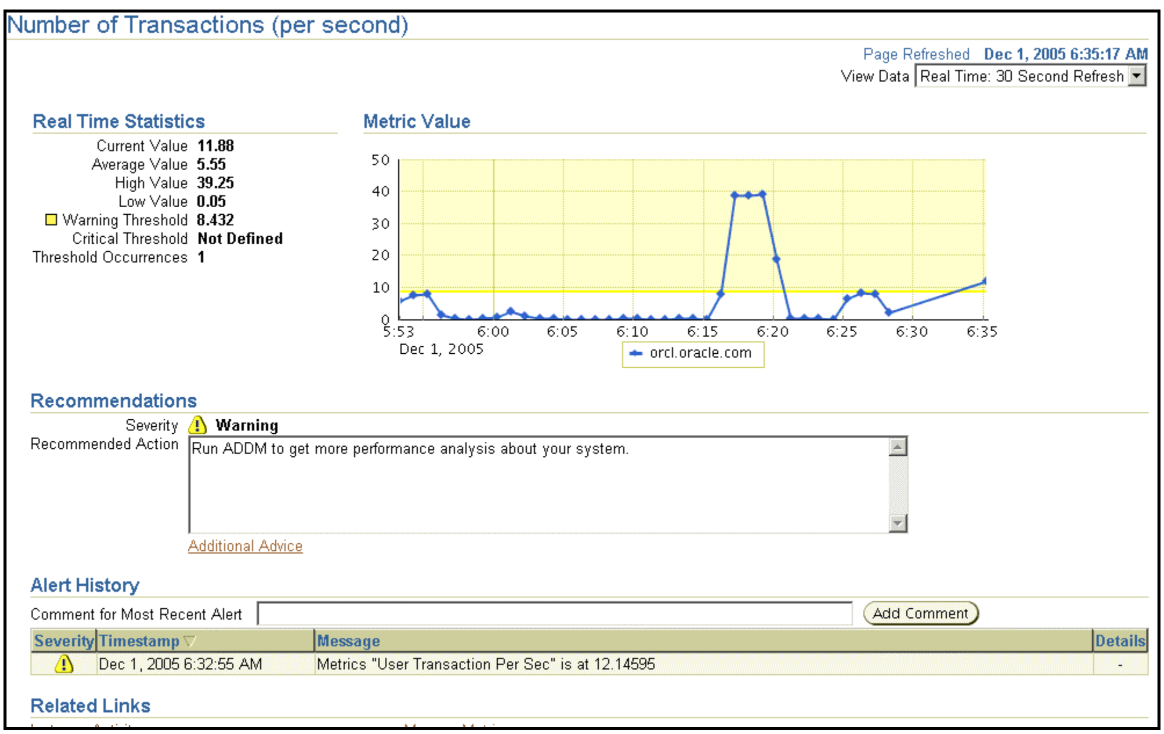


5. The spike indicates an alert. Go to your Database Home page to confirm that you are alerted.

Solutions for Practice 4-2: Working with Baselines (continued)



6. Click the corresponding alert link. This takes you to the Number of Transactions (per second) page. Select Real Time: 30 Second Refresh from the View Data drop-down list. You can confirm the rate and the alert.



7. Remove the `runload` file with `rm /home/oracle/workshops/runload`.

```
$ rm /home/oracle/workshops/runload
rm: remove regular empty file
`/home/oracle/workshops/runload'? y
```

7) From your Database Control Console, deactivate your metric baseline, and then disable metric baselines. Then log out of Enterprise Manager.

   Answer:

1. On the Database Control home page, click the Metric Baselines link.
2. On the Metric Baselines page, select "No active baseline", and click Apply
3. On the Confirmation page, click Yes.
4. On the Metric Baselines page, click the Disable Metric Baselines button.
5. On the Confirmation page, click Yes.
6. Click Logout in the upper right corner.

8) To clean up your environment, execute the `lab_04_02_08.sh` script. This script can take up to 15 minutes to complete.

   Answer:

   You can find the script in your `$HOME/labs` directory.

```
 $ ./lab_04_02_08.sh
TZ set to US/Pacific
Oracle Enterprise Manager 10g Database Control Release
10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation.  All rights
reserved.
http://edrsr3p1.us.oracle.com:1158/em/console/aboutApplica
tion
Stopping Oracle Enterprise Manager 10g Database Control
...
 ...  Stopped.

SQL*Plus: Release 10.2.0.1.0 - Production on Wed Jan 25
11:20:43 2006

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options
```

```
SQL> SQL> SQL> SQL>   2    3    4
PL/SQL procedure successfully completed.

SQL> SQL>
User dropped.

SQL> SQL> SQL> SQL> SQL> SQL> SQL> create type
AWRSQRPT_TEXT_TYPE
            *
ERROR at line 1:
ORA-00955: name is already used by an existing object


create type AWRSQRPT_TEXT_TYPE_TABLE
            *
ERROR at line 1:
ORA-00955: name is already used by an existing object


No errors.
No errors.
No errors.
No errors.
No errors.
No errors.
SQL> SQL> SQL> SQL> SQL> Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> SQL> ORACLE instance started.

Total System Global Area  629145600 bytes
Fixed Size                  1220988 bytes
Variable Size             318770820 bytes
Database Buffers          306184192 bytes
Redo Buffers                2969600 bytes
Database mounted.
Database opened.
SQL> SQL>   2    3    4
PL/SQL procedure successfully completed.

SQL> SQL>
PL/SQL procedure successfully completed.

SQL> SQL> Disconnected from Oracle Database 10g Enterprise
Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
TZ set to US/Pacific
Oracle Enterprise Manager 10g Database Control Release
10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation.  All rights
reserved.
```

```
http://edrsr3p1.us.oracle.com:1158/em/console/aboutApplica
tion
Starting Oracle Enterprise Manager 10g Database Control
............. started.
----------------------------------------------------------
Logs are generated in directory
/u01/app/oracle/product/10.2.0/db_1/edrsr3p1.us.oracle.com
_orcl/sysman/log

SQL*Plus: Release 10.2.0.1.0 - Production on Wed Jan 25
11:28:44 2006

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> SQL> ORACLE instance started.

Total System Global Area  285212672 bytes
Fixed Size                  1218992 bytes
Variable Size             117442128 bytes
Database Buffers          163577856 bytes
Redo Buffers                2973696 bytes
Database mounted.
Database opened.
SQL> SQL> Disconnected from Oracle Database 10g Enterprise
Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
TZ set to US/Pacific
Oracle Enterprise Manager 10g Database Control Release
10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation.  All rights
reserved.
http://edrsr3p1.us.oracle.com:5500/em/console/aboutApplica
tion
Starting Oracle Enterprise Manager 10g Database Control
......... started.
----------------------------------------------------------
Logs are generated in directory
/u01/app/oracle/product/10.2.0/db_1/edrsr3p1.us.oracle.com
_prod/sysman/log
 $
```

Practice Solutions for Lesson 5

During this practice, you are going to install Statspack and capture Statspack snapshots to analyze a workload running on your `orcl` database.

Solutions for Practice 5-1: *Installing Statspack*

The goal of this practice is to install the Statspack repository and packages in the `orcl` database. You need to install the Statspack repository in the `SYSAUX` tablespace. Make sure you use the `oracle` password for the `PERFSTAT` user.

1) Connected as `SYSDBA` using SQL*Plus, execute the `spcreate.sql` script to create the Statspack repository and packages.

   Answer:

   Set the `ORACLE_SID` environment variable to `orcl`, connect with `/ as sysdba`, and then execute the `spcreate.sql` script that is located in the `$ORACLE_HOME/rdbms/admin` directory.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Thu Jan 26
07:58:06 2006

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> @?/rdbms/admin/spcreate.sql

Choose the PERFSTAT user's password
-----------------------------------
Not specifying a password will result in the installation
FAILING

Enter value for perfstat_password: oracle
oracle


Choose the Default tablespace for the PERFSTAT user
---------------------------------------------------
Below is the list of online tablespaces in this database
which can
store user data.  Specifying the SYSTEM tablespace for the
user's
default tablespace will result in the installation
FAILING, as
using SYSTEM for performance data is not supported.

Choose the PERFSTAT users's default tablespace.  This is
the tablespace
```

```
in which the STATSPACK tables and indexes will be created.

TABLESPACE_NAME                CONTENTS  STATSPACK DEFAULT
TABLESPACE
------------------------------ --------- -----------------
-----------
EXAMPLE                        PERMANENT
SYSAUX                         PERMANENT *
USERS                          PERMANENT

Pressing <return> will result in STATSPACK's recommended
default
tablespace (identified by *) being used.

Enter value for default_tablespace: SYSAUX

Using tablespace SYSAUX as PERFSTAT default tablespace.


Choose the Temporary tablespace for the PERFSTAT user
-----------------------------------------------------
Below is the list of online tablespaces in this database
which can
store temporary data (e.g. for sort workareas).
Specifying the SYSTEM
tablespace for the user's temporary tablespace will result
in the
installation FAILING, as using SYSTEM for workareas is not
supported.

Choose the PERFSTAT user's Temporary tablespace.

TABLESPACE_NAME                CONTENTS  DB DEFAULT TEMP
TABLESPACE
------------------------------ --------- -----------------
---------
TEMP                           TEMPORARY *

Pressing <return> will result in the database's default
Temporary
tablespace (identified by *) being used.

Enter value for temporary_tablespace: TEMP

Using tablespace TEMP as PERFSTAT temporary tablespace.


... Creating PERFSTAT user


... Installing required packages
```

```
... Creating views


... Granting privileges

NOTE:
SPCUSR complete. Please check spcusr.lis for any errors.

...

NOTE:
SPCTAB complete. Please check spctab.lis for any errors.

...

NOTE:
SPCPKG complete. Please check spcpkg.lis for any errors.


SQL>
```

2) Make sure that you did not have errors while installing Statspack. Check the log files that were generated during the installation.

   Answer:

   Check the log files that were generated during the installation. The following is a command-line solution:

```
SQL> host ls
spcpkg.lis  spctab.lis  spcusr.lis

SQL> host grep error *.lis
spcpkg.lis:No errors.
spcpkg.lis:No errors.
spcpkg.lis:SPCPKG complete. Please check spcpkg.lis for
any errors.
spctab.lis:SPCTAB complete. Please check spctab.lis for
any errors.
spcusr.lis:SPCUSR complete. Please check spcusr.lis for
any errors.

SQL> exit
```

Solutions for Practice 5-2: *Creating Statspack Snapshots*

The goal of this practice is to run a workload on your `orcl` database and capture Statspack snapshots for that workload.

1) Before you capture your first Statspack snapshot, execute the `lab_05_02_01.sh` script located in your `$HOME/labs` directory. This script creates a tablespace and a user as the setup for the rest of the practice.

   Answer:

   The output of the `lab_05_02_01.sh` script:

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl

$ cd $HOME/labs

$ ./lab_05_02_01.sh

SQL*Plus: Release 10.2.0.1.0 - Production on Thu Jan 26
08:11:08 2006

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> SQL> SQL> SQL> drop tablespace tbsspc including
contents and datafiles
*
ERROR at line 1:
ORA-00959: tablespace 'TBSSPC' does not exist


SQL> SQL>   2    3    4    5
Tablespace created.

SQL> SQL>
PL/SQL procedure successfully completed.

SQL> SQL> drop user spc cascade
          *
ERROR at line 1:
ORA-01918: user 'SPC' does not exist


SQL> SQL>   2    3
User created.
```

```
SQL> SQL>
Grant succeeded.

SQL> SQL> Connected.
SQL> SQL> drop table spct purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
Table created.

SQL> SQL> > >
PL/SQL procedure successfully completed.

SQL> SQL> Disconnected from Oracle Database 10g Enterprise
Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
 $
```

2) Create a level 7 Statspack snapshot. Make sure that you retrieve the corresponding snapshot ID.

Answer:

Set the `i_snap_level` parameter of the `statspack.snap` procedure to 7.

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl

$ sqlplus perfstat/oracle

SQL*Plus: Release 10.2.0.1.0 - Production on Fri Nov 25
04:12:22 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> variable snap number;
SQL> begin
  2  :snap := statspack.snap(i_snap_level=>7);
  3  end;
  4  /

PL/SQL procedure successfully completed.

SQL> print snap
```

```
      SNAP
----------
         1

SQL> exit
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
[oracle@edrsr14p1 labs]$
```

3) Execute the `start_05_02_03.sh` script from the labs directory. This script starts the workload on your `orcl` database. The script takes about 1 minute to execute.

   Answer:

   The output of the `start_05_02_03.sh` script:

```
$ cd $HOME/labs
$ ./start_05_02_03.sh
$
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


$
```

4) After the workload finishes running, capture a new level 7 Statspack snapshot. Make sure that you retrieve the corresponding snapshot ID. The number is system assigned and is not controllable. The snap numbers are not necessarily consecutive.

Answer:

The commands are shown below. Alternatively, you can execute the `sol_05_02_04_01.sh` script to complete this step. **Note** the snap number.

```
$ cd $HOME/labs
$ sqlplus perfstat/oracle

SQL*Plus: Release 10.2.0.1.0 - Production on Fri Nov 25
04:18:27 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> variable snap number;
SQL> begin
  2    :snap := statspack.snap(i_snap_level=>7);
  3  end;
..4 /

PL/SQL procedure successfully completed.

SQL> print snap

      SNAP
----------
         2

SQL> exit
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
$
```

Solutions for Practice 5-3: *Generating Statspack Reports*

The goal of this lab is to generate a Statspack report, examine it, and fix any issues that you discover while interpreting the Statspack report.

1) Generate a Statspack report between the two previously captured snapshots.

   Answer:

   Generate the Statspack report by using the `spreport.sql` script. Use the snap number that you found in the previous practice.

```
$ cd $HOME/labs
$ sqlplus perfstat/oracle

SQL*Plus: Release 10.2.0.1.0 - Production on Fri Nov 25
04:18:27 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> set echo on

SQL> @?/rdbms/admin/spreport

...

Instances in this Statspack schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   DB Id    Inst Num DB Name      Instance     Host
----------- -------- ------------ ------------ -----------
-
 1090770270        1 ORCL         orcl         edrsr14p1

Using 1090770270 for database Id
Using          1 for instance number


Specify the number of days of snapshots to choose from
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Entering the number of days (n) will result in the most recent
(n) days of snapshots being listed.  Pressing <return> without
specifying a number lists all completed snapshots.


Listing all Completed Snapshots

                                                        Snap
Instance     DB Name        Snap Id   Snap Started    Level Comment
------------ ------------ --------- ----------------- ----- -----------
```

```
orcl          ORCL                   1 25 Nov 2005 04:16      7
                                     2 25 Nov 2005 04:19      7

Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap: 1
Begin Snapshot Id specified: 1

Enter value for end_snap: 2


Specify the Report Name
~~~~~~~~~~~~~~~~~~~~~~~
The default report file name is sp_1_2.  To use this name,
press <return> to continue, otherwise enter an
alternative.

Enter value for report_name: sp_1_2.lst

...

End of Report ( sp_1_2.lst )

SQL> exit
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
$
```

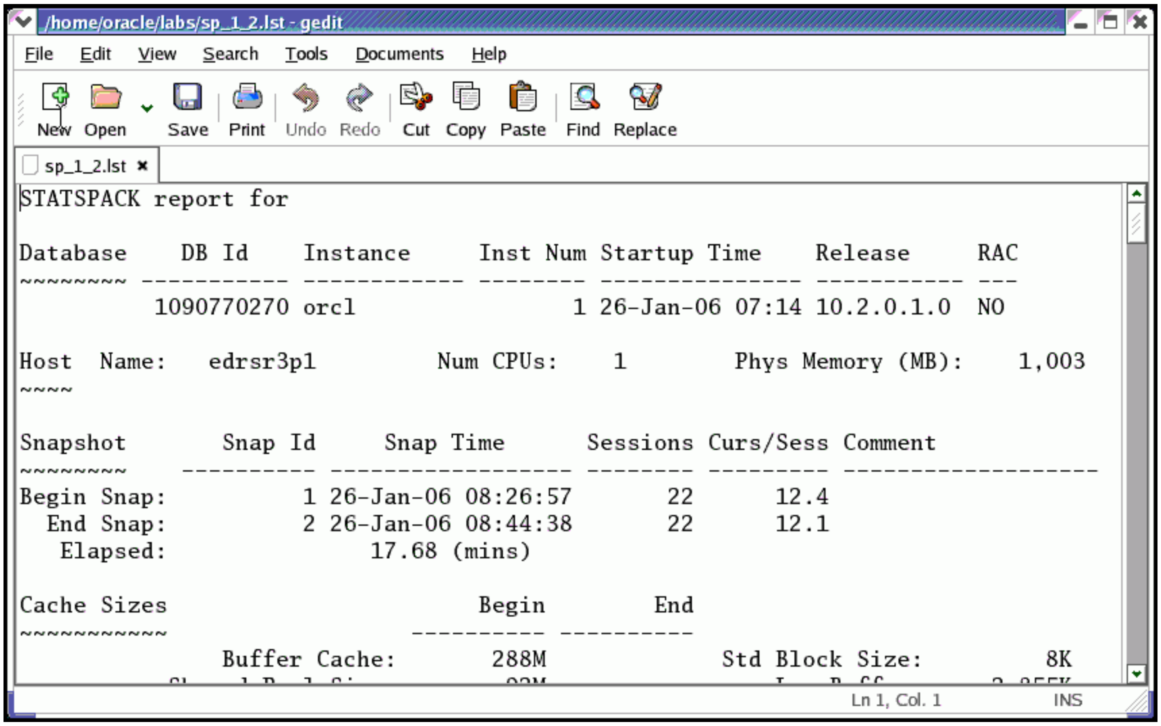
2) Examine the report. Use the `less` utility, which uses `vi` navigation commands, or use the `gedit` graphic text viewer. What are your conclusions?

Answer:

1. Invoke a viewer (`gedit` is shown). Navigation in `gedit` is similar to other graphic-based editors. To find a section of the report, use the menu Search > Find selection and the report section name.

```
$ cd $HOME/labs
$ gedit sp_1_2.lst
```

Solutions for Practice 5-3: Generating Statspack Reports (continued)




```
STATSPACK report for

Database    DB Id    Instance     Inst Num Startup Time    Release     RAC
~~~~~~~~ ----------- ------------ -------- --------------- ----------- ---
          1090770270 orcl                1 26-Jan-06 07:14 10.2.0.1.0  NO

Host  Name:   edrsr3p1         Num CPUs:    1        Phys Memory (MB):    1,003
~~~~

Snapshot       Snap Id     Snap Time      Sessions Curs/Sess Comment
~~~~~~~~    ---------- ------------------ -------- --------- -------------------
Begin Snap:          1 26-Jan-06 08:26:57       22      12.4
  End Snap:          2 26-Jan-06 08:44:38       22      12.1
   Elapsed:               17.68 (mins)

Cache Sizes                       Begin        End
~~~~~~~~~~~                  ---------- ----------
               Buffer Cache:       288M              Std Block Size:         8K
```

2. Scroll to the Top 5 Timed Events section. Buffer busy waits will be among the Top 5 Timed Events.

```
Top 5 Timed Events                                                Avg %Total
~~~~~~~~~~~~~~~~~~                                               wait   Call
Event                                            Waits    Time (s)   (ms)   Time
----------------------------------------- ------------ ----------- ------ ------
buffer busy waits                                  251          75    298   54.1
CPU time                                                        24          17.4
log file parallel write                          7,355          14      2   10.1
latch: In memory undo latch                         68          13    188    9.2
latch free                                          12           3    227    2.0
          -------------------------------------------------------------
```

3. Scrolling down to the SQL Ordered by Elapsed section of the report, you can see that there is an `INSERT` statement on the `SPCT` table that takes a lot of resources.

```
SQL ordered by Elapsed  DB/Inst: ORCL/orcl  Snaps: 1-2
-> Resources reported for PL/SQL code includes the resources used by
all SQL statements called by the code.
-> Total DB Time (s):               156
-> Captured SQL accounts for  170.6% of Total DB Time
-> SQL reported below exceeded  1.0% of Total DB Time

 Elapsed             Elap per         CPU                        Old
 Time(s) Executions  Exec (s)  %Total Time(s) Physical Reads Hash Value
--------- ---------- ---------- ------ -------- ----------- ----------
   150.96          8      18.87   96.8     17.56           0  721925790
Module: SQL*Plus
declare t number; begin for t in 1..2222 loop insert into spct v
alues (Null,'a'); commit; end loop; end;

   110.31     17,776       0.01   70.7      8.23           0 3783977436
Module: SQL*Plus
INSERT INTO SPCT VALUES (NULL,'a')
```

```
     3.67          1        3.67     2.4       3.24            12 2737202410
Module: SQL*Plus
begin  :snap := statspack.snap(i_snap_level=>7); end;

          -------------------------------------------------------------
```

4. This gives you an indication that the SPCT table could be the cause of the buffer busy waits. Looking at the Buffer Wait Statistics section, you can see that most of the waits are due to the data block category.

```
Buffer wait Statistics  DB/Inst: ORCL/orcl  Snaps: 51-52
-> ordered by wait time desc, waits desc

Class                        Waits Total Wait Time (s) Avg Time (ms)
---------------------- ----------- ------------------- -------------
data block                     226                  77           340
undo header                      6                   0             3
segment header                  19                   0             0
          -------------------------------------------------------------
```

5. Looking at the Segments by Buffer Busy Waits, SPCT is shown as the culprit segment.

```
Segments by Buffer Busy Waits  DB/Inst: ORCL/orcl  Snaps: 51-52
-> End Segment Buffer Busy Waits Threshold:       100

                                                           Buffer
                                  Subobject    Obj.          Busy   Pct
Owner   Tablespace Object Name       Name         Type         Waits Total
------- ---------- ----------------- ------------ ----- --------- -----
SPC     TBSSPC     SPCT                           TABLE       245 100.0
       -------------------------------------------------------
```

6. All this leads you to think that there is an issue with the SPCT table. You observed that there were a lot of buffer busy waits while inserting into the SPCT table. Looking at the table's definition, you discover that the table was defined without Automatic Segment Space Management.

```
$ sqlplus spc/spc

SQL*Plus: Release 10.2.0.1.0 - Production on Mon Nov 28
02:24:32 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options
SQL> SELECT segment_space_management
  2   FROM dba_tablespaces
  3   WHERE tablespace_name=(select tablespace_name
  4                             from user_tables);

SEGMEN
------
```

```
MANUAL

SQL> exit;
```

3) Fix the problem by implementing the recommendations from the previous step. Note that you can re-create the table for this purpose.

   Answer:

   You decide to re-create the table using Automatic Segment Space Management. This script is available as `sol_05_03_03_01.sql` in the `solutions` directory.

```
$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Fri Nov 25
08:42:55 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> drop tablespace tbsspc
  2  including contents and datafiles;

Tablespace dropped.

SQL> CREATE SMALLFILE TABLESPACE "TBSSPC"
  2  DATAFILE 'tbsspc1.dbf' SIZE 50M
  3  LOGGING
  4  EXTENT MANAGEMENT LOCAL
  5  SEGMENT SPACE MANAGEMENT AUTO;

Tablespace created.

SQL> connect spc/spc
Connected.
SQL> create table spct(id number, name varchar2(2000))
  2  tablespace tbsspc;

Table created.

SQL> exec DBMS_STATS.GATHER_TABLE_STATS(-
   >  ownname=>'SPC', tabname=>'SPCT',-
   >  estimate_percent=>DBMS_STATS.AUTO_SAMPLE_SIZE);

PL/SQL procedure successfully completed.

SQL> exit
```

4) Regenerate the same workload, and verify that the problem disappears.

   Answer:

   1. Take a new snapshot like you did in the previous practice. Run the same workload again and take another snapshot. This script is available as `sol_05_03_04_01.sh`.

```
$ export ORACLE_SID=orcl
$ sqlplus perfstat/oracle

SQL*Plus: Release 10.2.0.1.0 - Production on Fri Nov 25
08:50:54 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> variable snap number;
SQL> begin
  2   :snap := statspack.snap(i_snap_level=>7);
  3  end;
  4  /

PL/SQL procedure successfully completed.

SQL> print snap;
SQL>
      SNAP
----------
        11

SQL> exit
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
$ cd ../labs
$ ./start_04_2_03.sh
$
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

```
PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

$
$ sqlplus perfstat/oracle

SQL*Plus: Release 10.2.0.1.0 - Production on Fri Nov 25
08:50:54 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> variable snap number;
SQL> begin
  2    :snap := statspack.snap(i_snap_level=>7);
  3  end;
  4  /

PL/SQL procedure successfully completed.

SQL> print snap;

      SNAP
----------
        12

SQL> exit
```

2. Generate the corresponding Statspack report.

```
$ sqlplus perfstat/oracle

SQL*Plus: Release 10.2.0.1.0 - Production on Fri Nov 25
08:50:54 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to:
```

```
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> @?/rdbms/admin/spreport

Current Instance
~~~~~~~~~~~~~~~~

   DB Id    DB Name      Inst Num Instance
----------- ------------ -------- ------------
 1090770270 ORCL                1 orcl



Instances in this Statspack schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   DB Id    Inst Num DB Name      Instance     Host
----------- -------- ------------ ------------ -----------
-
 1090770270        1 ORCL         orcl         edrsr14p1

Using 1090770270 for database Id
Using          1 for instance number


Specify the number of days of snapshots to choose from
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Entering the number of days (n) will result in the most recent
(n) days of snapshots being listed.  Pressing <return> without
specifying a number lists all completed snapshots.



Listing all Completed Snapshots

                                                        Snap
Instance     DB Name        Snap Id   Snap Started    Level
Comment
------------ ------------ --------- ----------------- ----- ----
orcl         ORCL                 1 25 Nov 2005 04:16     7
                                  2 25 Nov 2005 04:19     7
                                 11 25 Nov 2005 07:27     7
                                 12 25 Nov 2005 07:29     7



Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap: 11
Begin Snapshot Id specified: 11

Enter value for end_snap: 12
…
```

```
Specify the Report Name
~~~~~~~~~~~~~~~~~~~~~~~
The default report file name is sp_11_12.  To use this
name,
press <return> to continue, otherwise enter an
alternative.

Enter value for report_name: sp_11_12.lst
…
End of Report ( sp_11_12.lst )

SQL>
```

3. Look at the generated report, using your preferred text browser. You can see that the number of buffer busy waits has been reduced.

```
…

Top 5 Timed Events                                          Avg %Total
~~~~~~~~~~~~~~~~~~                                         wait   Call
Event                                     Waits    Time (s)   (ms)   Time
-------------------------------- ------------ ----------- ------ ------
CPU time                                                20           23.2
log file parallel write                   6,152          12      2   14.0
buffer busy waits                            45          11    254   13.2
latch: In memory undo latch                  40           9    227   10.5
latch free                                   22           6    250    6.4
          ----------------------------------------------------


…

SQL ordered by Elapsed  DB/Inst: ORCL/orcl  Snaps: 61-62
-> Resources reported for PL/SQL code includes the resources used by
all SQL statements called by the code.
-> Total DB Time (s):             146
-> Captured SQL accounts for  150.5% of Total DB Time
-> SQL reported below exceeded  1.0% of Total DB Time

Elapsed              Elap per          CPU                          Old
Time (s) Executions  Exec (s)  %Total Time(s) Physical Reads Hash Value
-------- ---------- --------- ------ -------- -------------- ----------
  139.05          8     17.38   95.3    16.38              0  721925790
Module: SQL*Plus
declare t number; begin for t in 1..2222 loop insert into spct v
alues (Null,'a'); commit; end loop; end;

   73.16     17,776      0.00   50.1     7.59              0 3783977436
Module: SQL*Plus
INSERT INTO SPCT VALUES (NULL,'a')

…
          --------------------------------------------------------


…

Buffer wait Statistics  DB/Inst: ORCL/orcl  Snaps: 61-62
-> ordered by wait time desc, waits desc

Class                        Waits Total Wait Time (s) Avg Time (ms)
---------------------- ----------- ------------------- -------------
```

```
data block                       43                   12           277
undo header                       2                    0            15
          -------------------------------------------------------------

…

Segments by Buffer Busy Waits  DB/Inst: ORCL/orcl  Snaps: 61-62
-> End Segment Buffer Busy Waits Threshold:       100

                                                           Buffer
                                Subobject    Obj.            Busy   Pct
Owner Tablespace Object Name      Name         Type           Waits Total
----- ---------- ---------------- ------------ ----- ------------ -----
SPC   TBSSPC     SPCT                          TABLE            43 100.0
     -------------------------------------------------------------
```

Practice Solutions for Lesson 6

During this practice, you are going to use AWR, ADDM, and ASH reports to identify and fix the same issues as the ones you discovered in the previous practice.

Solutions for Practice 6-1: *Using AWR-Based Tools*

You are going to follow a similar sequence of steps as with the previous practice, but this time you will use AWR-based tools instead of Statspack. Note that during this practice, you explicitly capture AWR snapshots. However, by default, the system automatically captures those every hour.

1) Execute the `lab_06_01_01.sql` script to set up this practice.

   Answer:

   You can find this script in your `$HOME/labs` directory.

```
$ export ORACLE_SID=orcl
$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Tue Nov 29
01:15:57 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> @lab_06_01_01.sql
SQL> set echo on
SQL> drop tablespace tbsspc including contents and
datafiles;

Tablespace dropped.

SQL> CREATE SMALLFILE TABLESPACE "TBSSPC"
DATAFILE 'tbsspc1.dbf' SIZE 50M
LOGGING
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT MANUAL;
  2    3    4    5
Tablespace created.

SQL> execute
dbms_workload_repository.modify_snapshot_settings(interval
=> 1440);

PL/SQL procedure successfully completed.

SQL> exec
dbms_advisor.set_default_task_parameter('ADDM','DB_ACTIVIT
Y_MIN',30);

PL/SQL procedure successfully completed.
```

```
SQL> drop user spc cascade;

User dropped.

SQL> create user spc identified by spc
default tablespace tbsspc
temporary tablespace temp;
  2    3
User created.

SQL> grant connect, resource, dba to spc;

Grant succeeded.

SQL> connect spc/spc
Connected.
SQL> drop table spct purge;
create table spct(id number, name varchar2(2000));
drop table spct purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
Table created.

SQL> exec DBMS_STATS.GATHER_TABLE_STATS(-
ownname=>'SPC', tabname=>'SPCT',-
estimate_percent=>DBMS_STATS.AUTO_SAMPLE_SIZE);
> >
PL/SQL procedure successfully completed.

SQL> exit
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
$
```

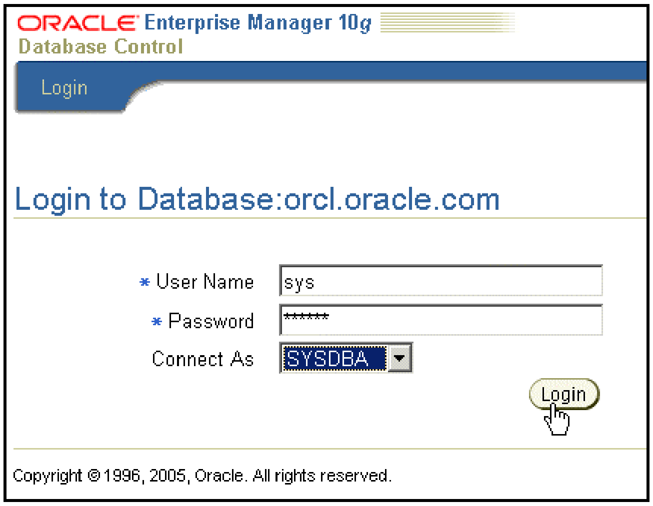

2) Take your first AWR snapshot. You can use either Enterprise Manager or the `DBMS_WORKLOAD_REPOSITORY` package.

Answer:

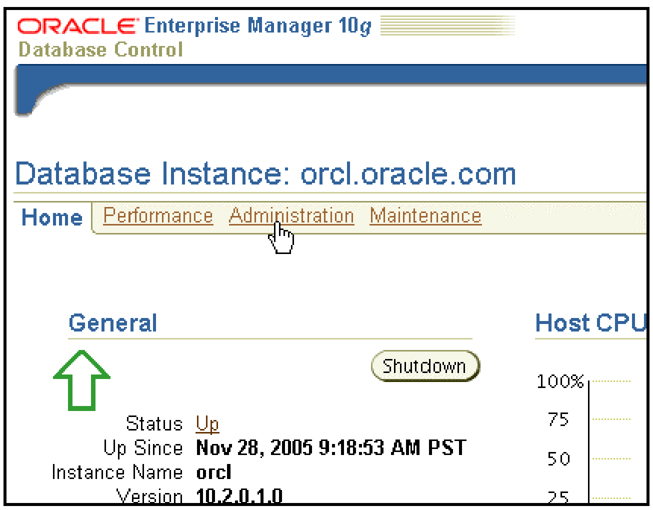
1. You can use the `sol_06_01_02.sh` script located in your `$HOME/solutions` directory to take an AWR snapshot by using the `DBMS_WORKLOAD_REPOSITORY` package. The solution in this document uses Enterprise Manager. First, open your browser by using a URL similar to

Solutions for Practice 6-1: Using AWR-Based Tools (continued)

`http://<machine_name>:1158/em`. This takes you to the EM Login page. Enter the credentials for the `SYS` user and click Login.

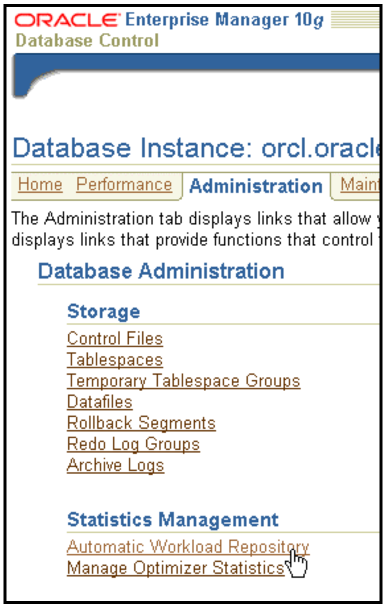


2. On the Database Instance page, click the Administration tab.

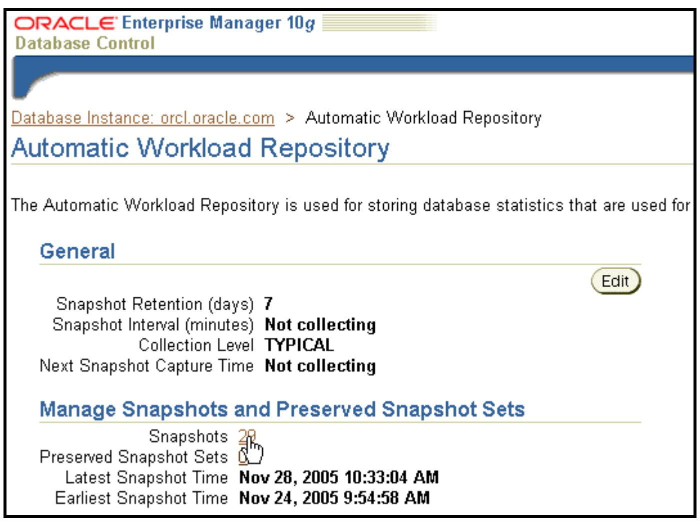


3. On the Administration page, click Automatic Workload Repository.

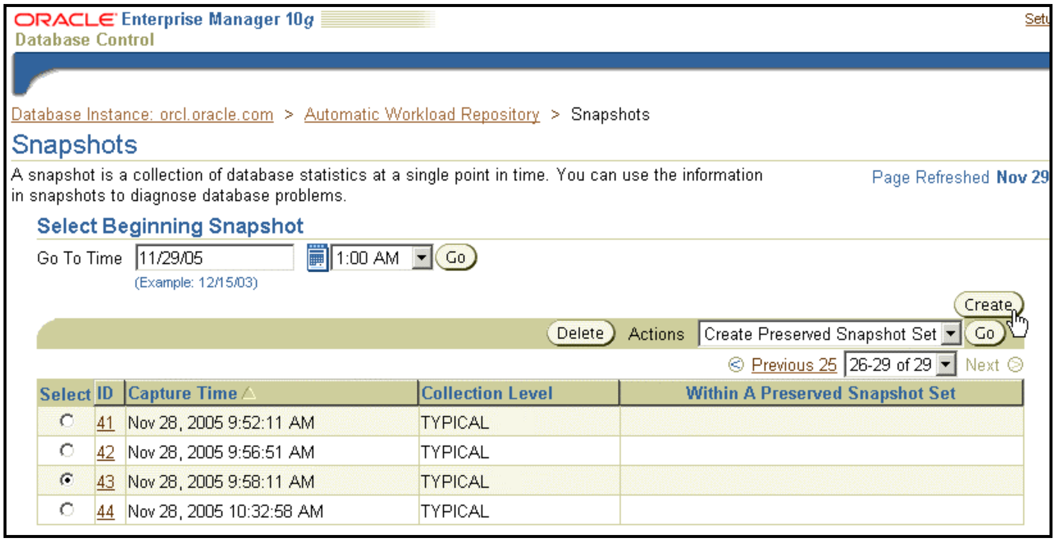
Solutions for Practice 6-1: Using AWR-Based Tools (continued)



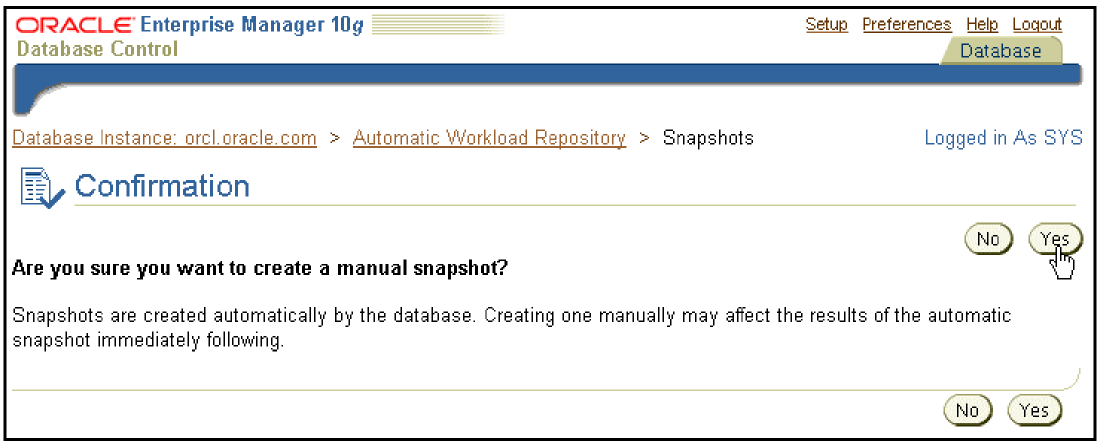
4. On the Automatic Workload Repository page, click the number corresponding to the Snapshots field.



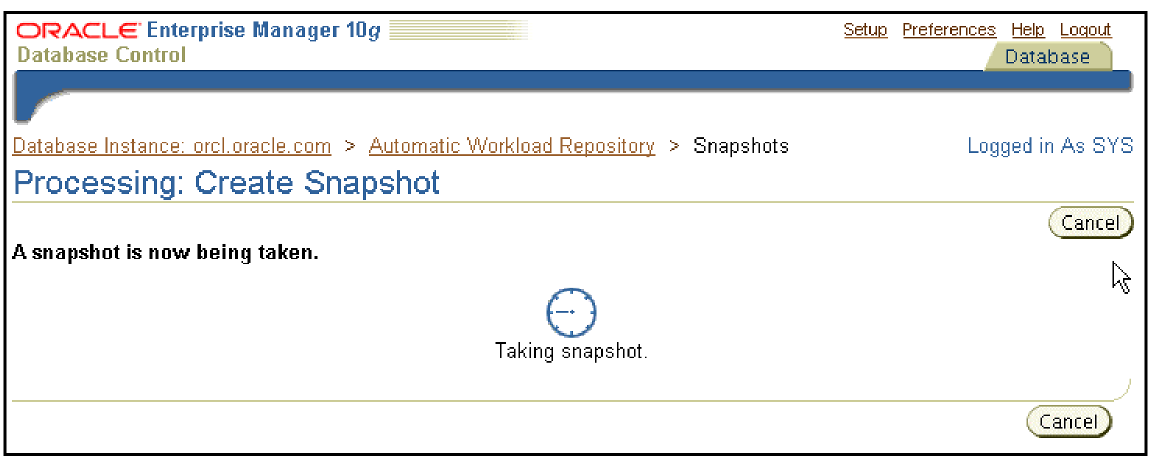
5. On the Snapshots page, click Create.



6. On the Confirmation page, click Yes.

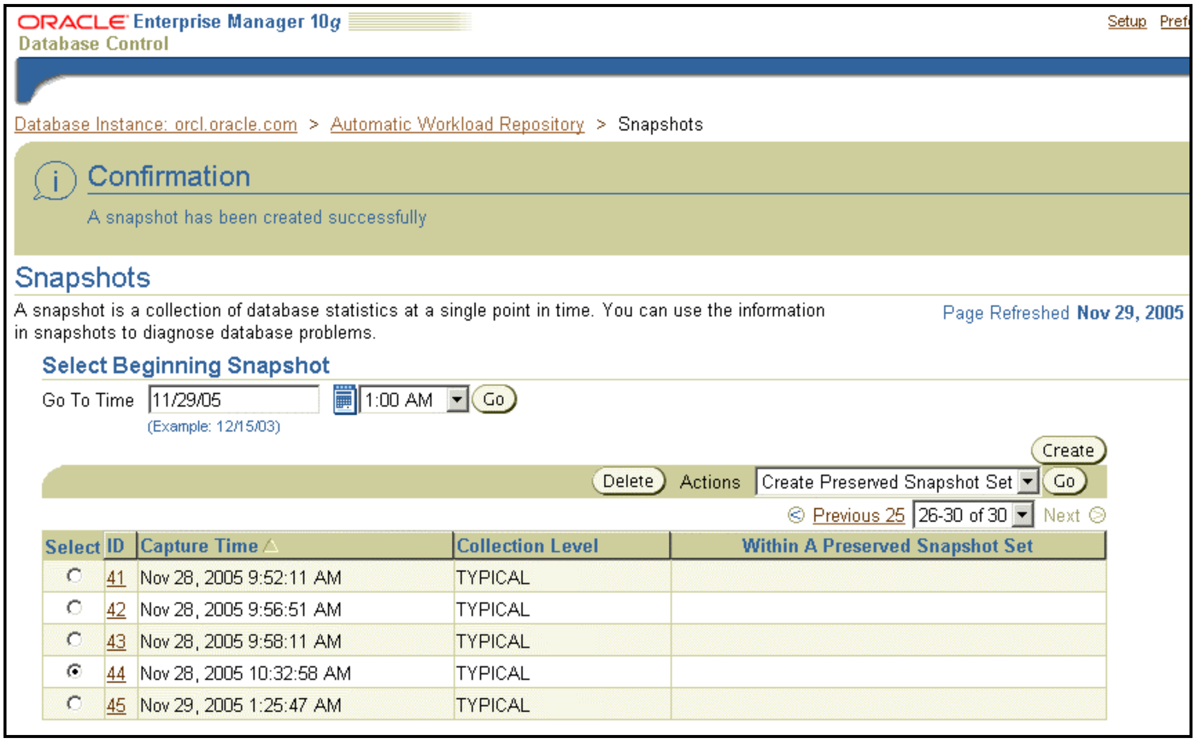


7. The processing page appears. Wait until the snapshot is created.



8. The Confirmation box appears on the Snapshots page. You can see that a new snapshot has been created (45 in the screenshot).

Solutions for Practice 6-1: Using AWR-Based Tools (continued)



3) Execute the `start_06_01_03.sh` script to generate the workload for this lab.

Answer:

The script is located in your `$HOME/labs` directory. Wait until the "Load is finished" message appears, and then continue. **Note:** The last "$" prompt does not appear until you press [Enter] one more time.

```
$ ./start_06_01_03.sh
$
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

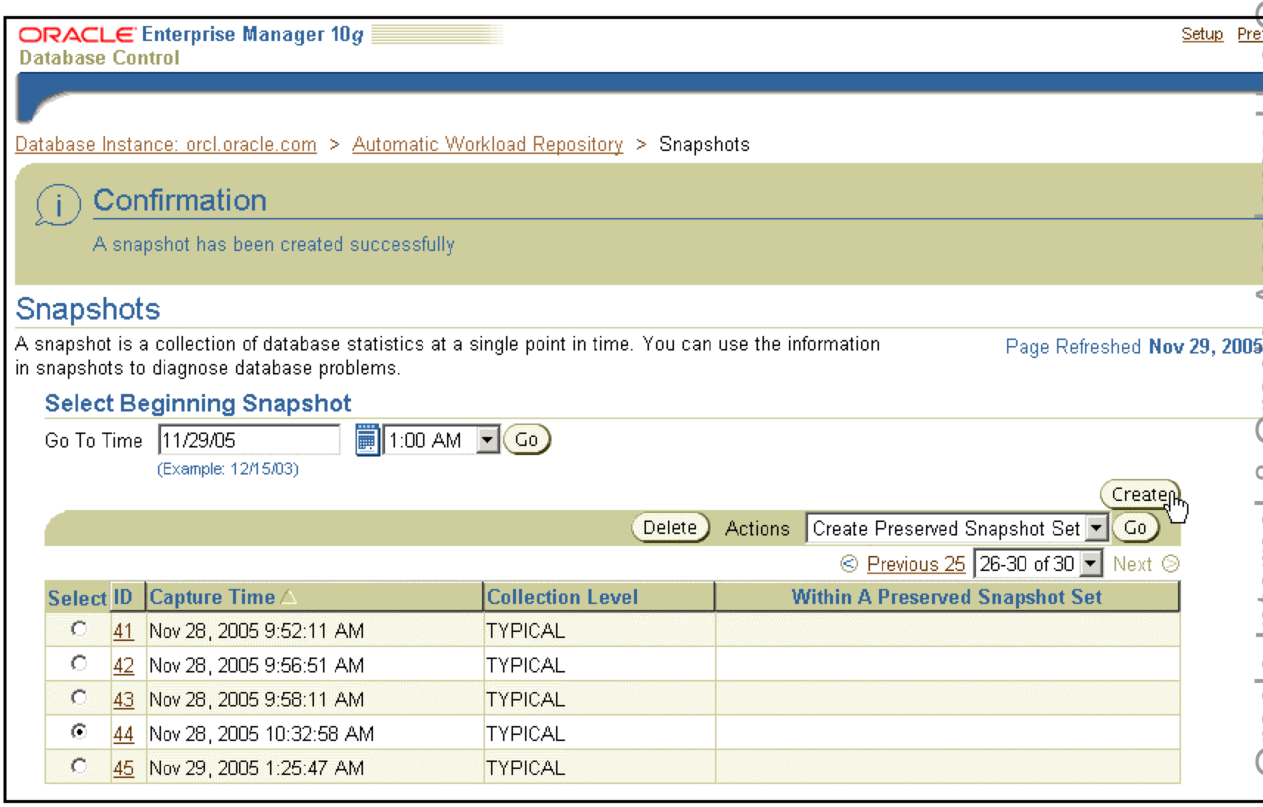
Solutions for Practice 6-1: Using AWR-Based Tools (continued)

```
PL/SQL procedure successfully completed.

Load is finished
$
```

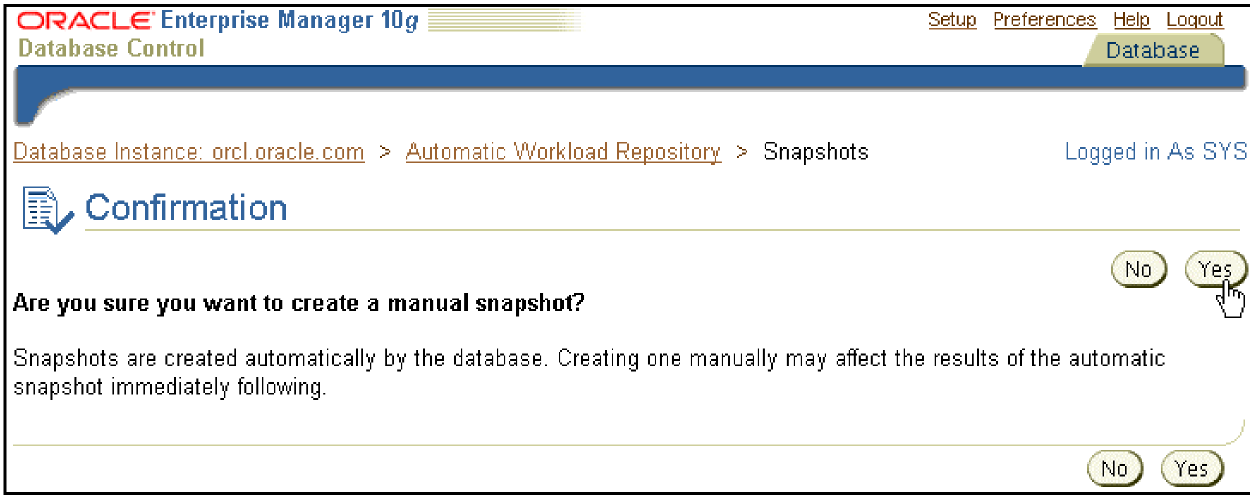
4) Take your second AWR snapshot.

Answer:

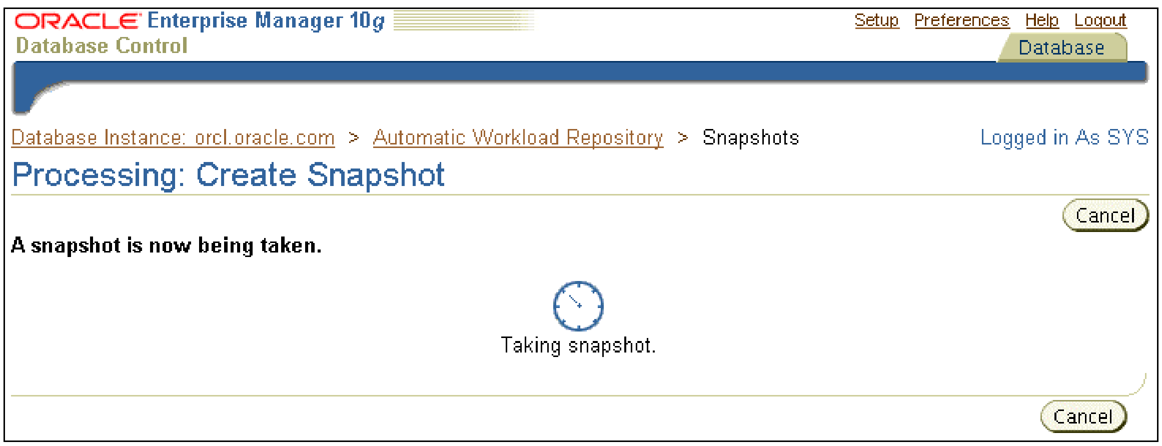
1. You can use the `sol_06_01_04.sh` script located in your `$HOME/solutions` directory to take an AWR snapshot by using the `DBMS_WORKLOAD_REPOSITORY` package. The solution in this document uses Enterprise Manager. On the Snapshots page, click Create.



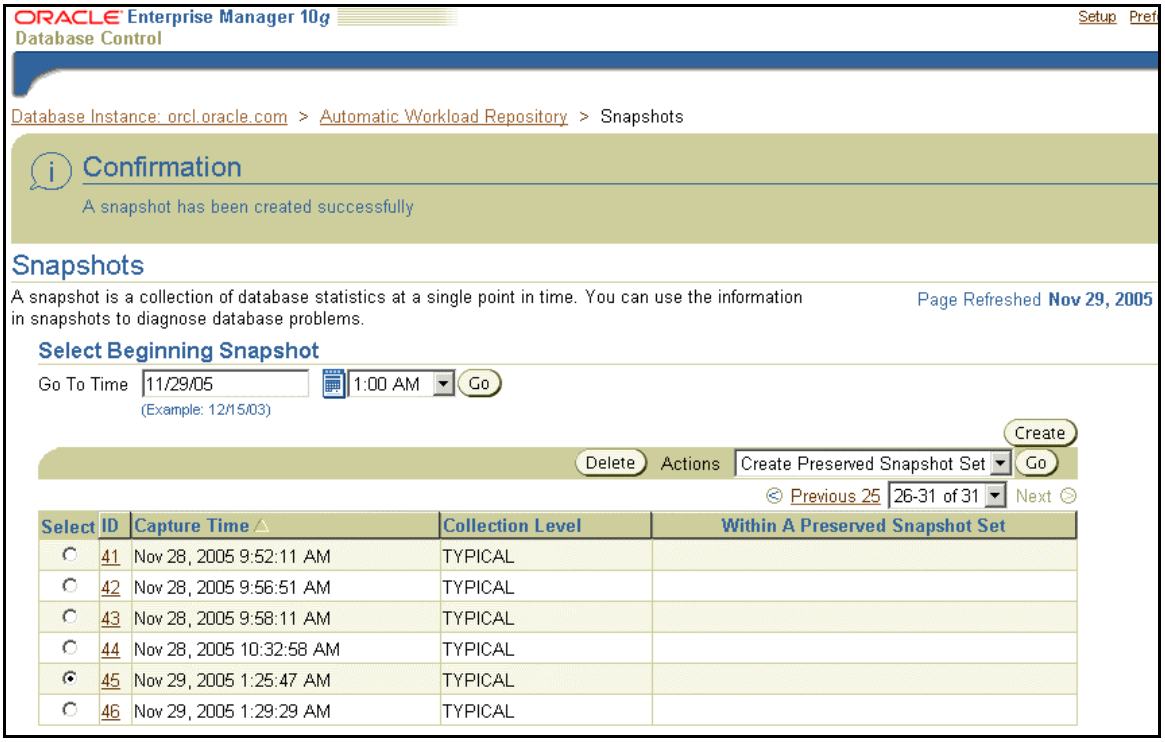
2. On the Confirmation page, click Yes.



3. The Processing page appears. Wait until the snapshot is created.



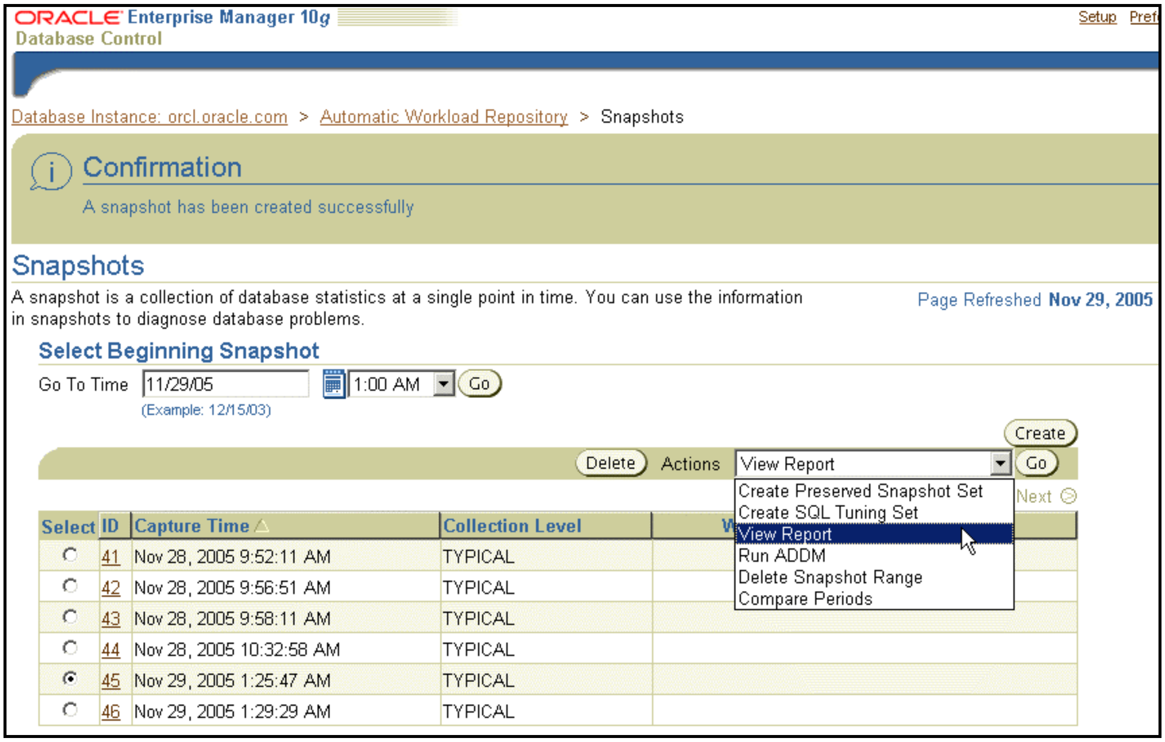
4. The Confirmation box appears on the Snapshots page, where you can see that a new snapshot has been created (46 in the screenshot).



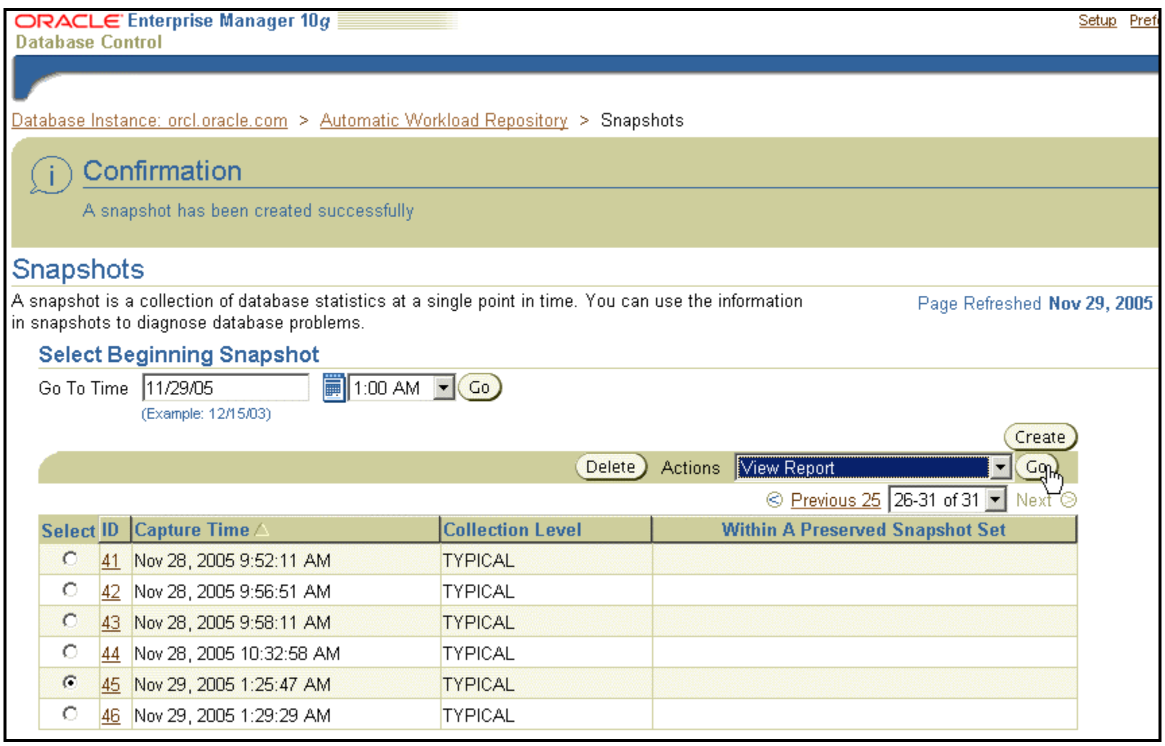
5) Look at the AWR and ADDM reports. What are your conclusions?

Answer:

1. You can use the `awrrpt.sql` script located in the `$ORACLE_HOME/rdbms/admin` directory to create an AWR report. The solution in this document uses Enterprise Manager. Still on the Snapshots page, select View Report from the Actions drop-down list. Make sure that your first AWR snapshot is selected (45 in the screenshot).

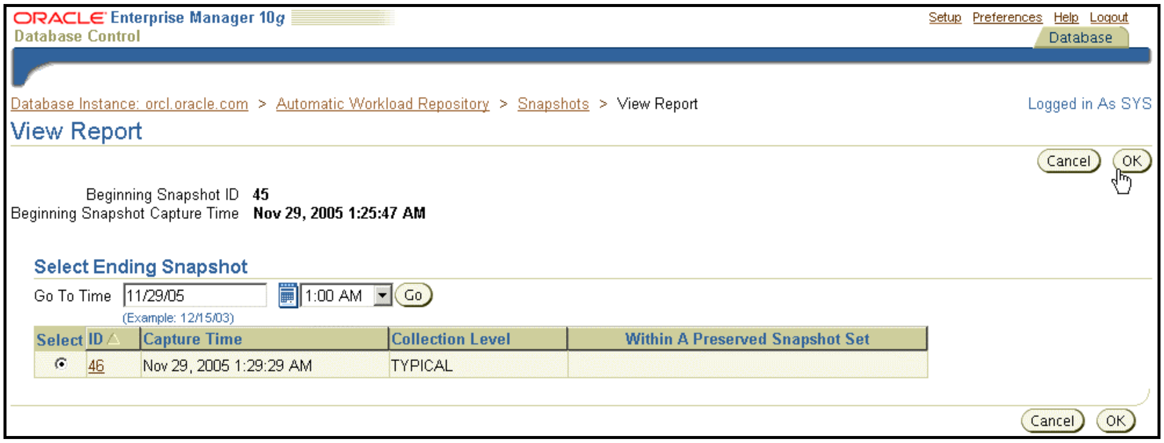


2. When done, click Go.

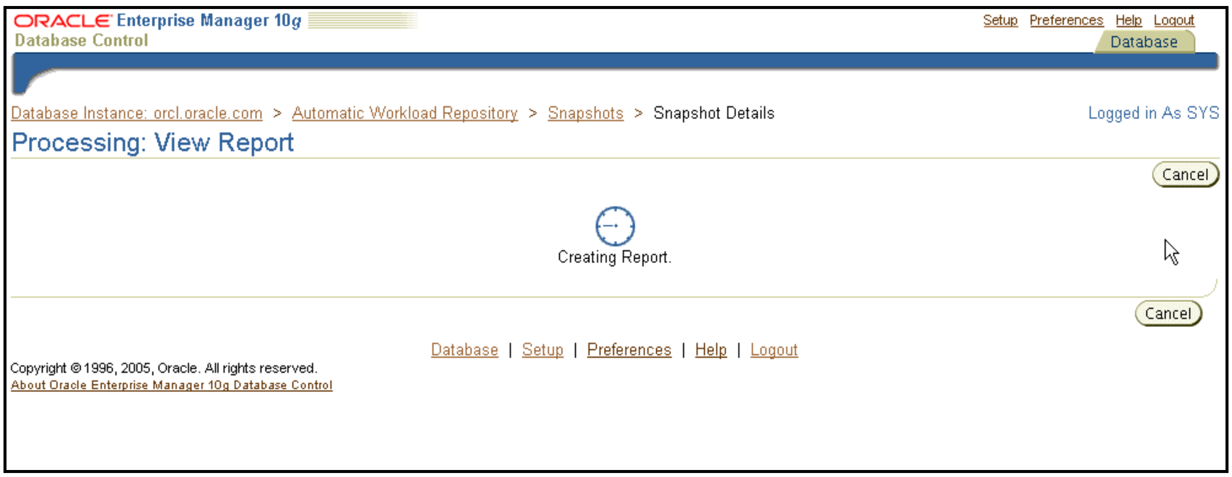


Solutions for Practice 6-1: Using AWR-Based Tools (continued)

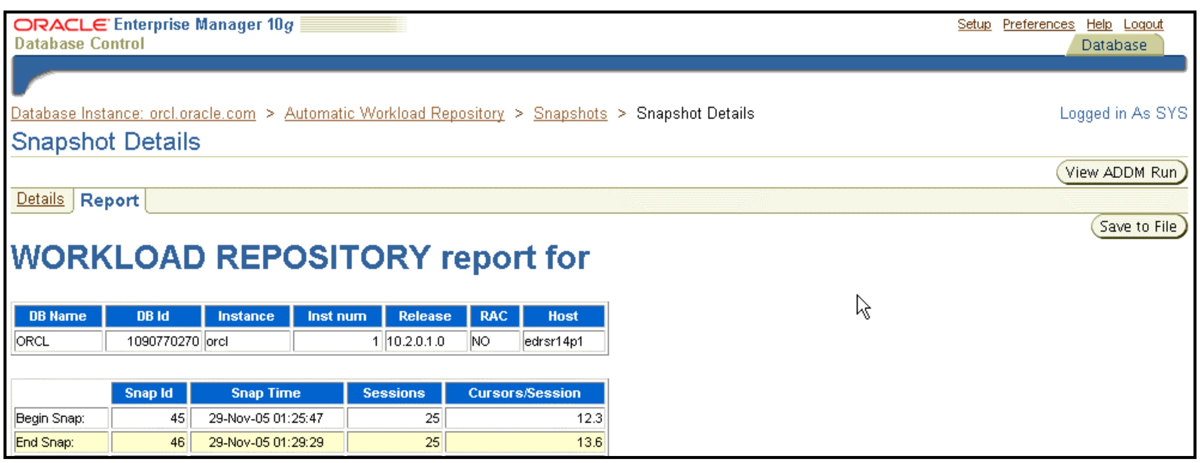
3. On the View Report page, make sure that your second AWR snapshot is selected, and click OK.



4. The processing page appears. Wait until the report is created.



5. After the report is created, you are taken to the Snapshot Details page where you can see the report.



6. Scroll down to the Top 5 Timed Events.

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
buffer busy waits	581	129	223	31.1	Concurrency
CPU time		52		12.6	
latch: In memory undo latch	269	47	174	11.2	Concurrency
log file parallel write	8,466	35	4	8.5	System I/O
latch free	19	15	770	3.5	Other

7. You can scroll down to the "SQL ordered by Elapsed Time" section or click the following links under Main Report: `SQL Statistics` and then `SQL ordered by Elapsed Time`.

SQL ordered by Elapsed Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied by 100

Elapsed Time (s)	CPU Time (s)	Executions	Elap per Exec (s)	% Total DB Time	SQL Id	SQL Module	SQL Text
328	40	8	40.96	78.77	2thp6rbungv9n	SQL*Plus	declare t number; begin for t ...
237	18	44,440	0.01	56.99	3csh3g3mjhmzh	SQL*Plus	INSERT INTO SPCT VALUES (NULL,...
35	1	4	8.63	8.30	6gvch1xu9ca3g		DECLARE job BINARY_INTEGER := ...
18	0	4	4.51	4.33	c8h3jdwaa532q	EM_PING	SELECT TO_NUMBER(PARAMETER_VAL...
14	0	4	3.43	3.30	6t7007dkc1abh	EM_PING	DELETE FROM MGMT_JOB_EMD_STATU...
8	7	1	8.31	2.00	8u809k64x3nzd	Admin Connection	begin DBMS_WORKLOAD_REPOSITORY...
4	0	4	0.94	0.90	cydnuss99swtd	OEM.SystemPool	BEGIN EM_PING.RECORD_BATCH_HEA...
1	1	1	1.31	0.31	bunssq950snhf		insert into wrh$_sga_target_ad...
1	0	69	0.01	0.19	cvn54b7yz0s8u		select /*+ index(idl_ub1$ i_id...
1	1	939	0.00	0.16	3c1kubcdjnppg		update sys.col_usage$ set eq...

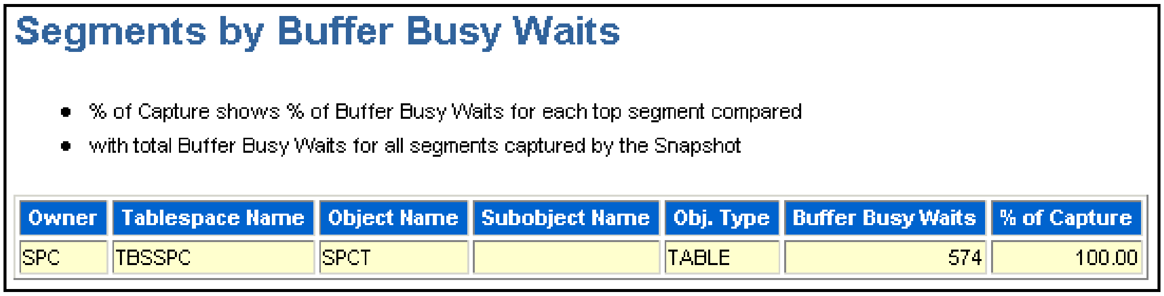
8. Scroll down to the Buffer Wait Statistics section, or click the following links in succession: `Back to Top`, `Wait Statistics`, and `Buffer Wait Statistics`.

Buffer Wait Statistics

- ordered by wait time desc, waits desc

Class	Waits	Total Wait Time (s)	Avg Time (ms)
data block	530	133	252
undo header	7	0	21
segment header	44	0	0

9. Scroll down to "Segments by Buffer Busy Waits," or click the following links in succession: `Back to Top`, `Segment Statistics`, and `Segments by Buffer Busy Waits`.

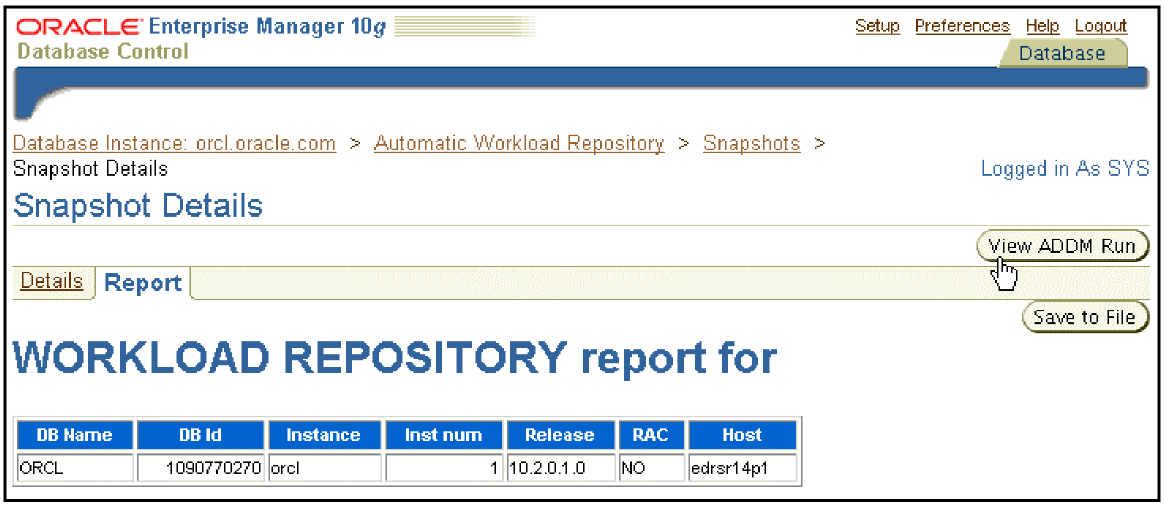


Segments by Buffer Busy Waits

- % of Capture shows % of Buffer Busy Waits for each top segment compared
- with total Buffer Busy Waits for all segments captured by the Snapshot

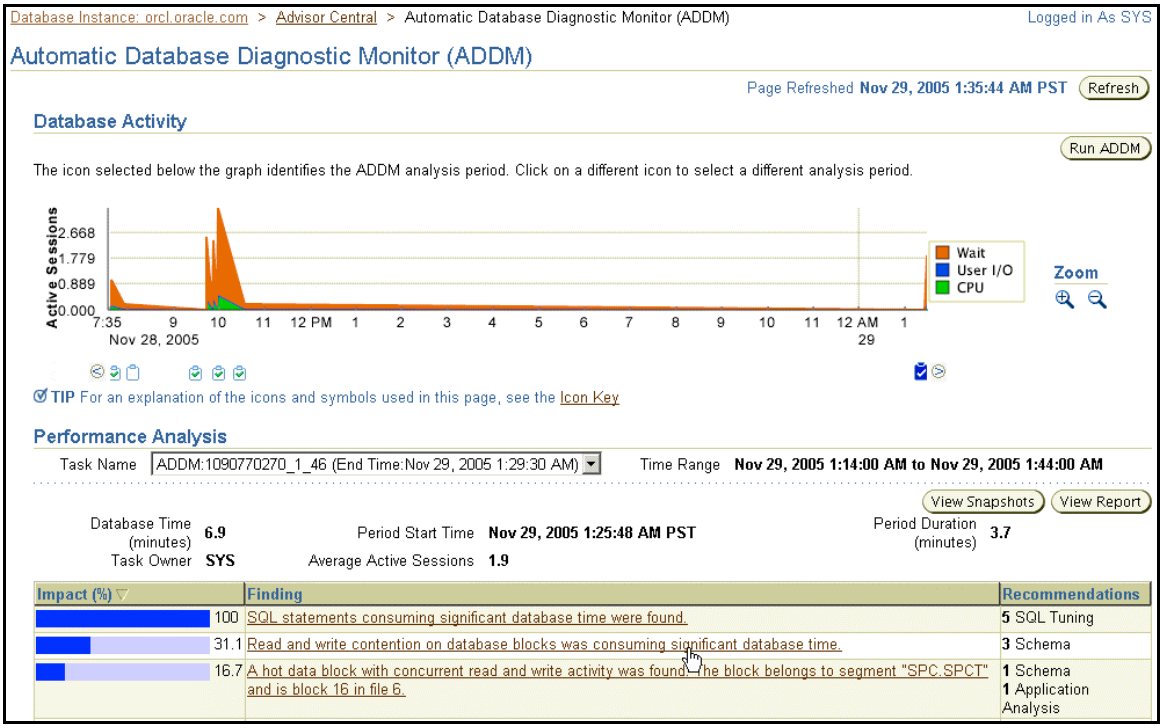
Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Buffer Busy Waits	% of Capture
SPC	TBSSPC	SPCT		TABLE	574	100.00

10. You can use the `addmrpt.sql` script located in the `$ORACLE_HOME/rdbms/admin` directory to generate an ADDM report. The solution in this document uses Enterprise Manager. Still on the Snapshot Details page, click View ADDM Run.

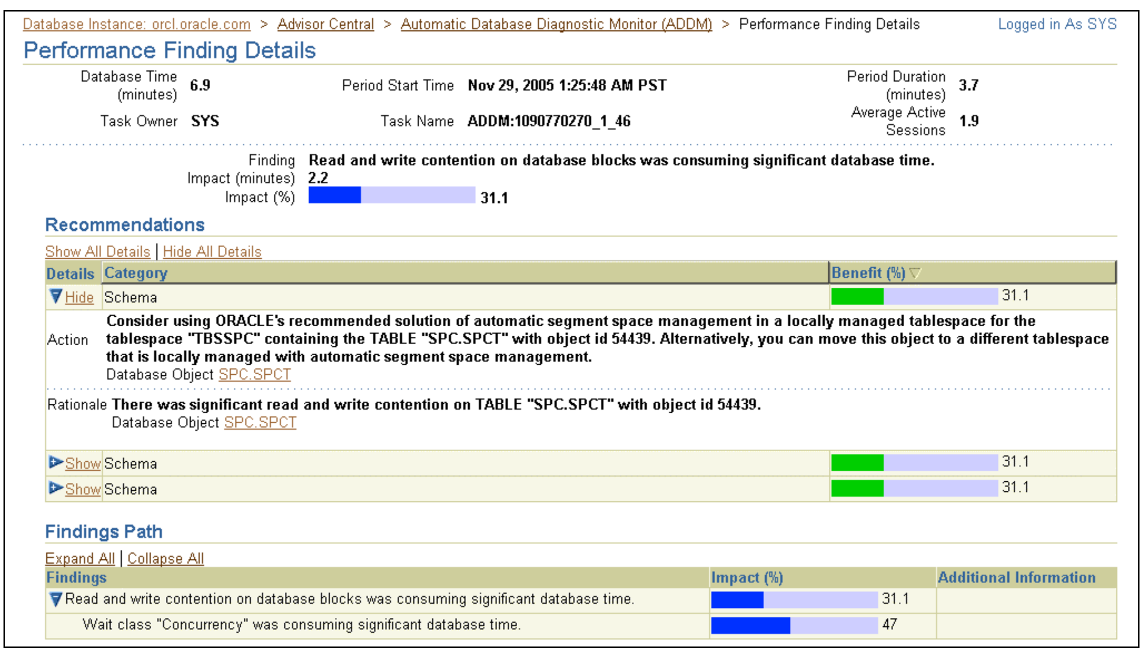


11. This takes you to the corresponding ADDM analysis page. On the Automatic Database Diagnostic Monitor (ADDM) page, click the “Read and write contention on database blocks” finding. The position of this finding may vary from the example shown, and the total number of findings may also vary.

Solutions for Practice 6-1: Using AWR-Based Tools (continued)



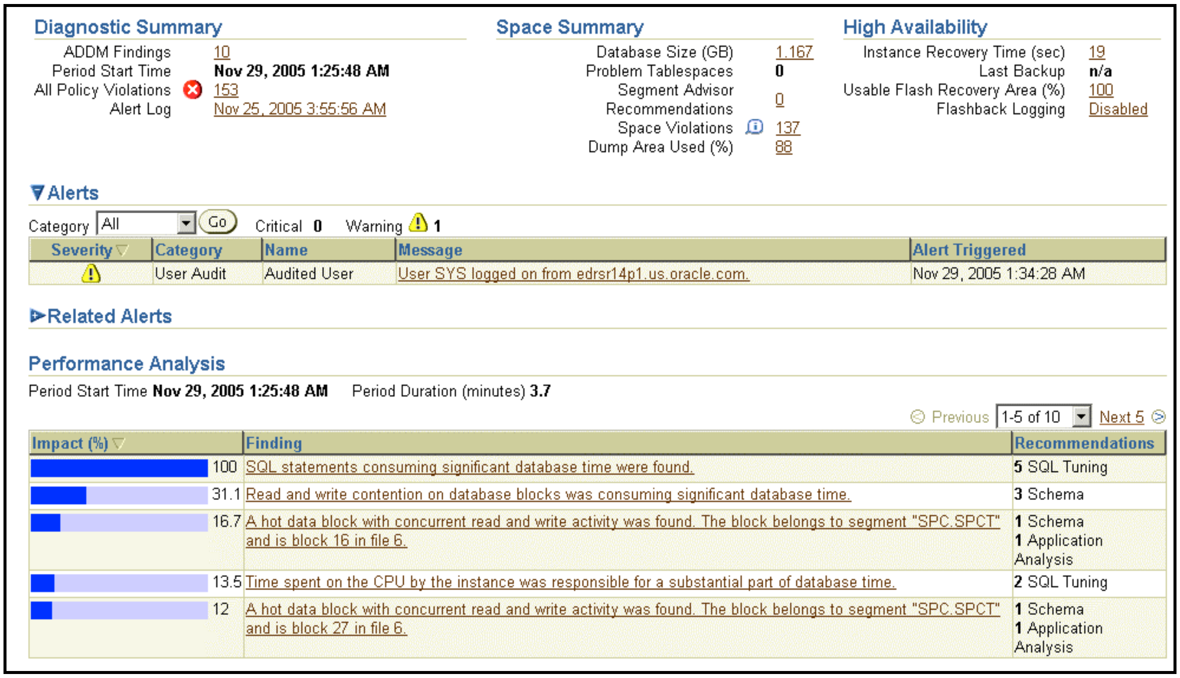
12. This takes you to the corresponding Performance Finding Details page. You can clearly see that ADDM recommends reorganizing the `SPCT` table to use Automatic Segment Space Management. Although you are able to infer this by looking at either a Statspack report or an AWR report, ADDM automatically tells you what to do. This recommendation may not appear. ADDM considers many factors before it gives a recommendation. A delay in creating the snapshot after the workload finishes is one reason the recommendation may not appear.



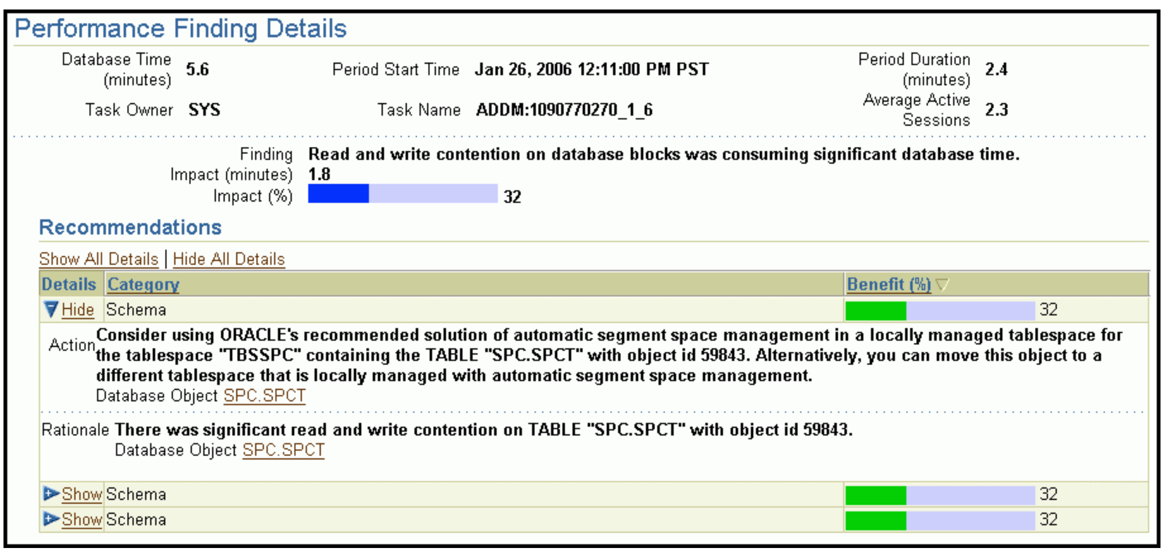
13. Click Database to return to the Database Instance Home page. The latest ADDM report may not be visible until a few automatic refreshes take place.

Solutions for Practice 6-1: Using AWR-Based Tools (continued)

14. After some automatic refreshes, you can see the latest performance analysis reported on the Home page.



15. In the Performance Analysis section, click the finding "Read and write contention on database blocks…." This shows the same page as in step 12.



6) To implement the suggested recommendation, execute the `lab_06_01_06.sql` script.

Answer:

You can locate the `lab_06_01_06.sql` script in your `$HOME/labs` directory.

```
$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Tue Nov 29
01:39:40 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL>@lab_06_01_06.sql

SQL> drop tablespace tbsspc
  2  including contents and datafiles;

Tablespace dropped.

SQL>
SQL> CREATE SMALLFILE TABLESPACE "TBSSPC"
  2  DATAFILE 'tbsspc1.dbf' SIZE 50M
  3  LOGGING
  4  EXTENT MANAGEMENT LOCAL
  5  SEGMENT SPACE MANAGEMENT AUTO;

Tablespace created.

SQL>
SQL> connect spc/spc
Connected.
SQL>
SQL> drop table spct purge;
drop table spct purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> create table spct(id number, name varchar2(2000))
  2  tablespace tbsspc;

Table created.

SQL>
SQL> exec DBMS_STATS.GATHER_TABLE_STATS(-
> ownname=>'SPC', tabname=>'SPCT',-
> estimate_percent=>DBMS_STATS.AUTO_SAMPLE_SIZE);

PL/SQL procedure successfully completed.
```

```
SQL>
SQL> exit;
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
$
```

7) Execute the `start_06_01_03.sh` script to start generating the workload for this lab again.

   Answer:

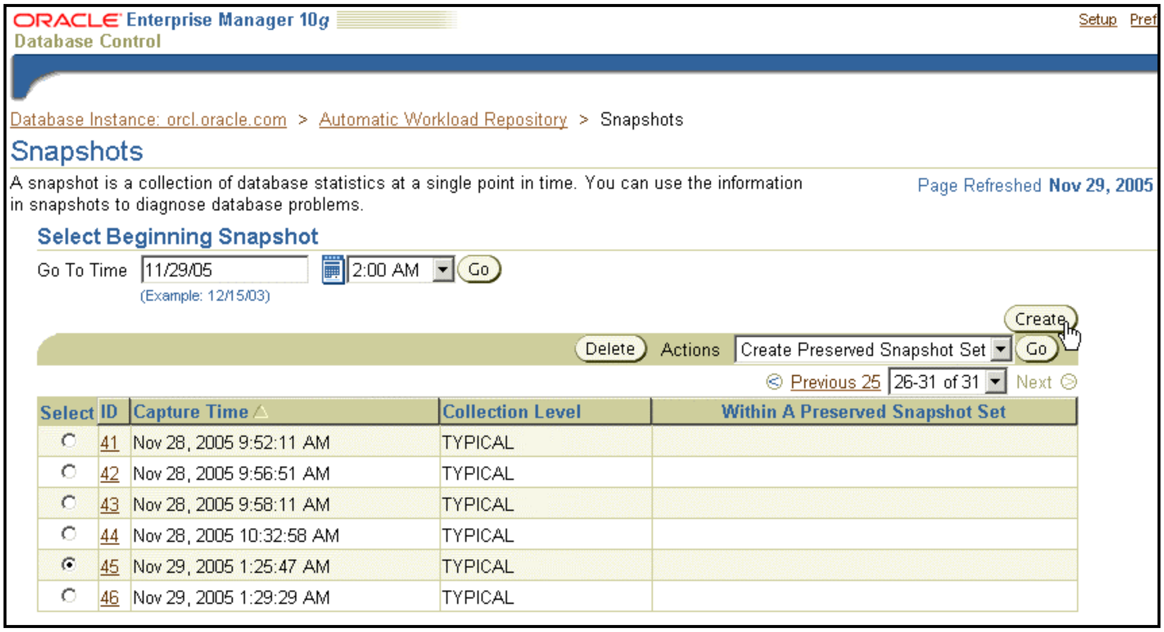
   This script is located in your `$HOME/labs` directory.

```
$ ./start_06_01_03.sh
$
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

$
```

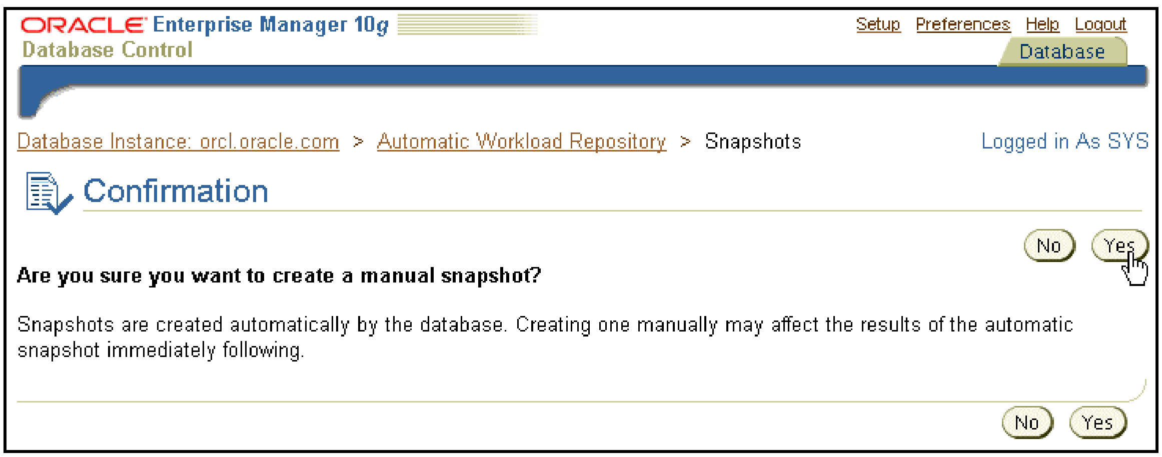
8) Take a third AWR snapshot.

   Answer:

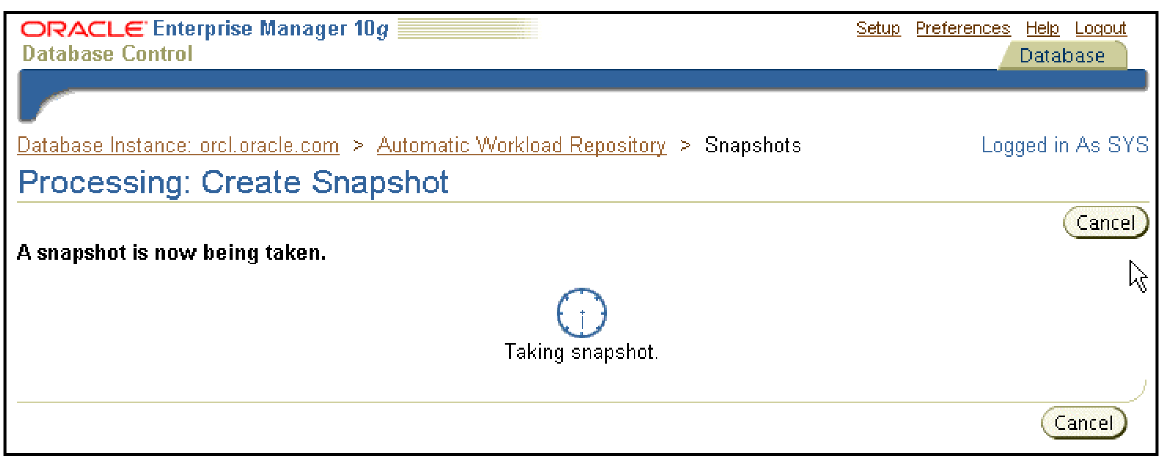
   1. You can use the `sol_06_01_08.sh` script located in your `$HOME/solutions` directory to take an AWR snapshot using the `DBMS_WORKLOAD_REPOSITORY` package. The solution in this document uses Enterprise Manager. On the Snapshots page, click Create.



2. On the Confirmation page, click Yes.

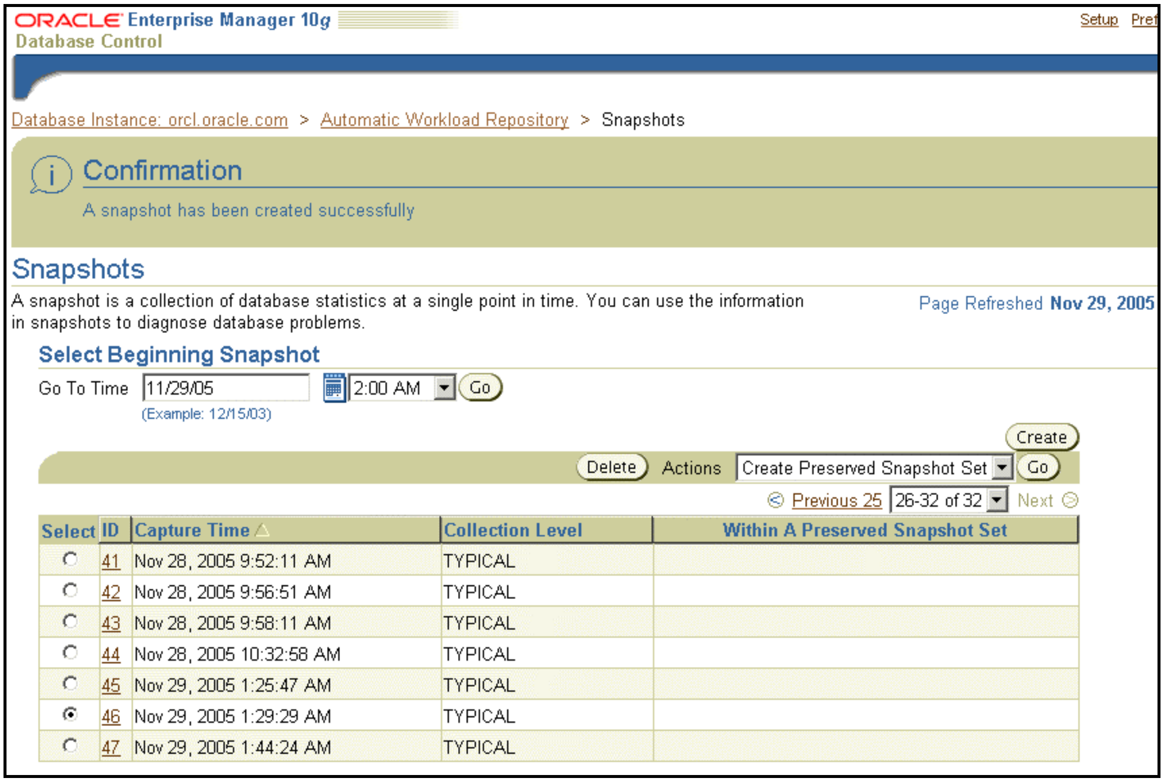


3. The Processing page appears. Wait until the snapshot is created.



4. The Confirmation box appears on the Snapshots page. You can clearly see that a third snapshot has been created (47 in the screenshot).

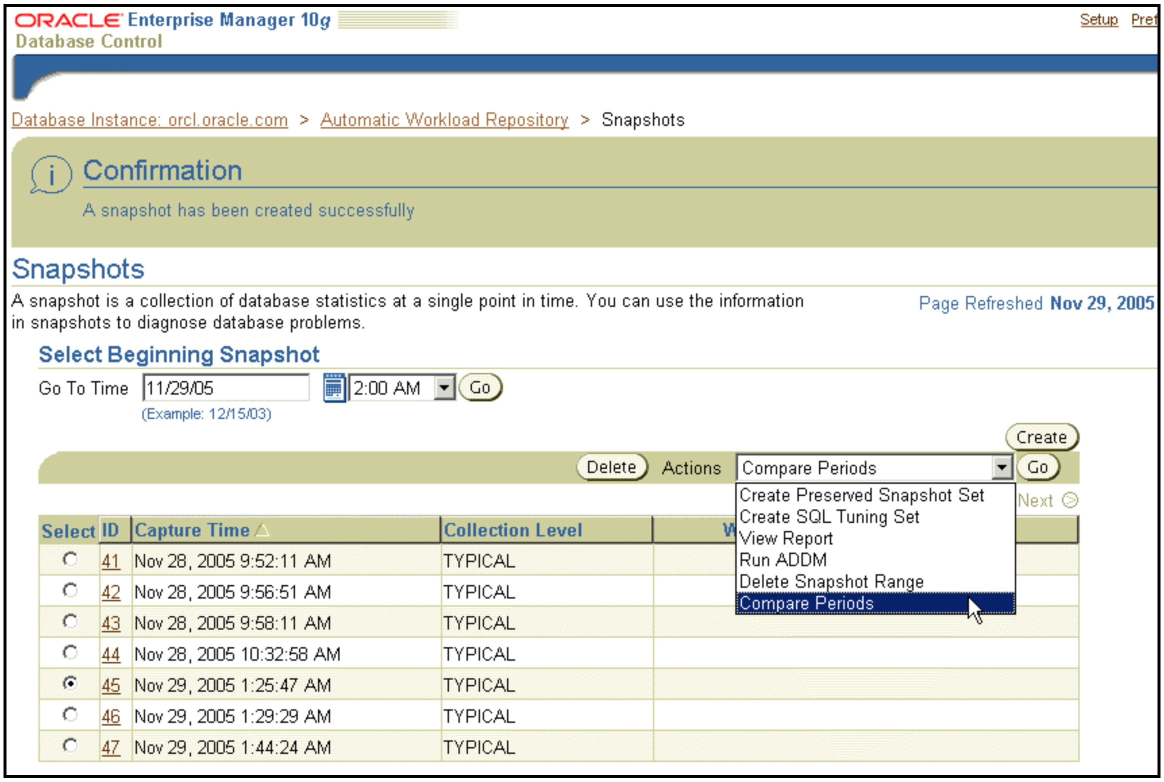
Solutions for Practice 6-1: Using AWR-Based Tools (continued)



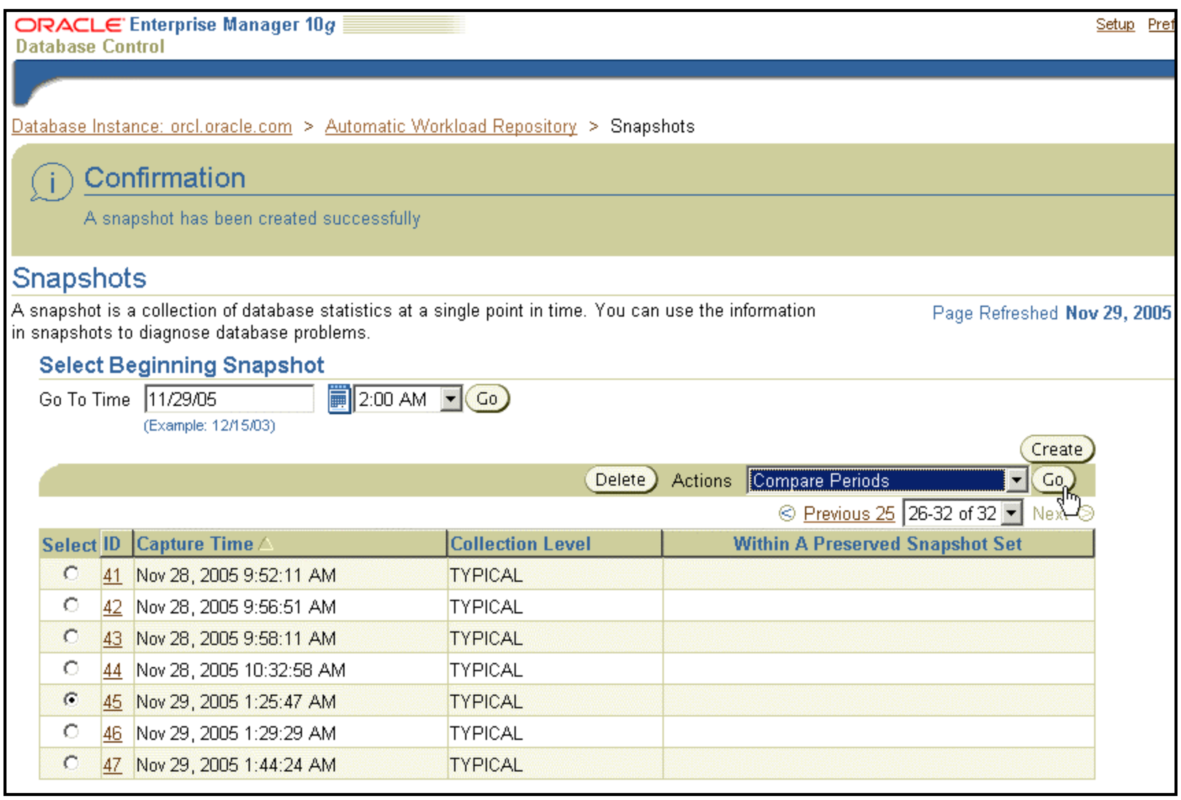
9) Generate and review a Compare Periods report, also known as a diff-diff report, comparing the periods between the three snapshots.

Answer:

1. You can use the `awrddrpt.sql` script located in your `$ORACLE_HOME/rdbms/admin` directory to generate the diff-diff report. The solution in this document uses Enterprise Manager. On the Snapshots page, select Compare Periods from the Actions drop-down list. Make sure that your first AWR snapshot is selected (45 in the screenshot).

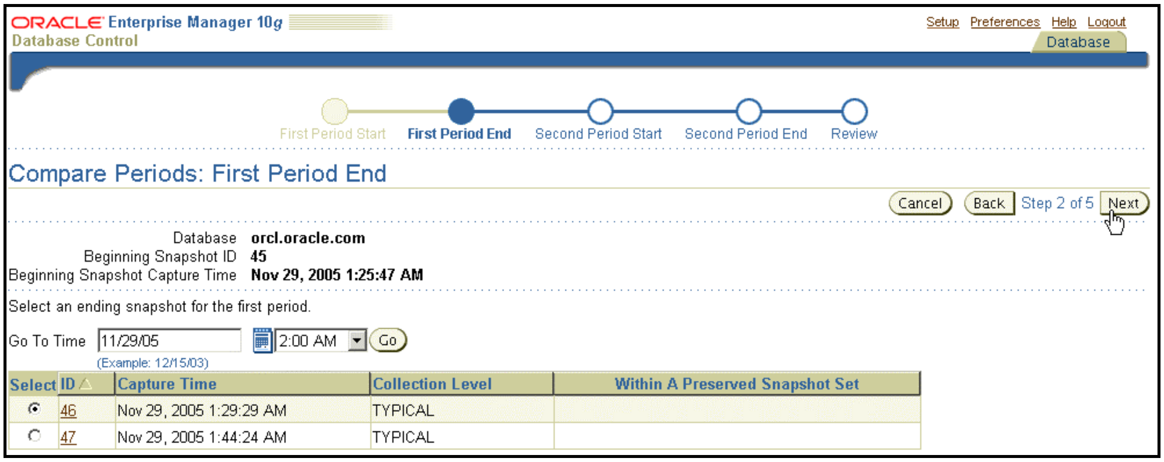

ORACLE Enterprise Manager 10g
Database Control
Setup Pref
Database Instance: orcl.oracle.com > Automatic Workload Repository > Snapshots
Confirmation
A snapshot has been created successfully
Snapshots
A snapshot is a collection of database statistics at a single point in time. You can use the information in snapshots to diagnose database problems.
Page Refreshed Nov 29, 2005
Select Beginning Snapshot
Go To Time 11/29/05 2:00 AM Go
(Example: 12/15/03)
Create
Delete Actions Compare Periods Go
Create Preserved Snapshot Set
Create SQL Tuning Set
View Report
Run ADDM
Delete Snapshot Range
Compare Periods
Next
Select ID Capture Time Collection Level
41 Nov 28, 2005 9:52:11 AM TYPICAL
42 Nov 28, 2005 9:56:51 AM TYPICAL
43 Nov 28, 2005 9:58:11 AM TYPICAL
44 Nov 28, 2005 10:32:58 AM TYPICAL
45 Nov 29, 2005 1:25:47 AM TYPICAL
46 Nov 29, 2005 1:29:29 AM TYPICAL
47 Nov 29, 2005 1:44:24 AM TYPICAL

2. Click Go.

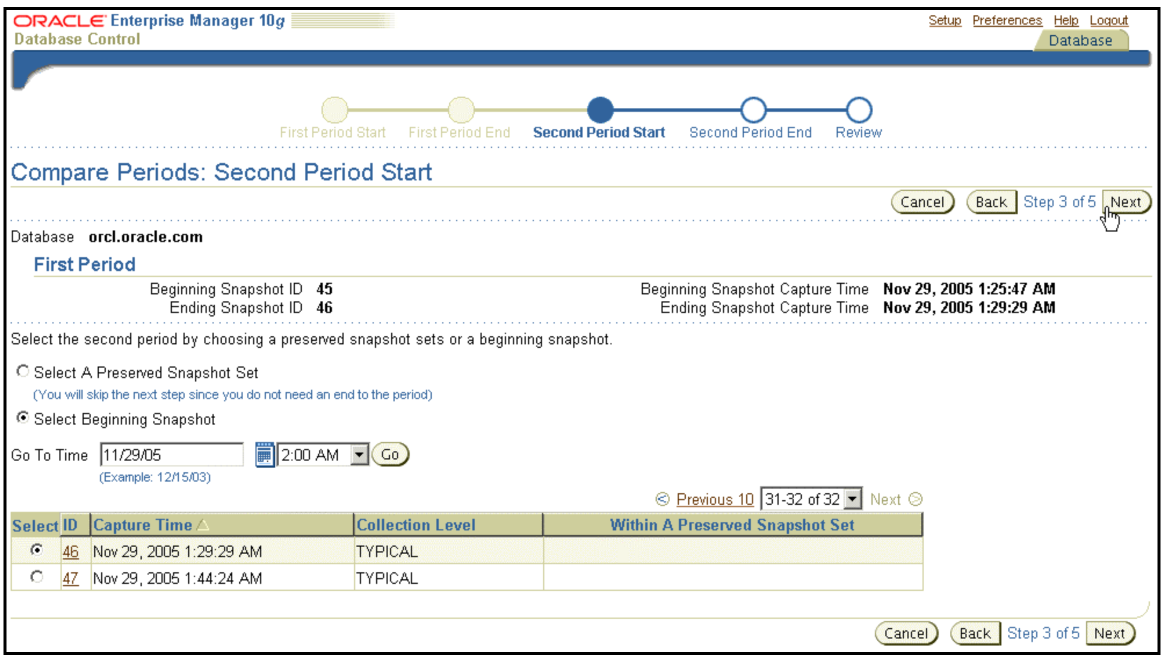

ORACLE Enterprise Manager 10g
Database Control
Setup Pref
Database Instance: orcl.oracle.com > Automatic Workload Repository > Snapshots
Confirmation
A snapshot has been created successfully
Snapshots
A snapshot is a collection of database statistics at a single point in time. You can use the information in snapshots to diagnose database problems.
Page Refreshed Nov 29, 2005
Select Beginning Snapshot
Go To Time 11/29/05 2:00 AM Go
(Example: 12/15/03)
Create
Delete Actions Compare Periods Go
Previous 25 26-32 of 32
Select ID Capture Time Collection Level Within A Preserved Snapshot Set
41 Nov 28, 2005 9:52:11 AM TYPICAL
42 Nov 28, 2005 9:56:51 AM TYPICAL
43 Nov 28, 2005 9:58:11 AM TYPICAL
44 Nov 28, 2005 10:32:58 AM TYPICAL
45 Nov 29, 2005 1:25:47 AM TYPICAL
46 Nov 29, 2005 1:29:29 AM TYPICAL
47 Nov 29, 2005 1:44:24 AM TYPICAL

Solutions for Practice 6-1: Using AWR-Based Tools (continued)

3. On the Compare Periods: First Period End page, select your second AWR snapshot (46 in the screenshot), and click Next.

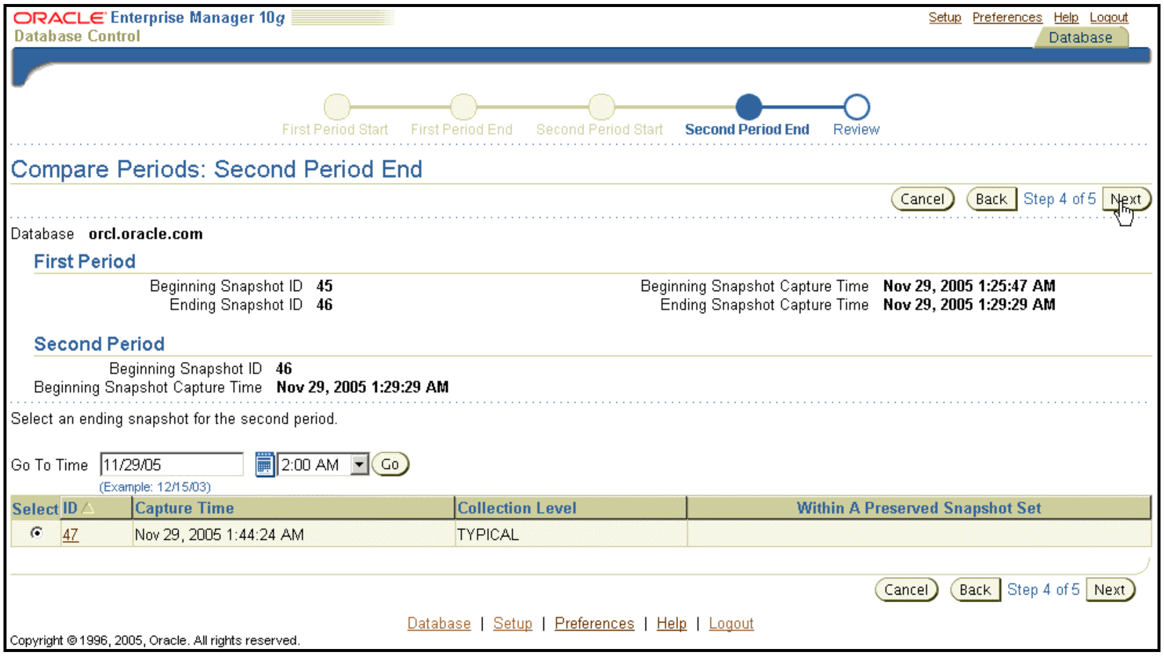


4. On the Compare Periods: Second Period Start page, select your second AWR snapshot (46 in the screenshot), and click Next.

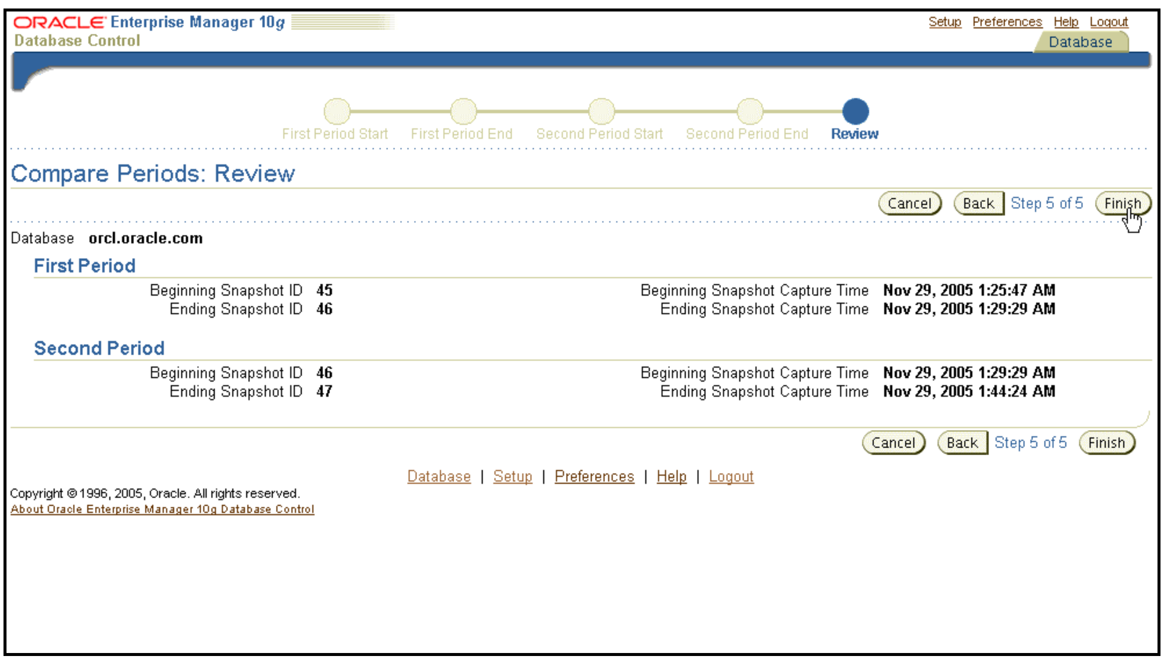


Solutions for Practice 6-1: Using AWR-Based Tools (continued)

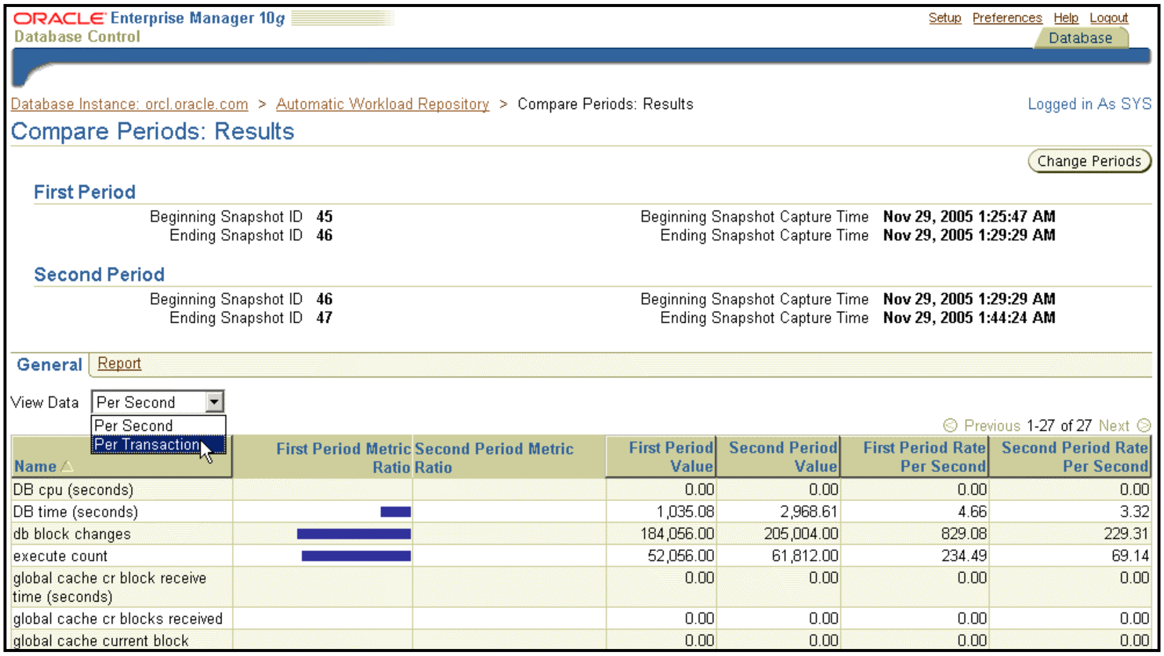
5. On the Compare Periods: Second Period End page, select your last AWR snapshot (47 in the screenshot), and click Next.



6. On the Compare Periods: Review page, click Finish.



7. On the Compare Periods: Results page, select Per Transaction from the View Data drop-down list on the General tabbed page.



ORACLE Enterprise Manager 10g
Database Control

Setup Preferences Help Logout
Database

Database Instance: orcl.oracle.com > Automatic Workload Repository > Compare Periods: Results

Logged in As SYS

Compare Periods: Results

Change Periods

First Period

Beginning Snapshot ID 45 Beginning Snapshot Capture Time Nov 29, 2005 1:25:47 AM
Ending Snapshot ID 46 Ending Snapshot Capture Time Nov 29, 2005 1:29:29 AM

Second Period

Beginning Snapshot ID 46 Beginning Snapshot Capture Time Nov 29, 2005 1:29:29 AM
Ending Snapshot ID 47 Ending Snapshot Capture Time Nov 29, 2005 1:44:24 AM

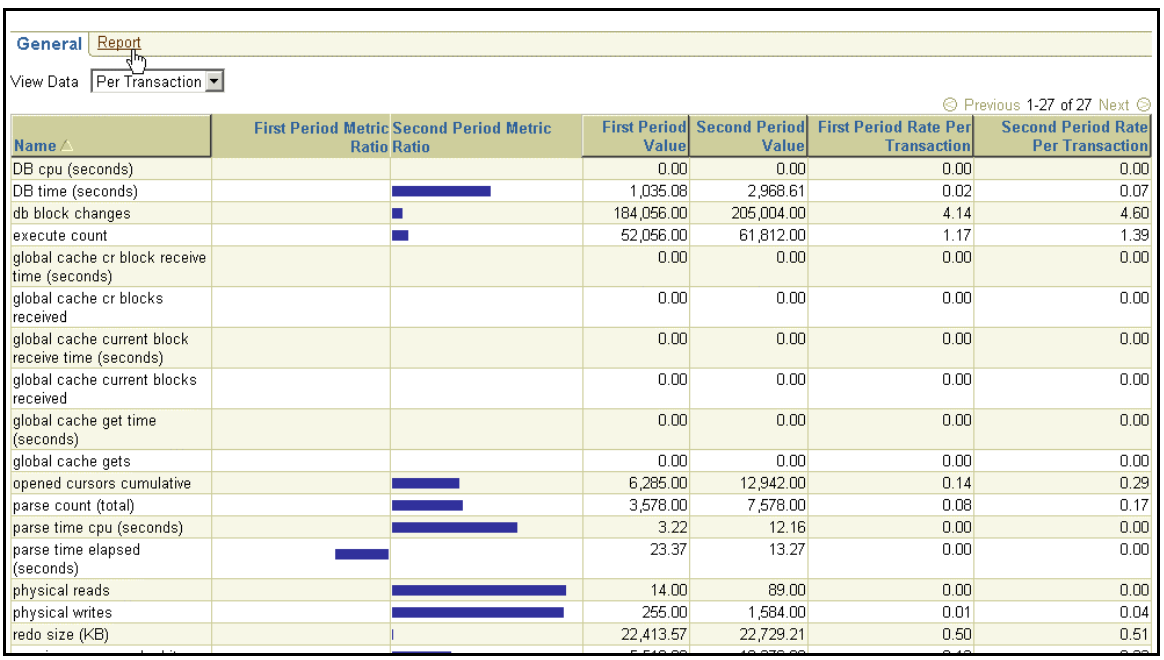
General Report

View Data Per Second
Per Second
Per Transaction

Previous 1-27 of 27 Next

Name	First Period Metric Ratio	Second Period Metric Ratio	First Period Value	Second Period Value	First Period Rate Per Second	Second Period Rate Per Second
DB cpu (seconds)			0.00	0.00	0.00	0.00
DB time (seconds)			1,035.08	2,968.61	4.66	3.32
db block changes			184,056.00	205,004.00	829.08	229.31
execute count			52,056.00	61,812.00	234.49	69.14
global cache cr block receive time (seconds)			0.00	0.00	0.00	0.00
global cache cr blocks received			0.00	0.00	0.00	0.00
global cache current block			0.00	0.00	0.00	0.00

8. You can see the performance differences between the two analyzed periods. (Your report will vary from the following example.) Click Report.



General Report

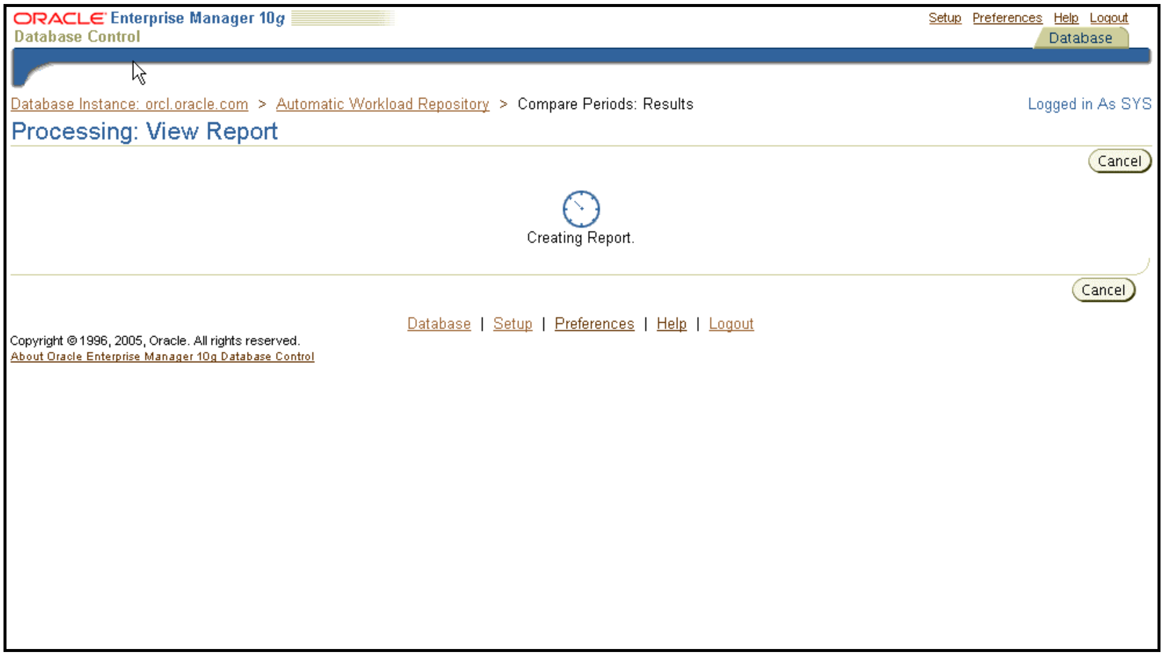
View Data Per Transaction

Previous 1-27 of 27 Next

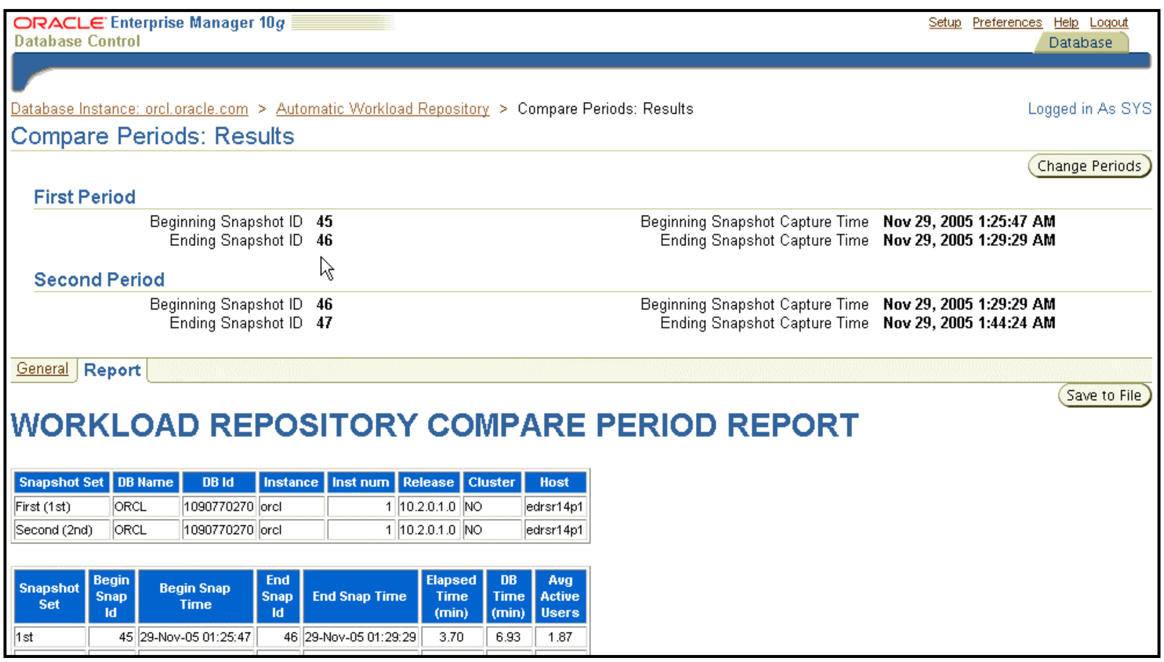
Name	First Period Metric Ratio	Second Period Metric Ratio	First Period Value	Second Period Value	First Period Rate Per Transaction	Second Period Rate Per Transaction
DB cpu (seconds)			0.00	0.00	0.00	0.00
DB time (seconds)			1,035.08	2,968.61	0.02	0.07
db block changes			184,056.00	205,004.00	4.14	4.60
execute count			52,056.00	61,812.00	1.17	1.39
global cache cr block receive time (seconds)			0.00	0.00	0.00	0.00
global cache cr blocks received			0.00	0.00	0.00	0.00
global cache current block receive time (seconds)			0.00	0.00	0.00	0.00
global cache current blocks received			0.00	0.00	0.00	0.00
global cache get time (seconds)			0.00	0.00	0.00	0.00
global cache gets			0.00	0.00	0.00	0.00
opened cursors cumulative			6,285.00	12,942.00	0.14	0.29
parse count (total)			3,578.00	7,578.00	0.08	0.17
parse time cpu (seconds)			3.22	12.16	0.00	0.00
parse time elapsed (seconds)			23.37	13.27	0.00	0.00
physical reads			14.00	89.00	0.00	0.00
physical writes			255.00	1,584.00	0.01	0.04
redo size (KB)			22,413.57	22,729.21	0.50	0.51

9. The Processing page appears. Wait until the report is generated.

Solutions for Practice 6-1: Using AWR-Based Tools (continued)



10. On the Report tabbed page, you can now see the Workload Repository Compare Period Report.



11. Scroll down to the Top 5 Timed Events section. You can see the difference between the two periods for the `buffer busy waits` event.

Top 5 Timed Events

1st					2nd				
Event	Waits	Time(s)	Percent Total DB Time	Wait Class	Event	Waits	Time(s)	Percent Total DB Time	Wait Class
buffer busy waits	581	129.3	31.08	Concurrency	CPU time		80.5	21.50	
CPU time		52.3	12.57		buffer busy waits	133	37.2	9.94	Concurrency
latch: In memory undo latch	269	46.8	11.25	Concurrency	log file parallel write	11,930	27.1	7.23	System I/O
log file parallel write	8,466	35.5	8.52	System I/O	*latch: redo copy	131	26.2	6.99	Configuration
*latch free	19	14.6	3.52	Other	latch: In memory undo latch	68	11.3	3.01	Concurrency
-latch: redo copy	6	3.6	.88	Configuration	-latch free	34	7.3	1.95	Other

12. Scroll down to the Buffer Wait Statistics section. Here also, you can clearly see the difference.

Buffer Wait Statistics

- Ordered by 'Diff' column of 'Wait Time % of DB time' descending

	Wait Time % DB Time			Total Wait Time (s)		# Waits		Avg Wait Time (ms)		
Class	1st	2nd	Diff	1st	2nd	1st	2nd	1st	2nd	%Diff
data block	0.32	0.10	-0.22	133.46	36.01	530	124	2.52	2.90	15.08
undo header	0.00	0.00	0.00	0.15	1.64	7	5	0.21	3.28	1,461.90
1st level bmb	0.00	0.00	0.00	0.00	0.61	0	4	0.00	1.53	100.00
segment header	0.00	0.00	0.00	0.00	0.00	44	0	0.00	0.00	0.00

13. Scroll down to the Top 5 Segments by Buffer Busy Waits section.

Top 5 Segments by Buffer Busy Waits

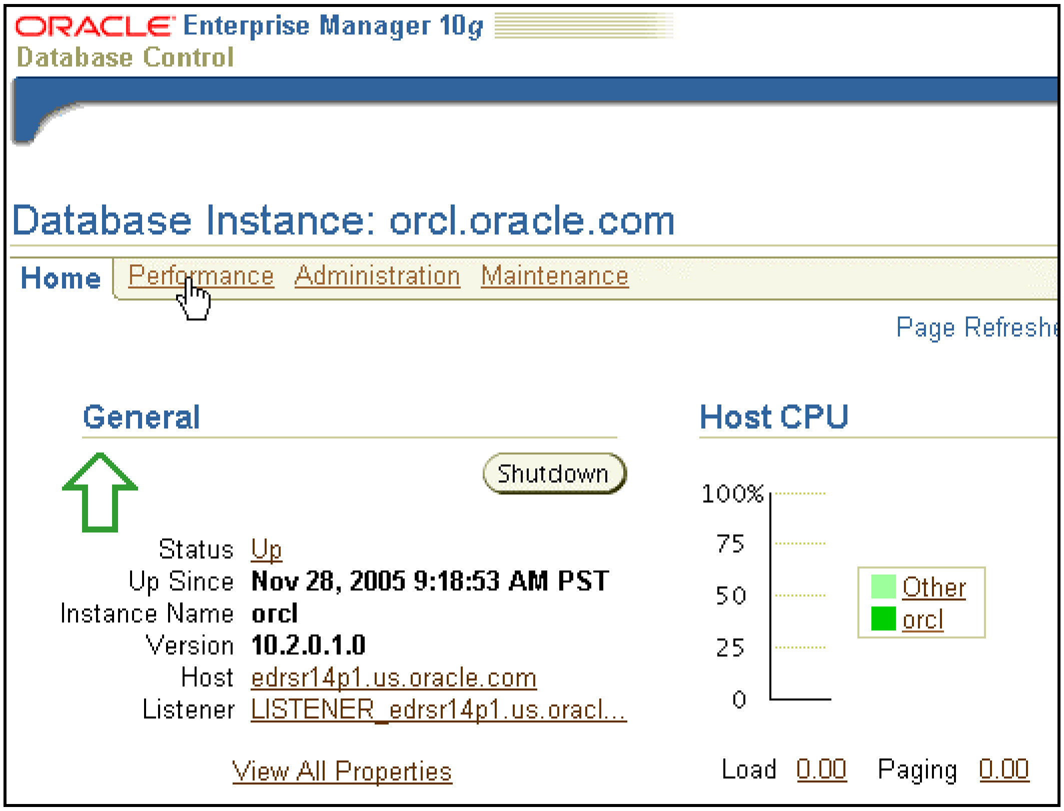
- Ordered by 'Diff' column of '% of Total Buffer Busy Waits'
- 'N/A' indicates no data was captured for the segment in the period

					% of Total Buffer Waits			Buffer Busy Waits	
Owner	Tablespace	Object Name	Subobject Name	Type	1st	2nd	Diff	1st	2nd
SPC	TBSSPC	SPCT		TABLE	98.80	96.24	-2.55	574	128

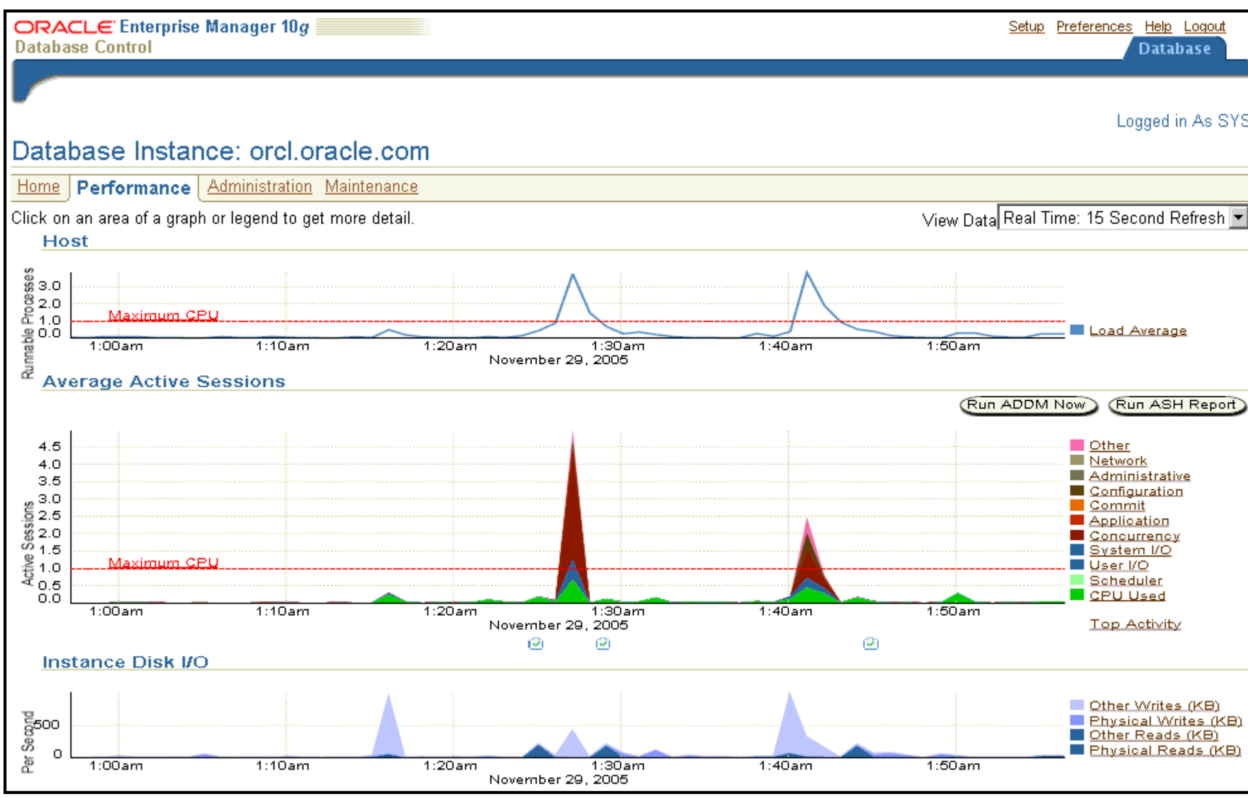
10) Generate the ASH Report from the Top Activity page for the hottest period.

Answer:

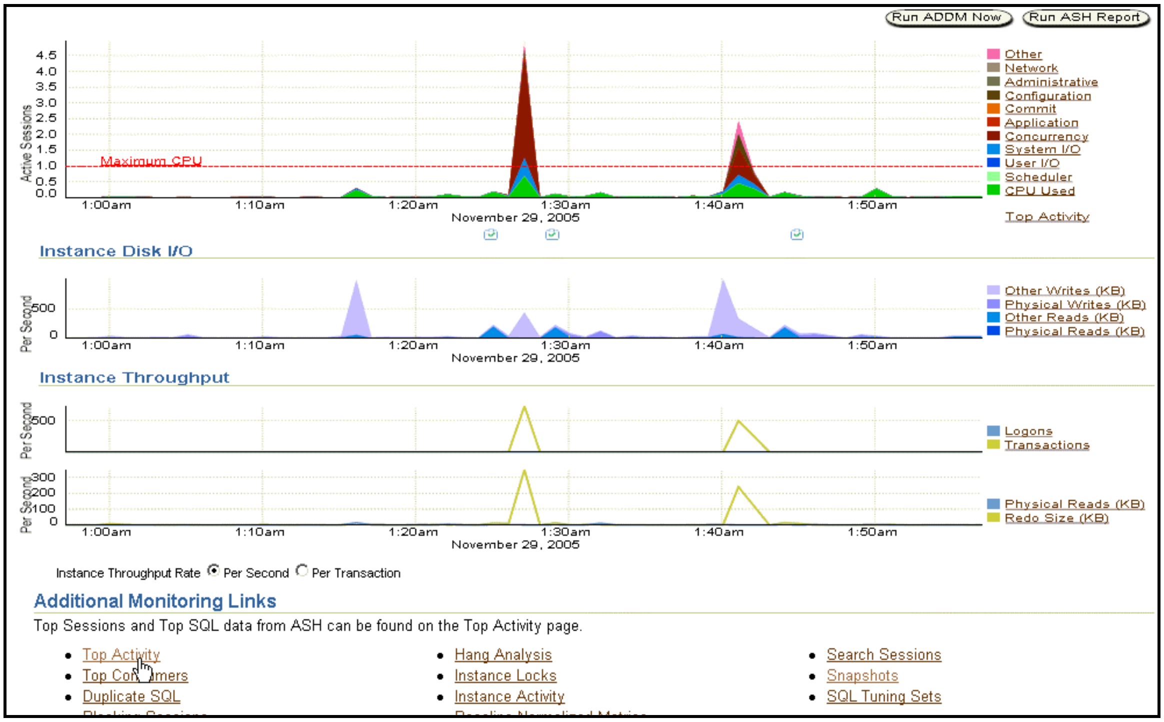
1. You can use the `ashrpt.sql` script located in the `$ORACLE_HOME/rdbms/admin` directory to generate the corresponding ASH Report. The solution in this document uses Enterprise Manager. On the Database Instance Home page, click Performance.



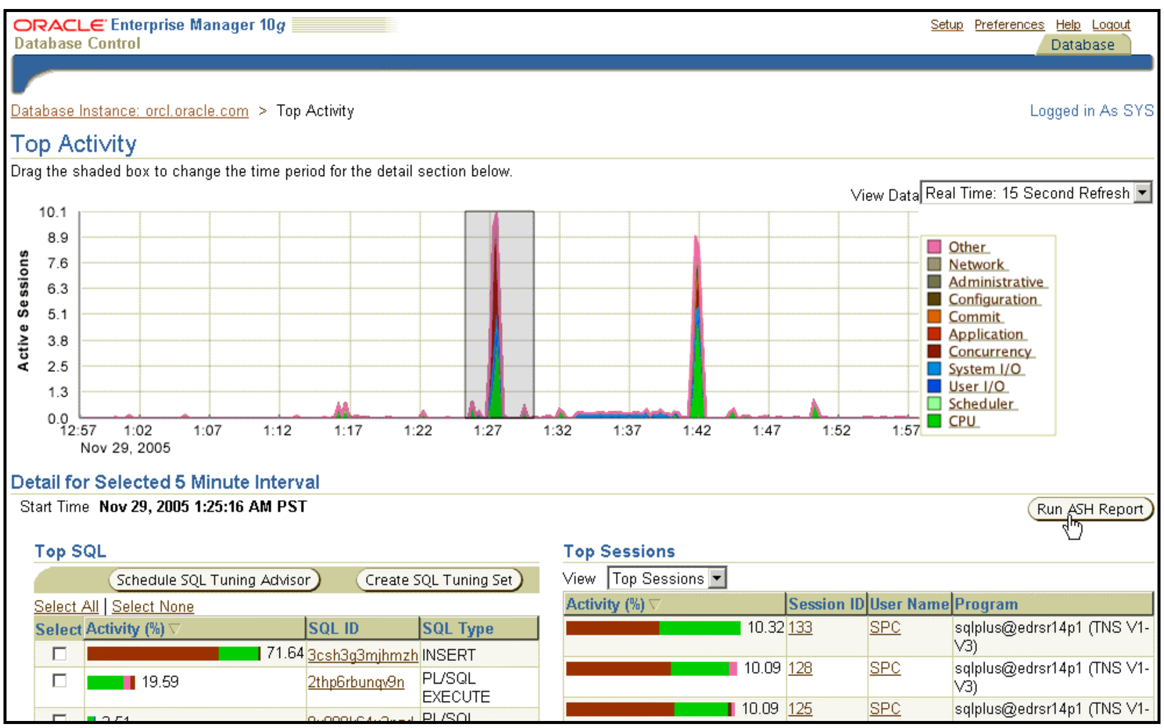
2. On the Performance page, you can clearly see the two peaks corresponding to the two periods.



3. Scroll down to the Additional Monitoring Links section, and click Top Activity.

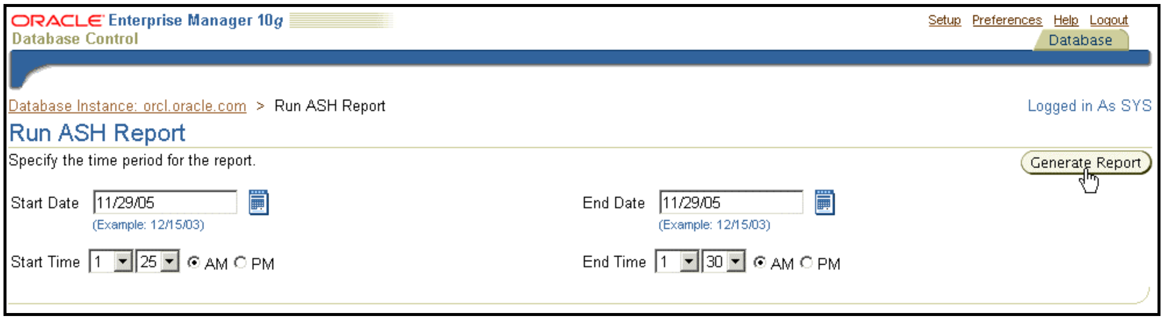


4. On the Top Activity page, slide the time window to the biggest peak on the Active Sessions graph, and click Run ASH Report.

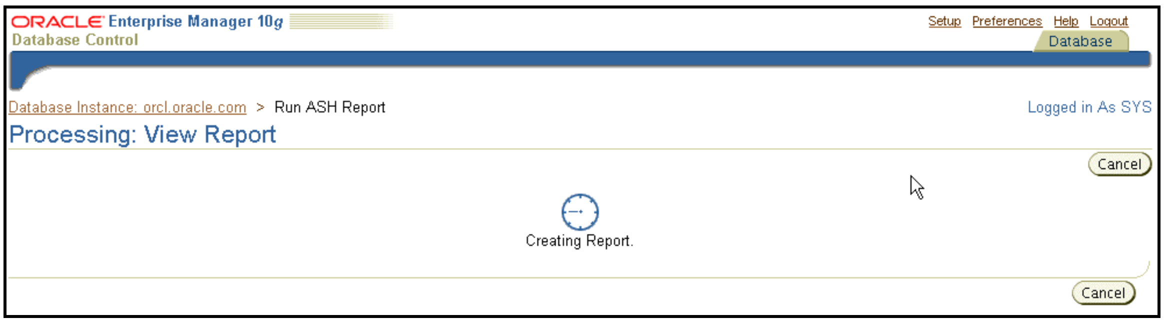


5. On the Run ASH Report page, click Generate Report.

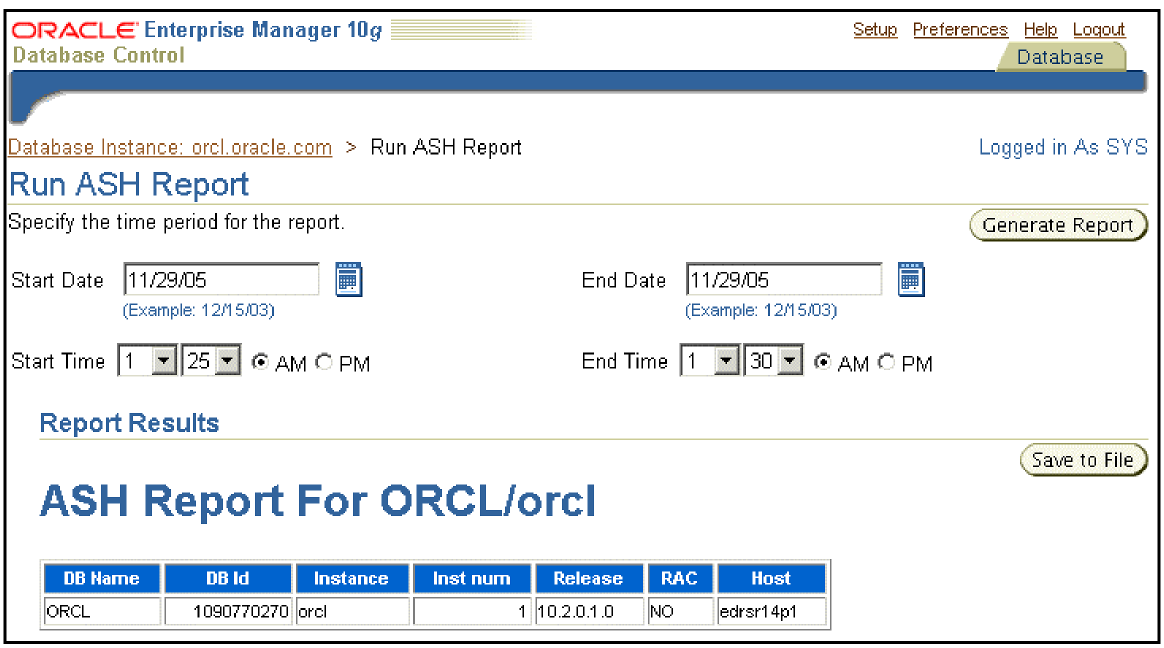
Solutions for Practice 6-1: Using AWR-Based Tools (continued)



6. The Processing page appears. Wait until the report is generated.



7. You are then directed to the Run ASH Report page, from where you can now see the Report Results.



8. From the first section, you can get various general details from the report.

ASH Report For ORCL/orcl

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
ORCL	1090770270	orcl	1	10.2.0.1.0	NO	edrsr14p1

CPUs	SGA Size	Buffer Cache	Shared Pool	ASH Buffer Size
1	272M (100%)	160M (58.8%)	95M (35.0%)	2.0M (0.7%)

	Sample Time	Data Source
Analysis Begin Time:	29-Nov-05 01:25:16	V$ACTIVE_SESSION_HISTORY
Analysis End Time:	29-Nov-05 01:30:16	V$ACTIVE_SESSION_HISTORY
Elapsed Time:	5.0 (mins)	
Sample Count:	436	
Average Active Sessions:	1.45	
Avg. Active Session per CPU:	1.45	
Report Target:	None specified	

9. Scroll down to the Top User Events section.

Top User Events

Event	Event Class	% Activity	Avg Active Sessions
CPU + Wait for CPU	CPU	38.30	0.56
buffer busy waits	Concurrency	27.52	0.40
latch: In memory undo latch	Concurrency	13.07	0.19
latch free	Other	3.44	0.05
latch: cache buffers chains	Concurrency	3.21	0.05

10. Scroll down to the Top SQL Command Types section.

Top SQL Command Types

- 'Distinct SQLIDs' is the count of the distinct number of SQLIDs with the given SQL Command Type found over all the ASH samples in the analysis period

SQL Command Type	Distinct SQLIDs	% Activity	Avg Active Sessions
INSERT	1	56.65	0.82
PL/SQL EXECUTE	6	19.95	0.29

11. Scroll down to the Top SQL Statements section.

Top SQL Statements

SQL ID	Planhash	% Activity	Event	% Event	SQL Text
3csh3g3mjhmzh		56.19	buffer busy waits	27.29	INSERT INTO SPCT VALUES (NULL,...
		56.19	CPU + Wait for CPU	12.84	INSERT INTO SPCT VALUES (NULL,...
		56.19	latch: In memory undo latch	12.61	INSERT INTO SPCT VALUES (NULL,...
2thp6rbunqv9n		15.37	CPU + Wait for CPU	11.47	declare t number; begin for t ...
		15.37	latch free	2.29	declare t number; begin for t ...
8u809k64x3nzd		2.75	CPU + Wait for CPU	2.52	begin DBMS_WORKLOAD_REPOSITORY...

12. Scroll down to the Activity Over Time section.

Activity Over Time

- Analysis period is divided into smaller time slots
- Top 3 events are reported in each of those slots
- 'Slot Count' shows the number of ASH samples in that slot
- 'Event Count' shows the number of ASH samples waiting for that event in that slot
- '% Event' is 'Event Count' over all ASH samples in the analysis period

Slot Time (Duration)	Slot Count	Event	Event Count	% Event
01:25:16 (44 secs)	13	CPU + Wait for CPU	12	2.75
		db file sequential read	1	0.23
01:26:00 (1.0 min)	7	CPU + Wait for CPU	3	0.69
		log file parallel write	2	0.46
		null event	2	0.46
01:27:00 (1.0 min)	407	CPU + Wait for CPU	149	34.17
		buffer busy waits	120	27.52
		latch: In memory undo latch	57	13.07
01:28:00 (1.0 min)	1	CPU + Wait for CPU	1	0.23
01:29:00 (1.0 min)	8	CPU + Wait for CPU	7	1.61
		log file parallel write	1	0.23

Practice Solutions for Lesson 7

The goal of this practice is to identify the sources of performance issues.

Solutions for Practice 7-1: *Using Enterprise Manager to Identify OS Issues*

The workload generator in this practice starts an application load on the `prod` database and an OS load.

1) In a terminal window, change directory to `/home/oracle/workshops`. Start the workload generator with the command: `./workgen 7 1`. Verify that the workload generator has started the workload processes by executing the `ps` command.

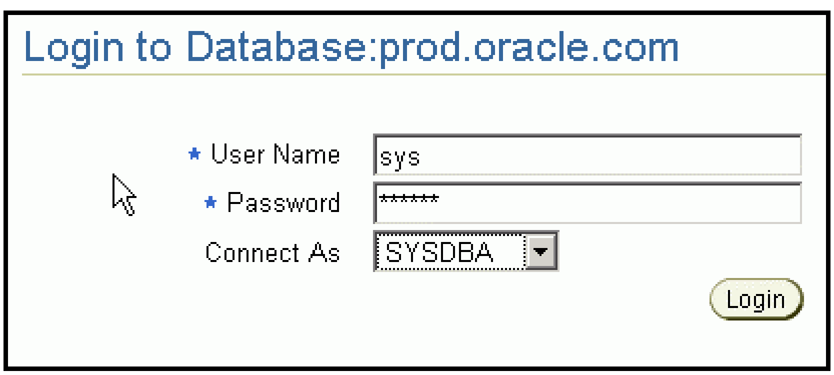
   Answer:

```
$ cd /home/oracle/workshops
$ . oraenv
ORACLE_SID = [orcl] ? prod

$ ./workgen 7 1
$ ps
  PID TTY          TIME CMD
23697 pts/4    00:00:00 bash
32415 pts/4    00:00:00 wkld_shell.sh
32419 pts/4    00:00:00 wkld_sess.sh
32420 pts/4    00:00:00 sqlplus
32421 pts/4    00:00:00 wkld_sess.sh
32422 pts/4    00:00:00 sqlplus
32423 pts/4    00:00:00 wkld_sess.sh
32425 pts/4    00:00:00 wkld_sess.sh
32426 pts/4    00:00:00 sqlplus
32427 pts/4    00:00:00 wkld_sess.sh
32428 pts/4    00:00:00 sqlplus
32429 pts/4    00:00:00 wkld_sess.sh
32430 pts/4    00:00:00 sqlplus
32433 pts/4    00:00:00 sqlplus
32446 pts/4    00:00:00 ps
```

2) Use Enterprise Manager to determine the source of the loads. In the browser, enter the URL http://localhost:5500/em to view the `prod` database information.

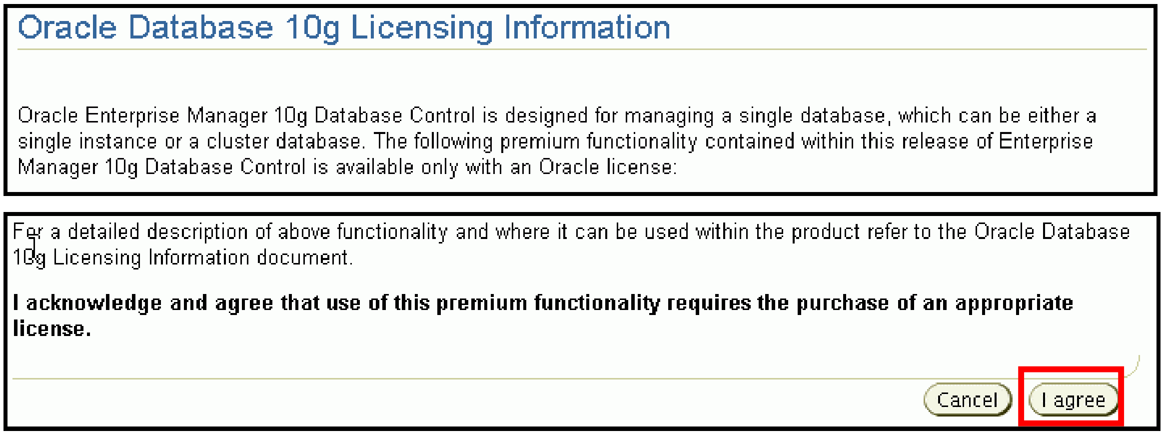
   Answer:

   1. On the login page, enter `SYS` as the username and `oracle` as password, select `SYSDBA` from the Connect As drop-down list, and then click Login..



Solutions for Practice 7-1: Using Enterprise Manager to Identify OS Issues (continued)

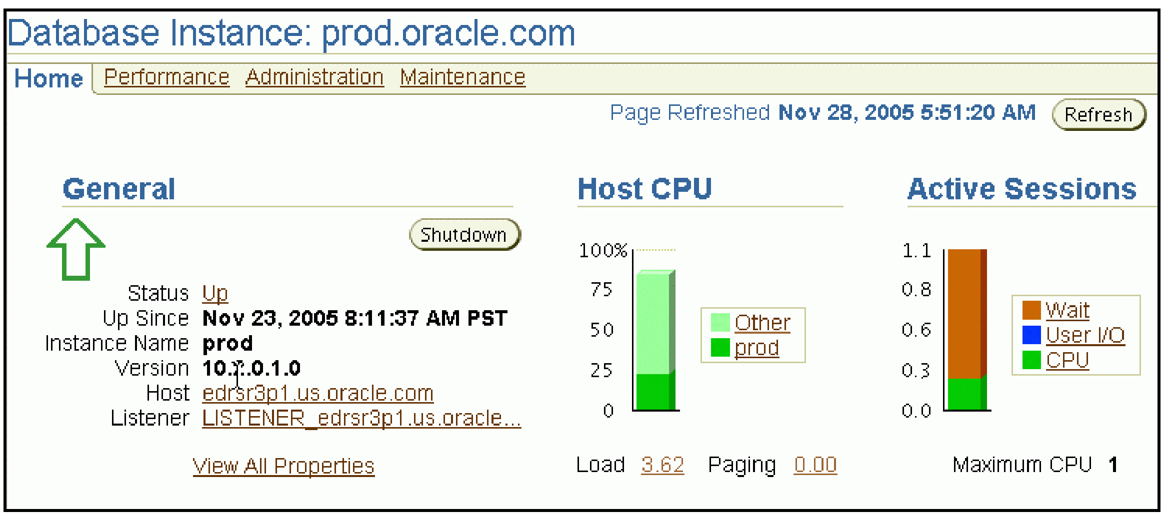
2. If this is the first time you have accessed this database through EM with this login, then the license information page will appear. Scroll to the bottom of the page and click "I agree" to accept the license.



3) Find the OS process that is consuming the most CPU.

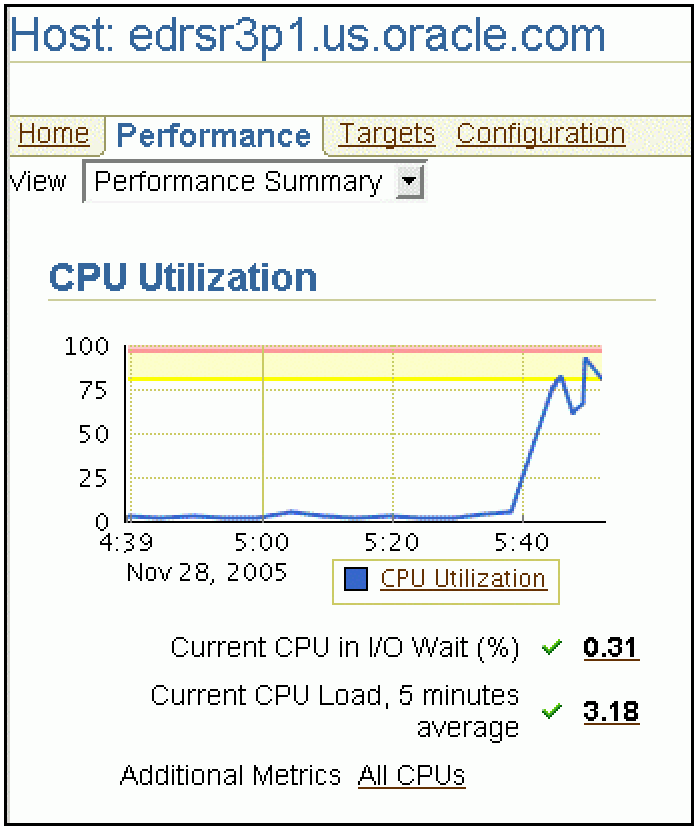
Answer:

1. The Database Home page for the `prod.oracle.com` database appears. Observe the Host CPU measurement.

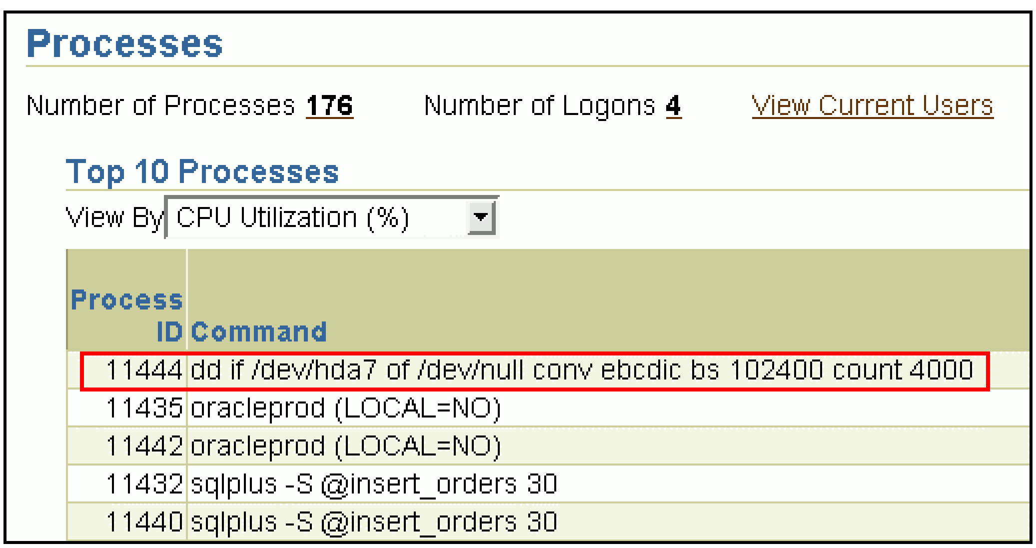


2. A significant percentage of the Host CPU is dedicated to **Other** processes. Click the Other link to view the Performance Summary page.

Solutions for Practice 7-1: Using Enterprise Manager to Identify OS Issues (continued)



3. Scroll down to view the Top 10 Processes by specifying View By CPU Utilization (%).
   Note: Refresh this view a few times until an OS process (not owned by `oracle`) appears. The `dd` process should appear. It may not be the top process.



Solutions for Practice 7-1: Using Enterprise Manager to Identify OS Issues (continued)

4) Stop the workload by removing the `runload` file from the `/home/oracle/workshops` directory.

   Answer:

```
cd /home/oracle/workshops
rm runload
rm: remove regular empty file `runload'? y
```

Solutions for Practice 7-2: *Using Enterprise Manager to Identify Database Performance Issues*

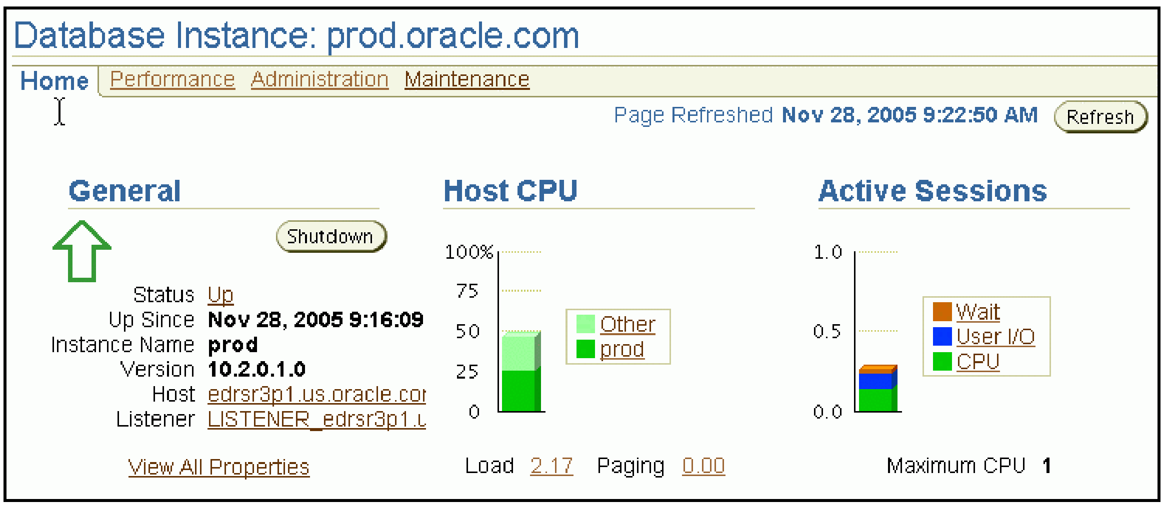
The goal of this practice is to use the EM performance pages to identify performance issues associated with an application workload. You will generate the same workload as in Practice 7-1, without the OS workload.

1) Start the workload generator with the command: `./workgen 7 2`.
   Navigate to the Database Home page in EM for the `prod` database. Use the URL http://localhost:5500/em. Allow the workload generator to run for at least 5 minutes.

   Answer:

   ```
   $ cd /home/oracle/workshops
   $ . oraenv
   ORACLE_SID = [orcl] ? prod

   $ ./workgen 7 2
   $
   ```

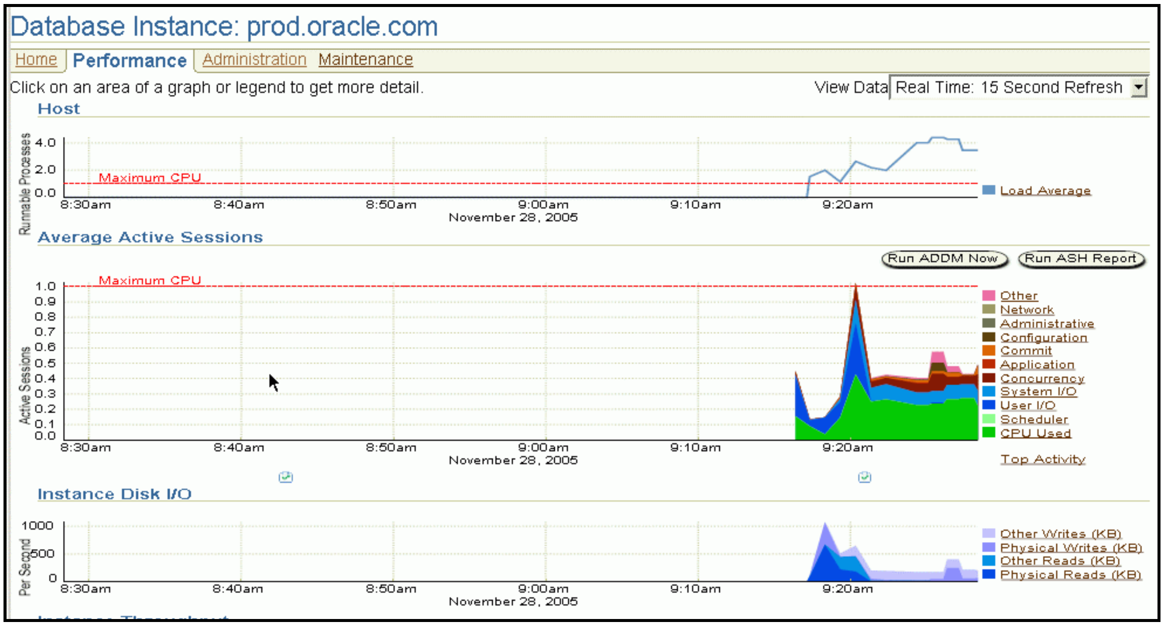
   Navigate to the `prod` database's Home page. Observe the values of Host CPU.

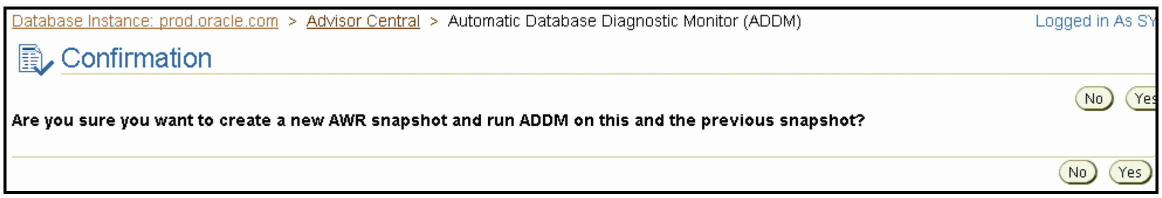
   

2) After about 5 minutes, take an ADDM snapshot.

   Answer:

   Navigate to the Performance page and click the Run ADDM Now button. When the Confirmation page appears, click Yes.

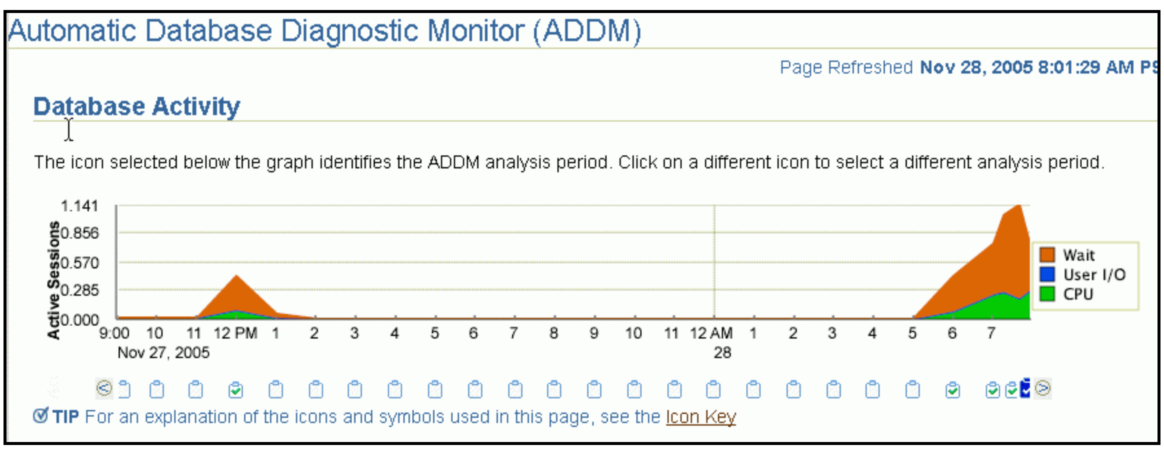




3) Find the top application performance issues by using ADDM.

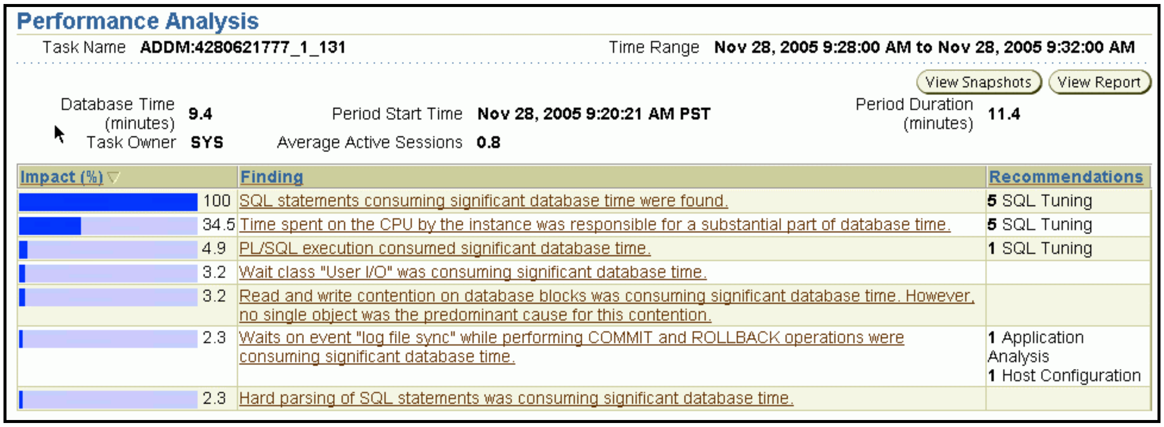
Answer:

1. The ADDM report page is displayed.

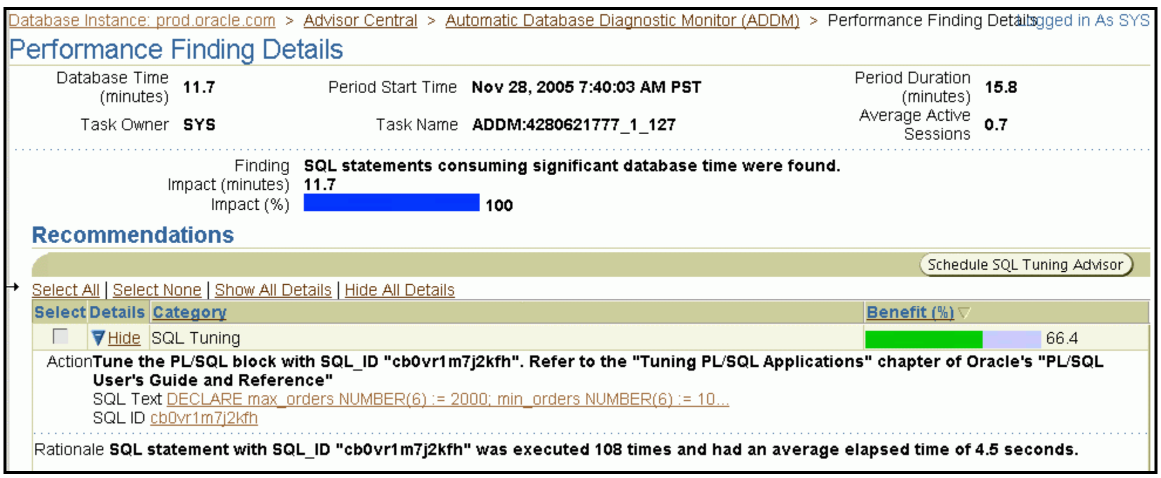


2. Scroll down to the Performance Analysis section. The number and order of findings in your results may vary from the example. Notice the finding “SQL statements consuming significant database time were found.” In a well-tuned database, this is often the first finding. SQL statements are doing the work in the database. Further investigation is still warranted.

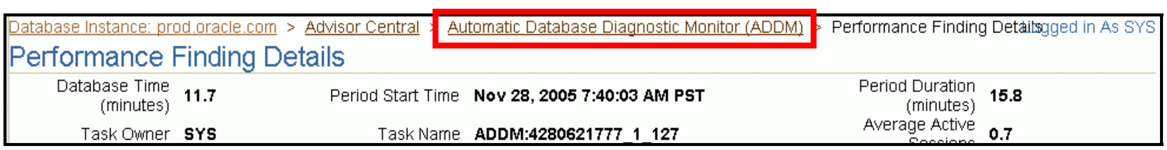
Solutions for Practice 7-2: Using Enterprise Manager to Identify Database Performance Issues (continued)



3. Click the finding to view the recommendations. In this case, the primary SQL statement consuming time is a PL/SQL block that is part of the workload generator. This situation would be similar to having an application that you are not allowed to change.



4. Return to the ADDM page to view Performance Analysis by using the path at the top of the page. Click Automatic Database Diagnostics Monitor (ADDM).



5. Other findings are possible that may not be shown. One may be “Time spent on the CPU by the instance was responsible for a substantial part of database time.” This finding will show a set of SQL statements. If the application is available to be changed, tuning the SQL statements generally gives the most performance benefits. If the statements are available for tuning, you can run the SQL Tuning Advisor. If the statements cannot be changed, implement SQL Profiles. SQL Profiles will often help performance without changing the code, by providing the optimizer with additional statistics.

4) Take a Statspack snapshot.

   Answer:

   Execute the Statspack snapshot procedure.

```
$ sqlplus perfstat/perfstat

SQL*Plus: Release 10.2.0.1.0 - Production on Mon Nov 28
09:36:27 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> exec statspack.snap;

PL/SQL procedure successfully completed.

SQL> exit
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
```

5) Stop the workload by removing the `runload` file from the `/home/oracle/workshops` directory.

   Answer:

```
$ cd /home/oracle/workshops
$ rm runload
rm: remove regular empty file `runload'? y
```

6) Find the top database issues by using Statspack.

   Answer:

   Run a Statspack report on the last two snapshots. The number of snapshots listed will vary.
   Observe the Top 5 Timed Events. The top wait events correspond to the ADDM findings. When `CPU time` is in the Top 5 Events, the next step is to determine what is consuming the CPU time. These are usually SQL statements and can be found in the SQL section of the Statspack report. Another event that may be seen in this report is `log file parallel write`. This corresponds to “Wait class user IO” in the ADDM findings that includes several I/O events. When tuning from the Statspack report, start with the Top 5 Timed Events.

Solutions for Practice 7-2: Using Enterprise Manager to Identify Database Performance Issues (continued)

```
$ sqlplus perfstat/perfstat

SQL*Plus: Release 10.2.0.1.0 - Production on Mon Nov 28 09:36:27
2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 -
Production
With the Partitioning, OLAP and Data Mining options

SQL>@?/rdbms/admin/spreport.sql

Current Instance
~~~~~~~~~~~~~~~~

   DB Id    DB Name      Inst Num Instance
----------- ------------ -------- ------------
 4280621777 PROD                1 prod



Instances in this Statspack schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   DB Id    Inst Num DB Name      Instance     Host
----------- -------- ------------ ------------ ------------
 4280621777        1 PROD         prod         edrsr3p1

Using 4280621777 for database Id
Using          1 for instance number


Specify the number of days of snapshots to choose from
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Entering the number of days (n) will result in the most recent
(n) days of snapshots being listed.  Pressing <return> without
specifying a number lists all completed snapshots.



Listing all Completed Snapshots

                                                           Snap
Instance     DB Name        Snap Id   Snap Started    Level
Comment
------------ ------------ --------- ----------------- ----- ----
prod         PROD                 1 28 Nov 2005 07:41     5
                                 11 28 Nov 2005 07:56     5
                                 21 28 Nov 2005 08:44     5

                                 31 28 Nov 2005 09:20     5
                                 32 28 Nov 2005 09:20     5
                                 33 28 Nov 2005 09:36     5
```

```
Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap: 32
Begin Snapshot Id specified: 32

Enter value for end_snap: 33
End   Snapshot Id specified: 33



Specify the Report Name
~~~~~~~~~~~~~~~~~~~~~~~
The default report file name is sp_32_33.  To use this name,
press <return> to continue, otherwise enter an alternative.

Enter value for report_name: w72_sprpt.txt

STATSPACK report for

Database    DB Id    Instance  Inst Num Startup Time    Release     RAC
~~~~~~~~ ----------- --------- -------- --------------- ----------- ---
         4280621777 prod              1 28-Nov-05 09:16 10.2.0.1.0  NO

Host Name:  edrsr3p1    Num CPUs: 1      Phys Memory(MB): 1,003
~~~~

Snapshot       Snap Id     Snap Time      Sessions Curs/Sess
Comment
~~~~~~~~    ---------- ------------------ -------- ---------
Begin Snap:         32 28-Nov-05 09:20:28       25       7.8
  End Snap:         33 28-Nov-05 09:36:40       25      10.1
   Elapsed:               16.20 (mins)

…

Top 5 Timed Events                                      Avg %Total
~~~~~~~~~~~~~~~~~~                                     wait   Call
Event                             Waits    Time (s)   (ms)   Time
------------------------- ------------ ----------- ------ ------
CPU time                                        260          55.4
log file parallel write         33,547           90      3   19.2
buffer busy waits                  222           26    115    5.5
log file sync                      354           18     51    3.9
latch: library cache               117           11     97    2.4
    ---------------------------------------------------------
…

End of Report ( w72_sprpt.txt )

SQL> exit
Disconnected from Oracle Database 10g Enterprise Edition Release
10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
```

Practice Solutions for Lesson 8

The goal of this practice is to observe and correct various shared pool issues.

Solutions for Practice 8-1: *Tuning a Hard Parse Workload*

In this section, you will run a hard parse workload, view the characteristic report events, and fix the problem. Set up the scenario by running the `./setup 8 1` script. Run the workload against the `prod` database. The workload is in the `/home/oracle/workshops` directory.

1) Set the environment variables for the `prod` database. Run the setup script.

   Answer:

   Change directory to the `/home/oracle/workshops` directory. Execute the `oraenv` script to set the environment variables.

```
$ cd /home/oracle/workshops
$ . oraenv
ORACLE_SID = [prod] ? prod
$ ./setup 8 1
Database closed.
Database dismounted.
ORACLE instance shut down.

ORACLE instance started.

Total System Global Area  314572800 bytes
Fixed Size                  1219160 bytes
Variable Size              96470440 bytes
Database Buffers          213909504 bytes
Redo Buffers                2973696 bytes
Database mounted.
Database opened.
Finished setup 8-1
$
```

2) Run the workload generator: `./workgen 8 1`.
   The workload generator runs runs a Statspack snapshot at the beginning and end. It also runs an ADDM snapshot at the beginning.

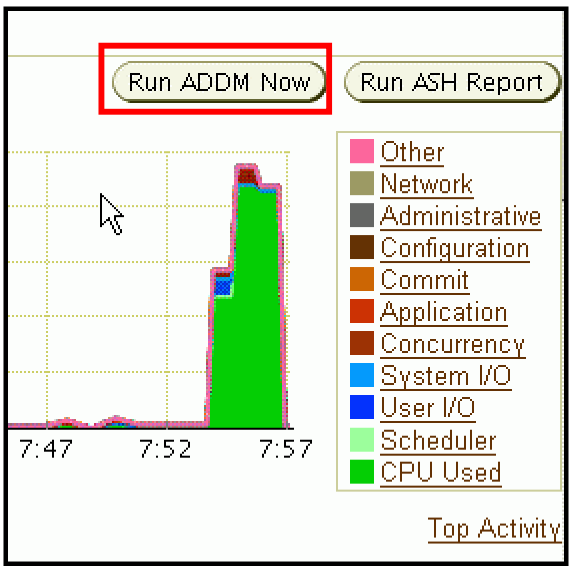
   Answer:

   1. Run the workload generator.

```
$ ./workgen 8 1

SQL*Plus: Release 10.2.0.1.0 - Production on Fri Jan 27
06:27:11 2006

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected.

PL/SQL procedure successfully completed.

Connected.
```

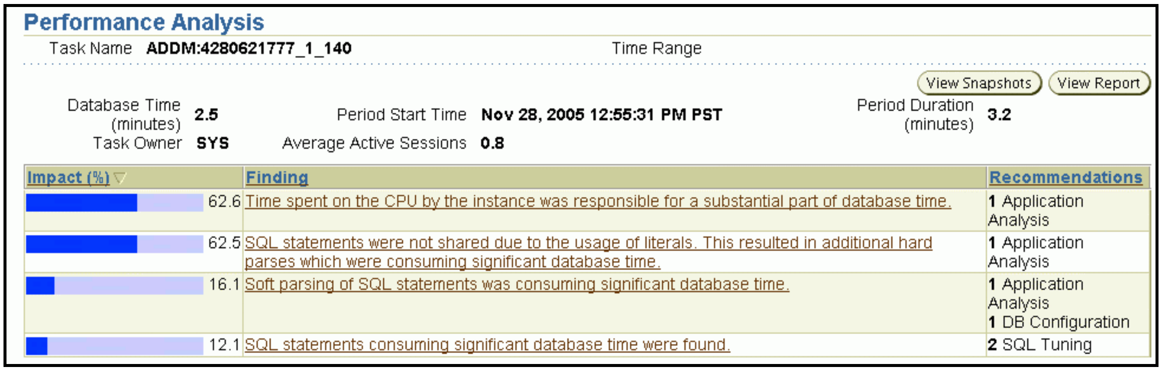
```
PL/SQL procedure successfully completed.

Connected.

System altered.

Connected.

PL/SQL procedure successfully completed.

Connected.
Connected.
Connected.
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

$
```

2. Navigate to the database performance page and wait until the workload begins to drop, and then click Run ADDM Now. Click Yes on the Confirmation page.

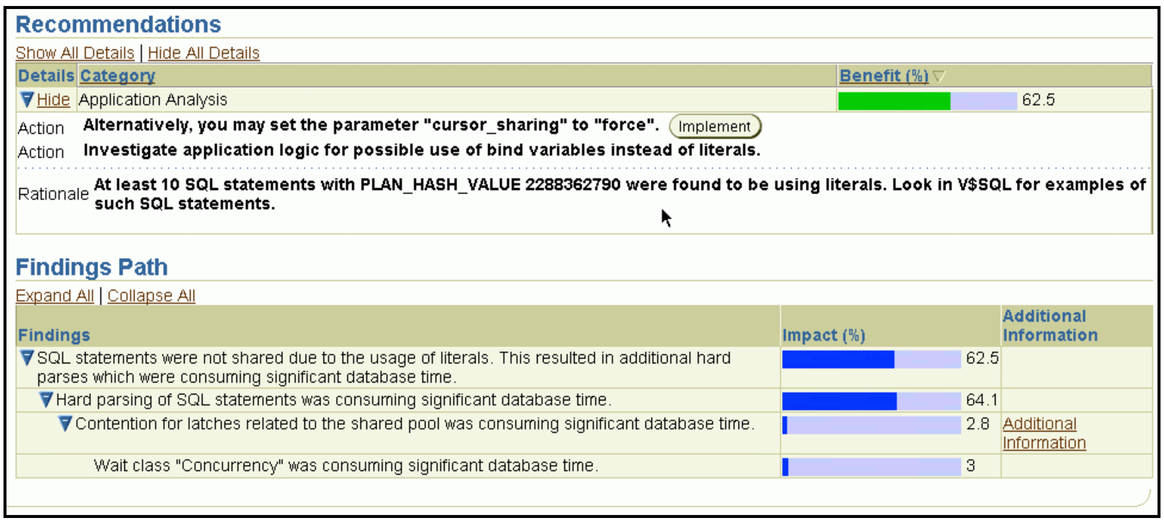


Solutions for Practice 8-1: Tuning a Hard Parse Workload (continued)

3) View the ADDM report.



4) Drill down to the recommendations. Click the finding: "SQL Statements were not shared due to the use of literals. This resulted in additional hard parses which were consuming significant database time."



5) Based on the ADDM report, what is the recommended action?

Answer:

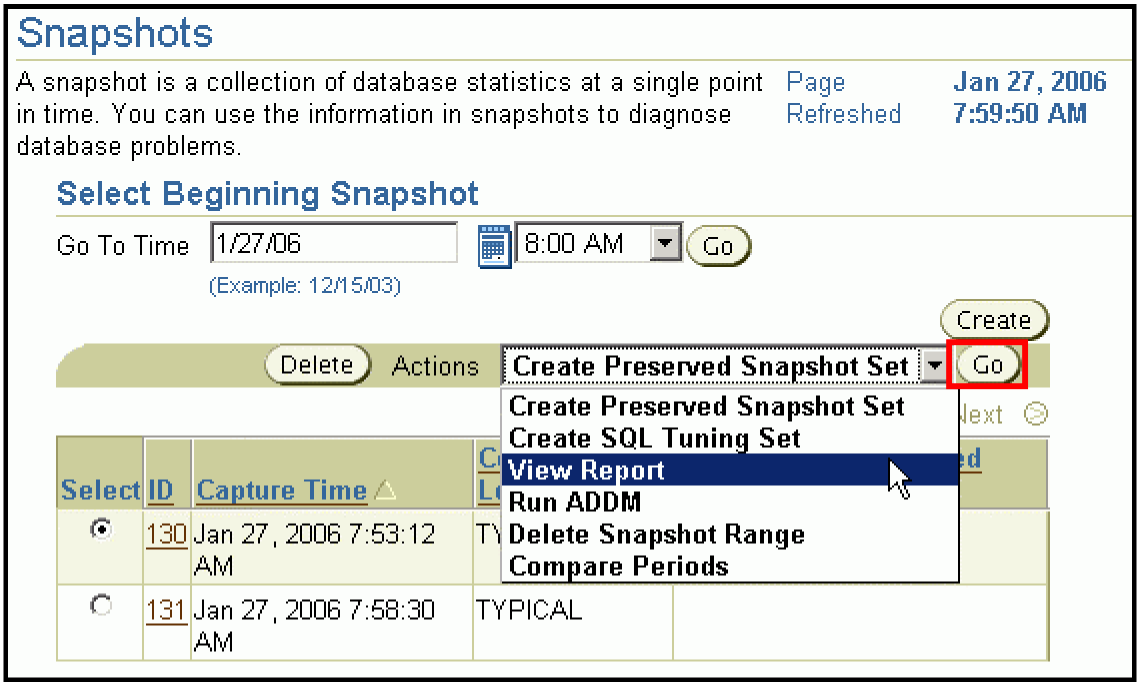
The recommendation is to "set the `CURSOR_SHARING` initialization parameter to `FORCE`."

6) Create a Statspack or AWR report from the last two snapshots.

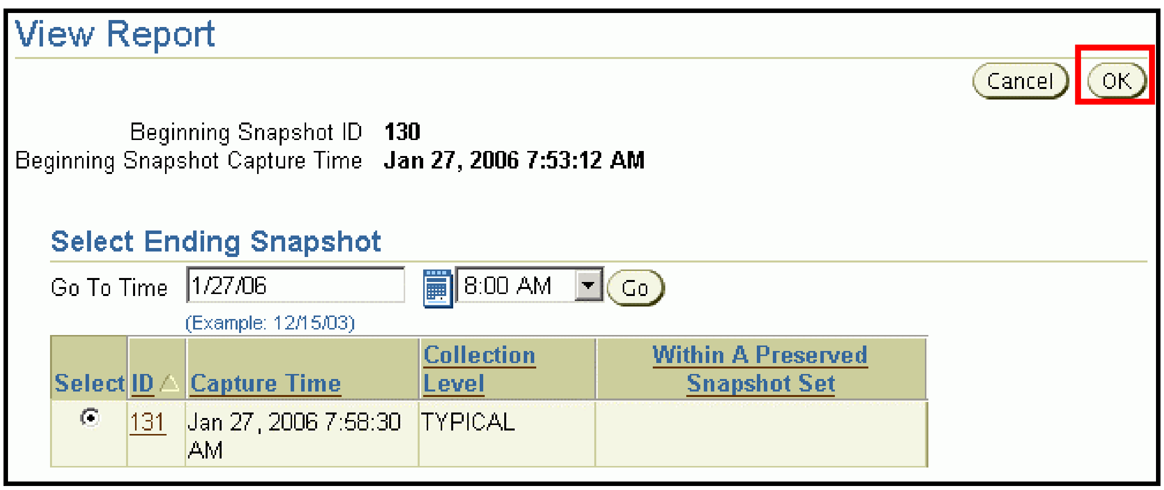
Answer:

1. The following shows the AWR report solution. From EM create an AWR report. Navigate to the Snapshots page. Select the next to last snapshot, and select View Report from the Actions drop-down list, and click Go.

Solutions for Practice 8-1: Tuning a Hard Parse Workload (continued)



2. On the View Report screen, select the last snapshot, and click OK.



7) Review the AWR report for the characteristics of a hard parse problem.

Answer:

The Top 5 Timed Events show that there are no significant wait events. `CPU Time` by itself does not indicate a problem. Further investigation into the Instance Efficiencies Percentages section shows `% Non-Parse CPU` is low, indicating that much of the CPU time is spent parsing. The `Soft Parse %:` shows that very few statements are found in the cache. Moving to the Load Profile section, compare `Parses per second` to `Hard parses per second`. A significant percentage of the parses are hard parses.

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
CPU time		137		77.4	
latch: library cache	61	5	74	2.6	Concurrency
db file scattered read	435	2	5	1.2	User I/O
db file sequential read	581	1	2	.7	User I/O
library cache pin	11	1	81	.5	Concurrency

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	100.00	Redo NoWait %:	99.97
Buffer Hit %:	99.74	In-memory Sort %:	100.00
Library Hit %:	90.72	Soft Parse %:	49.30
Execute to Parse %:	39.81	Latch Hit %:	100.00
Parse CPU to Parse Elapsd %:	75.36	% Non-Parse CPU:	30.99

Load Profile

	Per Second	Per Transaction
Redo size:	12,918.67	63,149.05
Logical reads:	3,264.06	15,955.38
Block changes:	36.05	176.23
Physical reads:	8.52	41.63
Physical writes:	2.84	13.86
User calls:	4.26	20.85
Parses:	59.46	290.66
Hard parses:	30.15	147.37
Sorts:	15.16	74.11
Logons:	0.06	0.28
Executes:	98.79	482.92
Transactions:	0.20	

% Blocks changed per Read:	1.10	Recursive Call %:	99.58
Rollback per transaction %:	6.15	Rows per Sort:	10.59

8) Implement the ADDM recommendation: either click the ADDM report Implement button or use the `ALTER SYSTEM SET CURSOR_SHARING = FORCE SCOPE=BOTH` command.

Solutions for Practice 8-1: Tuning a Hard Parse Workload (continued)

Answer:

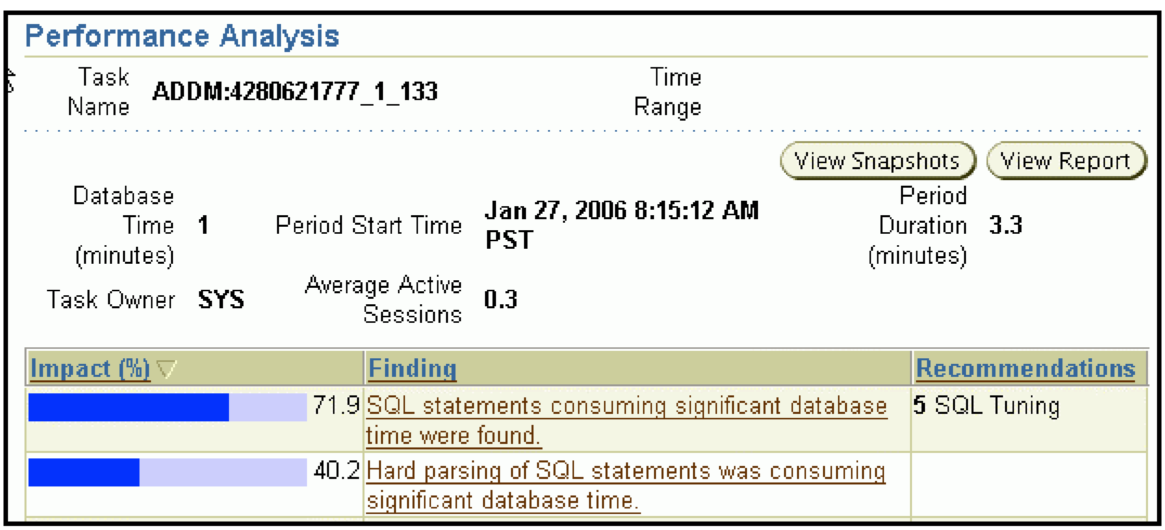
Using the SQL Plus interface, use the `ALTER SYSTEM SET CURSOR_SHARING = FORCE SCOPE=BOTH` command.

```
$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Fri Jan 27
08:13:01 2006

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> ALTER SYSTEM SET CURSOR_SHARING = FORCE SCOPE=BOTH;

System altered.

SQL> exit
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
$
```

9) Run the workload generator again and observe the differences in the tool of your choice.

Answer:

1. Using the ADDM tool, you see the impact of changing the `CURSOR_SHARING` parameter. The performance penalty of hard parsing has dropped significantly. The hard parse finding may not appear in the ADDM report.



Solutions for Practice 8-1: Tuning a Hard Parse Workload (continued)

2. Using the AWR report, review the significant statistics. Notice that the significant statistics show improvement. Notice that the hard parses are a much smaller percentage of the parses, the Soft Parse % is much improved, and % Non-Parse CPU is much higher. You may have that noticed the workload script ran more quickly as well.

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	99.99	In-memory Sort %:	100.00
Library Hit %:	90.32	Soft Parse %:	94.16
Execute to Parse %:	44.25	Latch Hit %:	100.00
Parse CPU to Parse Elapsd %:	68.25	% Non-Parse CPU:	80.72

Load Profile

	Per Second	Per Transaction
Redo size:	9,909.20	45,330.23
Logical reads:	5,138.76	23,507.53
Block changes:	18.87	86.30
Physical reads:	0.67	3.07
Physical writes:	0.71	3.23
User calls:	4.20	19.21
Parses:	77.12	352.77
Hard parses:	4.50	20.60
Sorts:	23.55	107.72
Logons:	0.09	0.42
Executes:	138.32	632.77
Transactions:	0.22	

Solutions for Practice 8-2: *Sizing the Shared Pool*

In this practice, you use the Statspack and ADDM reports to diagnose and fix the shared pool size.

1) In a terminal window, run the following command: `./setup 8 2`

   Answer:

```
$ cd /home/oracle/workshops
$ . oraenv
ORACLE_SID = [prod] ? prod
$ ./setup 8 2
ORACLE instance shut down.

ORACLE instance started.

Total System Global Area  314572800 bytes
Fixed Size                  1219160 bytes
Variable Size              96470440 bytes
Database Buffers          213909504 bytes
Redo Buffers                2973696 bytes
Database mounted.
Database opened.
Finished setup
$
```

2) In a terminal window, determine the machine time, run the workload generator, and check that the `wkld_one_sess.sh` processes are running by using the `ps` command. The `workgen` script creates an AWR snapshot and a Statspack snapshot, and then runs the workload. Record the time: _______________

```
date
./workgen 8 2
ps
```

   Answer:

```
$ date
Thu Dec  1 07:15:24 PST 2005
$ ./workgen 8 2

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

$ ps
  PID TTY          TIME CMD
13680 pts/3    00:00:01 bash
18351 pts/3    00:00:00 wkld_one_sess.s
18353 pts/3    00:00:00 wkld_one_sess.s
18354 pts/3    00:00:00 sqlplus
18355 pts/3    00:00:00 wkld_one_sess.s
```

```
18363 pts/3     00:00:00 sleep
18364 pts/3     00:00:00 sleep
18365 pts/3     00:00:00 ps
$
```

3) Wait about 5 minutes. In a terminal window, stop the workload with the `rm runload` command.

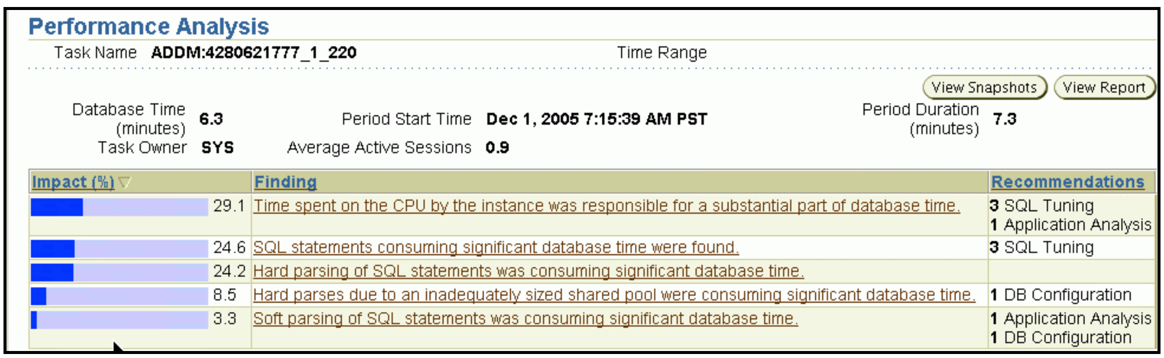
Answer:

```
 $ rm runload
rm: remove regular empty file `runload'? y
```

4) Navigate to the Performance page of the `prod` database. Create an ADDM report. Notice any findings referring to "Hard parses" (for example, "Hard parsing of SQL statement are consuming significant database time," or "Hard Parses due to an inadequately sized shared pool….").

Answer:

The particular findings and order of the findings may vary from the solution shown.



5) Create an AWR report between the last two snapshots.

Answer:

Navigate to the snapshots page. Using the View Report action, generate the AWR report. This is the same action as in Practice 8-1, Step 6.

6) View the AWR report and note the values for the Instance Efficiencies section, the Top 5 Timed Events, and the Library Cache Activity Statistics for the SQL AREA namespace.

Answer:

1. The Top 5 TimedEvents show time waiting on the library cache latch. This indicates contention in the lookup of SQL statements.

Solutions for Practice 8-2: Sizing the Shared Pool (continued)

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
CPU time		185		77.1	
db file sequential read	2,902	3	1	1.2	User I/O
latch: shared pool	40	3	67	1.1	Concurrency
os thread startup	5	3	533	1.1	Concurrency
latch: library cache	52	3	49	1.1	Concurrency

2. The latch contention taken with the hard parses shown in the Load Profile section shows a possible library cache or shared pool problem.

Load Profile

	Per Second	Per Transaction
Redo size:	4,479.46	19,497.51
Logical reads:	582.71	2,536.34
Block changes:	15.51	67.51
Physical reads:	16.44	71.55
Physical writes:	1.88	8.20
User calls:	37.40	162.78
Parses:	63.82	277.76
Hard parses:	3.89	16.95
Sorts:	19.71	85.80
Logons:	0.20	0.88
Executes:	74.07	322.39
Transactions:	0.23	

% Blocks changed per Read:	2.66	Recursive Call %:	96.76
Rollback per transaction %:	4.71	Rows per Sort:	6.84

3. The Reloads statistic of the SQL AREA namespace in the Library Cache Activity Statistics section indicates that the shared pool is too small. Reloads should be fewer than 1% of Pin Requests.

Library Cache Activity

- "Pct Misses" should be very low

Namespace	Get Requests	Pct Miss	Pin Requests	Pct Miss	Reloads	Invali- dations
BODY	212	10.85	981	6.32	38	0
CLUSTER	60	5.00	127	3.94	2	0
INDEX	172	15.12	236	16.95	14	0
SQL AREA	4,032	38.34	50,848	4.34	810	1
TABLE/PROCEDURE	4,998	21.19	12,252	28.77	880	0
TRIGGER	4	50.00	21	28.57	4	0

4. Further investigation of the dictionary cache shows a high number of misses.

Dictionary Cache Stats

- "Pct Misses" should be very low (< 2% in most cases)
- "Final Usage" is the number of cache entries being used

Cache	Get Requests	Pct Miss	Scan Reqs	Pct Miss	Mod Reqs	Final Usage
dc_awr_control	10	30.00	0		2	1
dc_free_extents	1	100.00	0		0	0
dc_global_oids	1,260	4.92	0		0	8
dc_histogram_data	5,219	29.34	0		0	505
dc_histogram_defs	11,999	42.47	0		0	1,042
dc_object_grants	1,169	16.17	0		0	51
dc_object_ids	13,420	7.54	0		0	337
dc_objects	5,288	26.13	0		0	297
dc_profiles	70	0.00	0		0	0
dc_rollback_segments	21	0.00	0		0	12
dc_segments	3,869	45.75	0		3	273
dc_sequences	6	100.00	0		6	0
dc_tablespaces	5,991	0.10	0		0	4
dc_usernames	2,199	0.59	0		0	2
dc_users	12,373	0.53	0		0	19
kqlsubheap_object	30	13.33	0		0	0

5. The Time Model gives the clearest indication of the impact of the problem. The parse time elapsed is taking a significant portion of the DBtime and hard parse elapsed accounts for the majority of the parse time elapsed.

Time Model Statistics

- Total time in database user-calls (DB Time): 240.4s
- Statistics including the word "background" measure background process time, and so do not contribute to the DB time statistic
- Ordered by % or DB time desc, Statistic name

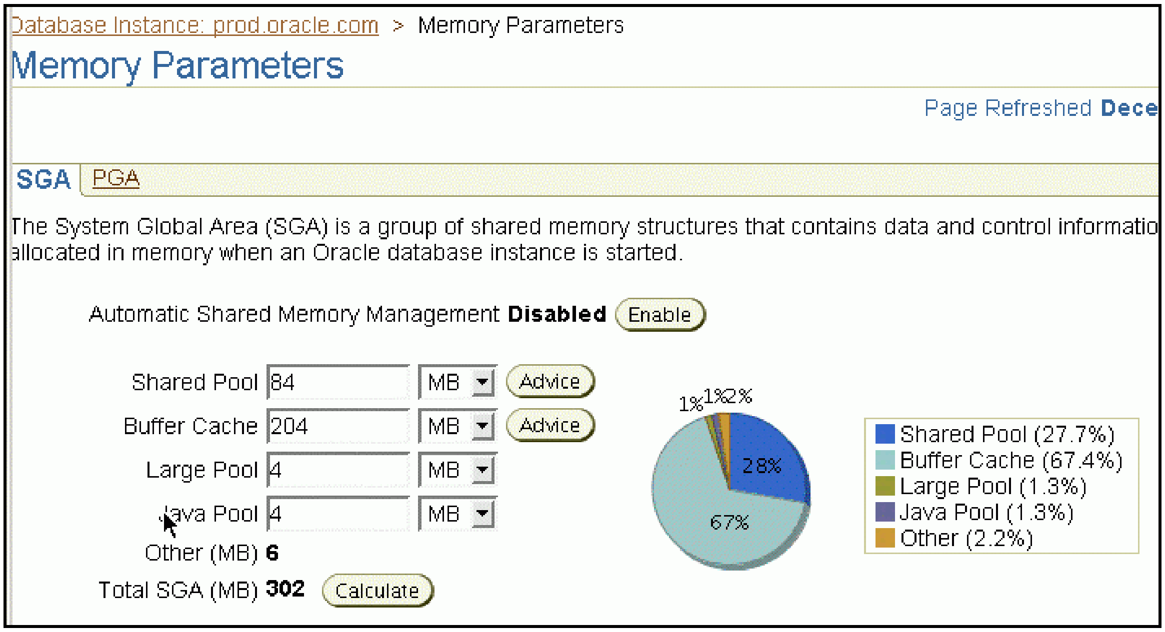
Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	229.91	95.63
DB CPU	185.44	77.14
parse time elapsed	63.63	26.47
hard parse elapsed time	56.69	23.58
PL/SQL compilation elapsed time	3.63	1.51
connection management call elapsed time	2.75	1.14
failed parse elapsed time	2.63	1.09
PL/SQL execution elapsed time	2.42	1.00
hard parse (sharing criteria) elapsed time	1.13	0.47
sequence load elapsed time	0.81	0.34
hard parse (bind mismatch) elapsed time	0.22	0.09
repeated bind elapsed time	0.07	0.03
DB time	240.41	
background elapsed time	8.60	
background cpu time	1.27	

7) Find an optimum shared pool size for this workload.

Answer:

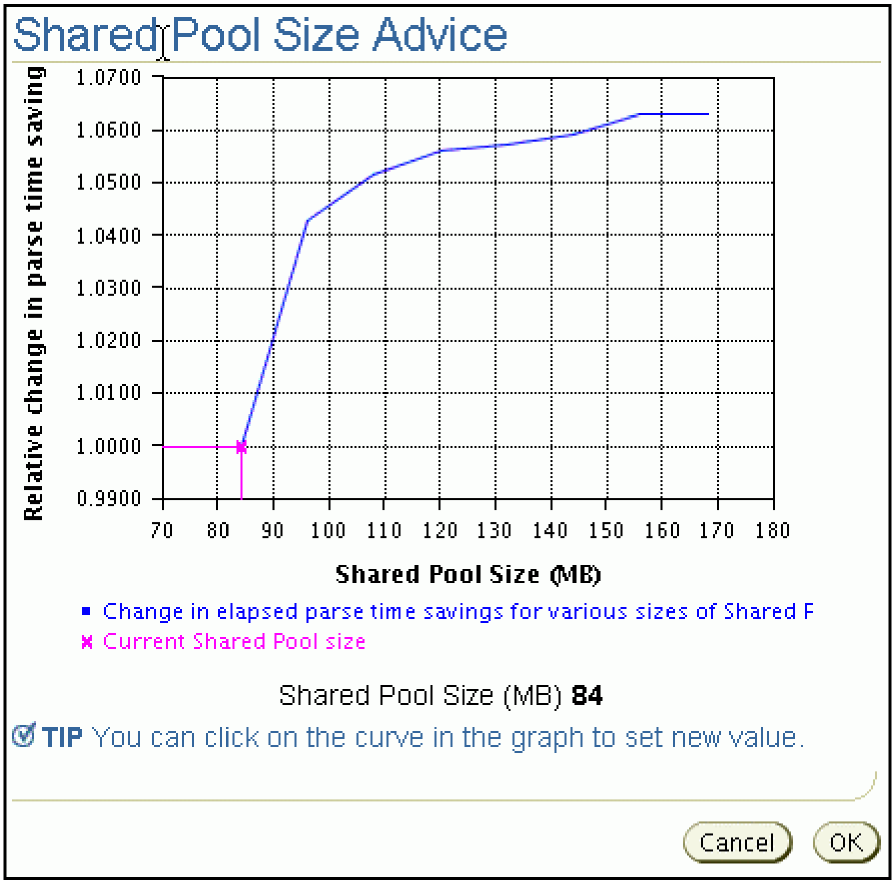
The solution can be obtained either through EM tools or using the Buffer Pool Advisory and Shared Pool Advisory in the Statspack report.

1. Navigate to the Memory Parameters page under the Administration tab.

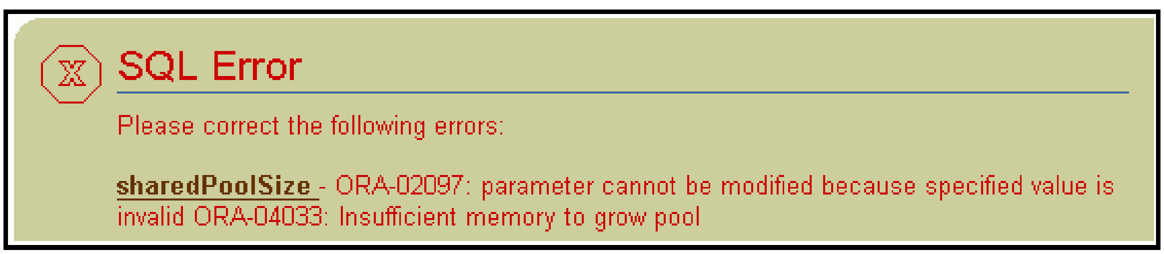


2. Click Shared Pool Advice. The graph shows performance versus shared pool size. The point the curve flattens is where there is no more improvement in performance as the shared pool size is increased. (The graph generated for your instance may vary.) Make a note of this point for your instance. Click Cancel. **Note:** This graph is based on estimates, on statistics collected during the previous workload. If the workload changes, so will the estimates.

Solutions for Practice 8-2: Sizing the Shared Pool (continued)

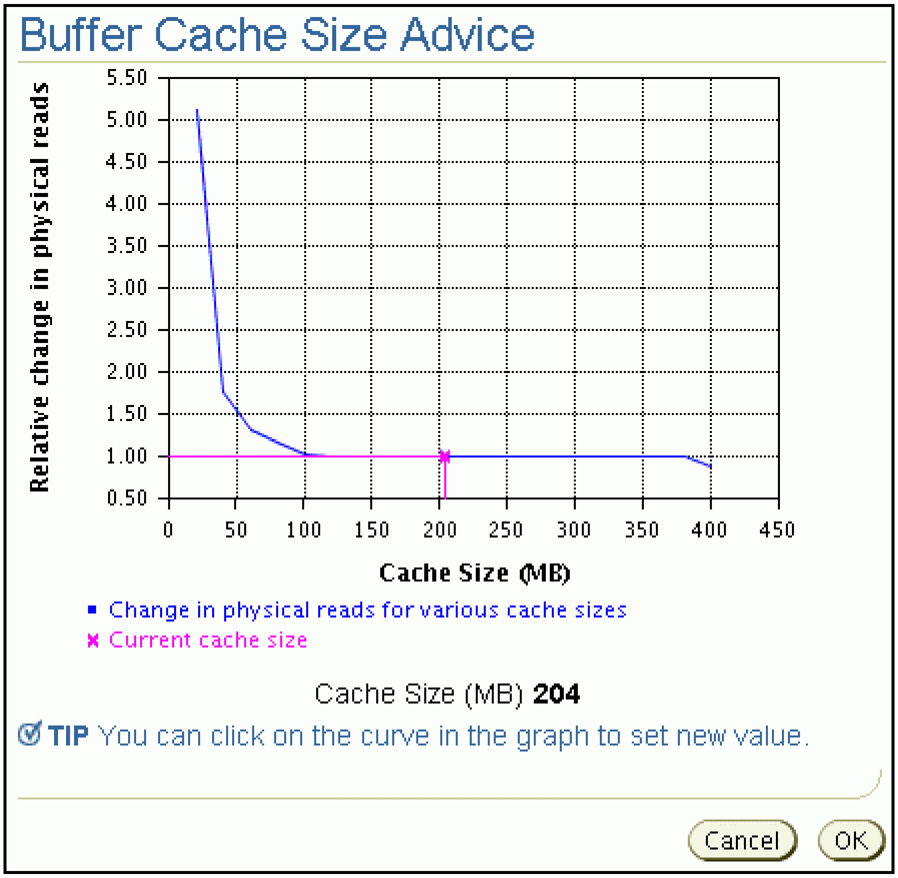


3. On the Memory Parameters Page, change the Shared Pool value to 156 MB. Click Apply. The following error message appears: “Insufficient memory to grow pool.” The Maximum SGA size is 300 MB. With this change, the new value for the SGA is 374 MB.



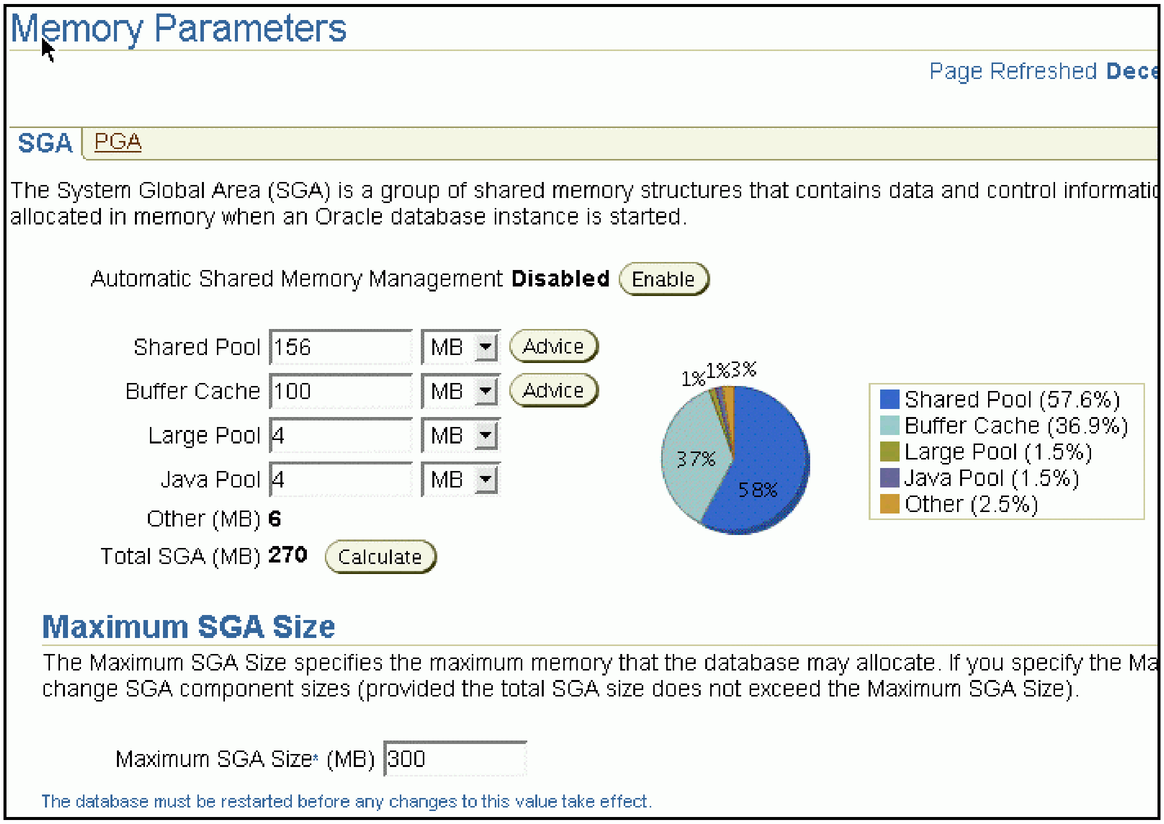
4. Click Buffer Cache Advice to determine how much memory can be taken from the buffer cache without seriously impacting performance. This graph shows the change in physical reads versus the cache size. You notice that the buffer cache could be reduced in size to about 100 MB with little or no additional physical reads. Click Cancel.

Solutions for Practice 8-2: Sizing the Shared Pool (continued)



5. Reduce the size of the database buffer cache to 100 MB on the Memory Parameters page. Increase the size of the shared pool to 156 MB and click Calculate. If the Total SGA is less than the Maximum SGA, then click Apply.

Solutions for Practice 8-2: Sizing the Shared Pool (continued)



8) Record the time: ________________
Run the workload generator again: `./workgen 8 2` for 5 minutes. Then stop the workload with the `rm runload` command.

Answer:

1. Find the system time. Start the workload.

```
  $ date
Thu Dec  1 08:10:03 PST 2005
 $ ./workgen 8 2

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

2. After 5 minutes, stop the workload.

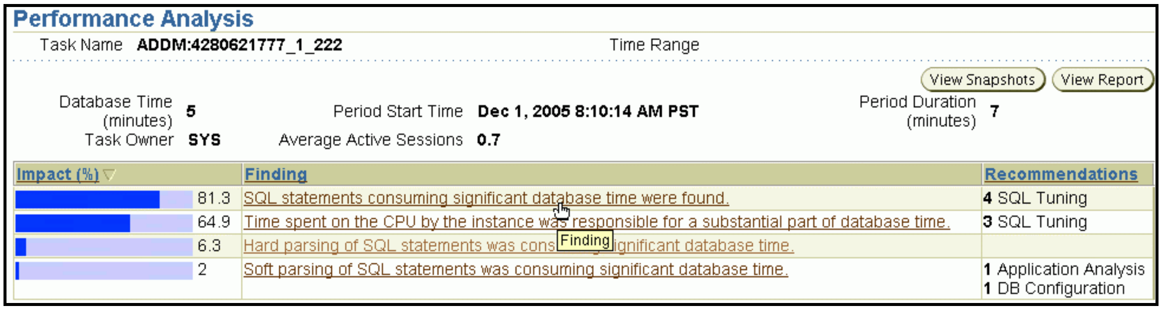
```
$ rm runload
rm: remove regular empty file `runload'? y
```

9) Navigate to the Performance page, wait until you see a drop in the Average Active Sessions graph, and then take another ADDM snapshot. What is the difference? What are the primary issues now?

Solutions for Practice 8-2: Sizing the Shared Pool (continued)

Answer:

Notice the findings. The hard parse findings has been reduced or eliminated from the findings.


Performance Analysis

Task Name **ADDM:4280621777_1_222** Time Range

View Snapshots View Report

Database Time (minutes) **5** Period Start Time **Dec 1, 2005 8:10:14 AM PST** Period Duration (minutes) **7**

Task Owner **SYS** Average Active Sessions **0.7**

Impact (%)	Finding	Recommendations
81.3	SQL statements consuming significant database time were found.	**4** SQL Tuning
64.9	Time spent on the CPU by the instance was responsible for a substantial part of database time.	**3** SQL Tuning
6.3	Hard parsing of SQL statements was cons Finding ignificant database time.	
2	Soft parsing of SQL statements was consuming significant database time.	**1** Application Analysis **1** DB Configuration

10) Navigate to the Snapshots page and create an AWR report.

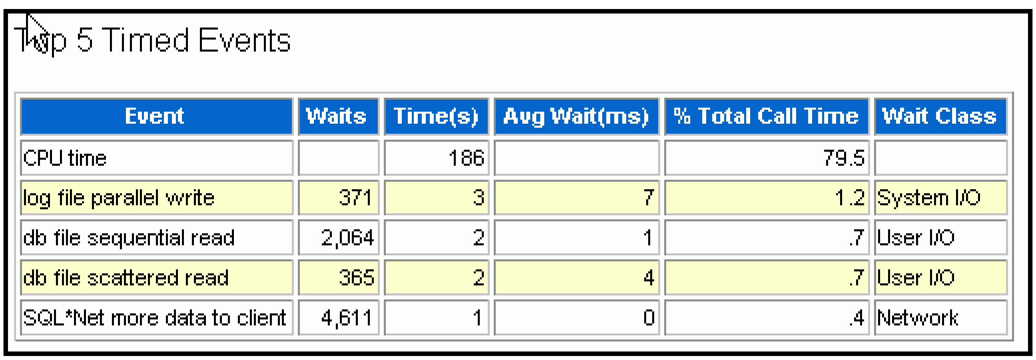
Answer:

Create an AWR report by using the same procedure as in Practice 8-1 Step 6.

11) View the AWR report and compare the values from the previous report. **Note:** The actual values you see will vary from those shown in the solutions.

Answer:

1. The % Total Call Time for Library cache and shared pool latches in the Top 5 Timed Events section has been reduced or eliminated.


Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
CPU time		186		79.5	
log file parallel write	371	3	7	1.2	System I/O
db file sequential read	2,064	2	1	.7	User I/O
db file scattered read	365	2	4	.7	User I/O
SQL*Net more data to client	4,611	1	0	.4	Network

2. The Load Profile section shows that hard parses are reduced (as are all parses).

Solutions for Practice 8-2: Sizing the Shared Pool (continued)

Load Profile

	Per Second	Per Transaction
Redo size:	5,430.60	23,764.08
Logical reads:	442.10	1,934.61
Block changes:	20.60	90.15
Physical reads:	10.00	43.76
Physical writes:	1.92	8.42
User calls:	38.13	166.86
Parses:	36.45	159.51
Hard parses:	2.02	8.83
Sorts:	7.87	34.44
Logons:	0.20	0.89
Executes:	42.61	186.44
Transactions:	0.23	

% Blocks changed per Read:	4.66	Recursive Call %:	92.71
Rollback per transaction %:	4.21	Rows per Sort:	9.27

3. Reloads for the SQL AREA namespace are now in an acceptable range.

Library Cache Activity

- "Pct Misses" should be very low

Namespace	Get Requests	Pct Miss	Pin Requests	Pct Miss	Reloads	Invali- dations
BODY	220	4.55	1,125	2.04	13	0
CLUSTER	21	0.00	39	2.56	1	0
INDEX	136	12.50	228	7.46	0	0
SQL AREA	2,602	33.55	32,149	4.26	433	0
TABLE/PROCEDURE	2,599	17.85	9,136	17.36	396	0
TRIGGER	6	16.67	75	6.67	4	0

4. The Dictionary Cache Stats section shows a much-reduced percentage of misses.

Solutions for Practice 8-2: Sizing the Shared Pool (continued)

Dictionary Cache Stats

- "Pct Misses" should be very low (< 2% in most cases)
- "Final Usage" is the number of cache entries being used

Cache	Get Requests	Pct Miss	Scan Reqs	Pct Miss	Mod Reqs	Final Usage
dc_awr_control	12	0.00	0		2	1
dc_free_extents	2	50.00	0		0	1
dc_global_oids	1,338	2.91	0		0	45
dc_histogram_data	3,563	17.43	0		0	1,065
dc_histogram_defs	8,899	27.81	0		0	3,627
dc_object_grants	979	7.46	0		0	110
dc_object_ids	8,739	5.44	0		0	761
dc_objects	3,528	19.64	0		0	977
dc_profiles	78	0.00	0		0	1
dc_rollback_segments	42	0.00	0		0	12
dc_segments	3,003	24.61	0		5	987
dc_sequences	14	42.86	0		14	6
dc_tablespaces	4,883	0.00	0		0	15
dc_usernames	1,081	0.46	0		0	9
dc_users	7,938	0.40	0		0	45
kqlsubheap_object	11	18.18	0		0	2
outstanding_alerts	2	50.00	0		2	28

5. The Time Model shows that hard parses are still contributing to parse elapsed time, but `% of DB Time` is reduced from the previous report.

Time Model Statistics

- Total time in database user-calls (DB Time): 233.3s
- Statistics including the word "background" measure background process time, and so do not contribute to the DB time statistic
- Ordered by % or DB time desc, Statistic name

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	227.37	97.44
DB CPU	185.57	79.52
parse time elapsed	29.03	12.44
hard parse elapsed time	24.11	10.33
failed parse elapsed time	2.82	1.21
hard parse (sharing criteria) elapsed time	2.48	1.06
PL/SQL execution elapsed time	1.53	0.66
connection management call elapsed time	0.98	0.42
PL/SQL compilation elapsed time	0.74	0.32
sequence load elapsed time	0.06	0.03
hard parse (bind mismatch) elapsed time	0.03	0.01
repeated bind elapsed time	0.00	0.00
DB time	233.35	
background elapsed time	10.01	
background cpu time	3.81	

12) Clean up the database by running: `./cleanup 8 2`

Answer:

```
$ ./cleanup 8 2
Cleanup Finished
```

Solutions for Practice 8-3: *Keeping Objects in the Shared Pool*

The performance of some applications is hindered by having to reload frequently used objects into the shared pool. If an object (such as a procedure, package, or sequence) is used often but not frequently enough to prevent aging out of the shared pool, keeping it will reduce the number of reloads. Keeping objects that are used very frequently is not necessary, but it does not hurt performance, because those objects will stay in the shared pool without being kept.

1) Start the workload generator: `./workgen 8 3`

2) After the workload has run for a few minutes, choose a candidate object to `KEEP` in the shared pool. Write and save a query to find a candidate object. Write the query to test whether the object has been `KEPT`. A suggested query is:

```
SELECT name, type, sharable_mem, kept
    FROM v$db_object_cache
    WHERE sharable_mem > 4000
    AND EXECUTIONS > 5
    AND (type='FUNCTION' OR type='PROCEDURE')
/
```

   This query is available in the `labs` directory as `sp_objects.sql`.

   Answer:

   The results of the query will vary. The `DEPLETE_INV` procedure should appear in all queries.

```
$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Wed Nov 30 13:40:26 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
SQL> set pagesize 100

SQL> SELECT name, type, sharable_mem, kept
  2  FROM v$db_object_cache
  3  WHERE sharable_mem > 4000
  4  AND EXECUTIONS > 5
  5  AND (type='FUNCTION' OR type='PROCEDURE')
  6  /

NAME
----------------------
TYPE                         SHARABLE_MEM KEP
---------------------------- ------------ ---
SETEMUSERCONTEXT
PROCEDURE                            4455 NO
```

```
TGT_SYSDATE_WGUID
FUNCTION                              4456 NO

GETEMKEY
FUNCTION                              4447 NO

TGT_SYSDATE_WTZ
FUNCTION                              4454 NO

DEPLETE_INV
PROCEDURE                            12638 NO

SQL>
```

3) Use the KEEP procedure to keep the DEPLETE_INV PROCEDURE object in the shared pool.

Answer:

```
SQL> EXECUTE dbms_shared_pool.keep('OE.DEPLETE_INV');

PL/SQL procedure successfully completed.

SQL> @sp_objects.sql

SQL> SELECT owner, name, type, sharable_mem, kept
  2  FROM v$db_object_cache
  3  WHERE sharable_mem > 4000
  4  AND EXECUTIONS > 5
  5  AND (type='FUNCTION' OR type='PROCEDURE')
  6  /

OWNER
--------------------
NAME
----------------------------------------
TYPE                         SHARABLE_MEM KEP
---------------------------- ------------ ---
SYSMAN
SETEMUSERCONTEXT
PROCEDURE                            4455 NO

SYSMAN
TGT_SYSDATE_WGUID
FUNCTION                             4456 NO

SYSMAN
GETEMKEY
FUNCTION                             4447 NO

SYSMAN
TGT_SYSDATE_WTZ
```

```
FUNCTION                                  4454 NO

OE
DEPLETE_INV
PROCEDURE                                12638 YES
```

4) Flush the shared pool and observe the state of `KEPT` objects.

Answer:

```
SQL> ALTER SYSTEM FLUSH SHARED_POOL;

System altered.

SQL> SELECT owner, name, type, sharable_mem, kept
  2  FROM v$db_object_cache
  3  WHERE sharable_mem > 4000
  4  AND EXECUTIONS > 5
  5  AND (type='FUNCTION' OR type='PROCEDURE')
  6  /

OWNER
--------------------------
NAME
----------------------------------------------------------
TYPE                         SHARABLE_MEM KEP
---------------------------- ------------ ---
OE
DEPLETE_INV
PROCEDURE                           12638 YES

SQL> exit
```

5) Stop the workload generator by executing the `rm runload` command.

Answer:

```
 $ rm runload
rm: remove regular empty file `runload'? y
```

Practice Solutions for Lesson 9

The goal of this practice is to observe the symptoms of, diagnose, and apply solutions for buffer cache problems.

Solutions for Practice 9-1: *Using the DB Cache Advisor*

1) Complete the setup for this scenario by running: `./setup 9 1`

   Answer:

```
 $ ./setup 9 1
ORACLE instance shut down.

ORACLE instance started.

Total System Global Area  335544320 bytes
Fixed Size                  1219280 bytes
Variable Size             117441840 bytes
Database Buffers          213909504 bytes
Redo Buffers                2973696 bytes
Database mounted.
Database opened.
Finished setup 9-1
```

2) Start the workload generator with `./workgen 9 1`.

   Answer:

   The `workgen` script creates an AWR snapshot and a Statspack snapshot before starting the workload in the background. If you run the `ps` command, you can see the background workload processes.

```
$ ./workgen 9 1

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

 $ ps
  PID TTY          TIME CMD
13680 pts/3    00:00:01 bash
32277 pts/3    00:00:00 wkld_sess.sh
32282 pts/3    00:00:00 wkld_sess.sh
32287 pts/3    00:00:00 wkld_sess.sh
32306 pts/3    00:00:00 wkld_sess.sh
32320 pts/3    00:00:00 wkld_sess.sh
32329 pts/3    00:00:00 wkld_sess.sh
32591 pts/3    00:00:00 sleep
32628 pts/3    00:00:00 sleep
32641 pts/3    00:00:00 sleep
32642 pts/3    00:00:00 sleep
32683 pts/3    00:00:00 sleep
32688 pts/3    00:00:00 sqlplus
32691 pts/3    00:00:00 ps
```

Solutions for Practice 9-1: Using the DB Cache Advisor (continued)

3) Wait about 5 minutes, and then in a terminal window, stop the workload with the `rm runload` command.

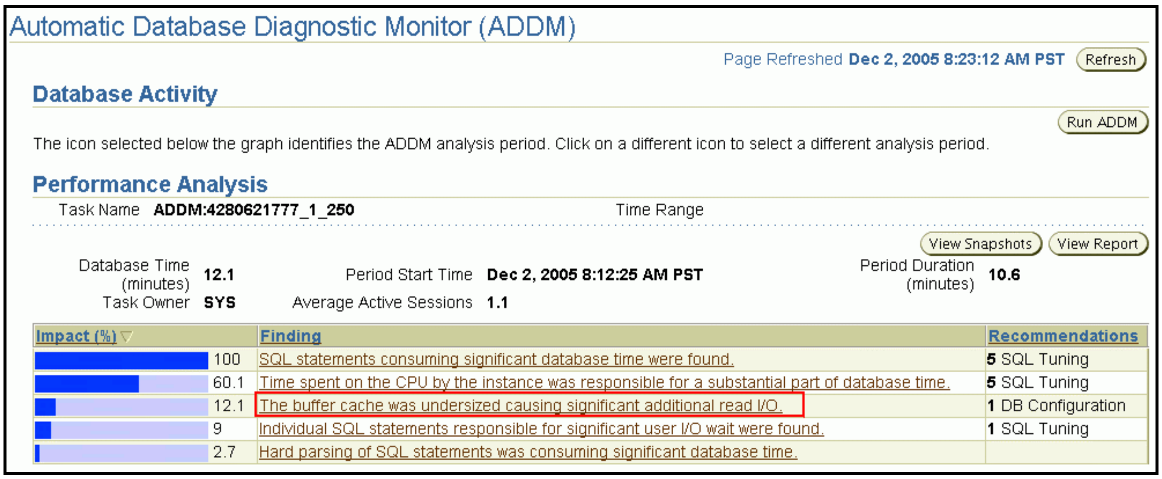
Answer:

```
$ rm runload
rm: remove regular empty file `runload'? y
```

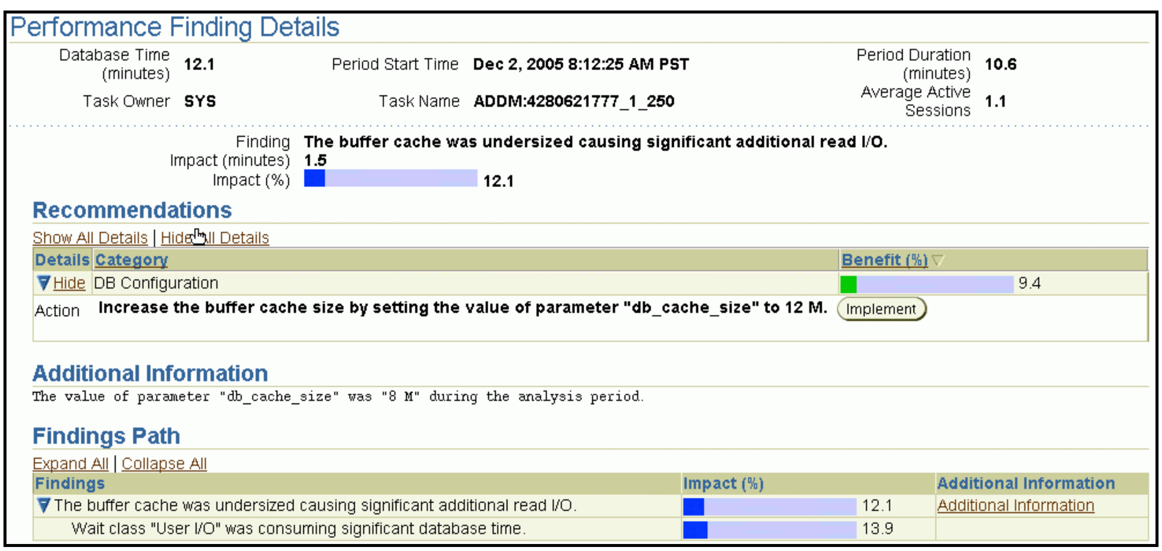
4) Create an ADDM report. View the Performance Analysis; check the performance findings. Find the top finding that you can address.

Answer:

1. Review ADDM Performance Analysis.



2. Drill down on the finding “The buffer cache was undersized causing significant additional I/O.” Notice the recommendation to increase the buffer cache size to 12 MB. This value is limited by the advisor, which takes estimates only from 50% to 200% of the current setting.



Solutions for Practice 9-1: Using the DB Cache Advisor (continued)

5) Navigate to the Snapshots page; create an AWR report. Save the report as `awr_91_1.html`.

Answer:

1. This follows the same procedure as the solution to Practice 8-1 step 6.
2. At the top-right of the report, click “Save to File.” When prompted, specify the name of the report. Notice which directory it is being created in so you can find it later.

6) Review the AWR report and identify the primary performance issue.

Answer:

1. The Top 5 Timed Events show one or more of the events that point to the buffer cache size: db file sequential read, db file scattered read, and latch: cache buffer lru chain.

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
CPU time		438		56.7	
db file sequential read	1,168,155	75	0	9.7	User I/O
db file scattered read	11,553	14	1	1.8	User I/O
log file sync	201	5	24	.6	Commit
latch: library cache	46	4	86	.5	Concurrency

2. Notice the buffer cache size in the report.

Report Summary

Cache Sizes

	Begin	End		
Buffer Cache:	8M	8M	Std Block Size:	8K
Shared Pool Size:	104M	104M	Log Buffer:	2,904K

3. The Load Profile section shows a high number and rate of physical reads.

Solutions for Practice 9-1: Using the DB Cache Advisor (continued)

Load Profile

	Per Second	Per Transaction
Redo size:	8,342.64	3,377.92
Logical reads:	21,038.09	8,518.28
Block changes:	49.39	20.00
Physical reads:	1,970.36	797.79
Physical writes:	6.21	2.51
User calls:	14.10	5.71
Parses:	22.09	8.95
Hard parses:	0.78	0.32
Sorts:	9.23	3.74
Logons:	0.30	0.12
Executes:	670.66	271.55
Transactions:	2.47	

4. The Instance Efficiency section shows a low buffer cache hit ratio.

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	90.63	In-memory Sort %:	100.00
Library Hit %:	99.47	Soft Parse %:	96.47
Execute to Parse %:	96.71	Latch Hit %:	100.00
Parse CPU to Parse Elapsd %:	58.23	% Non-Parse CPU:	98.35

5. The Buffer Pool Statistics section shows a high number of physical reads.

Buffer Pool Statistics

- Standard block size Pools D: default, K: keep, R: recycle
- Default Pools for other block sizes: 2k, 4k, 8k, 16k, 32k

P	Number of Buffers	Pool Hit%	Buffer Gets	Physical Reads	Physical Writes	Free Buff Wait	Writ Comp Wait	Buffer Busy Waits
D	998	91	13,980,211	1,309,203	4,122	0	0	0

6. The Latch Sleep Breakdown section shows some (and, in the following example, really insignificant) misses and sleeps for latches related to the buffer cache. The number of misses and sleeps is small enough in comparison to the get requests that they would not be considered at all except, because they are the top in the latch sleep breakdown, they are additional evidence pointing to the buffer cache.

Latch Sleep Breakdown

- ordered by misses desc

Latch Name	Get Requests	Misses	Sleeps	Spin Gets	Sleep1	Sleep2	Sleep3
cache buffers lru chain	1,335,352	68	68	0	0	0	0
cache buffers chains	29,287,839	49	50	0	0	0	0
library cache	1,077,870	46	46	0	0	0	0
simulator lru latch	1,161,866	12	12	0	0	0	0
library cache pin	976,480	6	6	0	0	0	0
KWQMN job cache list latch	285	4	4	0	0	0	0
object queue header operation	2,644,917	4	4	0	0	0	0
row cache objects	249,030	4	4	0	0	0	0
shared pool	80,977	2	2	0	0	0	0
In memory undo latch	15,238	1	1	0	0	0	0
redo allocation	7,562	1	1	0	0	0	0
session idle bit	22,082	1	1	0	0	0	0

7) Using the Buffer Pool Advisory in the AWR report, what is an appropriate size for the database buffer cache?

Answer:

In the following advisory, the number of physical reads would be reduced to 0.14 for the recommendation of 16 MB. **Note:** Due to the limitations of the advisor, the maximum estimated cache is only 200% of the current size. A larger buffer pool may provide even more benefit.

Buffer Pool Advisory

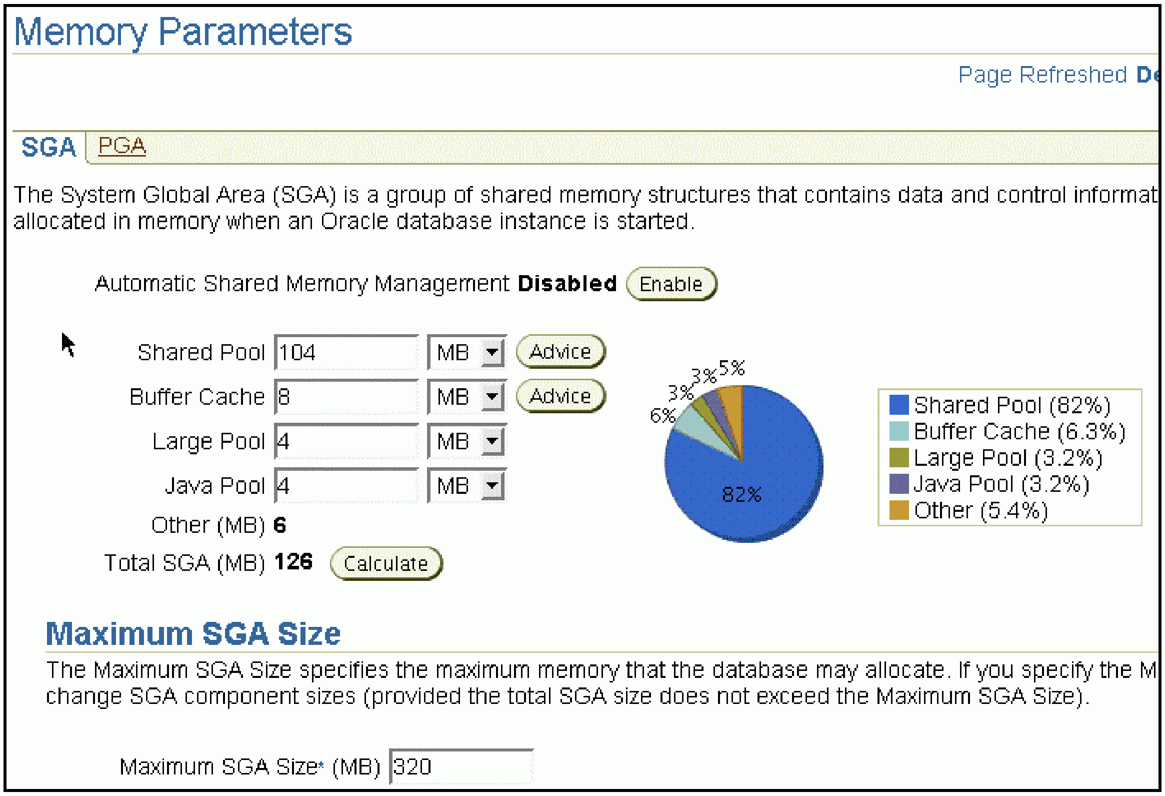
- Only rows with estimated physical reads >0 are displayed
- ordered by Block Size, Buffers For Estimate

P	Size for Est (M)	Size Factor	Buffers for Estimate	Est Phys Read Factor	Estimated Physical Reads
D	4	0.50	499	1.73	2,267,100
D	8	1.00	998	1.00	1,311,922
D	12	1.50	1,497	0.34	447,099
D	16	2.00	1,996	0.14	185,665

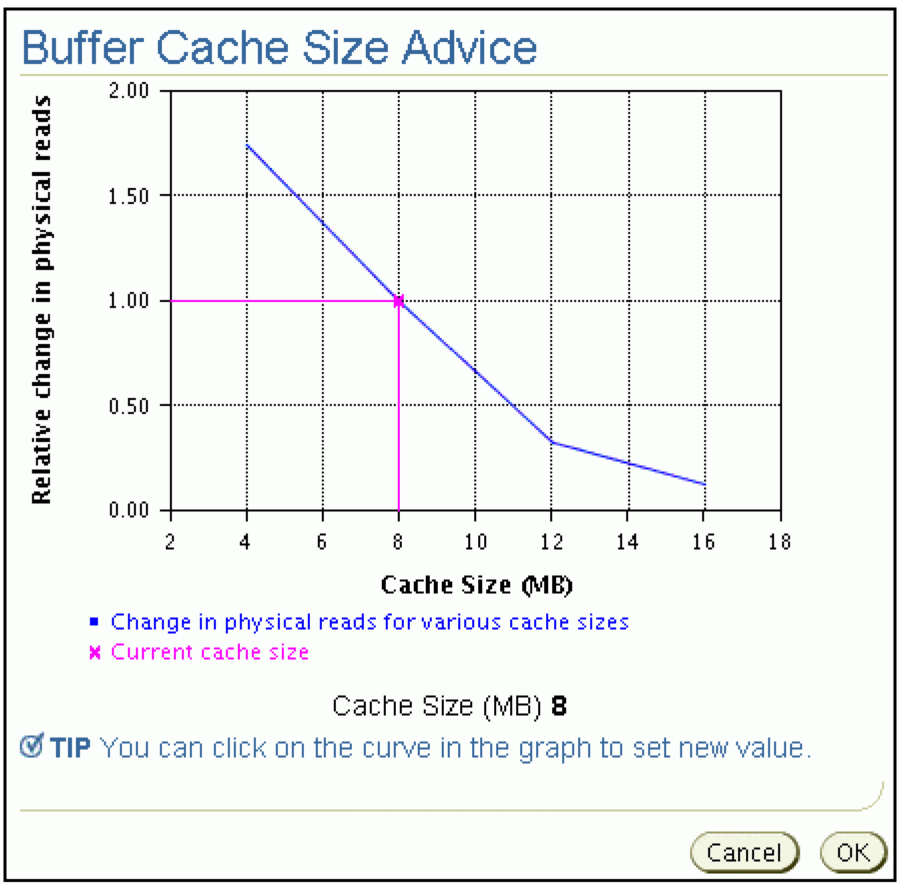
8) Navigate to the Memory Parameters page. Use the Buffer Cache Advisor. What is the appropriate buffer cache size?

Answer:

Due to the limitations of the advisor, the recommendation is limited to 16M.



View the Buffer Cache Advice.

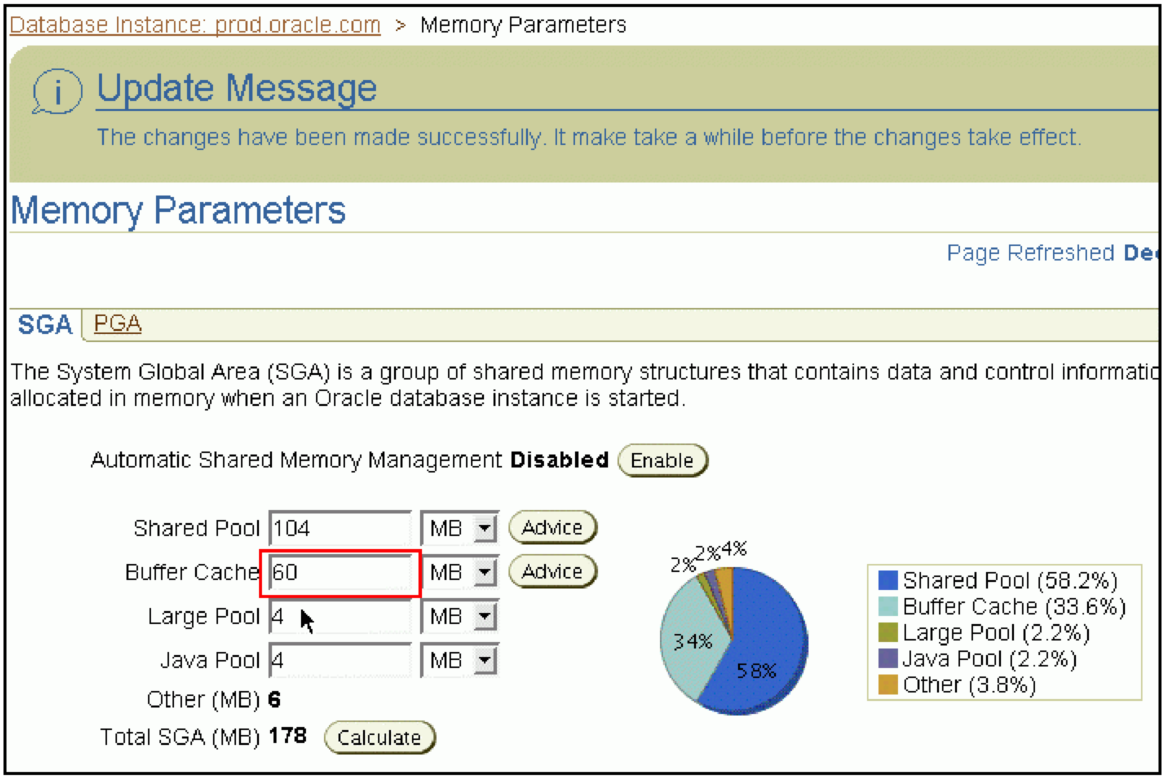


Because the curve is still descending at the point where `DB_CACHE_SIZE` is 16 MB, the optimum `DB_CACHE_SIZE` is somewhat larger.

9) Change the buffer cache size. To reduce the number of trials, increase `DB_CACHE_SIZE` to 60 MB.

Answer:

From the advisor you can see that the optimal size is greater than 16 MB. By changing the `DB_CACHE_SIZE` to 60 MB, the next trial should allow the advisor to show a better value.



10) Start the workload generator again: `./workgen 9 1`

Answer:

```
 $ ./workgen 9 1

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

11) After about 5 minutes, stop the workload with the `rm runload` command.

Answer:

```
$ rm runload
rm: remove regular empty file `runload'? y
```

Solutions for Practice 9-1: Using the DB Cache Advisor (continued)

12) Create an ADDM report. View the Performance Analysis; check the performance findings. Did the primary issues change?

Answer:

Yes, the issues have changed and the impact of the remaining issues has been reduced. **Note:** The remaining issues are related to SQL tuning, I/O, or CPU.

Performance Analysis

Task Name **ADDM:4280621777_1_252** Time Range

View Snapshots | View Report

Database Time (minutes) **6.4** Period Start Time **Dec 2, 2005 9:39:08 AM PST** Period Duration (minutes) **6.2**

Task Owner **SYS** Average Active Sessions **1**

Impact (%)	Finding	Recommendations
100	SQL statements consuming significant database time were found.	**5** SQL Tuning
60.5	Time spent on the CPU by the instance was responsible for a substantial part of database time.	**3** SQL Tuning
2.4	PL/SQL execution consumed significant database time.	**1** SQL Tuning

13) Navigate to the Snapshots page, and create an AWR report.

Answer:

1. This follows the same procedure as the solution to Practice 8-1 step 6.
2. In another window or tab display the `awr_91_1.html file`, so that you can compare.

14) Review the AWR report and identify the primary performance changes.

Answer:

1. In the Top 5 Timed Events, the order of the events and the amount of time waiting changed. The number of waits recorded against db_sequential_reads has dropped now that more blocks are being held in the cache. Waits for the other buffer cache–related items have dropped. Overall, more time is going to useful work (SQL execution) and less time to physical reads.

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
CPU time		303		59.6	
db file sequential read	3,615	19	5	3.7	User I/O
log file parallel write	703	3	4	.6	System I/O
latch: library cache	46	2	53	.5	Concurrency
db file scattered read	97	2	19	.4	User I/O

2. The physical reads as shown in the Load Profile section have drop significantly when the buffer cache size is increased to 60 MB.

Solutions for Practice 9-1: Using the DB Cache Advisor (continued)

Report Summary

Cache Sizes

	Begin	End		
Buffer Cache:	60M	60M	Std Block Size:	8K
Shared Pool Size:	104M	104M	Log Buffer:	2,904K

Load Profile

	Per Second	Per Transaction
Redo size:	7,746.52	3,642.79
Logical reads:	28,180.65	13,251.89
Block changes:	41.70	19.61
Physical reads:	9.15	4.30
Physical writes:	2.04	0.96
User calls:	11.31	5.32
Parses:	7.77	3.65
Hard parses:	0.03	0.01
Sorts:	4.71	2.22
Logons:	0.32	0.15
Executes:	807.84	379.88
Transactions:	2.13	

% Blocks changed per Read:	0.15	Recursive Call %:	98.69
Rollback per transaction %:	0.20	Rows per Sort:	10.87

3. The Buffer Pool Statistics section shows a better buffer cache hit ratio and a much lower number of physical reads.

Buffer Pool Statistics

- Standard block size Pools D: default, K: keep, R: recycle
- Default Pools for other block sizes: 2k, 4k, 8k, 16k, 32k

P	Number of Buffers	Pool Hit%	Buffer Gets	Physical Reads	Physical Writes	Free Buff Wait	Writ Comp Wait	Buffer Busy Waits
D	7,485	100	13,569,983	4,454	981	0	0	1

4. The following Buffer Cache Advisory example indicates that a buffer cache size of 44 MB is an optimal value for this workload. This is the point where additional memory does not reduce the Estimated Physical Reads. The optimal value for your instance may vary.

Solutions for Practice 9-1: Using the DB Cache Advisor (continued)

Buffer Pool Advisory

- Only rows with estimated physical reads >0 are displayed
- ordered by Block Size, Buffers For Estimate

P	Size for Est (M)	Size Factor	Buffers for Estimate	Est Phys Read Factor	Estimated Physical Reads
D	4	0.07	499	414.13	1,962,971
D	8	0.13	998	232.50	1,102,042
D	12	0.20	1,497	69.81	330,892
D	16	0.27	1,996	23.84	113,000
D	20	0.33	2,495	12.96	61,438
D	24	0.40	2,994	3.05	14,451
D	28	0.47	3,493	1.63	7,746
D	32	0.53	3,992	1.11	5,252
D	36	0.60	4,491	1.08	5,136
D	40	0.67	4,990	1.07	5,087
D	44	0.73	5,489	1.00	4,740
D	48	0.80	5,988	1.00	4,740
D	52	0.87	6,487	1.00	4,740
D	56	0.93	6,986	1.00	4,740
D	60	1.00	7,485	1.00	4,740
D	64	1.07	7,984	1.00	4,740
D	68	1.13	8,483	1.00	4,740
D	72	1.20	8,982	1.00	4,740
D	76	1.27	9,481	1.00	4,740
D	80	1.33	9,980	1.00	4,740

5. The latch statistics show a reduction in the number of sleeps and misses, which is due to the increase in the buffer cache size.

Latch Sleep Breakdown

- ordered by misses desc

Latch Name	Get Requests	Misses	Sleeps	Spin Gets	Sleep1	Sleep2	Sleep3
library cache	854,766	38	46	0	0	0	0
cache buffers chains	27,124,119	15	14	1	0	0	0
simulator lru latch	1,149,363	6	6	0	0	0	0
library cache pin	817,216	3	3	0	0	0	0
cache buffers lru chain	8,358	2	2	0	0	0	0
row cache objects	122,805	2	2	0	0	0	0
In memory undo latch	9,906	1	1	0	0	0	0
enqueues	10,559	1	1	0	0	0	0

Solutions for Practice 9-1: Using the DB Cache Advisor (continued)

15) Run the cleanup script: `./cleanup 9 1`

Answer:

```
 $ ./cleanup 9 1
Finished Cleanup 9-1
```

Solutions for Practice 9-2: *Using the Keep Pool*

In this scenario, assume the workload as in scenario 9-1 but you are constrained on memory resources. You are allowed only 112 MB of memory. The shared pool is already set as small as possible at 80 MB. Can you find a way to achieve a reasonable performance for both shared pool and database buffer cache?

1) Run the setup script for scenario 9-2: `./setup 9 2`

   Answer:

```
 $ ./setup 9 2
Database closed.
Database dismounted.
ORACLE instance shut down.

ORACLE instance started.

Total System Global Area  339738624 bytes
Fixed Size                  1219280 bytes
Variable Size             117441840 bytes
Database Buffers          213909504 bytes
Redo Buffers                7168000 bytes
Database mounted.
Database opened.
Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.

Total System Global Area  121634816 bytes
Fixed Size                  1218028 bytes
Variable Size              92277268 bytes
Database Buffers           20971520 bytes
Redo Buffers                7168000 bytes
Database mounted.
Database opened.
Finished setup 9-2
```

2) Run the workload generator for Practice 9-1: `./workgen 9 1`

   Answer:

```
 $ ./workgen 9 1

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

3) Wait about 5 minutes, and then end the workload by deleting the `runload` file. While you are waiting, examine the performance graph. Notice the peak values and the I/O per second values.

Solutions for Practice 9-2: Using the Keep Pool (continued)

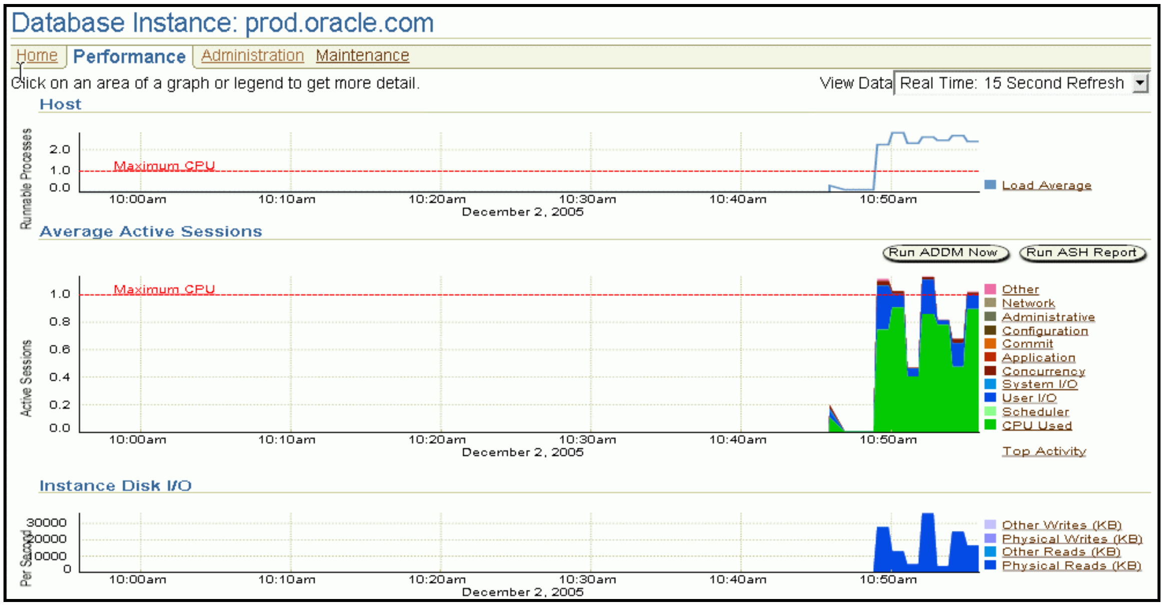
Answer:

```
$ rm runload
rm: remove regular empty file `runload'? y
```

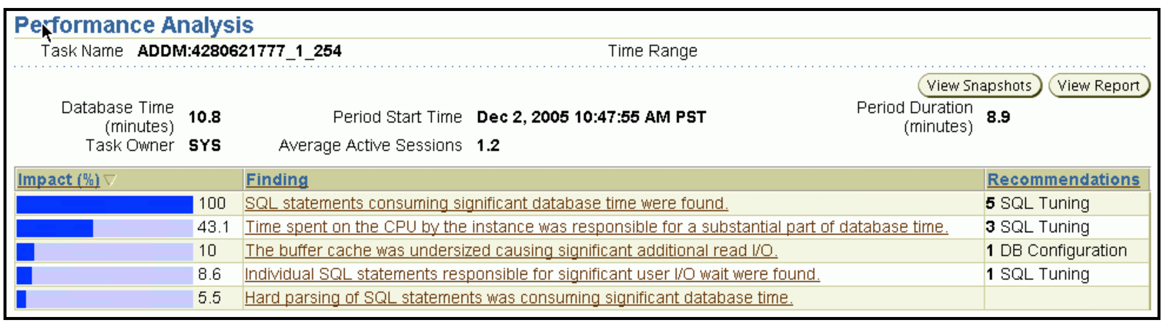
4) Create an ADDM report.

Answer:

1. The performance graph will look something like the following.



2. The ADDM Performance Analysis report will be similar to this.



Performance Analysis

Task Name **ADDM:4280621777_1_254** Time Range

View Snapshots | View Report

Database Time (minutes) **10.8** Period Start Time **Dec 2, 2005 10:47:55 AM PST** Period Duration (minutes) **8.9**

Task Owner **SYS** Average Active Sessions **1.2**

Impact (%)	Finding	Recommendations
100	SQL statements consuming significant database time were found.	**5** SQL Tuning
43.1	Time spent on the CPU by the instance was responsible for a substantial part of database time.	**3** SQL Tuning
10	The buffer cache was undersized causing significant additional read I/O.	**1** DB Configuration
8.6	Individual SQL statements responsible for significant user I/O wait were found.	**1** SQL Tuning
5.5	Hard parsing of SQL statements was consuming significant database time.	

5) Create an AWR report and save it as `awr_92_1.html`.

Answer:

1. This follows the same procedure as the solution to Practice 8-1 Step 6.
2. At the top-right of the report, click "Save to File." When prompted specify the name of the report. Notice which directory it is being created in so you can find it later.

6) Examine the AWR report. Are there tables that should be cached in a keep pool?

Answer:

1. The Top 5 Timed Events show one or more of the events that point to the buffer cache size: db file sequential read, db file scattered read, and latch: cache buffer lru chain. From the following example report, we can judge the possible impact of the buffer cache problem. If all waits associated with the physical reads are eliminated, then you can expect about a 10% increase in performance. Of course, removing all the waits for physical reads is not possible. **Note:** Your results will vary from the examples shown.

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
CPU time		280		55.8	
db file sequential read	877,679	49	0	9.7	User I/O
db file scattered read	7,500	7	1	1.4	User I/O
log file parallel write	1,999	7	3	1.3	System I/O
latch: library cache	40	3	75	.6	Concurrency

2. Notice the buffer cache size in the report. The Load Profile shows a high number and rate of physical reads. Notice the transaction rate and physical reads per transaction.

Solutions for Practice 9-2: Using the Keep Pool (continued)

Report Summary

Cache Sizes

	Begin	End		
Buffer Cache:	8M	8M	Std Block Size:	8K
Shared Pool Size:	80M	80M	Log Buffer:	2,904K

Load Profile

	Per Second	Per Transaction
Redo size:	17,787.79	2,592.68
Logical reads:	20,293.23	2,957.86
Block changes:	118.36	17.25
Physical reads:	2,377.91	346.59
Physical writes:	11.85	1.73
User calls:	16.99	2.48
Parses:	15.45	2.25
Hard parses:	0.32	0.05
Sorts:	12.92	1.88
Logons:	0.50	0.07
Executes:	652.37	95.09
Transactions:	6.86	

% Blocks changed per Read:	0.58	Recursive Call %:	97.85
Rollback per transaction %:	0.14	Rows per Sort:	6.64

3. The Instance Efficiency section shows a low buffer cache hit ratio.

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	88.28	In-memory Sort %:	100.00
Library Hit %:	99.87	Soft Parse %:	97.96
Execute to Parse %:	97.63	Latch Hit %:	100.00
Parse CPU to Parse Elapsd %:	77.66	% Non-Parse CPU:	98.91

4. The Buffer Pool Statistics section shows a high number of physical reads.

Buffer Pool Statistics

- Standard block size Pools D: default, K: keep, R: recycle
- Default Pools for other block sizes: 2k, 4k, 8k, 16k, 32k

P	Number of Buffers	Pool Hit%	Buffer Gets	Physical Reads	Physical Writes	Free Buff Wait	Writ Comp Wait	Buffer Busy Waits
D	998	88	8,290,390	971,388	4,840	0	0	2

5. The Latch Sleep Breakdown section shows some (in the example insignificant) misses and sleeps. The number of Misses and Sleeps are small enough in

comparison to the Get Requests that they should not be considered at all, because they are not in the top wait events.

Latch Sleep Breakdown

- ordered by misses desc

Latch Name	Get Requests	Misses	Sleeps	Spin Gets	Sleep1	Sleep2	Sleep3
library cache	636,025	39	40	0	0	0	0
cache buffers lru chain	994,798	30	30	0	0	0	0
cache buffers chains	17,663,359	27	27	0	0	0	0
simulator lru latch	676,666	13	13	0	0	0	0
library cache pin	583,733	11	11	0	0	0	0
redo allocation	10,151	3	3	0	0	0	0
In memory undo latch	25,102	1	1	0	0	0	0
object queue header operation	1,966,472	1	1	0	0	0	0
shared pool	24,724	1	1	0	0	0	0

6. How many full table scans are being performed? Check the instance activity statistics.

Instance Activity Stats

Statistic	Total	per Second	per Trans
CPU used by this session	53,502	69.22	30.42
CPU used when call started	53,586	69.33	30.46
CR blocks created	72	0.09	0.04
Cached Commit SCN referenced	2,959,132	3,828.31	1,682.28
Commit SCN cached	10	0.01	0.01
DB time	237,110	306.76	134.80

table fetch by rowid	4,500,068	11,014.62	1,605.45
table fetch continued row	15	0.04	0.01
table scan blocks gotten	664,314	1,626.01	237.00
table scan rows gotten	135,143,527	330,784.98	48,213.89
table scans (long tables)	225	0.55	0.08
table scans (short tables)	110,817	271.24	39.54

7. Find the SQL statements that have the highest number of physical reads.

SQL ordered by Reads

- Total Disk Reads: 971,505
- Captured SQL account for 99.6% of Total

Physical Reads	Executions	Reads per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
865,462	2	432,731.00	89.08	177.89	277.18	f88x40hstdhbb	SQL*Plus	DECLARE CURSOR C2 IS SELECT ...
860,798	122,486	7.03	88.60	152.00	231.75	431mwkyt65jbb	SQL*Plus	SELECT SUM(AMOUNT_SOLD) TOTAL ...
85,752	65	1,319.26	8.83	16.31	39.67	2bf83kpy24bj1	SQL*Plus	DECLARE CURSOR C1 IS SELECT ...
82,798	65	1,273.82	8.52	10.02	25.75	d4z6h7cjjyytn	SQL*Plus	SELECT * FROM USER_TABLES
5,160	39	132.31	0.53	11.[illegible]8	30.42	4yvdq7tc0gdq3	SQL*Plus	DECLARE max_orders NUM...
4,290	9	476.67	0.44	61.39	123.45	315vdwcszqnch	SQL*Plus	DECLARE CURSOR C1 IS SELECT ...
3,155	2,039	1.55	0.32	3.55	8.28	b83t16nqg7vt9	SQL*Plus	INSERT INTO ORDER_ITEMS(ORDER...
3,018	144	20.96	0.31	2.65	3.34	0u9dmyxy318w0	SQL*Plus	SELECT SUM(AMOUNT_SOLD) TOTAL ...
2,687	22	122.14	0.28	0.80	0.82	8fx6pqqbpra0s	OEM.DefaultPool	begin emd_database.getDBSiteMa...
2,449	2,871	0.85	0.25	1.70	3.61	av6kf9w9m362a	SQL*Plus	SELECT COUNT(*) FROM ORDER_ITE...

8. Check some of the SQL IDs for the full text of the statements with highest number of physical reads to determine the segments involved. The table names are highlighted.

f88x40hstdhbb	DECLARE CURSOR C2 IS SELECT CUST_ID FROM CUSTOMERS; CURSOR C1 IS SELECT PROD_ID FROM PRODUCTS; TYPE emp_list IS TABLE OF VARCHAR2(30); parameters emp_list; j NUMBER(10); tot_amount NUMBER(20); BEGIN FOR C2_ROW in C2 LOOP SELECT SUM(AMOUNT_SOLD) total INTO TOT_AMOUNT FROM sh.sales WHERE cust_id = C2_ROW.cust_id; END LOOP; FOR C1_ROW in C1 LOOP SELECT SUM(AMOUNT_SOLD) total INTO TOT_AMOUNT FROM sh.sales WHERE prod_id = C1_ROW.prod_id; END LOOP; END;

431mwkyt65jbb	SELECT SUM(AMOUNT_SOLD) TOTAL FROM SH.SALES WHERE CUST_ID = :B1

315vdwcszqnch	DECLARE CURSOR C1 IS SELECT * FROM orders CURSOR C2 IS SELECT * FROM order_items; CURSOR C3 IS SELECT * from CUSTOMERS; TYPE list IS TABLE OF VARCHAR2(30); parameters list; total_qoh NUMBER(20); tot_items NUMBER(20); BEGIN FOR c2row in C2 loop SELECT SUM(QUANTITY_ON_HAND) into TOTAL_QOH FROM INVENTORIES WHERE product_id = c2row.product_id; END LOOP; FOR c3row IN C3 LOOP SELECT count(*) into tot_items FROM order_items, orders WHERE order_items.order_id = orders.order_id AND c3row.customer_id = orders.customer_id; END LOOP; END;

gvmv1th76ysz0	SELECT SUM(QUANTITY_ON_HAND) FROM INVENTORIES WHERE PRODUCT_ID = :B1

2bf83kpy24bj1	DECLARE CURSOR C1 IS SELECT * FROM hr.employees; CURSOR C2 IS SELECT * FROM hr.Departments; CURSOR C3 IS SELECT * FROM hr.LOcations; CURSOR C4 IS SELECT * FROM hr.regions; CURSOR C5 IS SELECT * FROM user_tables; total NUMBER(10); v_dept Varchar2(30); v_region VARCHAR2(25); v_city VARCHAR2(30); v_country VARCHAR2(40); BEGIN FOR c5row in C5 LOOP SELECT count(index_name) into total from user_indexes WHERE table_name = c5row.table_name; END LOOP; FOR c1row in C1 LOOP BEGIN SELECT D.Department_name, R.region_name, L.City, C.country_name INTO v_dept, v_region, v_city, v_country FROM departments D, locations L, regions R, countries C WHERE D.location_id = L.location_id AND C.region_id = R.region_id AND L.country_id = C.country_id AND D.department_id = c1row.department_id; EXCEPTION WHEN NO_DATA_FOUND THEN NULL; END; END LOOP; FOR c1row in C1 LOOP BEGIN SELECT count (*) into total FROM JOB_HISTORY WHERE employee_id = c1row.employee_id; EXCEPTION WHEN NO_DATA_FOUND THEN NULL; END; END LOOP; END;

9. Find the segments that have the highest physical reads. The small tables that should be placed in the keep pool are not shown here. Knowledge of the application is needed to properly choose the tables and indexes to keep. The following example may not match your results. The results will vary by machine type.

Segments by Physical Reads

- Total Physical Reads: 971,505
- Captured Segments account for 28.1% of Total

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Physical Reads	%Total
SH	EXAMPLE	SALES	ES_Q4_1999	TABLE PARTITION	64,002	6.59
SH	EXAMPLE	SALES	ES_Q3_1999	TABLE PARTITION	62,823	6.47
SH	EXAMPLE	SALES	ES_Q1_1999	TABLE PARTITION	62,490	6.43
SH	EXAMPLE	SALES	ES_Q4_2001	TABLE PARTITION	53,849	5.54
OE	EXAMPLE	ITEM_ORDER_IX		INDEX	2,437	0.25

7) The developers have provided a script named `w92_keep_pool.sql` that will create a small keep pool of 4 MB, assign some small but very active tables to the keep pool, and load those tables into the pool. Review this script, and then run it.

Answer:

1. Review the script.

```
 $ less w92_keep_pool.sql
-- create a KEEP pool
-- and set the table properties to make use of it
set feedback off
set termout off
set pagesize 1000
spool w92_keep.lst REPLACE

connect / as sysdba

ALTER SYSTEM SET db_keep_cache_size = 4M;

connect hr/hr

ALTER TABLE EMPLOYEES STORAGE(BUFFER_POOL KEEP);
```

```
ALTER TABLE COUNTRIES STORAGE(BUFFER_POOL KEEP);
ALTER TABLE DEPARTMENTS STORAGE(BUFFER_POOL KEEP);
ALTER TABLE LOCATIONS STORAGE(BUFFER_POOL KEEP);
ALTER TABLE REGIONS STORAGE(BUFFER_POOL KEEP);
ALTER TABLE JOB_HISTORY STORAGE(BUFFER_POOL KEEP);

select * from employees;
select * from departments;
select * from countries;
select * from locations;
select * from regions;
select * from job_history;

connect oe/oe

ALTER TABLE orders STORAGE(BUFFER_POOL KEEP);
ALTER TABLE order_items STORAGE(BUFFER_POOL KEEP);
ALTER TABLE inventories STORAGE(BUFFER_POOL KEEP);
ALTER TABLE customers STORAGE(BUFFER_POOL KEEP);
ALTER TABLE product_information STORAGE(BUFFER_POOL KEEP);

select * from orders;
select * from order_items;
select * from inventories;
select * from customers;
select * from product_information;

SPOOL OFF
exit
```

2. Run the script.

```
 $ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Fri Dec 2
13:20:22 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> @w92_keep_pool.sql
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
SQL> exit
```

8) Test this new configuration with the workload from scenario 9-1.

Answer:

```
 $ ./workgen 9 1

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

9) Wait about 5 minutes, and then end the workload by deleting the `runload` file. While you are waiting, examine the performance graph. Notice the peak values and the I/O per second values.

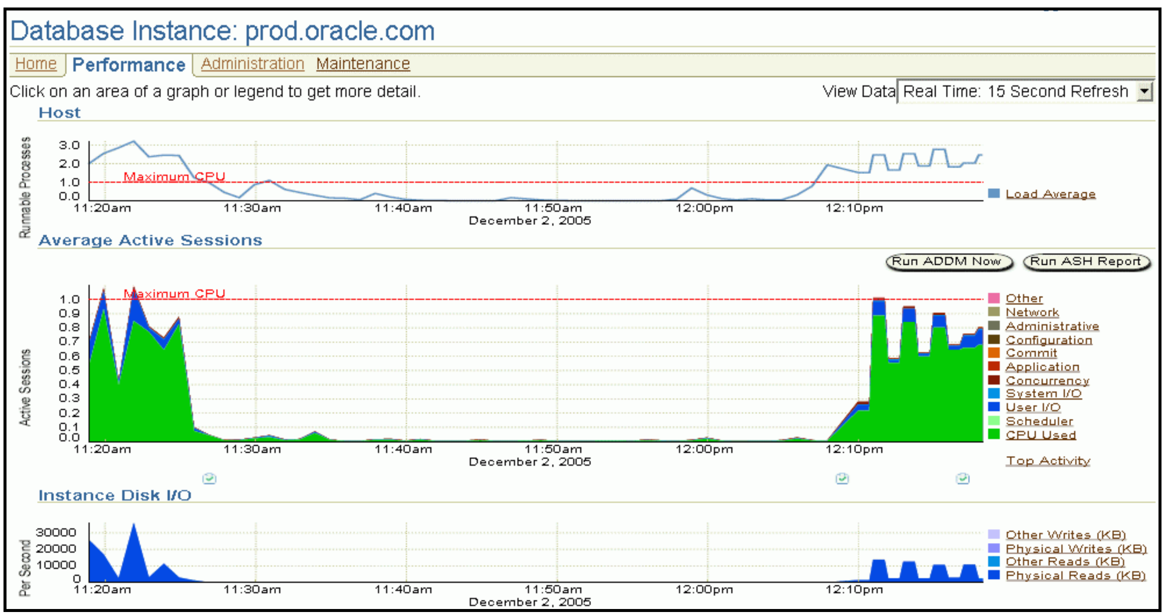
Answer:

```
 $ rm runload
rm: remove regular empty file `runload'? y
```

10) Create an ADDM report.

Answer:

1. The performance graphs will vary from the following example.



2. The ADDM finding will look something like this. The undersized buffer cache finding should have been eliminated. The number and order of the findings will vary.

Solutions for Practice 9-2: Using the Keep Pool (continued)

Performance Analysis

Task Name **ADDM:4280621777_1_258** Time Range

View Snapshots | View Report

Database Time (minutes)	**8.1**	Period Start Time	**Dec 2, 2005 12:09:38 PM PST**	Period Duration (minutes)	**7.4**
Task Owner	**SYS**	Average Active Sessions	**1.1**		

Impact (%)	Finding	Recommendations
100	SQL statements consuming significant database time were found.	**5** SQL Tuning
65.7	Time spent on the CPU by the instance was responsible for a substantial part of database time.	**3** SQL Tuning
5.7	Wait class "User I/O" was consuming significant database time.	
2.4	Hard parsing of SQL statements was consuming significant database time.	
2.1	PL/SQL execution consumed significant database time.	**1** SQL Tuning

11) Create an AWR report.

Answer:

This follows the same procedure as the solution to Practice 8-1 step 6.

12) Examine the AWR report and compare it with the previous report.
Answer the following questions:

a) What is the difference in the Top 5 Timed Events?

Answer:

The db file sequential reads and db file scattered reads may not have changed or may appear to have gotten worse. When the buffer cache is more efficient, more transactions (useful work) can be completed. But the more useful work will generate more reads.

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
CPU time		231		54.5	
db file sequential read	598,544	34	0	8.0	User I/O
latch: library cache	49	6	114	1.3	Concurrency
db file scattered read	5,686	5	1	1.2	User I/O
log file parallel write	1,522	5	3	1.2	System I/O

b) Is there a difference in physical reads and transactions?

Answer:

The number of transactions per second has increased, showing that useful work has increased. The expectation is that physical reads are reduced per transaction. This expectation may not always be supported in the statistics, especially with short duration samples (as you are doing in this practice).

Load Profile

	Per Second	Per Transaction
Redo size:	19,793.76	2,845.69
Logical reads:	21,526.24	3,094.77
Block changes:	124.04	17.83
Physical reads:	2,035.28	292.61
Physical writes:	12.03	1.73
User calls:	16.47	2.37
Parses:	15.81	2.27
Hard parses:	0.37	0.05
Sorts:	13.55	1.95
Logons:	0.50	0.07
Executes:	716.04	102.94
Transactions:	6.96	

% Blocks changed per Read:	0.58	Recursive Call %:	98.07
Rollback per transaction %:	0.00	Rows per Sort:	7.32

c) Is there a difference in Buffer Hit %?

Answer:

The overall Buffer Hit % may change very little. The buffer cache hit ratio is a combination of the pools.

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	90.55	In-memory Sort %:	100.00
Library Hit %:	99.87	Soft Parse %:	97.67
Execute to Parse %:	97.79	Latch Hit %:	100.00
Parse CPU to Parse Elapsd %:	56.05	% Non-Parse CPU:	98.76

d) Is the keep pool being used?

Answer:

Yes, the Buffer Pool Statistics section shows 100% hit ratio. The frequently used blocks are moved to the keep pool and a large number of hits are removed from the default pool calculation, so you would expect the hit ratio of the default pool to decline.

Solutions for Practice 9-2: Using the Keep Pool (continued)

Buffer Pool Statistics

- Standard block size Pools D: default, K: keep, R: recycle
- Default Pools for other block sizes: 2k, 4k, 8k, 16k, 32k

P	Number of Buffers	Pool Hit%	Buffer Gets	Physical Reads	Physical Writes	Free Buff Wait	Writ Comp Wait	Buffer Busy Waits
D	998	90	6,371,263	643,358	2,303	0	0	1
K	499	100	647,472	8	8	0	0	0

e) What is the difference in the table scan statistics?

Answer:

No significant difference. The scan statistics count the requests for a full table scan, not whether it results in a physical read.

13) Based on the reports, what are some possible next steps?

Answer:

1. Increase the size of the buffer cache. The advisory goes to only 16 MB, but it appears that more memory may still help reduce the number of physical reads.
2. Tune the SQL and the application. At this point, the SQL and application are using almost all the database time. This indicates that SQL tuning is the area that must be addressed to get any significant additional performance benefits.

14) Run the cleanup script for Practice 9-2. From the `/home/oracle/workshops` directory, execute: `./cleanup 9 2`

Answer:

```
$ cd /home/oracle/workshops
$ ./cleanup 9 2
$
```

Practice Solutions for Lesson 10

The goal of this practice is to use the Automatic SGA Tuning capability of Oracle Database 10*g*.

Solutions for Practice 10-1: *Enabling Automatic Shared Memory*

1) Prepare the instance for this practice.

   a) Run the setup script `./setup 10 1`.

   Answer:

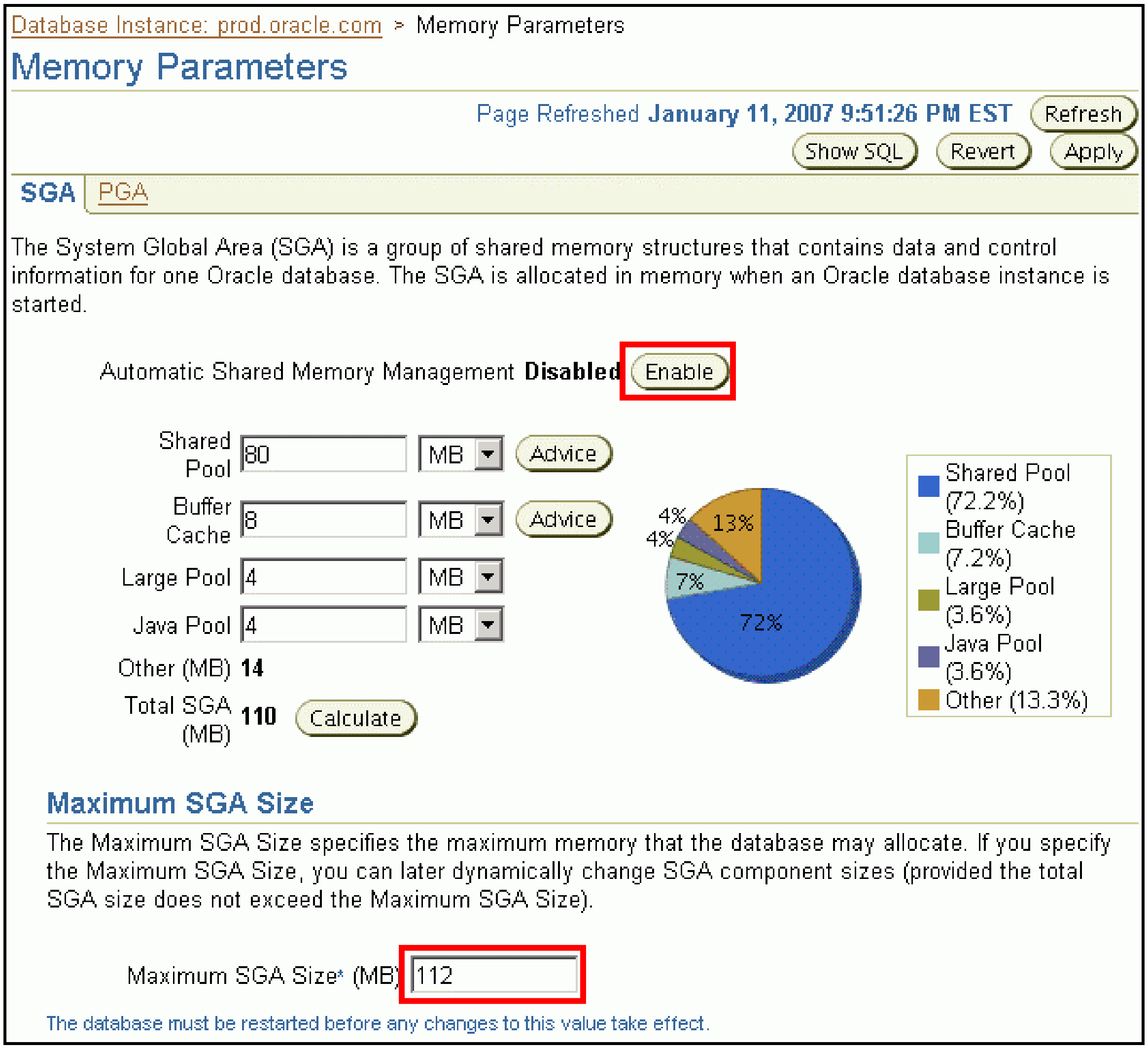
```
$ ./setup 10 1
Database closed.
Database dismounted.
ORACLE instance shut down.
ORACLE instance started.

Total System Global Area  117440512 bytes
Fixed Size                  1218052 bytes
Variable Size              92277244 bytes
Database Buffers           16777216 bytes
Redo Buffers                7168000 bytes
Database mounted.
Database opened.
$
```

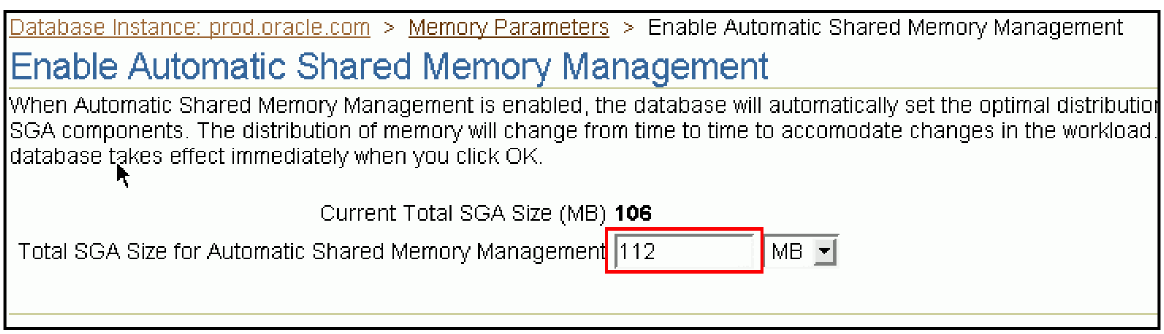
   b) Enable the Automatic Shared Memory Management feature in Enterprise Manager. Assume the same situation as in Practice 9-2 (that SGA memory is constrained to 112 MB). Ensure that Maximum SGA Size is set to 112 MB.

   Answer:

   Navigate to the Memory Parameters page. Make sure that Maximum SGA Size is set to 112 MB. Enable Automatic Shared Memory Management.

Solutions for Practice 10-1: Enabling Automatic Shared Memory (continued)





2) Run the workload generator for Practice 9-1 again: `./workgen 9 1`

Answer:

```
$ ./workgen 9 1

PL/SQL procedure successfully completed.
```

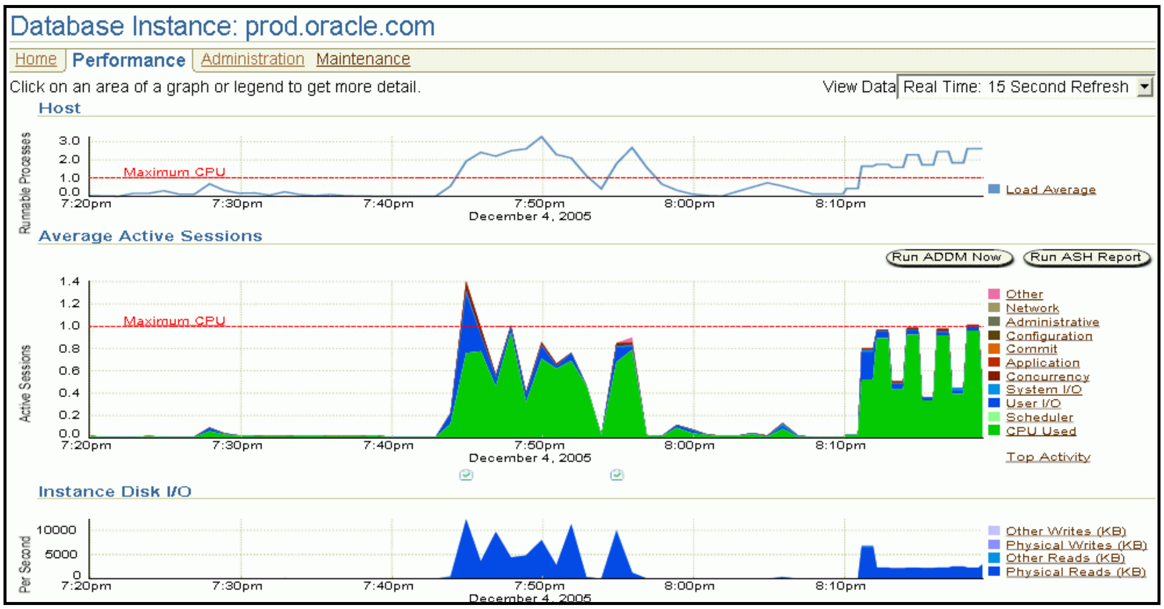
Solutions for Practice 10-1: Enabling Automatic Shared Memory (continued)

```
PL/SQL procedure successfully completed.
```

3) Observe the Performance graph. Is there any difference from the previous run of this workload?

   Answer:

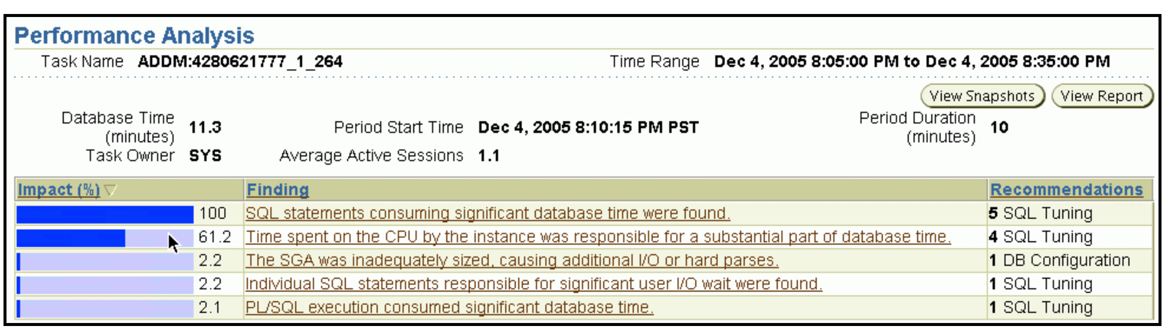
   Because the memory is constrained, there may be very little difference. You could expect that the peak may be somewhat lower and the physical IOs may be reduced.



4) After about 5 minutes, run an ADDM report. For any findings related to memory sizing, drill down to the details.

   Answer:

   1. View the ADDM report. The top findings should be related to SQL statements and CPU usage, such as “SQL statements consuming significant database time…” or “Time spent on the CPU by the instance…” or “Host CPU was a bottleneck….”



Performance Analysis

Task Name ADDM:4280621777_1_264 — Time Range Dec 4, 2005 8:05:00 PM to Dec 4, 2005 8:35:00 PM

View Snapshots | View Report

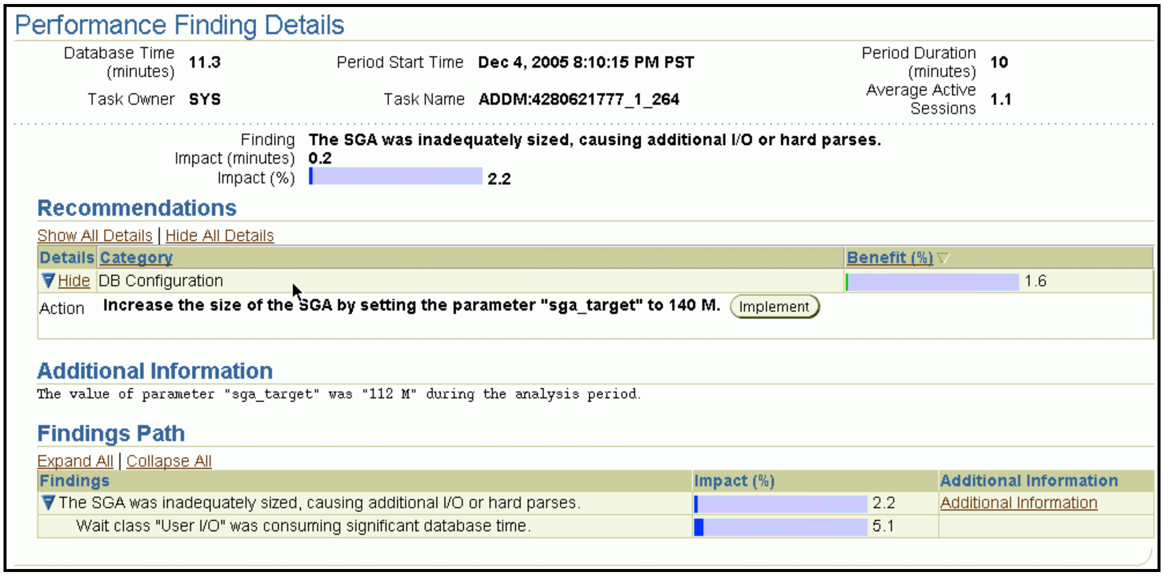
Database Time (minutes) 11.3 — Period Start Time Dec 4, 2005 8:10:15 PM PST — Period Duration (minutes) 10

Task Owner SYS — Average Active Sessions 1.1

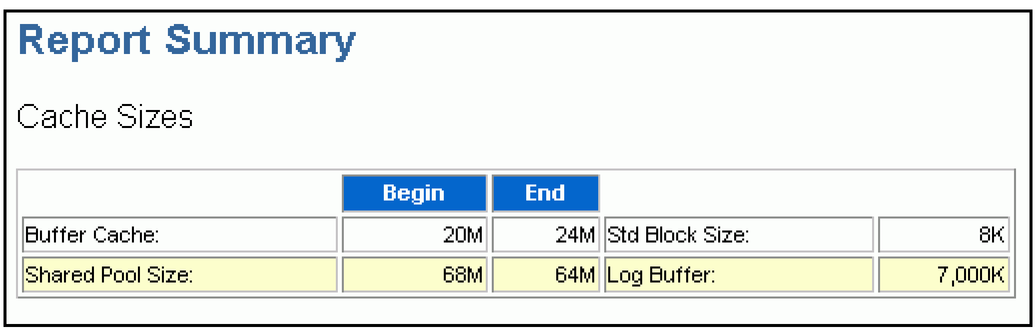
Impact (%)	Finding	Recommendations
100	SQL statements consuming significant database time were found.	5 SQL Tuning
61.2	Time spent on the CPU by the instance was responsible for a substantial part of database time.	4 SQL Tuning
2.2	The SGA was inadequately sized, causing additional I/O or hard parses.	1 DB Configuration
2.2	Individual SQL statements responsible for significant user I/O wait were found.	1 SQL Tuning
2.1	PL/SQL execution consumed significant database time.	1 SQL Tuning

Solutions for Practice 10-1: Enabling Automatic Shared Memory (continued)

2. Drill down on the finding “The SGA was inadequately sized, causing additional I/O or hard parses” and view the recommendation. This finding may not appear.



3. If the finding “The SGA was inadequately sized, causing additional I/O or hard parses” does not appear, click View Snapshots and then click the Report tab. Notice the Buffer cache and shared pool sizes: the buffer cache size started at 8 MB and the shared pool started at 80 MB. The Automatic Shared Memory Management feature adjusted the memory to maintain the best performance with available memory.



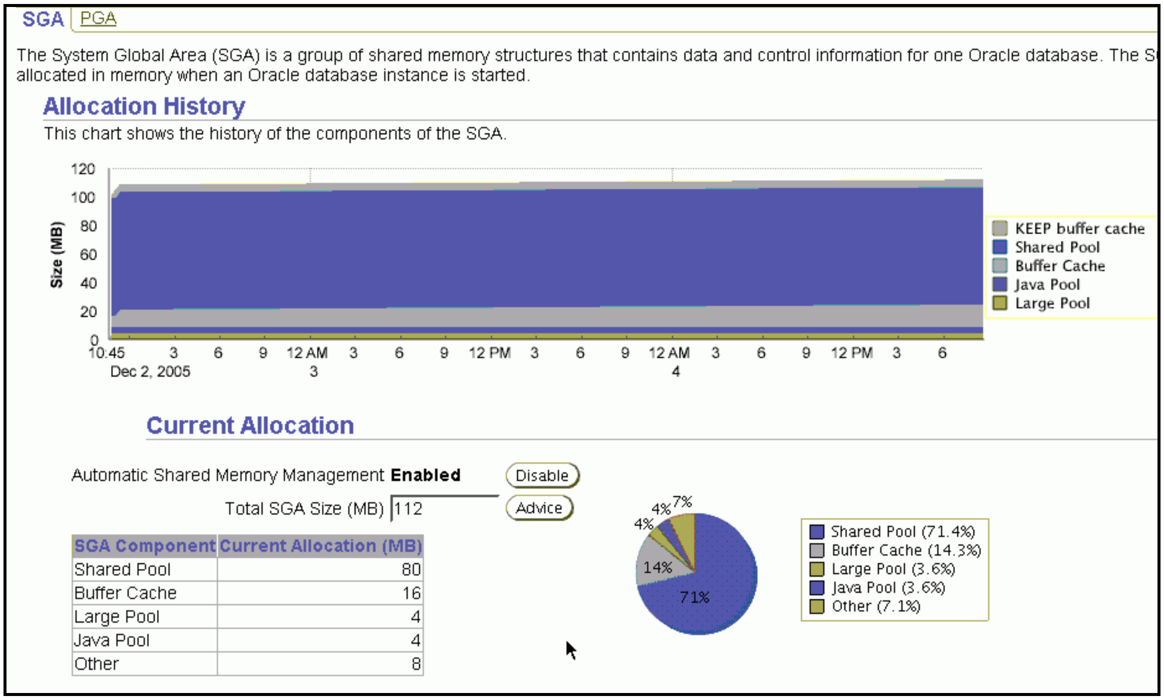
	Begin	End		
Buffer Cache:	20M	24M	Std Block Size:	8K
Shared Pool Size:	68M	64M	Log Buffer:	7,000K

5) What are the memory parameter settings?

Answer:

The memory parameter settings may vary depending on the machine you are using. An expected set of values are the following: Buffer Cache between 8 MB and 28 MB, Shared Pool between 60 MB and 80 MB, Large Pool = 4 MB, Java Pool = 4 MB, and Other = 8 MB to 16 MB. Note that the EM Memory Parameter page does not show the value of the keep pool separately. It is included in Other.

Solutions for Practice 10-1: Enabling Automatic Shared Memory (continued)



6) Check the Total SGA Size Advisory for an optimal SGA size.

Answer:

Click Advice on the Memory Parameters page. The advisory shows that increasing the SGA size will give some improvement in performance. The amount of memory recommended will vary with the machine type.

7) Check the Total SGA Size Advisory for an optimal SGA size.

Answer:

Click Advice on the Memory Parameters page. The advisory shows that increasing the SGA size will give some improvement in performance. The amount of memory recommended will vary with the machine type.

8) Stop the workload with the `rm runload` command.

Answer:

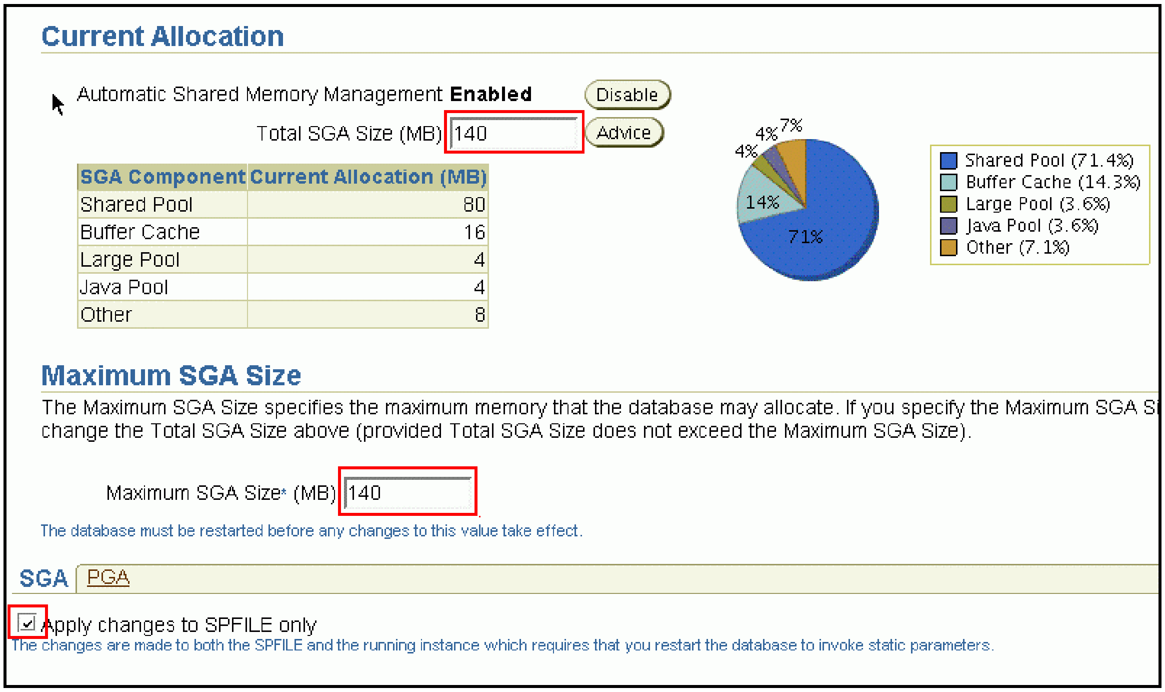
```
$ rm runload
rm: remove regular empty file `runload'? y
```

9) Based on the advisor, the memory constraint was raised to 140 MB. Make this change. (This will require a database shutdown to change the `SGA_MAX_SIZE` parameter.)

Solutions for Practice 10-1: Enabling Automatic Shared Memory (continued)

Answer:

1. On the Memory Parameters page, change Total SGA Size to 140 MB and Maximum SGA Size to 140 MB, and select the “Apply changes to SPFILE only” check box. Click Apply.



2. Restart the instance.

```
 $ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Mon Dec 5
06:09:45 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup
ORACLE instance started.

Total System Global Area  146800640 bytes
Fixed Size                  1218220 bytes
Variable Size              88082772 bytes
```

Solutions for Practice 10-1: Enabling Automatic Shared Memory (continued)

```
Database Buffers            54525952 bytes
Redo Buffers                 2973696 bytes
Database mounted.
Database opened.
SQL> exit
```

10) Run the `wksh_10_01_09.sql` script from SQL*Plus.

```
$sqlplus / as sysdba @wksh_10_01_09.sql
```

Answer:

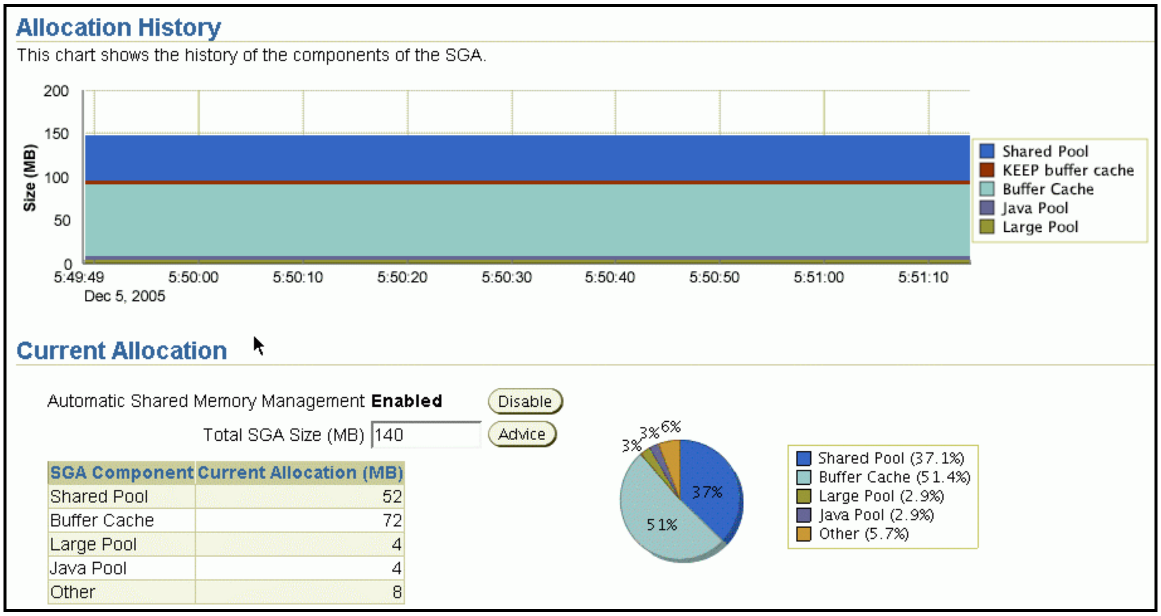
```
$ cd /home/oracle/workshops
$ sqlplus / as sysdba @wksh_10_01_09.sql

SQL*Plus: Release 10.2.0.1.0 - Production on Fri Jan 12
00:42:41 2007

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

Connected.
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
$
```

Solutions for Practice 10-1: Enabling Automatic Shared Memory (continued)

11) Observe the initial settings for the SGA on the Memory Parameters page.

Answer:

Notice the values of Buffer Cache and Shared Pool. The values for the shared pool and buffer cache can vary based on the machine you are using.



12) Start the workload for Practice 9-1 again: `./workgen 9 1`

Answer:

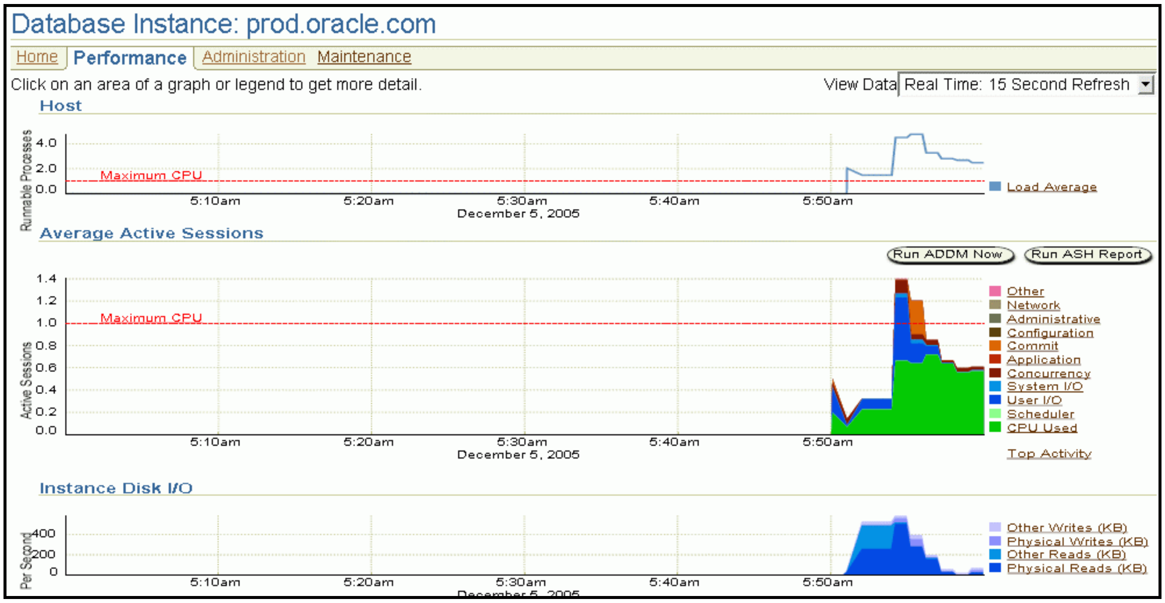
```
 $ ./workgen 9 1

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

$
```

13) Observe the Performance graph. What are the major differences between this and the previous graph?

Answer:

The expected result is that CPU usage is lower, as shown by the Host average load graph and the CPU portion of the Average Active Sessions graph, after an initial adjustment period. The Instance Disk I/O graph may show lower activity.

Solutions for Practice 10-1: Enabling Automatic Shared Memory (continued)



14) After about 5 minutes, stop the workload.

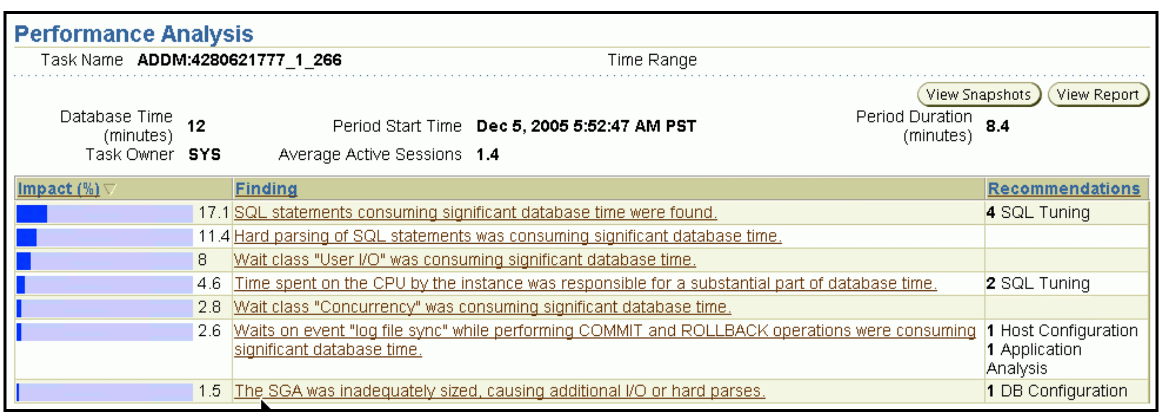
Answer:

```
 $ rm runload
rm: remove regular empty file `runload'? y
```

15) Run an ADDM report. View the results and drill down to the detail pages

Answer:

1. The problem issues have been reduced and the impact of the remaining issues is reduced.



Performance Analysis

Task Name **ADDM:4280621777_1_266** Time Range

View Snapshots | View Report

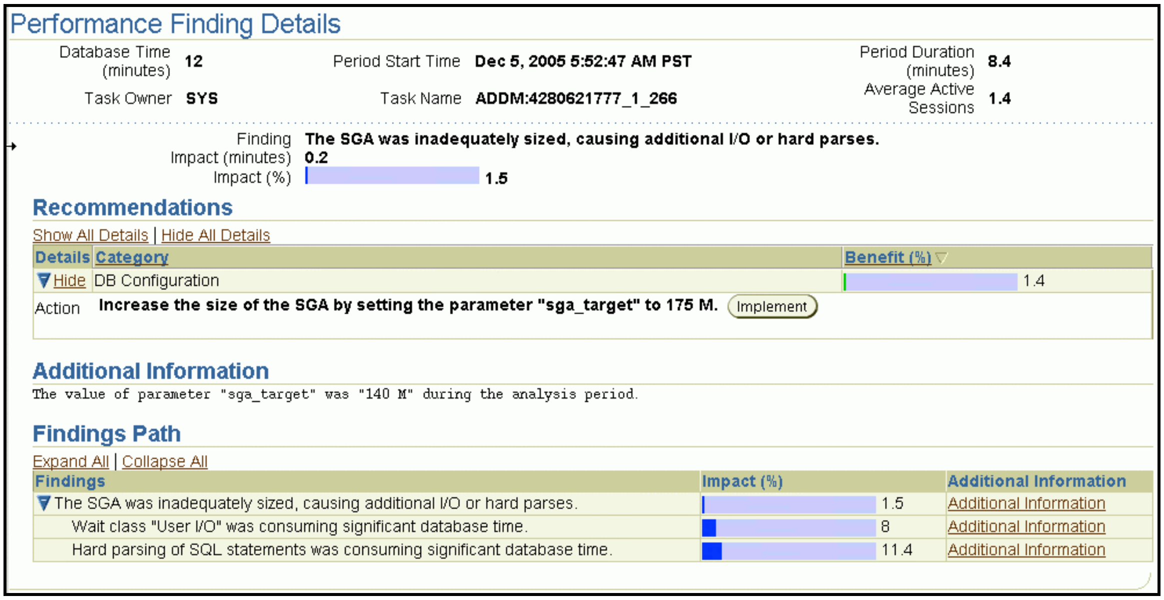
Database Time (minutes) **12** Period Start Time **Dec 5, 2005 5:52:47 AM PST** Period Duration (minutes) **8.4**

Task Owner **SYS** Average Active Sessions **1.4**

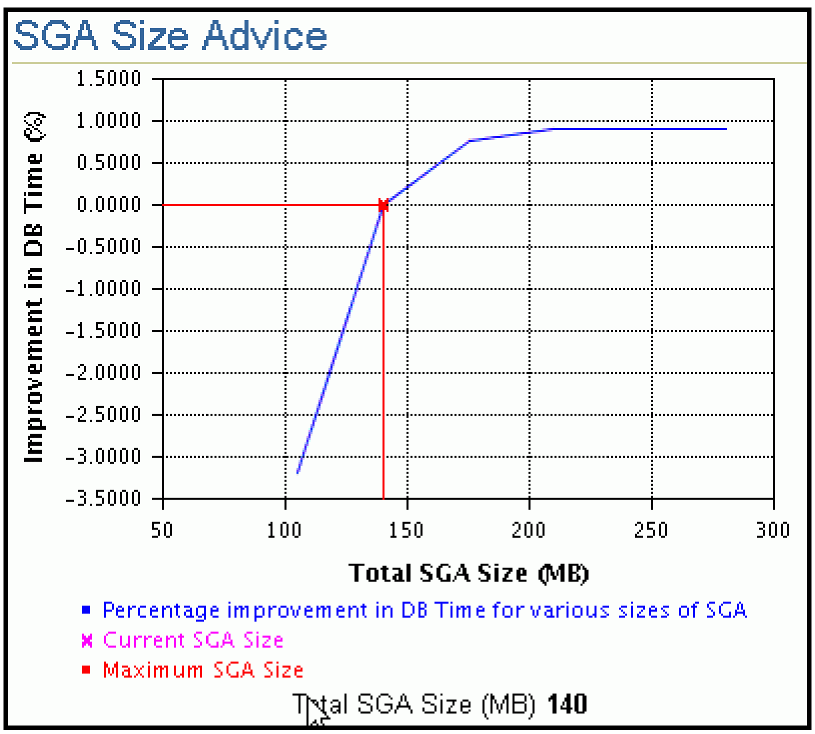
Impact (%)	Finding	Recommendations
17.1	SQL statements consuming significant database time were found.	**4** SQL Tuning
11.4	Hard parsing of SQL statements was consuming significant database time.	
8	Wait class "User I/O" was consuming significant database time.	
4.6	Time spent on the CPU by the instance was responsible for a substantial part of database time.	**2** SQL Tuning
2.8	Wait class "Concurrency" was consuming significant database time.	
2.6	Waits on event "log file sync" while performing COMMIT and ROLLBACK operations were consuming significant database time.	**1** Host Configuration **1** Application Analysis
1.5	The SGA was inadequately sized, causing additional I/O or hard parses.	**1** DB Configuration

2. The finding "The SGA was inadequately sized..." may still be reported. The recommendation is to increase the SGA to 175 MB. The impact of this issue as shown in the screenshot is only 1.5%. Increasing the SGA memory allocation to 175 MB will give only a very minor benefit.

Solutions for Practice 10-1: Enabling Automatic Shared Memory (continued)



3. Check the SGA Memory Advisory. It shows that 175 MB would give some additional benefit. You decide that 140 MB is adequate.



Solutions for Practice 10-1: Enabling Automatic Shared Memory (continued)

16) What would be a reasonable next tuning step based on the ADDM report?

Answer:

The number and ranking of the issues shown in the ADDM reports of various machine will vary.

1. The issue with the largest impact is the time taken by the SQL statements. This requires access to the application to perform SQL tuning.
2. Other issues are User I/O and Commit. Both, when examined, are essentially problems of SQL tuning.
3. The details for concurrency are nonspecific.
4. The log file sync issue is examined in a later lesson.
5. The sizing of the SGA issue has had a surprisingly large impact on a variety of other issues, compared to the impact rating assigned to it. It is reasonable to assume that if the SGA is increased again, there will be further improvements in performance.

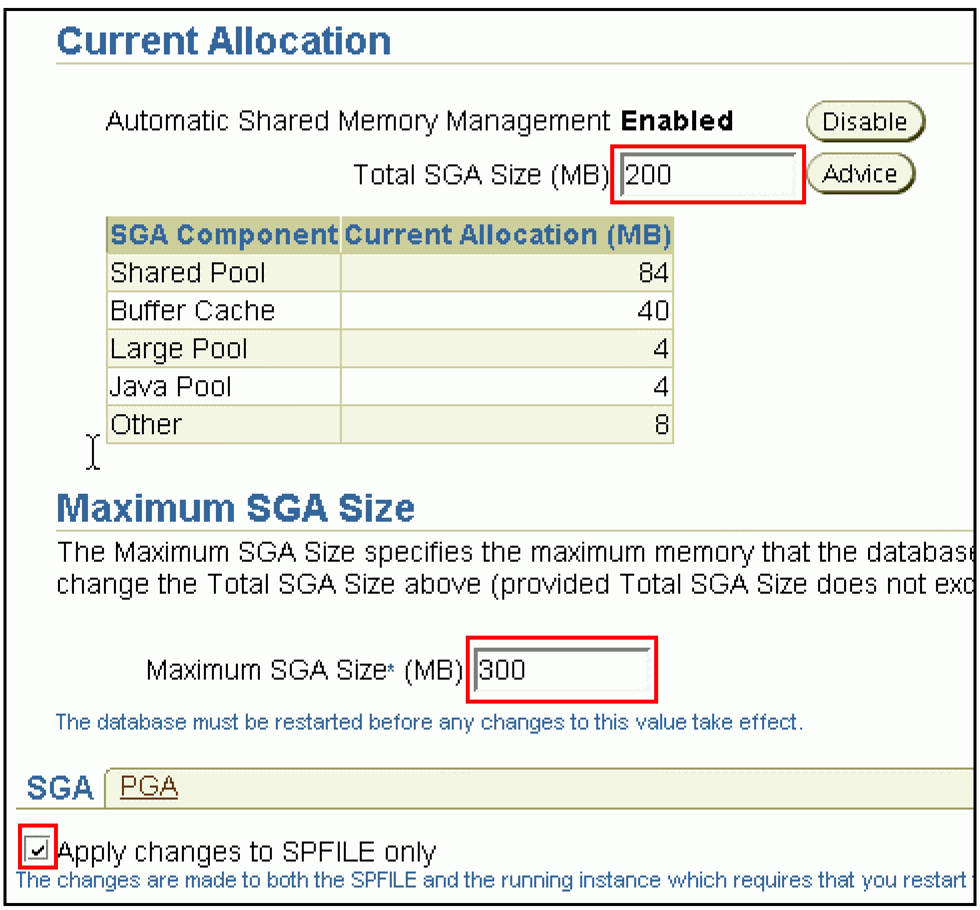
Solutions for Practice 10-2: *Adjusting Memory As Workloads Change*

On the basis of the recommendation of the ADDM report in the previous scenario, the memory constraint was raised to 140 MB. The current application has seen an increase in activity and now another application is being added to the database. With it comes an additional allocation of SGA memory. How do you allocate this additional memory to give the best performance? There is the additional constraint being added that the database must be more available. You are not allowed to shut down during your tuning sessions. You are allowed one shutdown to adjust the `SGA_MAX_SIZE`.

1) You are given an absolute maximum of 300 MB of SGA and are instructed to use as little as required to get reasonable performance. (This is not a well-formed goal.) Set the maximum SGA size to 300 MB. Use a reasonable guess, and set the SGA size to 200 MB for the starting point for the new workload.

   Answer:

   1. Use the Memory Parameters page to set the new values for Maximum SGA Size to 300 MB, and Total SGA Size to 200 MB. Make the changes to the SPFILE only.



2. Restart the instance so the changes are applied.

```
 $ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Mon Dec 5
07:36:57 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.

SQL> startup
ORACLE instance started.

Total System Global Area  146800640 bytes
Fixed Size                  1218220 bytes
Variable Size              88082772 bytes
Database Buffers           54525952 bytes
Redo Buffers                2973696 bytes
Database mounted.
Database opened.
SQL>
```

2) Navigate to the Memory Parameters page and observe the initial memory settings.

Answer:

Buffer Cache is ________ MB.

Shared Pool is _________ MB.

3) Start the workload generator. The new workload is simulated by:
`./workgen 10 2`

Answer:

```
$ ./workgen 10 2

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

$
```

Solutions for Practice 10-2: Adjusting Memory As Workloads Change (continued)

4) Observe the Performance graphs for about 5 minutes and then stop the workload.

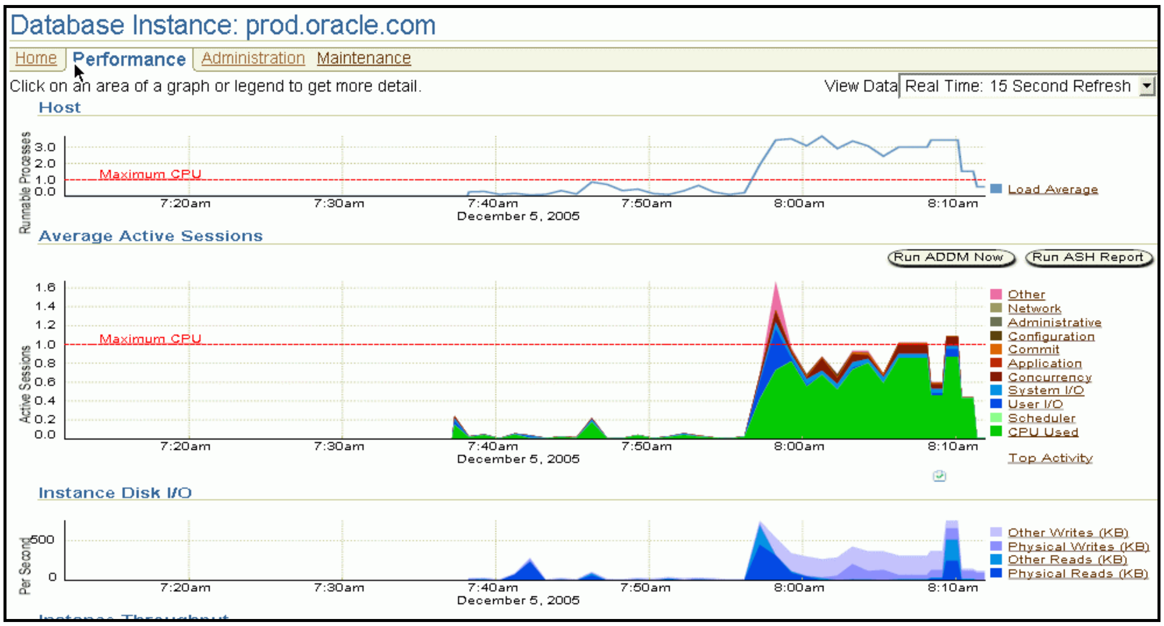
Answer:

```
$ rm runload
rm: remove regular empty file `runload'? y
$
```

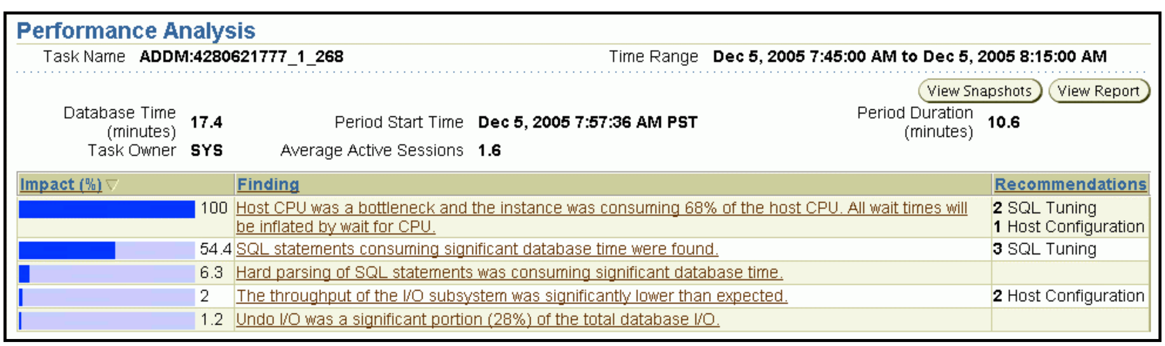
5) What did you observe in the Performance graphs?

Answer:

1. Notice that the Host average load graph, the CPU portion of the Instance Active Sessions graph, and the Instance Disk I/O graph, all have increased values.



2. Create an ADDM report. Depending on the machine type, there are a variety of findings. Hard Parsing shows up in almost all of them. The specific findings and order of the findings will vary.



Performance Analysis

Task Name **ADDM:4280621777_1_268** Time Range **Dec 5, 2005 7:45:00 AM to Dec 5, 2005 8:15:00 AM**

View Snapshots | View Report

Database Time (minutes) **17.4** Period Start Time **Dec 5, 2005 7:57:36 AM PST** Period Duration (minutes) **10.6**

Task Owner **SYS** Average Active Sessions **1.6**

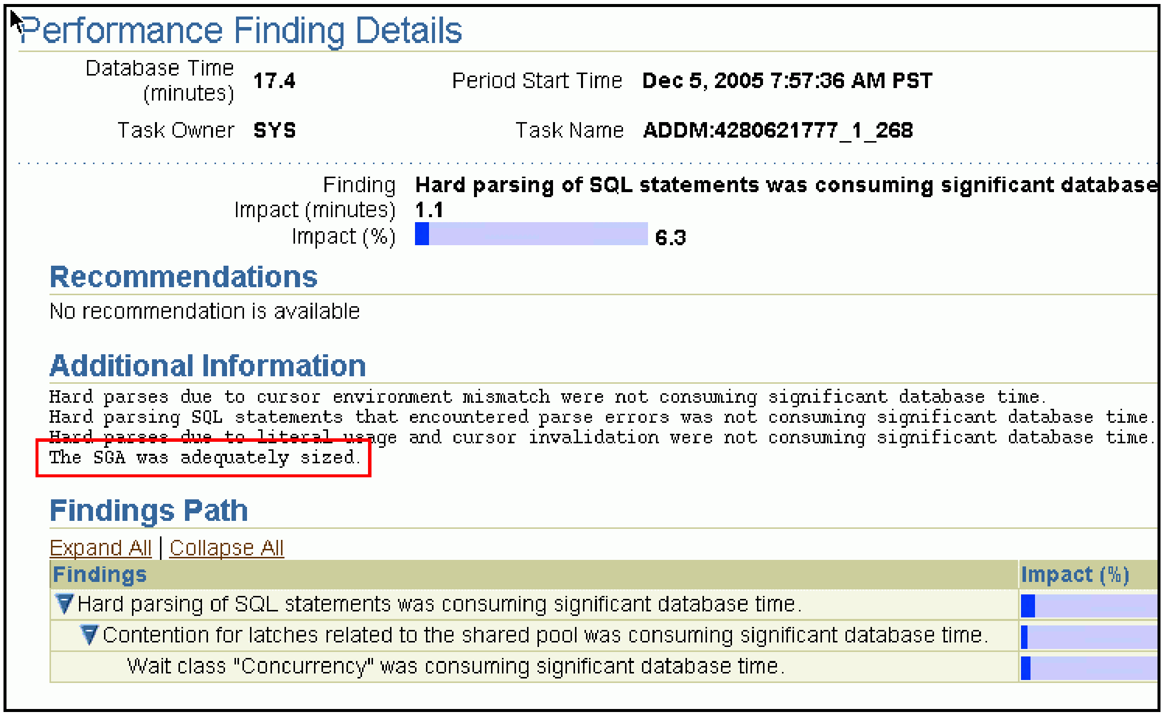
Impact (%)	Finding	Recommendations
100	Host CPU was a bottleneck and the instance was consuming 68% of the host CPU. All wait times will be inflated by wait for CPU.	**2** SQL Tuning **1** Host Configuration
54.4	SQL statements consuming significant database time were found.	**3** SQL Tuning
6.3	Hard parsing of SQL statements was consuming significant database time.	
2	The throughput of the I/O subsystem was significantly lower than expected.	**2** Host Configuration
1.2	Undo I/O was a significant portion (28%) of the total database I/O.	

6) View the ADDM findings. Is there a recommendation for SGA size?

Solutions for Practice 10-2: Adjusting Memory As Workloads Change (continued)

Answer:

1. The finding "Hard parse of SQL statements was consuming significant database time." is affected by the size of the SGA. Drill down to the details of this finding.
2. The details say that the SGA is adequately sized.

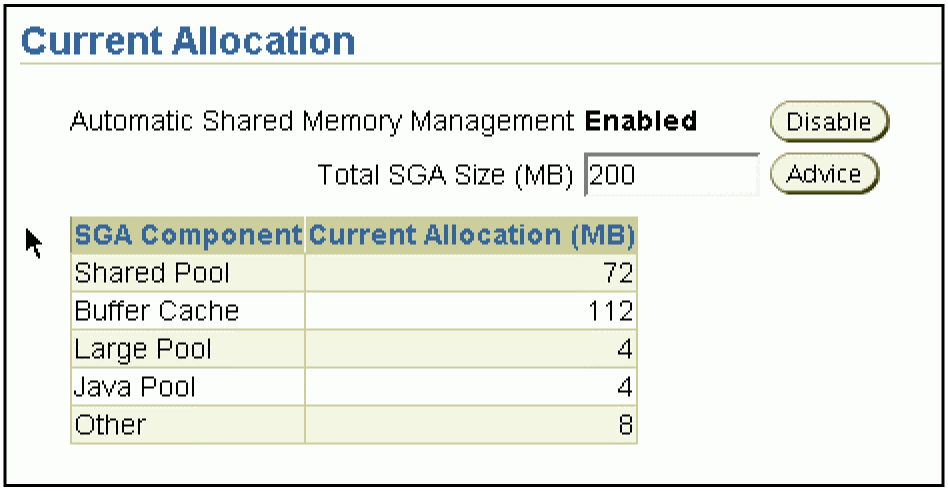


7) Check the changes to the memory parameters during the time the workload was running. What were the final buffer cache size and shared pool size? How much tuning was required to find an adequately sized SGA?

Answer:

1. The Buffer Cache ended at 112 MB and the Shared Pool at 72 MB in the example. The values on your machine are:
   Buffer Cache: ___________MBytes
   Shared Pool: ___________MBytes

Solutions for Practice 10-2: Adjusting Memory As Workloads Change (continued)



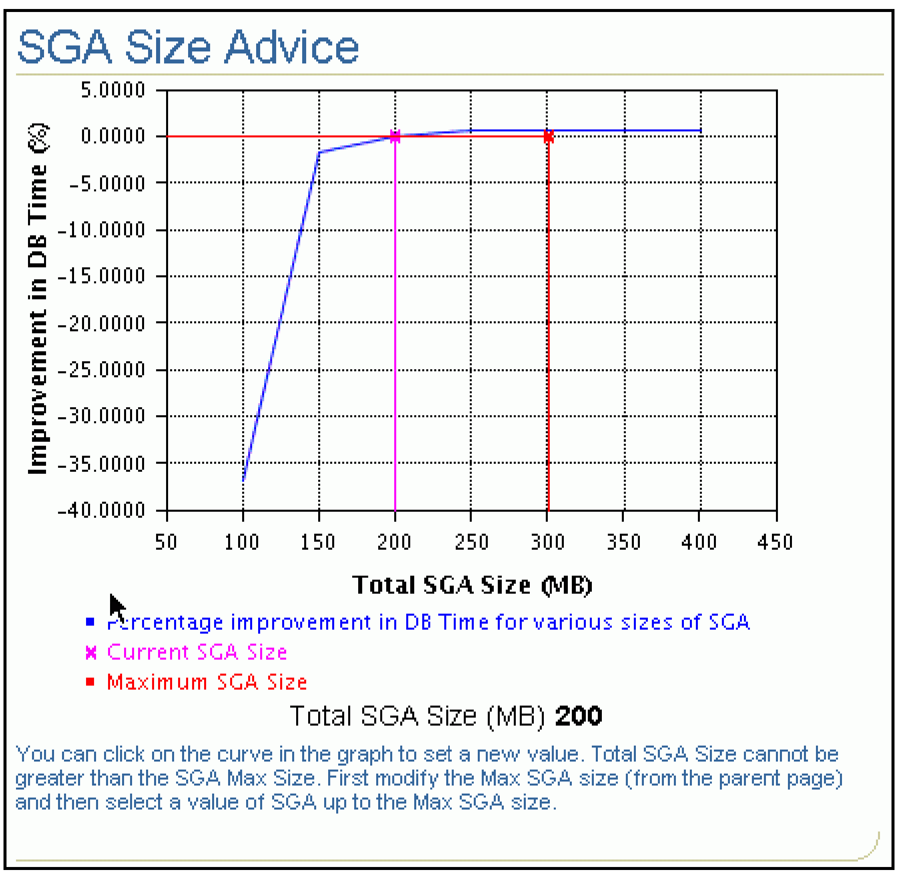
SGA Component	Current Allocation (MB)
Shared Pool	72
Buffer Cache	112
Large Pool	4
Java Pool	4
Other	8

2. The adequately sized SGA was automatically sized based on the `SGA_TARGET_SIZE` parameter, set by Total SGA Size in EM.

8) How would you find a possible smaller SGA size that would still be adequate?

Answer:

1. Check the SGA Size Advisor on the EM Memory Parameters page. It shows that there would be some impact by reducing the SGA size to 150 MB, and very little gain by increasing the size above 200 MB.



Solutions for Practice 10-2: Adjusting Memory As Workloads Change (continued)

2. Create an AWR report from the last two snapshots and use the Buffer Pool Advisor to find another trial value for SGA_TARGET_SIZE. Based on the current settings, a slight reduction in the Buffer Cache would not impact performance much. If memory was severely constrained, reducing the SGA_TARGET_SIZE by 20 to 30 MB could be justified from this report.

Buffer Pool Statistics

- Standard block size Pools D: default, K: keep, R: recycle
- Default Pools for other block sizes: 2k, 4k, 8k, 16k, 32k

P	Number of Buffers	Pool Hit%	Buffer Gets	Physical Reads	Physical Writes	Free Buff Wait	Writ Comp Wait	Buffer Busy Waits
D	12,974	100	13,987,385	3,008	3,269	0	0	12
K	499		0	0	0	0	0	0

Buffer Pool Advisory

- Only rows with estimated physical reads >0 are displayed
- ordered by Block Size, Buffers For Estimate

P	Size for Est (M)	Size Factor	Buffers for Estimate	Est Phys Read Factor	Estimated Physical Reads
D	8	0.08	998	67.00	1,059,988
D	16	0.15	1,996	7.35	116,263
D	24	0.23	2,994	2.92	46,143
D	32	0.31	3,992	1.43	22,648
D	40	0.38	4,990	1.25	19,844
D	48	0.46	5,988	1.13	17,887
D	56	0.54	6,986	1.07	16,908
D	64	0.62	7,984	1.03	16,316
D	72	0.69	8,982	1.02	16,098
D	80	0.77	9,980	1.01	15,953
D	88	0.85	10,978	1.01	15,929
D	96	0.92	11,976	1.00	15,820
D	104	1.00	12,974	1.00	15,820
D	112	1.08	13,972	1.00	15,820
D	120	1.15	14,970	1.00	15,820
D	128	1.23	15,968	1.00	15,820
D	136	1.31	16,966	1.00	15,820
D	144	1.38	17,964	1.00	15,820
D	152	1.46	18,962	1.00	15,820
D	160	1.54	19,960	1.00	15,820

Solutions for Practice 10-2: Adjusting Memory As Workloads Change (continued)

9) Exceute a cleanup script before the next practice; run `./cleanup 10 2`.

Answer:

```
$ ./cleanup 10 2

Finished Cleanup 10-2
```

Practice Solutions for Lesson 11

In this practice, you observe the symptoms of checkpoint and redo performance issues. You also diagnose and correct those issues.

Solutions for Practice 11-1: *Diagnosing Checkpoint and Redo*

Use a workload generator to create a scenario that demonstrates checkpoint and redo tuning. This scenario uses the Resource Manager to limit the active session pool. The Resource Manager prevents the workload scripts from overloading the CPU and reduces contention in the database for buffer cache blocks, so that the same scripts run with similar results on different type of machines. You will see waits for the Resource Manager as shown in the Scheduler wait class.

1) Set up the database by running the `./setup 11 1` script from the `/home/oracle/workshops` directory.

   Answer:

```
 $ ./setup 11 1

System altered.


PL/SQL procedure successfully completed.

ORACLE instance shut down.

ORACLE instance started.

Total System Global Area  314572800 bytes
Fixed Size                  1219184 bytes
Variable Size             197133712 bytes
Database Buffers          113246208 bytes
Redo Buffers                2973696 bytes
Database mounted.
Database opened.
--   Target database:   prod.oracle.com
--   Script generated at:       28-JAN-2006   11:54
Starting reorganization
Executing as user: SYS
ALTER TABLE "OE"."INVENTORIES" MOVE  PCTFREE 50
ALTER INDEX "OE"."INVENTORY_IX" REBUILD
ALTER INDEX "OE"."INV_PRODUCT_IX" REBUILD
BEGIN DBMS_STATS.GATHER_TABLE_STATS('"OE"',
'"INVENTORIES"',
estimate_percent=>NULL, cascade=>TRUE); END;
Completed Reorganization.  Starting cleanup phase.
Starting cleanup of recovery tables
Completed cleanup of recovery tables
Starting cleanup of generated procedures
Completed cleanup of generated procedures
Script execution complete

Table altered.


Table altered.
```

```

Index altered.


Index altered.


Index altered.


Sequence altered.

Finished Setup 11-1
$
```

2) Start a workload by running: `./workgen 11 1`

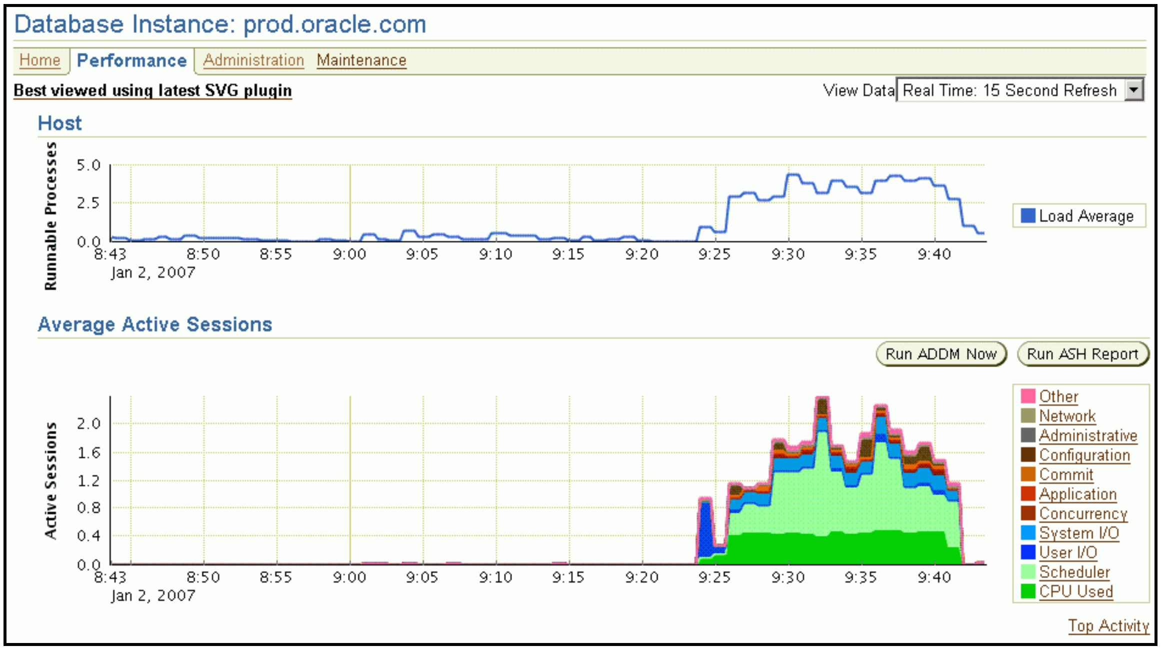
   Answer:

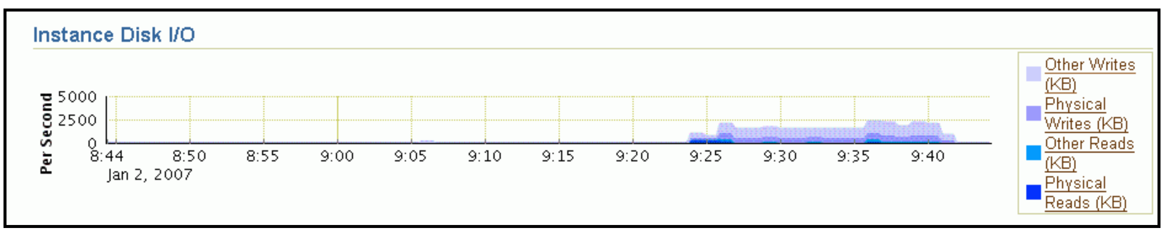
```
 $ ./workgen 11 1

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

 $
```

3) Observe the graphs on the Performance page. High I/O rates are always a pointer to a possible performance issue. What processes are contributing to the high rate of writes?

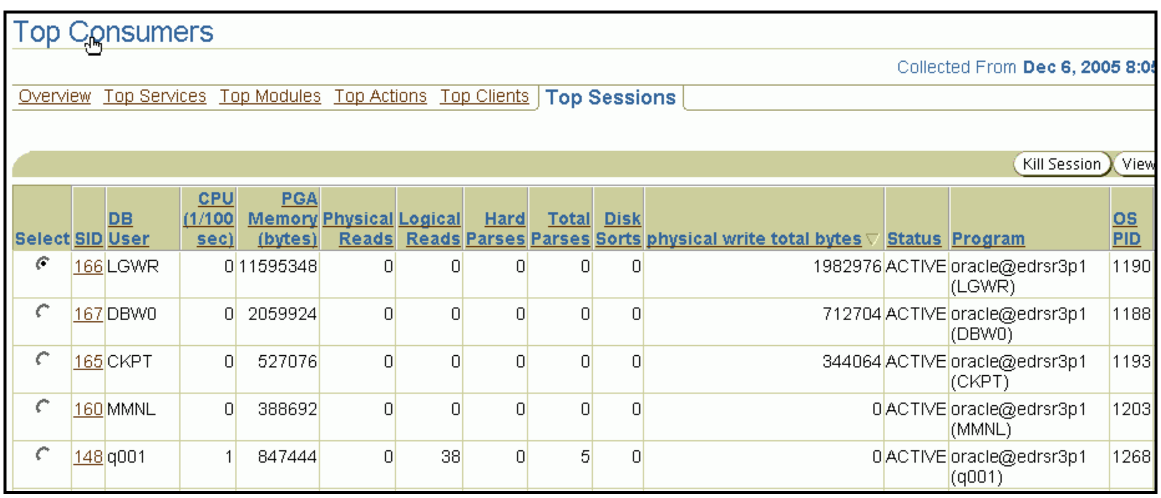
   Answer:

   1. View the graphs on the Performance page. Notice that the Instance Disk I/O graph has a high rate of Other Writes. The scale on the Instance Disk I/O graph may vary from the one shown.





2. Click the Other Writes link to view the processes that are producing the writes. Notice that LGWR, CKPT, and DBW0 are producing the writes.



Select	SID	DB User	CPU (1/100 sec)	PGA Memory (bytes)	Physical Reads	Logical Reads	Hard Parses	Total Parses	Disk Sorts	physical write total bytes	Status	Program	OS PID
◉	166	LGWR	0	11595348	0	0	0	0	0	1982976	ACTIVE	oracle@edrsr3p1 (LGWR)	1190
○	167	DBW0	0	2059924	0	0	0	0	0	712704	ACTIVE	oracle@edrsr3p1 (DBW0)	1188
○	165	CKPT	0	527076	0	0	0	0	0	344064	ACTIVE	oracle@edrsr3p1 (CKPT)	1193
○	160	MMNL	0	388692	0	0	0	0	0	0	ACTIVE	oracle@edrsr3p1 (MMNL)	1203
○	148	q001	1	847444	0	38	0	5	0	0	ACTIVE	oracle@edrsr3p1 (q001)	1268

4) After about 5 minutes, stop the workload.

Answer:

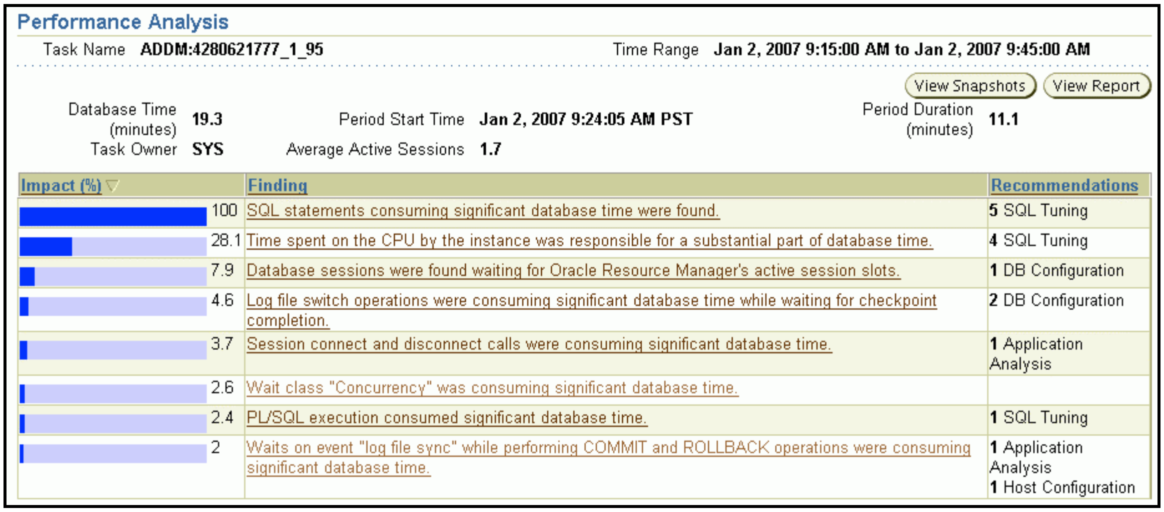
```
 $ rm runload
rm: remove regular empty file `runload'? y
```

Solutions for Practice 11-1: Diagnosing Checkpoint and Redo (continued)

5) Create an ADDM report.

Answer:

Create the ADDM report. Notice the following two wait events:
“Log file switch operations…” and “Waits on event “log file sync…”
There may also be findings concerning contention on buffer cache–related events such as latches, hot blocks, and general concurrency.



6) Create an AWR report.

Answer:

This follows the same procedure as the solution to Practice 8-1 step 6.

7) Review the AWR report.

a) Review the Load profile. What kind of load is being performed?

Answer:

The Redo size statistic dominates this load profile. This workload is doing a large number of DML operations. This load could be considered as an OLTP load.

Report Summary

Cache Sizes

	Begin	End		
Buffer Cache:	112M	116M	Std Block Size:	8K
Shared Pool Size:	72M	68M	Log Buffer:	7,000K

Load Profile

	Per Second	Per Transaction
Redo size:	1,277,897.87	2,516.35
Logical reads:	25,991.50	51.18
Block changes:	10,135.70	19.96
Physical reads:	7.32	0.01
Physical writes:	126.96	0.25
User calls:	121.47	0.24
Parses:	165.70	0.33
Hard parses:	1.24	0.00
Sorts:	638.93	1.26
Logons:	4.11	0.01
Executes:	2,836.89	5.59
Transactions:	507.84	

% Blocks changed per Read:	39.00	Recursive Call %:	97.55
Rollback per transaction %:	0.00	Rows per Sort:	3.86

b) Review the Top 5 Timed Events. What are the top events? Are these events expected for this type of workload?

Answer:

The Resource Manager is collecting a large number of waits because it limits the active session pool, forcing sessions to wait to submit the next SQL request. The large amount of redo is causing a large amount of redo to be written to the log file, so the bottleneck is indicated by log file parallel write and log file switch completion. With so much redo log information being written, the question can be raised: "How often are the log switches occurring?"

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
resmgr:cpu quantum	2,471	413	167	35.7	Scheduler
CPU time		295		25.5	
log file parallel write	158,072	107	1	9.3	System I/O
resmgr:become active	210	92	436	7.9	Scheduler
log file switch (checkpoint incomplete)	96	40	421	3.5	Configuration

c) Review the Time Model Statistics section. Does this section give more insight?

Answer:

The Time Model System Stats is not very helpful. If the waits are because of redo issues, the LGWR time is hidden in the background elapsed time.

Time Model Statistics

- Total time in database user-calls (DB Time): 1157.6s
- Statistics including the word "background" measure background
- Ordered by % or DB time desc, Statistic name

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	1,022.39	88.32
DB CPU	294.67	25.46
connection management call elapsed time	42.98	3.71
PL/SQL execution elapsed time	35.35	3.05
parse time elapsed	18.24	1.58
hard parse elapsed time	12.13	1.05
sequence load elapsed time	10.46	0.90
PL/SQL compilation elapsed time	1.25	0.11
hard parse (sharing criteria) elapsed time	0.06	0.01
repeated bind elapsed time	0.04	0.00
hard parse (bind mismatch) elapsed time	0.00	0.00
DB time	1,157.56	
background elapsed time	144.09	
background cpu time	30.33	

d) What sections would you check next?

Answer:

1. Scan the wait statistics for log file–related waits. There are several. Some did not make the top five timed events. The Instance Activity Statistics also point to a redo problem.

Wait Events

- s - second
- cs - centisecond - 100th of a second
- ms - millisecond - 1000th of a second
- us - microsecond - 1000000th of a second
- ordered by wait time desc, waits desc (idle events last)

Event	Waits	%Time -outs	Total Wait Time (s)	Avg wait (ms)	Waits /txn
resmgr:cpu quantum	2,471	0.00	413	167	0.01
log file parallel write	158,072	0.00	107	1	0.73
resmgr:become active	210	0.00	92	436	0.00
log file switch (checkpoint incomplete)	96	27.08	40	421	0.00
log file sync	1,904	0.00	23	12	0.01
db file sequential read	3,158	0.00	15	5	0.01
buffer busy waits	446	0.00	13	30	0.00
log file switch completion	222	0.90	12	56	0.00
buffer deadlock	21	100.00	8	376	0.00
latch: library cache	183	0.00	8	41	0.00
enq: SQ - contention	9	0.00	6	679	0.00
latch: enqueue hash chains	25	0.00	4	147	0.00
enq: HW - contention	26	0.00	3	135	0.00
control file parallel write	1,163	0.00	3	3	0.01
LGWR wait for redo copy	1,217	9.45	3	3	0.01

2. Examine the Instance Activity Stats – Thread Activity section, to determine the log file switch rate. This example of an AWR report shows a per hour rate of 314. You could also examine the alert log; a sample of an alert log is showing log switches about every 90 seconds, which is about 40 per hour.

Instance Activity Stats - Thread Activity

- Statistics identified by '(derived)' come from sources other than SYSSTAT

Statistic	Total	per Hour
log switches (derived)	58	314.57

```
alert_prod.log
Tue Dec  6 21:09:42 2005
Thread 1 advanced to log sequence 117
  Current log# 6 seq# 117 mem# 0:
/u01/app/oracle/oradata/prod/disk1/redo06.log
Tue Dec  6 21:11:12 2005
Thread 1 advanced to log sequence 118
```

```
  Current log# 4 seq# 118 mem# 0:
/u01/app/oracle/oradata/prod/disk1/redo04.log
Tue Dec  6 21:12:40 2005
Thread 1 advanced to log sequence 119
  Current log# 5 seq# 119 mem# 0:
/u01/app/oracle/oradata/prod/disk1/redo05.log
Tue Dec  6 21:14:11 2005
Thread 1 advanced to log sequence 120
  Current log# 6 seq# 120 mem# 0:
/u01/app/oracle/oradata/prod/disk1/redo06.log
Tue Dec  6 21:15:41 2005
Thread 1 advanced to log sequence 121
  Current log# 4 seq# 121 mem# 0:
/u01/app/oracle/oradata/prod/disk1/redo04.log
Tue Dec  6 21:17:13 2005
Thread 1 advanced to log sequence 122
  Current log# 5 seq# 122 mem# 0:
/u01/app/oracle/oradata/prod/disk1/redo05.log
Tue Dec  6 21:18:38 2005
Thread 1 advanced to log sequence 123
  Current log# 6 seq# 123 mem# 0:
/u01/app/oracle/oradata/prod/disk1/redo06.log
```

3. Examine the redo log file size in `V$LOG`. The sizes of the log files are 10 MB each.

```
SQL> select * from v$log;

    GROUP#    THREAD#  SEQUENCE#      BYTES    MEMBERS ARC STATUS
---------- ---------- ---------- ---------- ---------- --- ------------
FIRST_CHANGE# FIRST_TIM
------------- ---------
         4          1        221   10485760          1 NO  ACTIVE
      7307988 28-JAN-06

         5          1        222   10485760          1 NO  CURRENT
      7316888 28-JAN-06

         6          1        220   10485760          1 NO  INACTIVE
      7298858 28-JAN-06


SQL>
```

8) Based on the finding of many log switches and the recommendation that the log should switch about three to four times per hour, what is the next tuning step?

Answer:

Increase the size of the redo log files. To find the optimum size, calculate 10 MB × number of switches per hour / 4 switches per hour. The size that you find may be quite large. To create a consistent practice, run the `bigredo.sql` SQL script to create a set of large redo logs (200 MB each). This size may not be the optimum size that you found, but it will show the impact of increasing the log file size.

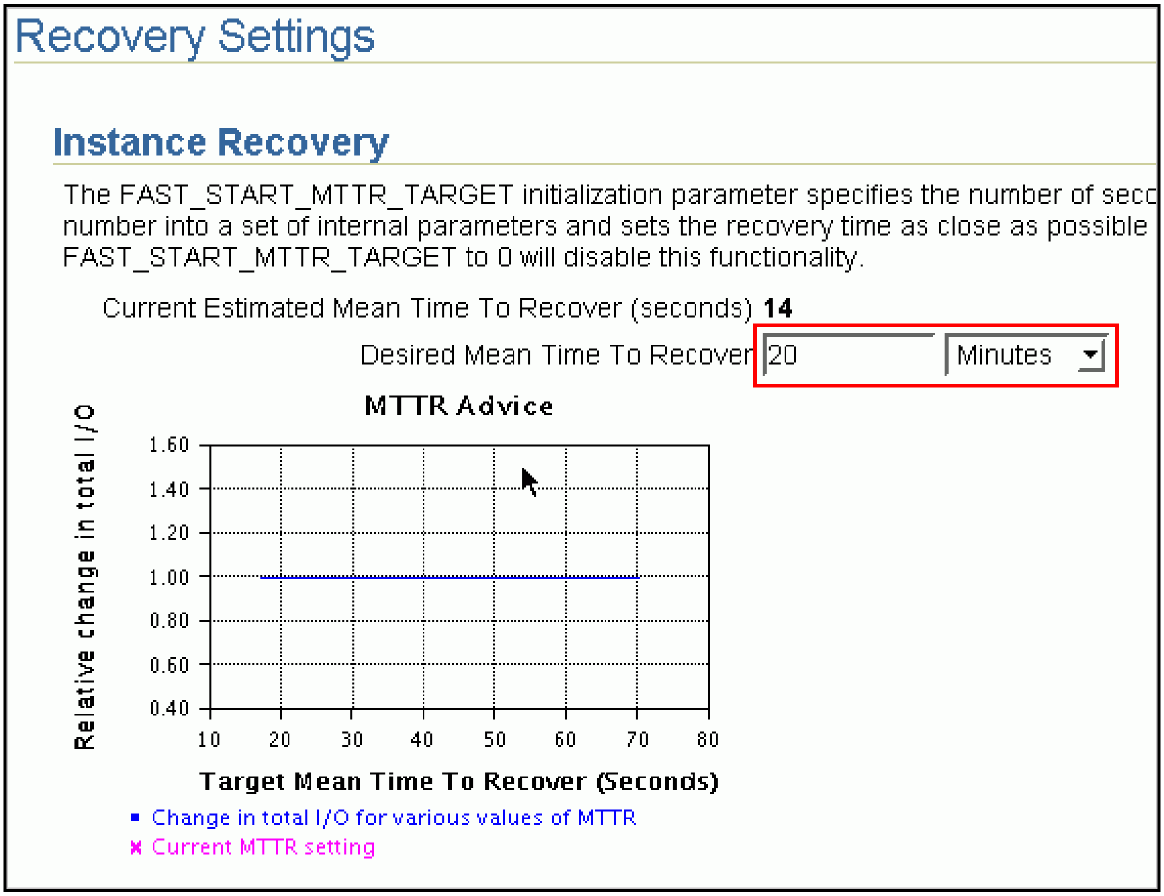
```
 $ sqlplus / as sysdba @bigredo

SQL*Plus: Release 10.2.0.1.0 - Production on Tue Jan 2
11:13:11 2007

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

Connected.
SQL>
SQL> ALTER DATABASE ADD LOGFILE GROUP 7
'/u01/app/oracle/oradata/prod/disk1/redo07.log' SIZE 200M
REUSE;

Database altered.

SQL>
SQL> execute DBMS_LOCK.SLEEP(20);

PL/SQL procedure successfully completed.

SQL>
SQL> ALTER DATABASE ADD LOGFILE GROUP 8
'/u01/app/oracle/oradata/prod/disk1/redo08.log' SIZE 200M
REUSE;

Database altered.

SQL>
SQL> execute DBMS_LOCK.SLEEP(20);

PL/SQL procedure successfully completed.

SQL>
SQL> ALTER DATABASE ADD LOGFILE GROUP 9
'/u01/app/oracle/oradata/prod/disk1/redo09.log' SIZE 200M
REUSE;

Database altered.

SQL>
SQL> execute DBMS_LOCK.SLEEP(20);

PL/SQL procedure successfully completed.

SQL>
SQL> ALTER SYSTEM SWITCH LOGFILE;
```

```
System altered.

SQL> execute DBMS_LOCK.SLEEP(5);

PL/SQL procedure successfully completed.

SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.

SQL> execute DBMS_LOCK.SLEEP(5);

PL/SQL procedure successfully completed.

SQL> ALTER SYSTEM SWITCH LOGFILE;

System altered.

SQL>
SQL> execute DBMS_LOCK.SLEEP(10);

PL/SQL procedure successfully completed.

SQL>
SQL> ALTER SYSTEM CHECKPOINT;

System altered.

SQL>
SQL> ALTER DATABASE DROP LOGFILE GROUP 4;

Database altered.

SQL> ALTER DATABASE DROP LOGFILE GROUP 5;

Database altered.

SQL> ALTER DATABASE DROP LOGFILE GROUP 6;

Database altered.

SQL>
SQL> exit
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
```

9) The service level agreement requires a recovery time of 20 minutes. Find the current recovery time and adjust the `FAST_START_MTTR_TARGET` to 1200 seconds.

Answer:

In Enterprise Manager, navigate to the Recovery Settings page. From the Database Home page, click the Maintenance tab and then Recovery Settings in the Backup/Recovery Settings section. Notice the Current Estimated Mean Time to Recover. __________. Change the setting to 20 Minutes and click Apply.



10) Start the workload generator again: `./workgen 11 1`

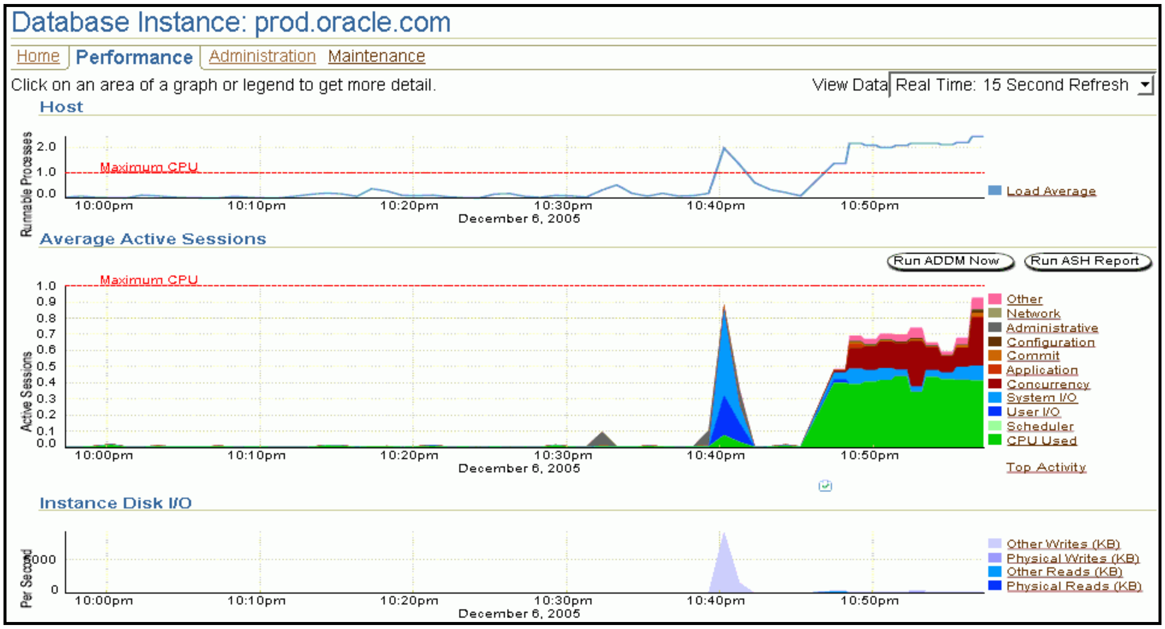
Answer:

```
$ ./workgen 11 1

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

$
```

11) Observe the graphs on the Performance page.

Answer:

The Host load average graph shows a much-reduced level. The Average Active Sessions has a lower peak and a more uniform level of activity.



12) After about 5 minutes, stop the workload.

Answer:

```
 $ rm runload
rm: remove regular empty file `runload'? y
```

13) Create an AWR report.

Answer:

This follows the same procedure as the solution to Practice 8-1 step 6.

14) Review the AWR report.

a) Review the Load Profile. Have the load characteristics changed?

Answer:

In the example, the load profile is almost the same. The example report shows an increase in the amount of useful work. The number of transactions per second has increased and the amount of redo generated per second has increased. Your results may vary and may not show any significant increase. The automatic checkpoint tuning provided in Oracle Database 10*g* reduces the need to tune redo log file size as long as the files are adequately sized.

Report Summary

Cache Sizes

	Begin	End		
Buffer Cache:	120M	112M	Std Block Size:	8K
Shared Pool Size:	64M	72M	Log Buffer:	7,000K

Load Profile

	Per Second	Per Transaction
Redo size:	1,564,485.93	2,334.14
Logical reads:	32,080.46	47.86
Block changes:	13,025.90	19.43
Physical reads:	1.67	0.00
Physical writes:	83.14	0.12
User calls:	157.20	0.23
Parses:	147.87	0.22
Hard parses:	1.81	0.00
Sorts:	839.40	1.25
Logons:	5.36	0.01
Executes:	3,667.38	5.47
Transactions:	670.26	

b) Review the Top 5 Timed Events section. What has changed?

Answer:

The order and the specific events will change. In the example, the checkpoint incomplete event has dropped off the list, and log file parallel write is taking more time.

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
resmgr:cpu quantum	5,452	1,006	184	40.2	Scheduler
CPU time		705		28.2	
log file parallel write	325,607	351	1	14.1	System I/O
resmgr:become active	216	88	408	3.5	Scheduler
log file sync	4,786	72	15	2.9	Commit

c) Does the Time Model give any additional information?

Answer:

No.

15) Clean up by running: `./cleanup 11 1`

Answer:

```
$ ./cleanup 11 1
System altered.


PL/SQL procedure successfully completed.

Finished cleanup 11-1
$
```

Practice Solutions for Lesson 12

This lesson has a demonstration of diagnosing I/O performance problems and possible changes to mitigate those problems.

Solutions for Practice 12-1: *Tuning I/O: A Demonstration*

This practice consists of a series of viewlets. These viewlets display some relatively severe I/O performance issues. The system has been tuned to eliminate as many other issues as possible. Automatic shared memory management is enabled with as `SGA_MAX_SIZE` and `SGA_TARGET` set to 600 MB. The demonstration system has a slower CPU and disk drives than are available in most systems, but the relative improvements in I/O performance are valid for any system. The demonstration system has the following specifications:

- RH AS 4
- 2 CPU INTEL 386 – 666 MHz
- 1 GB RAM
- 2 SCSI-3 controllers with a 40 MB/sec Transfer Rate
- 4 SCSI-3 disk drives 18 GB each, with a 40 MB/sec transfer rate
- The disks are assigned 2 per controller
- Each disk is partitioned into four partitions of 4 GB each.
- The first partition on each disk (the fastest), located on the outside edge of the disk, is not used.
- The second partition of each disk is assigned to the ASM disk group +DG2.
- The third partition of each disk is mounted as /a3, /b3, /c3, and /d3. These are the partitions used for the single disk and manual striping demonstrations
- The fourth partitions on each disk are only used for incidental activity, such as the flash recovery area. These partitions are the slowest.
- The insert activity is against the `LOADTEST*` tablespaces There are 10 tablespaces. The setup script for each test drops and recreates the `LOADTEST*` tablespaces.
- There will be significant undo and redo generated during each test.
- The source of data for the inserts is the example tablespace. There will be significant reads against the `EXAMPLE` tablespace.

The workload consists of 10 sessions each simulating a different user each user is assigned to a different tablespace. Each user creates tables from existing tables in the `EXAMPLE` tablespace, and then inserts additional rows, updates those rows, and deletes a fraction of the rows. Each user runs the workload once and exits. The start time and end time for each user session is recorded in the table `SYSTEM.CYCLES`. The elapsed time for each session, and the average time for all session provide a clear indicator of the performance differences.

In a terminal window issue the command:

```
mozilla file:///home/oracle/labs/lab_12_01.html
```

Click the links in this file to play the viewlets.

Solutions for Practice 12-1: Tuning I/O: A Demonstration (continued)

1) View the I/O Performance Test with Single Disk (`SINGLE_DISK_PERF_TEST`). The configuration for this test has all control files and data files on one disk (/a3) and the redo log files on another disk (/b3). Notice the ADDM findings record the redo average write speed _______________ and the data file write throughput speed_________________. Record the average session time _______________.

2) View the I/O Performance Test with Manual Striping (`MANUAL_STRIPE_PERF_TEST`). The configuration for this test has the `SYSTEM`, `UNDO`, `SYSAUX`, and `EXAMPLE` tablespaces each on a different disk. The four redo log files are each on a different disk. The `LOADTEST*` tablespaces each have four data files; they are arranged so that the first data file of each is on a different disk. When there are multiple data files, each new extent that is allocated will be allocated on the next data file. The behavior of extent allocation will give a striping across all the data files for the `LOADTEST*` tablespaces. Notice the ADDM findings record the redo average write speed _______________ and the data file write throughput speed_________________. Record the average session time _______________.

3) View the I/O Performance Test with ASM. (`ASM_PERF_TEST`). Notice the ADDM findings record the redo average write speed _______________ and the data file write throughput speed_________________. Record the average session time _______________.

4) What conclusions can you draw from these demonstrations?

   Answer:

   1. Manual striping gives a significant benefit over a single disk by increased bandwidth. Manual striping can be labor intensive, requiring down time to reorganize the data for best performance.
   2. ASM management uses the available disk bandwidth effectively by spreading the data layout across all the available disks.

Practice Solutions for Lesson 13

In this practice, you use the appropriate tools to adjust the PGA memory usage and temporary tablespace properties for improved performance.

Solutions for Practice 13-1: *Tuning `PGA_AGGREGATE_TARGET`*

You have a DSS system that has a variety of queries being performed. Users are complaining that the queries are slow. `PGA_AGGREGATE_TARGET` is set at 20 MB. Determine the smallest setting that eliminates all multipass work areas.

1) Be sure that the environment variables are set to access the `prod` database and the working directory is `/home/oracle/workshops`. Execute the setup script for this practice: `./setup 13 1`.

   Answer:

   Change directory to `/home/oracle/workshops`. Set the environment variables with `oraenv`. Execute `./setup 13 1`.

```
$ cd /home/oracle/workshops
$ . oraenv
ORACLE_SID = [prod] ? prod
$ ./setup 13 1
TZ set to US/Pacific
Oracle Enterprise Manager 10g Database Control Release
10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation.  All rights
reserved.
http://edrsr9p1.us.oracle.com:5500/em/console/aboutApplica
tion
Stopping Oracle Enterprise Manager 10g Database Control
...
 ...  Stopped.

ORACLE instance shut down.

ORACLE instance started.

Total System Global Area  314572800 bytes
Fixed Size                  1219184 bytes
Variable Size             209716624 bytes
Database Buffers          100663296 bytes
Redo Buffers                2973696 bytes
Database mounted.
Database opened.

TZ set to US/Eastern
Oracle Enterprise Manager 10g Database Control Release
10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation.  All rights
reserved.
http://edt3r20p1.us.oracle.com:5500/em/console/aboutApplic
ation
Starting Oracle Enterprise Manager 10g Database Control
....................... started.
----------------------------------------------------------
```

Solutions for Practice 13-1: Tuning PGA_AGGREGATE_TARGET (continued)

```
Logs are generated in directory
/u01/app/oracle/product/10.2.0/db_1/edt3r20p1.us.oracle.co
m_prod/sysman/log

Finished setup 13-1
$
```

2) Start the workload generator with `./workgen 13 1`. This workload consists of several sessions doing a variety of queries that require different sizes of work areas.

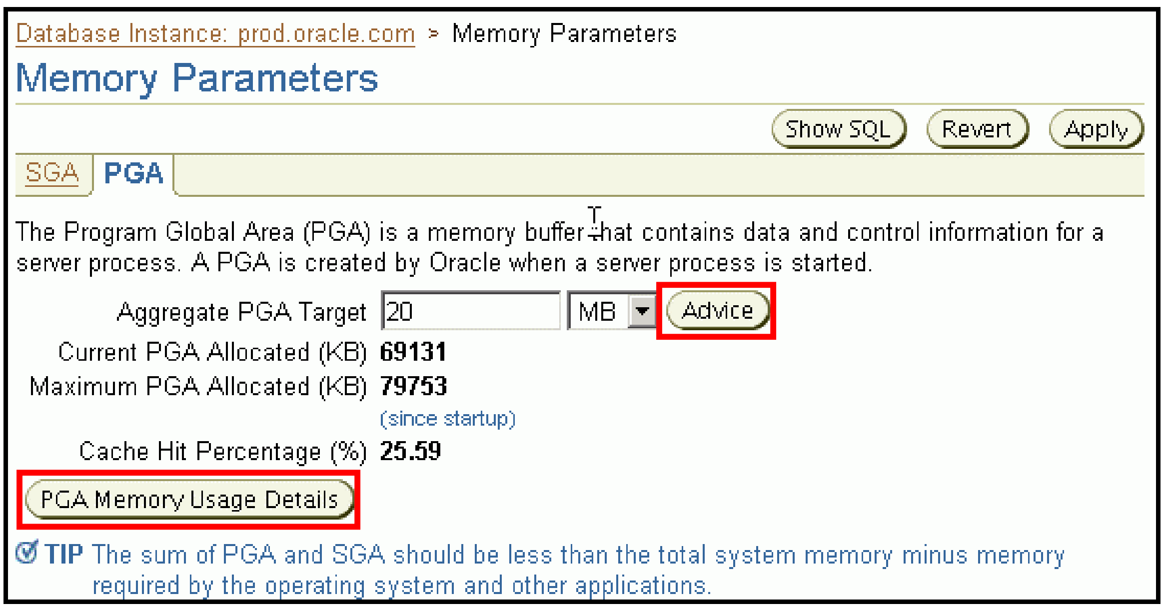
   Answer:

```
$ ./workgen 13 1

PL/SQL procedure successfully completed.

$
```

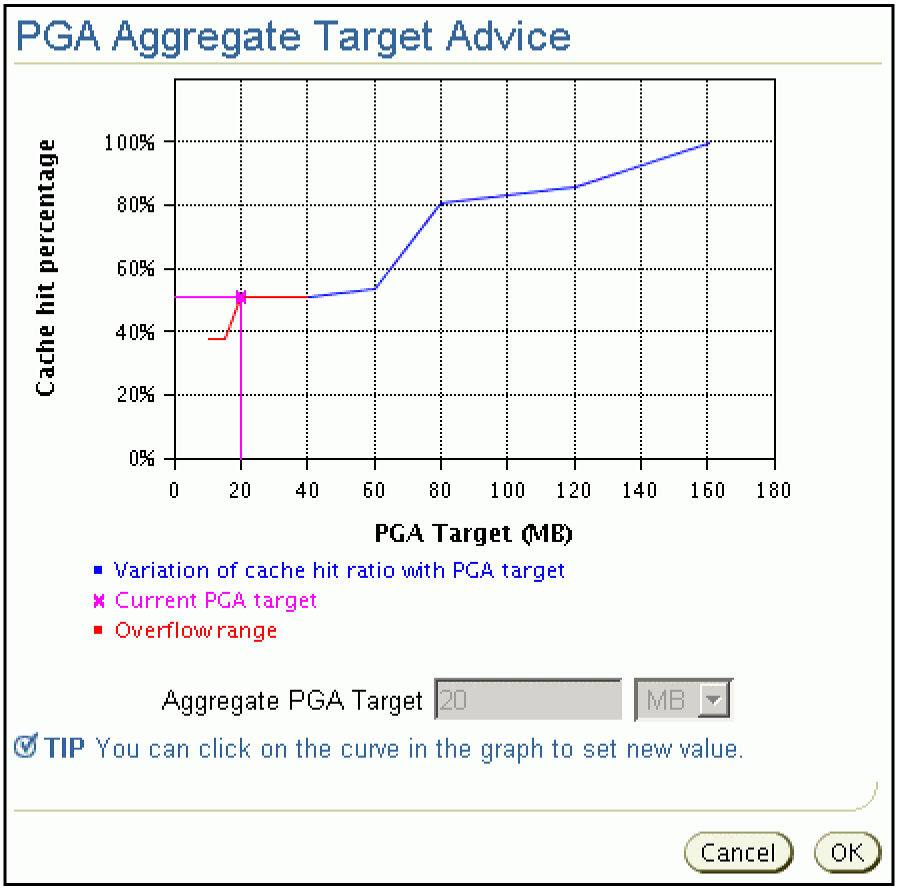
3) Wait a minute or two and then, while the workload is running, use the PGA Advisor and PGA Memory Usage Details to select a new `PGA_AGGREGATE_TARGET`.

   Answer:

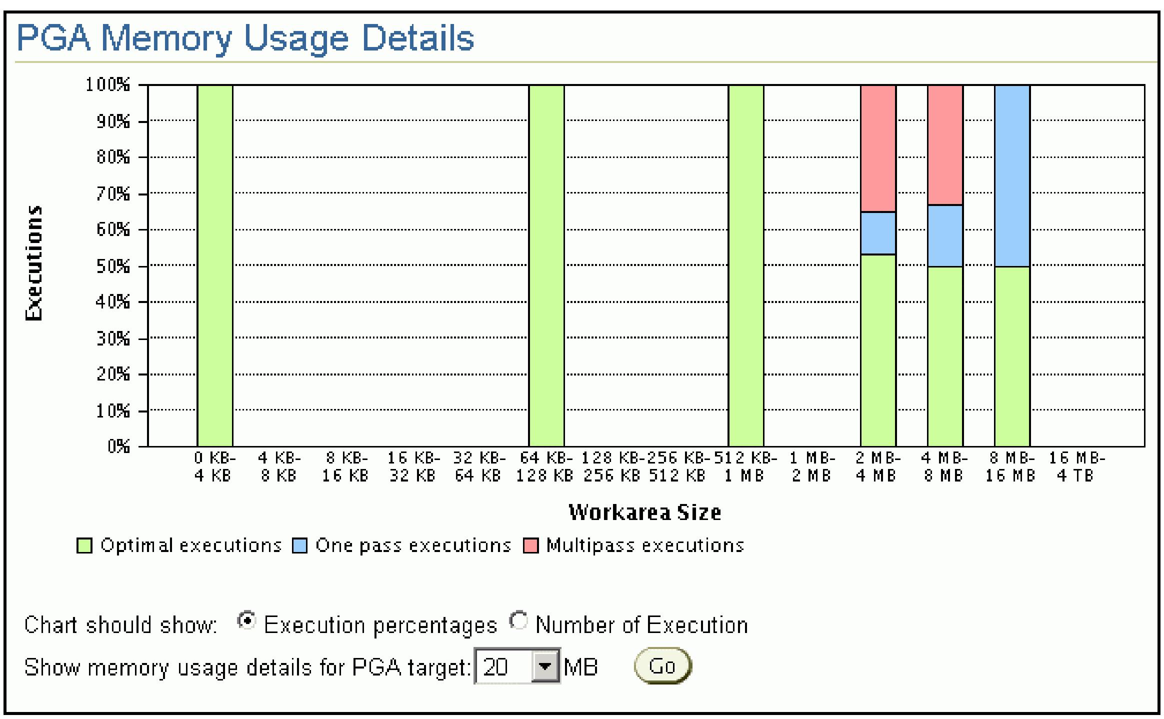
   1. Navigate to the PGA Memory Parameter page. Click the Administration tab, and then Memory Parameters. Then click the PGA tab on the Memory Parameters page.



   2. On the PGA Aggregate Target Advice page, click Advice to view the advisor recommendations. Notice the first inflection point is at 80 MB. This is the point where the multipass work areas will be eliminated.

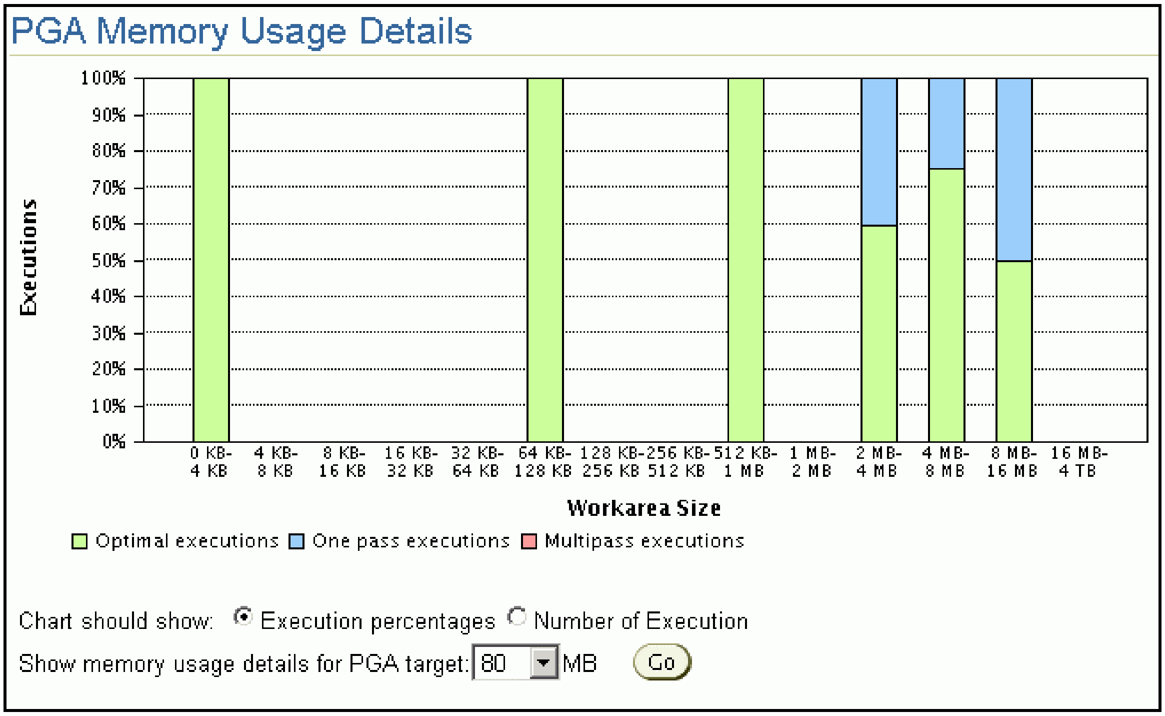


3. Return to the PGA Memory Parameters Page. Click PGA Memory Usage Details.



Solutions for Practice 13-1: Tuning PGA_AGGREGATE_TARGET (continued)

4. Use the value of the first inflection point on the advice graph to estimate the work area usage. The PGA Memory Usage Details shows no multipass work areas at this point. You will use 80 MB for the value of `PGA_AGGREGATE_TARGET` on the next workload test.



4) After the workload has been running about 5 minutes, stop the workload with `rm runload`.

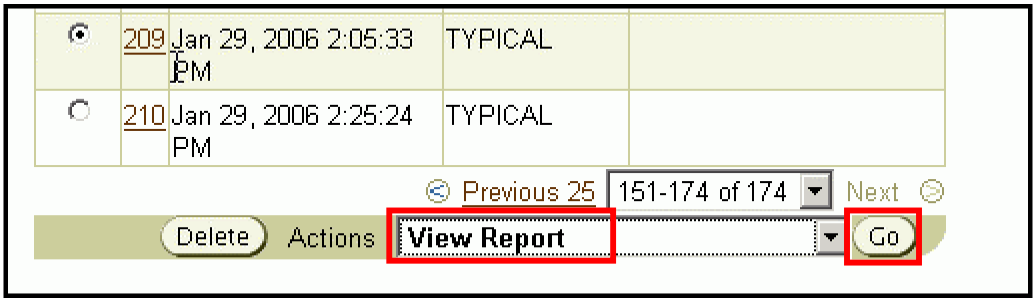
Answer:

```
$ rm runload
rm: remove regular empty file `runload'? y
```

5) Create an AWR snapshot and an AWR report for the last two snapshots.

Answer:

1. Navigate to the AWR Snapshots page and create a new snapshot.
2. When you return to the Snapshots page, select View Report from the Actions drop-down list and click Go.

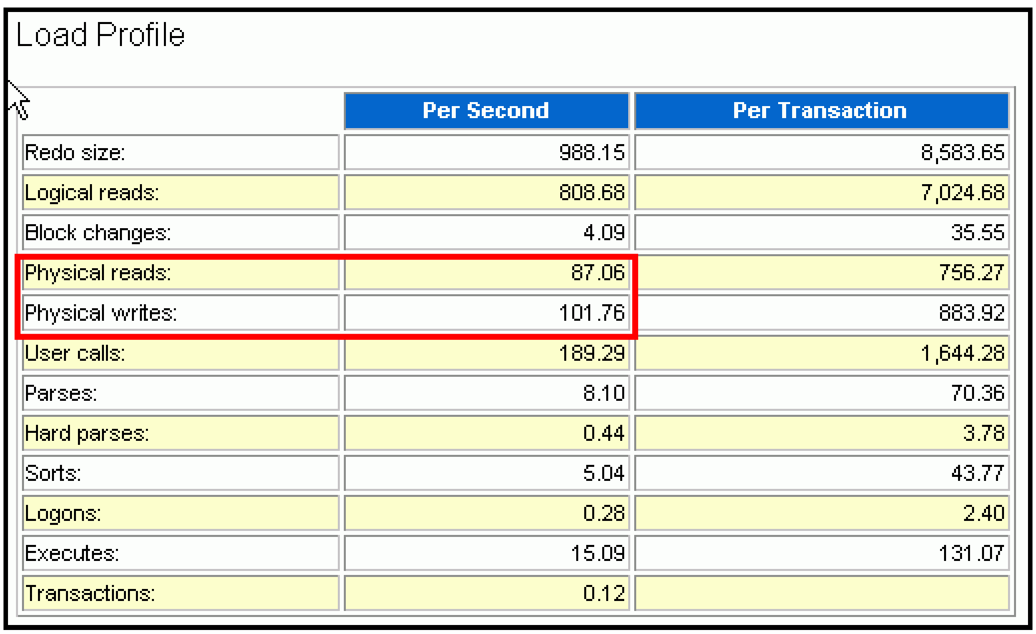


6) Review the AWR report for symptoms and recommendations concerning PGA area sizing. Save the report.

   a) Check Load Profile for physical reads and writes.

   Answer:

   At this point, all you know is that these queries seem to be slow. Make note of the physical reads and physical writes per second.
   Physical reads per second: ____________
   Physical writes per second: ____________



Load Profile

	Per Second	Per Transaction
Redo size:	988.15	8,583.65
Logical reads:	808.68	7,024.68
Block changes:	4.09	35.55
Physical reads:	87.06	756.27
Physical writes:	101.76	883.92
User calls:	189.29	1,644.28
Parses:	8.10	70.36
Hard parses:	0.44	3.78
Sorts:	5.04	43.77
Logons:	0.28	2.40
Executes:	15.09	131.07
Transactions:	0.12	

   b) Determine the SQL with the greatest number of reads. View the SQL statements to determine the cause of the high number of reads.

   Answer:

   1. The “SQL Orderd by Reads” section of the report shows the SQL statements with the greatest reads. Note and record the SQL ID and “Reads per Exec” for the SQL statements that show a significant number of reads per execution.

      SQL ID _________________________ Reads per Exec ___________________

      SQL ID _________________________ Reads per Exec ___________________

SQL ID ________________________ Reads per Exec __________________

SQL ordered by Reads

- Total Disk Reads: 14,215
- Captured SQL account for 136.2% of Total

Physical Reads	Executions	Reads per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
8,934	2	4,467.00	62.85	3.04	435.81	b57rvwy9jbvv6	SQL*Plus	select * from customers order ...
2,618	2	1,309.00	18.42	2.95	338.61	3bq3pfux13pkr	SQL*Plus	select * from (select * from ...
2,393	2	1,196.50	16.83	2.74	336.82	bxvwcp4pjr43n	SQL*Plus	select * from (select * from ...
2,080	2	1,040.00	14.63	2.61	332.21	fkmrxvv6fxqar	SQL*Plus	select * from (select * from ...
1,165	3	388.33	8.20	3.06	264.95	cssuux75dw74c	SQL*Plus	select * from (select * from ...
1,048	4	262.00	7.37	3.45	335.83	54dxhddg0mfdt	SQL*Plus	select * from (select * from ...
758	4	189.50	5.33	0.41	33.89	85h35h24mxy40	SQL*Plus	select distinct country_name ...
232	4	58.00	1.63	0.46	19.72	brjahu0up7grh	SQL*Plus	select distinct country_name, ...
81	0		0.57	5.45	83.29	8u809k64x3nzd	Admin Connection	begin DBMS_WORKLOAD_REPOSITORY...
69	484	0.14	0.49	0.31	18.99	db78fxqxwxt7r		select /*+ rule */ bucket, en...

2. The top SQL statements have an `ORDER BY` clause or an access path that requires a sort. The `ORDER BY` clause forces sorts and additional reads if the PGA is not properly sized. The SQL statements are:

b57rwy9jbw6	select * from customers order by cust_gender, cust_marital_status

3bq3pfux13pkr	select * from (select * from dba_objects order by timestamp) where rownum < 80000

bxwcp4pjr43n	select * from (select * from dba_objects order by timestamp) where rownum < 40000

c) Determine the extra writes that are caused by the current PGA sizing.

Answer:

The PGA Aggr Summary shows the extra work area (in MB) that is read and written. The example indicates that 4,500 MB were written because the work area could not be done in memory. Make note of PGA Cache Hit % for your test: _____________.

PGA Aggr Summary

- PGA cache hit % - percentage of W/A (WorkArea) data processed only in-memory

PGA Cache Hit %	W/A MB Processed	Extra W/A MB Read/Written
24.49	1,464	4,514

d) Determine the recommended PGA sizing from the AWR report.

Answer:

Move the AWR report to the Advisories section and find the PGA Memory Advisory. To find the minimum size with no multipass work areas, choose the smallest value in which Estd PGA Overalloc Count is 0. Using the example, the size is 60 MB. The estimated extra work area read and written to disk does not drop significantly until the PGA is 80 MB. Your results may vary.

PGA Memory Advisory

- When using Auto Memory Mgmt, minimally choose a pga_aggregate_target value where Estd PGA Overalloc Count is 0

PGA Target Est (MB)	Size Factr	W/A MB Processed	Estd Extra W/A MB Read/ Written to Disk	Estd PGA Cache Hit %	Estd PGA Overalloc Count
10	0.50	748.41	1,061.81	41.00	78
15	0.75	748.41	1,061.81	41.00	78
20	1.00	748.41	614.29	55.00	77
24	1.20	748.41	614.29	55.00	77
28	1.40	748.41	614.29	55.00	76
32	1.60	748.41	614.29	55.00	71
36	1.80	748.41	614.29	55.00	66
40	2.00	748.41	614.29	55.00	62
60	3.00	748.41	560.05	57.00	0
80	4.00	748.41	154.99	83.00	0
120	6.00	748.41	110.29	87.00	0
160	8.00	748.41	0.00	100.00	0

e) Observe the number of multipass and one-pass work areas that are being used.

Answer:

Find the PGA Aggr Target Histogram in the Advisories section of the AWR report. Work areas that cannot be contained in memory write to disk (temporary tablespace) and are recorded as one-pass or multipass work areas. In the PGA Aggr Target Histogram, the column on the right shows the minimum requested size for a one-pass execution. The example shows a number of multipass executions and one-pass executions for sorts larger than 2 MB.

PGA Aggr Target Histogram

- Optimal Executions are purely in-memory operations

Low Optimal	High Optimal	Total Execs	Optimal Execs	1-Pass Execs	M-Pass Execs
2K	4K	1,792	1,792	0	0
64K	128K	3	3	0	0
512K	1024K	125	125	0	0
2M	4M	46	0	12	34
4M	8M	32	0	4	28
8M	16M	2	0	2	0

Solutions for Practice 13-1: Tuning PGA_AGGREGATE_TARGET (continued)

7) Make a new setting for PGA_AGGREGATE_TARGET on the basis of the value you determined.

   Answer:

   In a terminal window, connect as sysdba, and change the PGA_AGGREGATE_TARGET to 80 MB.

```
$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Sun Jan 29
14:51:04 2006

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> ALTER SYSTEM SET PGA_AGGREGATE_TARGET = 80M;

System altered.

SQL> EXIT
```

8) Execute the workload again.

   Answer:

```
$ ./workgen 13 1

PL/SQL procedure successfully completed.

$
```

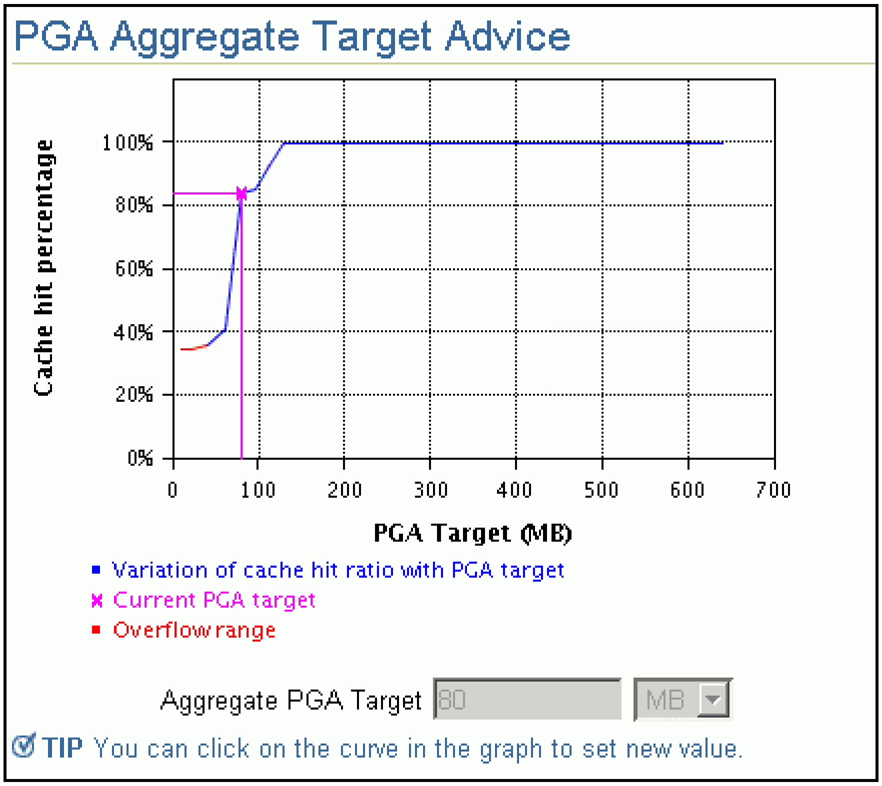
9) After a few minutes, check the PGA Advisor and the PGA Memory Usage Details pages.

   a) View the PGA Advice page; have the recommended values changed?

   Answer:

   Yes, the advice graph has changed, but the first inflection point is still at 80 MB.

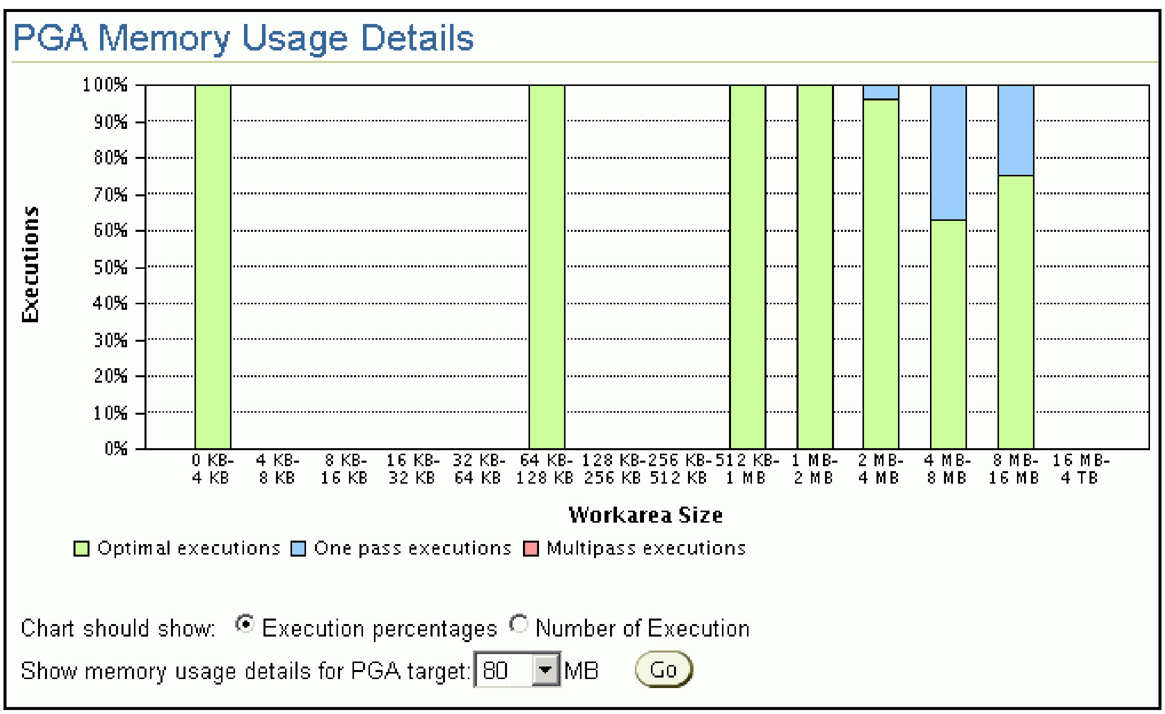
Solutions for Practice 13-1: Tuning PGA_AGGREGATE_TARGET (continued)



b) View the PGA Memory Usage Details page. Are there any multipass work areas?

Answer:

No. The PGA Memory Usage Details histogram does not show any multipass work areas at 80 MB.



Solutions for Practice 13-1: Tuning PGA_AGGREGATE_TARGET (continued)

10) After the workload has been running about 5 minutes, stop the workload with `rm runload`.

Answer:

```
$ rm runload
rm: remove regular empty file `runload'? y
```

11) Create an AWR snapshot and an AWR report for the last two snapshots.

Answer:

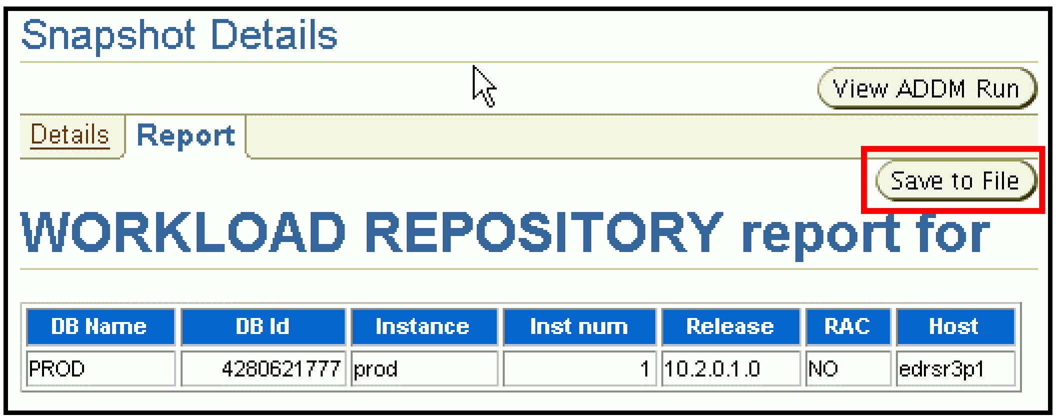
Same procedure as in step 5.

12) Review the AWR report for symptoms, recommendations concerning PGA area sizing, and changes that show improved performance.

a) Save this report to file for later viewing. Name the file `awr13_1_12.html`.

Answer:

Click "Save to File" and provide a file name.



b) Check Load Profile for physical reads and writes per second.

Answer:

Physical reads and writes per second have dropped significantly in the example.

Load Profile

	Per Second	Per Transaction
Redo size:	27,707.73	83,987.33
Logical reads:	1,066.85	3,233.82
Block changes:	182.98	554.65
Physical reads:	35.29	106.98
Physical writes:	50.53	153.18
User calls:	188.18	570.40
Parses:	18.42	55.83
Hard parses:	1.27	3.84
Sorts:	6.75	20.47
Logons:	0.28	0.85
Executes:	31.70	96.10
Transactions:	0.33	

% Blocks changed per Read:	17.15	Recursive Call %:	54.41
Rollback per transaction %:	5.58	Rows per Sort:	813.77

c) Have the SQL statements that perform the most reads changed?

Answer:

1. Find the SQL statements identified in step 6b that have the most reads. The order may have changed. The SQL ID is the hash value of the SQL statement and does not change between runs. Use the SQL ID to find the statements. Notice that the reads per execution have dropped significantly. Users would notice that the statements are responding more quickly based on this example.

SQL ordered by Reads

- Total Disk Reads: 15,184
- Captured SQL account for 91.4% of Total

Physical Reads	Executions	Reads per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Module	SQL Text
4,218	11	383.45	27.78	13.13	203.34	fkmrxvv6fxqar	SQL*Plus	select * from (select * from ...
4,019	7	574.14	26.47	9.85	117.75	3bq3pfux13pkr	SQL*Plus	select * from (select * from ...
3,358	7	479.71	22.12	8.94	126.86	bxvwcp4pjr43n	SQL*Plus	select * from (select * from ...
1,778	2	889.00	11.71	2.19	35.47	b57rvwy9jbvv6	SQL*Plus	select * from customers order ...
302	8	37.75	1.99	2.66	38.32	6gvch1xu9ca3g		DECLARE job BINARY_INTEGER := ...
297	8	37.13	1.96	7.31	259.00	cssuux75dw74c	SQL*Plus	select * from (select * from ...
85	4	21.25	0.56	12.59	733.56	8vjh4nbv1phu3	SQL*Plus	select sum(s.AMOUNT_sold) - su...
40	12	3.33	0.26	9.04	430.77	54dxhddg0mfdt	SQL*Plus	select * from (select * from ...
28	0		0.18	4.18	20.29	8u809k64x3nzd	Admin Connection	begin DBMS_WORKLOAD_REPOSITORY...
12	6	2.00	0.08	2.27	55.59	1rtzcdny0u5wu	SQL*Plus	select SUM(amount_sold) from s...

Solutions for Practice 13-1: Tuning PGA_AGGREGATE_TARGET (continued)

2. The top statement from the previous test has changed in the example, but the second and third statements are the same.

d) Determine the improvement in the PGA cache hit percentage. How could the extra write statistic be decieving?

Answer:

PGA Cache Hit % has increased, showing that more sorting was done in memory. Extra Work Area MB Read/Written could show an increase in I/O from the previous report because the statistic is a total over the report interval (especially if the durations of the reports are different).

PGA Aggr Summary

- PGA cache hit % - percentage of W/A (WorkArea) data processed only in-memory

PGA Cache Hit %	W/A MB Processed	Extra W/A MB Read/Written
63.86	472	267

e) Determine the recommended PGA sizing for the AWR report.

Answer:

Following the process of finding the first value with the Estd PGA Overalloc Amount equal to 0, the recommended value is still 60 MB. But the first value that shows a signficant improvement is 80 MB.

PGA Memory Advisory

- When using Auto Memory Mgmt, minimally choose a pga_aggregate_target value where Estd PGA Overalloc Count is 0

PGA Target Est (MB)	Size Factr	W/A MB Processed	Estd Extra W/A MB Read/ Written to Disk	Estd PGA Cache Hit %	Estd PGA Overalloc Count
10	0.13	479.95	972.75	33.00	52
20	0.25	479.95	972.75	33.00	52
40	0.50	479.95	967.58	33.00	46
60	0.75	479.95	885.35	35.00	0
80	1.00	479.95	133.50	78.00	0
96	1.20	479.95	133.50	78.00	0
112	1.40	479.95	70.52	87.00	0
128	1.60	479.95	0.00	100.00	0
144	1.80	479.95	0.00	100.00	0
160	2.00	479.95	0.00	100.00	0
240	3.00	479.95	0.00	100.00	0
320	4.00	479.95	0.00	100.00	0
480	6.00	479.95	0.00	100.00	0
640	8.00	479.95	0.00	100.00	0

f) Were there any multipass work areas in the reported period?

Answer:

No multipass work areas are reported. The number of one-pass work areas cannot be directly compared due to the differences in time periods of the reports and the multipass sorts that are now one-pass. There is the additional complication that because the system is more efficient, more statements—and thus more sorts—can be performed in the same period.

PGA Aggr Target Histogram

- Optimal Executions are purely in-memory operations

Low Optimal	High Optimal	Total Execs	Optimal Execs	1-Pass Execs	M-Pass Execs
2K	4K	2,854	2,854	0	0
64K	128K	3	3	0	0
512K	1024K	253	253	0	0
1M	2M	86	86	0	0
2M	4M	40	38	2	0
4M	8M	138	36	102	0
8M	16M	2	0	2	0

Solutions for Practice 13-2: *Tuning Temporary Tablespace Performance*

The current temporary tablespace is a locally managed temporary tablespace. To compare the performance with that of a dictionary-managed temporary tablespace, create a dictionary-managed temporary tablespace and make it the database default tablespace. Execute the workload for Practice 13-1 again. Compare the AWR reports.

1) The `setup` script for this practice creates a dictionary-managed temporary tablespace named `TEMP_D` with a 150 MB data file named `temp_D01`. Set the data file properties to autoextend on, maxsize 300 MB, and next 1 MB. Then make it the default temporary tablespace for the database. Run `./setup 13 2`.

   Answer:

```
$ ./setup 13 2

SQL*Plus: Release 10.2.0.1.0 - Production on Wed Jan 3
10:34:54 2007

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

SQL>
SQL> connect / as sysdba
Connected.
SQL>
SQL>
SQL> CREATE TABLESPACE TEMP_D
  2  DATAFILE
'/u01/app/oracle/oradata/prod/disk1/temp_D01.dbf'
  3  SIZE 150M AUTOEXTEND ON NEXT 1M MAXSIZE 300M
  4  EXTENT MANAGEMENT DICTIONARY
  5  DEFAULT STORAGE( INITIAL 64K NEXT 64K PCTINCREASE 0
MAXEXTENTS UNLIMITED)
  6  TEMPORARY;

Tablespace created.

SQL>
SQL> ALTER DATABASE DEFAULT TEMPORARY TABLESPACE "TEMP_D";

Database altered.

SQL>
SQL>
SQL> exit
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
$
```

Solutions for Practice 13-2: Tuning Temporary Tablespace Performance (continued)

2) Generate the Practice 13-1 workload with `./workgen 13 1`.

Answer:

```
$ ./workgen 13 1

PL/SQL procedure successfully completed.

$
```

3) After the workload has been running about 5 minutes, stop the workload with `rm runload`.

Answer:

```
$ rm runload
rm: remove regular empty file `runload'? y
```

4) Create an AWR snapshot and create an AWR report.

Answer:

1. This procedure is the same as Practice 13-1, step 5.

5) Compare the AWR report for this set of snapshots to the previous AWR report in Practice 13-1, step 12. You saved this report to `awr13_1_12.html`.

a) Compare the Load Profile.

Answer:

The physical reads and writes per second may have increased or remained the same.

Load Profile

	Per Second	Per Transaction
Redo size:	5,629.61	39,133.62
Logical reads:	996.28	6,925.52
Block changes:	29.04	201.86
Physical reads:	29.50	205.07
Physical writes:	46.14	320.77
User calls:	201.68	1,401.93
Parses:	28.69	199.45
Hard parses:	0.79	5.51
Sorts:	5.52	38.41
Logons:	0.30	2.10
Executes:	37.42	260.12
Transactions:	0.14	

% Blocks changed per Read:	2.91	Recursive Call %:	62.70
Rollback per transaction %:	0.00	Rows per Sort:	1142.69

Solutions for Practice 13-2: Tuning Temporary Tablespace Performance (continued)

b) Compare the Top 5 Timed Events.

Answer:

A wait event, “enq ST contention,” may appear. An enqueue is a lock on a resource. The ST enqueue is a space management transaction. For most machine types there is nothing that indicates a problem in the Top 5 Timed Events.

Top 5 Timed Events

Event	Waits	Time(s)	Avg Wait(ms)	% Total Call Time	Wait Class
CPU time		229		4.0	
latch: library cache	123	143	1,163	2.5	Concurrency
latch: cache buffers chains	25	76	3,057	1.3	Concurrency
latch free	43	46	1,059	.8	Other
control file parallel write	240	35	146	.6	System I/O

c) Has there been a change in the PGA work area statistics?

Answer:

1. PGA Cache Hit % is virtually unchanged or may show a slight reduction.

PGA Aggr Summary

- PGA cache hit % - percentage of W/A (WorkArea) data processed only in-memory

PGA Cache Hit %	W/A MB Processed	Extra W/A MB Read/Written
60.74	1,765	1,141

2. No multipass work areas are reported.

PGA Aggr Target Histogram

- Optimal Executions are purely in-memory operations

Low Optimal	High Optimal	Total Execs	Optimal Execs	1-Pass Execs	M-Pass Execs
2K	4K	1,915	1,915	0	0
64K	128K	3	3	0	0
256K	512K	2	2	0	0
512K	1024K	138	138	0	0
1M	2M	36	36	0	0
2M	4M	16	10	6	0
4M	8M	62	6	56	0

Solutions for Practice 13-2: Tuning Temporary Tablespace Performance (continued)

d) Is there a change in the elaped time per execution of the longest-running SQL statements?

Answer:

1. The "SQL Ordered by Elapsed Time" report from example `awr13_1_12.html` shows the elapsed time per execution of the longest-running SQL.

SQL ordered by Elapsed Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied by 100

Elapsed Time (s)	CPU Time (s)	Executions	Elap per Exec (s)	% Total DB Time	SQL Id	SQL Module	SQL Text
987	49	2	493.61	19.93	cj4bx12rruv4p	SQL*Plus	select s.prod_id, SUM(s.amoun...
734	13	4	183.39	14.81	8vjh4nbv1phu3	SQL*Plus	select sum(s.AMOUNT_sold) - su...
569	15	5	113.80	11.49	6rz119f0r1wqy	SQL*Plus	select s.prod_id, sum(s.amoun...
431	9	12	35.90	8.70	54dxhddg0mfdt	SQL*Plus	select * from (select * from ...
259	7	8	32.38	5.23	cssuux75dw74c	SQL*Plus	select * from (select * from ...
226	5	4	56.55	4.57	3g36d1uum3662	SQL*Plus	select PROD_ID, SUM(amount_so...
203	13	11	18.49	4.11	fkmrxvv6fxqar	SQL*Plus	select * from (select * from ...
127	9	7	18.12	2.56	bxvwcp4pjr43n	SQL*Plus	select * from (select * from ...
118	10	7	16.82	2.38	3bg3pfux13pkr	SQL*Plus	select * from (select * from ...
57	2	4	14.31	1.16	8rtsdaxm7apg7	SQL*Plus	select SUM(amount_sold), PROD...
57	2	2	28.36	1.15	8fx6pqqbpra0s	OEM.DefaultPool	begin emd_database.getDBSiteMa...
56	2	6	9.26	1.12	1rtzcdny0u5wu	SQL*Plus	select SUM(amount_sold) from s...
52	1	2	26.13	1.06	b07usvw8cn92f	OEM.DefaultPool	SELECT DECODE(LOG_MODE, 'NOAR...

2. When you compare the two reports, the elapsed time per execution is less in almost every SQL statement that is reported.

SQL ordered by Elapsed Time

- Resources reported for PL/SQL code includes the resources used by all SQL statements called by the code.
- % Total DB Time is the Elapsed Time of the SQL statement divided into the Total Database Time multiplied by 100

Elapsed Time (s)	CPU Time (s)	Executions	Elap per Exec (s)	% Total DB Time	SQL Id	SQL Module	SQL Text
3,406	80	6	567.73	17.79	cj4bx12rruv4p	SQL*Plus	select s.prod_id, SUM(s.amoun...
3,002	72	41	73.22	15.67	8vjh4nbv1phu3	SQL*Plus	select sum(s.AMOUNT_sold) - su...
2,716	65	56	48.49	14.18	2rmw64nrra33j	SQL*Plus	select s.prod_id, sum(s.amoun...
2,202	53	33	66.73	11.50	6rz119f0r1wqy	SQL*Plus	select s.prod_id, sum(s.amoun...
1,710	58	155	11.03	8.93	54dxhddg0mfdt	SQL*Plus	select * from (select * from ...
1,272	37	3	423.92	6.64	589mxw1dz9mhn	SQL*Plus	select s.prod_id, sum(s.amoun...
952	38	90	10.58	4.97	cssuux75dw74c	SQL*Plus	select * from (select * from ...
855	21	31	27.59	4.47	3g36d1uum3662	SQL*Plus	select PROD_ID, SUM(amount_so...
835	52	83	10.07	4.36	3bq3pfux13pkr	SQL*Plus	select * from (select * from ...
718	37	75	9.57	3.75	fkmrxvv6fxqar	SQL*Plus	select * from (select * from ...
703	42	72	9.76	3.67	bxvwcp4pjr43n	SQL*Plus	select * from (select * from ...
232	11	56	4.14	1.21	1rtzcdny0u5wu	SQL*Plus	select SUM(amount_sold) from s...

6) Save this AWR report as `awr13_2_5.html`.

Answer:

Use the same procedure as in Practice 13-1, step 12.

7) The primary difference between the locally managed tablespace and the dictionary-managed temporary tablespace is in space allocation. The dictionary-managed tablespace uses locks and enqueues to manage space allocation in extent tables in the system tablespace, while the locally managed temporary tablespace uses bitmaps for extent management. Check the difference in Enqueue Activity in the Wait Statistics section of each report.

Answer:

1. The Enqueue Activity from the `awr13_1_12.html` report shows few or no Enqueue waits.

Enqueue Activity

- only enqueues with waits are shown
- Enqueue stats gathered prior to 10g should not be compared with 10g data
- ordered by Wait Time desc, Waits desc

Enqueue Type (Request Reason)	Requests	Succ Gets	Failed Gets	Waits	Wt Time (s)	Av Wt Time(ms)
SQ-Sequence Cache	14	14	0	2	0	90.00

2. The Enqueue Activity from `awr13_2_5.html` shows increased enqueue activity and waits, thus explaining the increase in elapsed time of the SQL statements.

Enqueue Activity

- only enqueues with waits are shown
- Enqueue stats gathered prior to 10g should not be compared with 10g data
- ordered by Wait Time desc, Waits desc

Enqueue Type (Request Reason)	Requests	Succ Gets	Failed Gets	Waits	Wt Time (s)	Av Wt Time(ms)
SS-Sort Segment	2	2	0	1	2	1,670.00
ST-Space Transaction	708	708	0	9	0	31.11
SQ-Sequence Cache	106	106	0	3	0	33.33

8) Run the cleanup script for 13-2: `./cleanup 13 2`

Answer:

Execute the cleanup script.

```
$ ./cleanup 13 2

Database altered.


Tablespace dropped.

$
```

Practice Solutions for Lesson 14

In this practice, you change tablespace properties that are impacting performance. These changes do not apply to every database; they are specific to certain types of applications.

Solutions for Practice 14-1: *Tuning Database Space Usage*

In this practice, you convert a dictionary-managed tablespace to a locally managed tablespace. You will also observe performance differences between dictionary-managed and locally managed tablespaces.

The workload for this scenario is a high-volume insert with multiple processes. This load requires many new extents. This scenario uses the `prod` database; therefore, set the environment variables appropriately.

1) Set the environment variables. Run the setup script for Practice 14-1 found in the `/home/oracle/workshops` directory.

```
$ cd /home/oracle/workshops
$ ./setup 14 1
```

The setup script moves the OE schema to the `USERS` tablespace. The `USERS` tablespace is dictionary managed.

Answer:

Execute the setup script for Practice 14-1.

```
$ . oraenv
ORACLE_SID = [orcl] ? prod
$ cd /home/oracle workshops
$ ./setup 14 1
TZ set to US/Pacific
Oracle Enterprise Manager 10g Database Control Release
10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation.  All rights
reserved.
http://edrsr3p1.us.oracle.com:1158/em/console/aboutApplica
tion
Stopping Oracle Enterprise Manager 10g Database Control
...
 ...  Stopped.

SQL*Plus: Release 10.2.0.1.0 - Production on Fri Feb 3
06:33:18 2006

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> Database closed.
Database dismounted.
ORACLE instance shut down.
```

```
SQL> Disconnected from Oracle Database 10g Enterprise
Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
Database closed.
Database dismounted.
ORACLE instance shut down.

ORACLE instance started.

Total System Global Area  629145600 bytes
Fixed Size                  1220988 bytes
Variable Size             171970180 bytes
Database Buffers          452984832 bytes
Redo Buffers                2969600 bytes
Database mounted.
Database opened.

Database altered.


Database altered.

Setup 14-1 finished

exit
$
```

2) Run the workload. Use the `workgen` script:

   `$ ./workgen 14 1`

   Run the ADDM report as soon as the `take ADDM snapshot now` prompt appears. Do *not* convert the tablespace to a locally managed tablespace until the script is finished.

   Answer:

   Run the workload generator. This workload will run about 3 minutes and stop. While it is running, go to the Performance page and observe the performance graphs.

```
$ ./workgen 14 1

Table dropped.


Table created.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

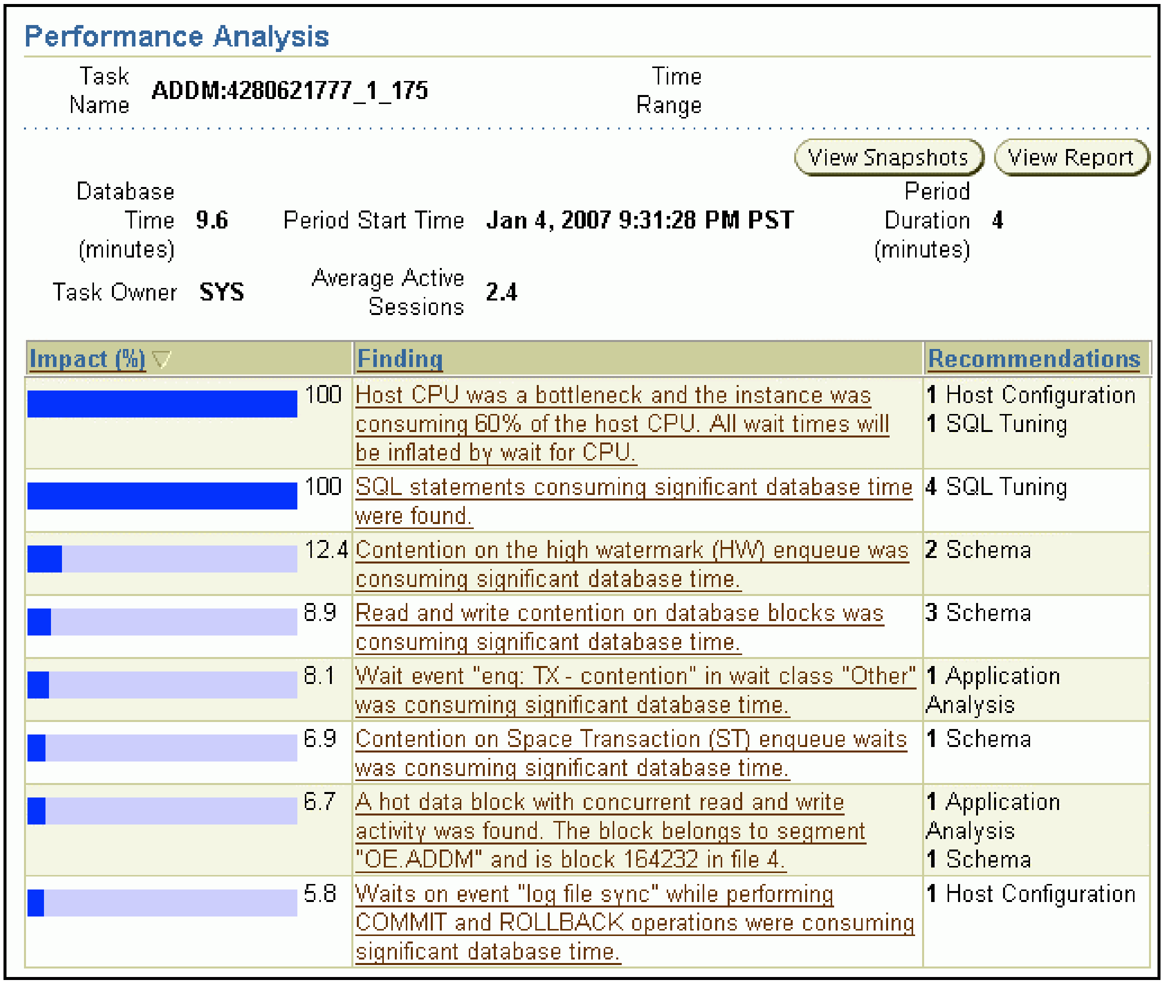
```
PL/SQL procedure successfully completed.

 Workload is finishing, take ADDM snapshot now.

Table dropped.


Table created.


PL/SQL procedure successfully completed.

 Workload is finished
$
```

3) The workload will end after 3 minutes. Then run an ADDM report and view the findings. Drill down on findings related to database blocks.

   Answer:

   1. From the Performance page, run an ADDM report.
   2. View the Performance findings. There may be several different findings. The ones that you should explore are “Contention on highwater mark HW enqueue…,” “Read and write contention on database blocks…,” and “Contention on Space Transaction (ST) enqueue waits….” Not all of these findings will appear on all machine types.

Solutions for Practice 14-1: Tuning Database Space Usage (continued)



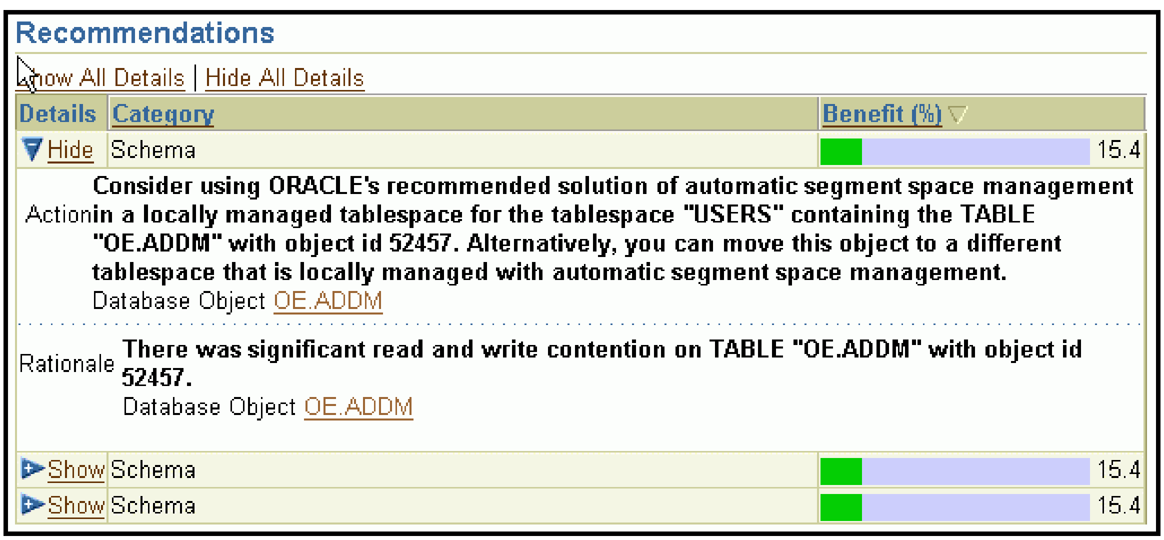
Performance Analysis

Task Name	ADDM:4280621777_1_175	Time Range	
Database Time (minutes)	9.6	Period Start Time	Jan 4, 2007 9:31:28 PM PST
Period Duration (minutes)	4		
Task Owner	SYS	Average Active Sessions	2.4

View Snapshots | View Report

Impact (%)	Finding	Recommendations
100	Host CPU was a bottleneck and the instance was consuming 60% of the host CPU. All wait times will be inflated by wait for CPU.	1 Host Configuration 1 SQL Tuning
100	SQL statements consuming significant database time were found.	4 SQL Tuning
12.4	Contention on the high watermark (HW) enqueue was consuming significant database time.	2 Schema
8.9	Read and write contention on database blocks was consuming significant database time.	3 Schema
8.1	Wait event "enq: TX - contention" in wait class "Other" was consuming significant database time.	1 Application Analysis
6.9	Contention on Space Transaction (ST) enqueue waits was consuming significant database time.	1 Schema
6.7	A hot data block with concurrent read and write activity was found. The block belongs to segment "OE.ADDM" and is block 164232 in file 4.	1 Application Analysis 1 Schema
5.8	Waits on event "log file sync" while performing COMMIT and ROLLBACK operations were consuming significant database time.	1 Host Configuration

3. Drill down on the one of the findings. In the example “Read and write contention on database blocks was consuming significant database time.” is shown. Your results may not include this recommendation.



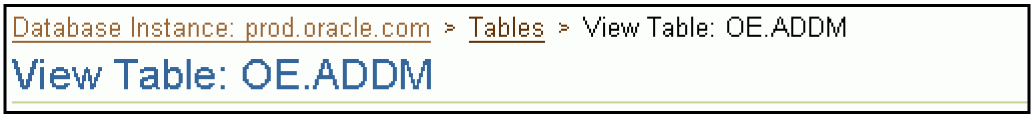
Recommendations

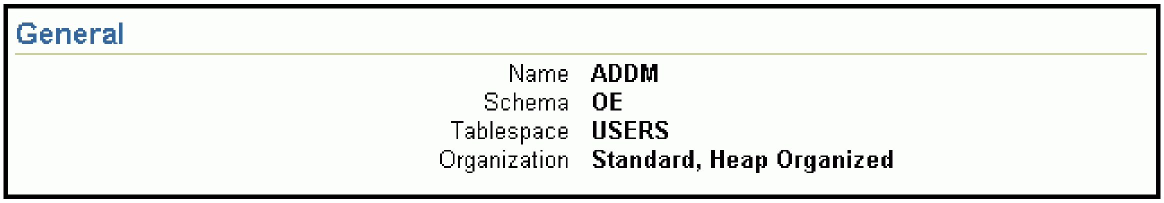
Show All Details | Hide All Details

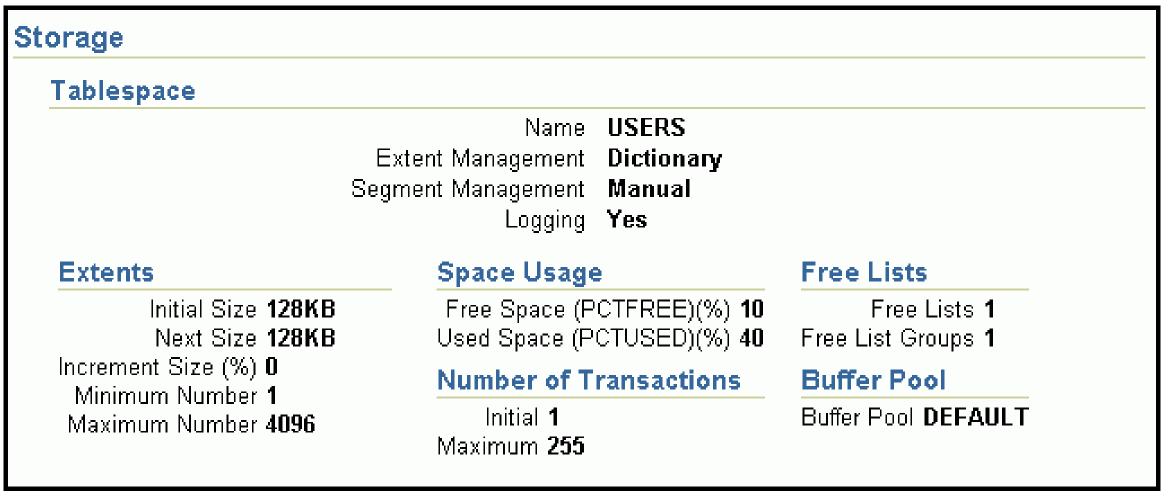
Details	Category	Benefit (%)
Hide	Schema	15.4
Action: **Consider using ORACLE's recommended solution of automatic segment space management in a locally managed tablespace for the tablespace "USERS" containing the TABLE "OE.ADDM" with object id 52457. Alternatively, you can move this object to a different tablespace that is locally managed with automatic segment space management.** Database Object OE.ADDM **Rationale:** **There was significant read and write contention on TABLE "OE.ADDM" with object id 52457.** Database Object OE.ADDM		
Show	Schema	15.4
Show	Schema	15.4

4. The recommendations for these findings say that the `OE.ADDM` should be placed in a locally managed tablespace. You know that the `USERS` tablespace is

dictionary managed. Portions of the View Table: OE.ADDM page are shown here.

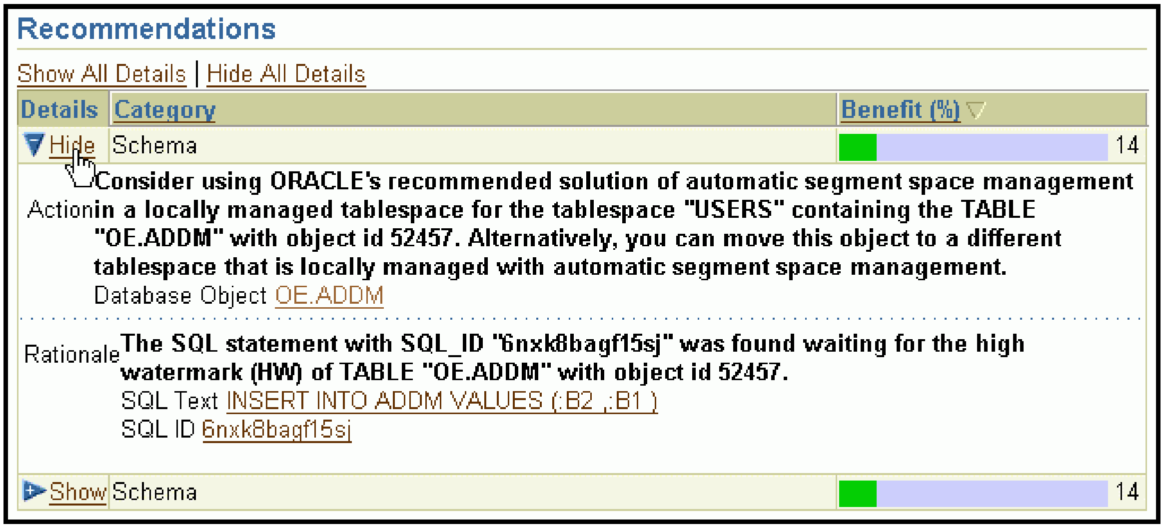




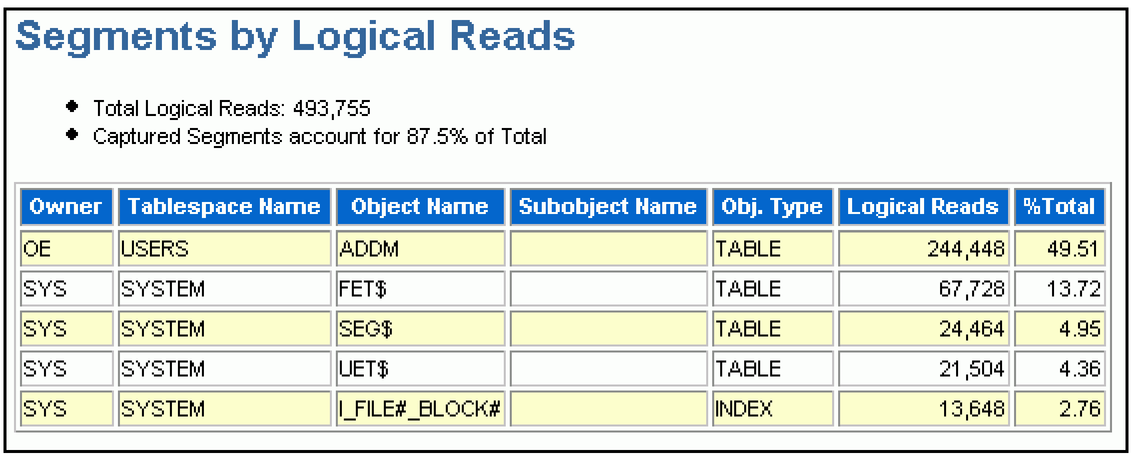


5. Consider the findings “Contention on the highwater mark (HW) enqueue…” and Contention on Space Transaction (ST) enqueue waits….” These finding indicate, respectively, very rapid growth of segments in the database and many operations to allocated and deallocate extents. If the segment or segments responsible can be isolated, you can make the tablespace locally managed to reduce the ST and possibly HW contention. Drill down on the finding. ADDM may recommend the use of locally managed tablespaces. This recommendation may not always appear. ADDM considers more factors than just a single finding.

Solutions for Practice 14-1: Tuning Database Space Usage (continued)



6. Generate an AWR report. You find that the UET$ and FET$ tables are listed in the segments with the highest logical reads. These tables are the ones accessed for every extent allocation when the tablespace is dictionary managed.



Segments by Logical Reads

- Total Logical Reads: 493,755
- Captured Segments account for 87.5% of Total

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Logical Reads	%Total
OE	USERS	ADDM		TABLE	244,448	49.51
SYS	SYSTEM	FET$		TABLE	67,728	13.72
SYS	SYSTEM	SEG$		TABLE	24,464	4.95
SYS	SYSTEM	UET$		TABLE	21,504	4.36
SYS	SYSTEM	I_FILE#_BLOCK#		INDEX	13,648	2.76

4) Convert the `USERS` tablespace to locally managed. Use the Enterprise Manager Database Control to convert the tablespace to locally managed.

Answer:

1. There are multiple ways to convert a tablespace to a locally managed one. The three main methods are:

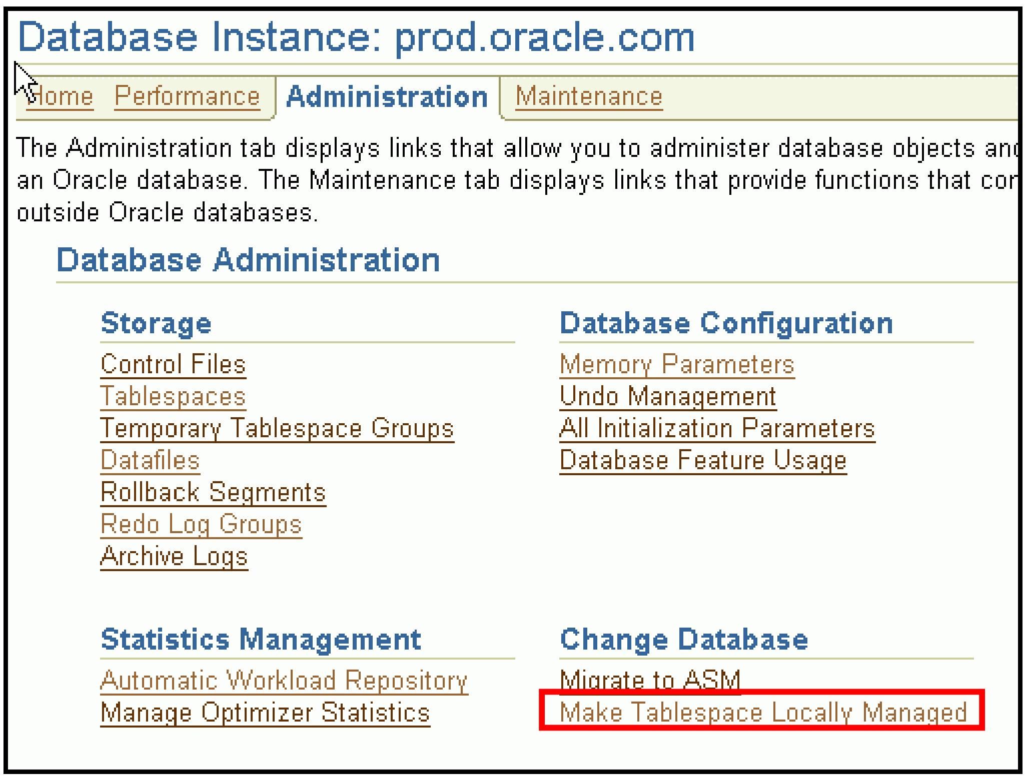
   a. Create a locally managed tablespace, move the data objects into the tablespace, drop the original tablespace, and rename the new tablespace to the original name. This method reorganizes the data objects and ensures good space usage. This method can be performed with the data available (on-line reorganization).

   b. Export the data objects, drop the dictionary-managed tablespace, create a locally managed tablespace, and import into the new tablespace. This method reorganizes the data objects and ensures good space usage. The data is not available during the operation.

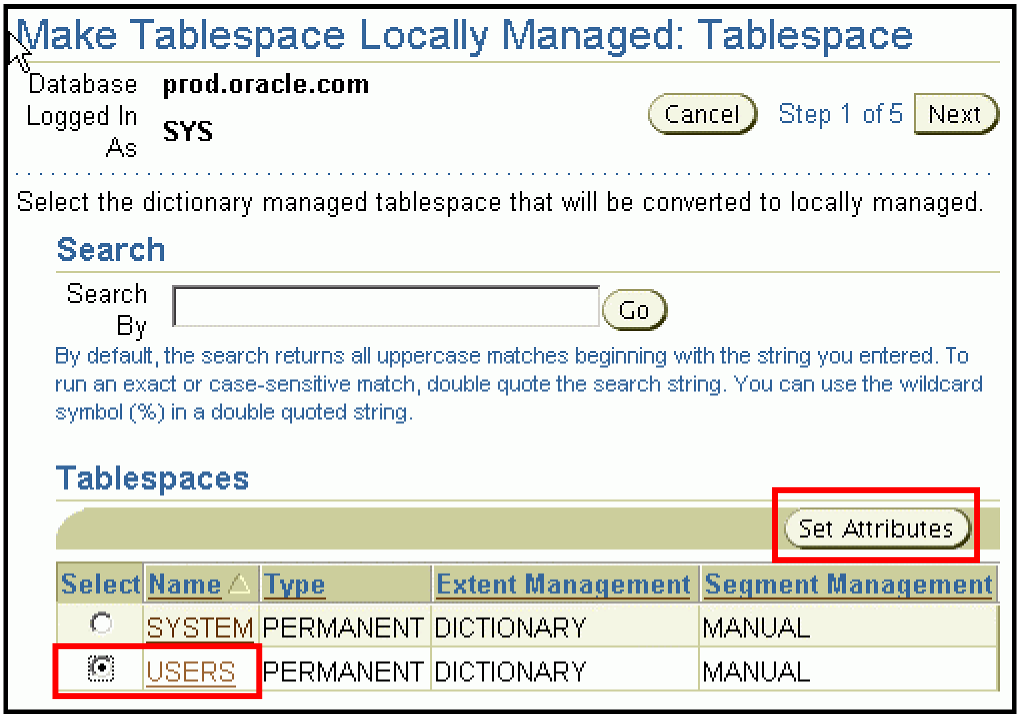
c. Convert the tablespace in-place using the `DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL` procedure. This procedure does not reorganize the objects, and requires space in the existing tablespace to create the bitmaps used for local management. The existing extents are not changed, so the full benefits of a locally managed tablespace are not seen.

2. This solution uses EM Database Control and method a.

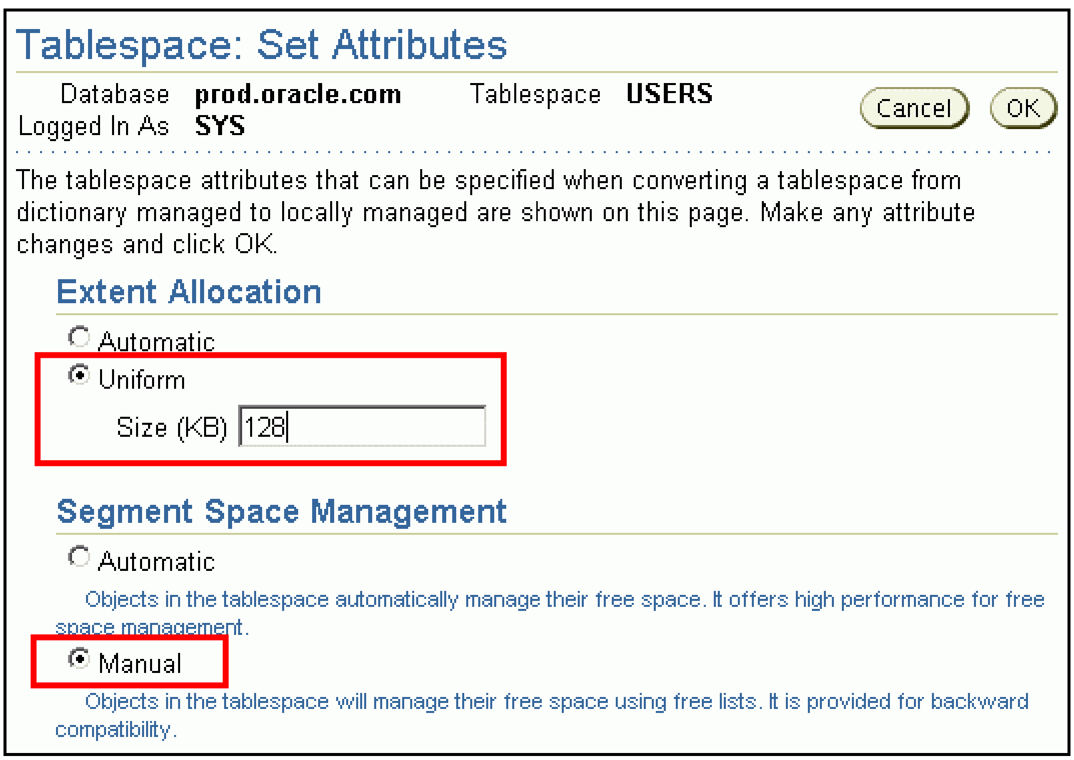
a. Click Make Tablespace Locally Managed in the Change Database section of the Administration page.



b. Select the `USERS` tablespace. Click Set Attributes.



c. On the Tablespace: Set Attributes page, verify that Segment Space Management is set to Manual. Set Extent Allocation to Uniform with 128 KB extent size. Click OK.



d. Return to the Make Tablespace Locally Managed: Tablespace page, and then click Next.

e. On the Make Tablespace Locally Managed: Options page, click Next.

f. When the Make Tablespace Locally Managed: Impact Report page appears, check for messages that would stop the process, and then click Next.

g. On the Make Tablespace Locally Managed: Schedule page, check that the job is scheduled to start immediately, verify that the host credentials are correct, and then click Next.

h. On the Make Tablespace Locally Managed: Review page, you can check or save the script. Click Submit Job.

i. On the Job Activity page, you can monitor the progress of the job. Click the job name link in the Confirmation section, and then refresh the page until the job shows "Status: Succeeded." The job takes about 2 minutes to complete.

5) Run the workload again.

Answer:

Run the workload generator. The workload stops after about 3 minutes.

```
$ ./workgen 14 1

Table dropped.


Table created.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

 Workload is finishing, take ADDM snapshot now.

Table dropped.


Table created.


PL/SQL procedure successfully completed.

 Workload is finished
$
```

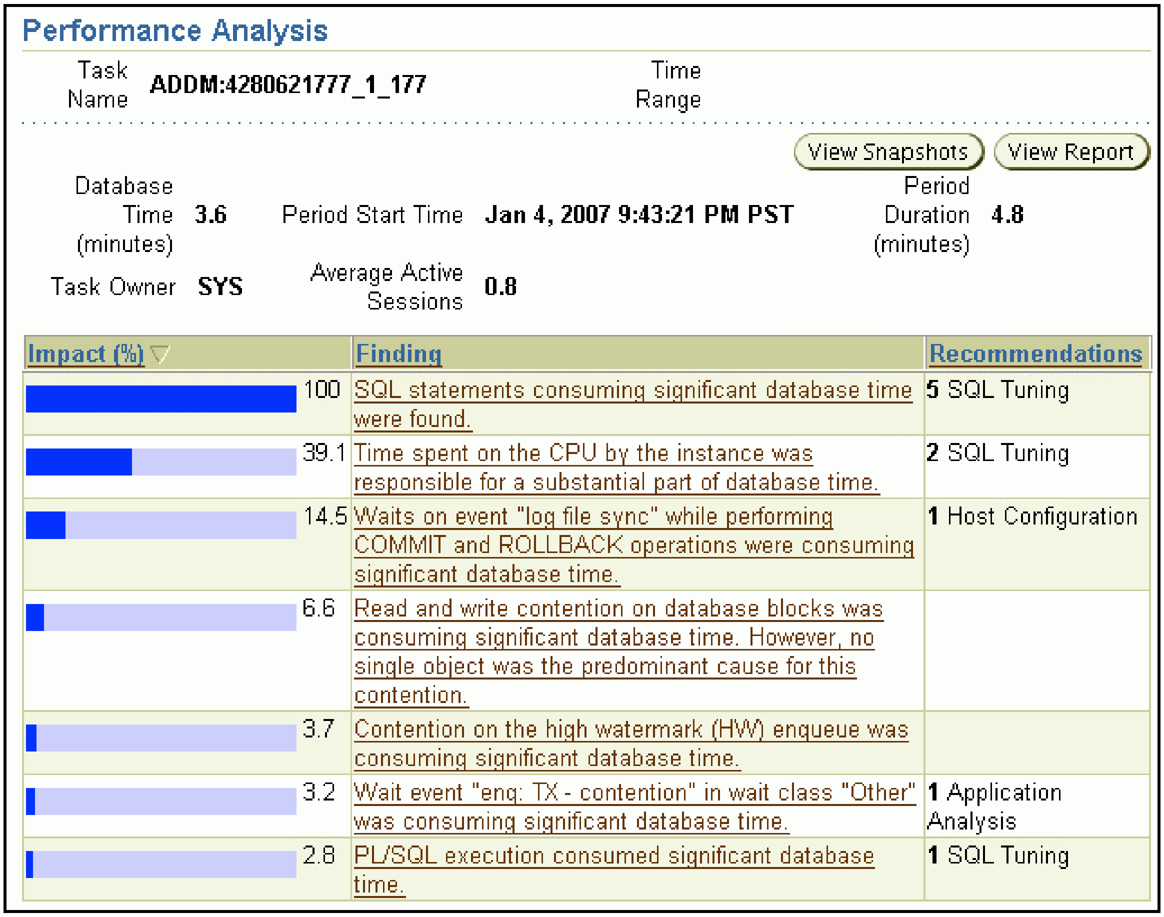
6) Navigate to the Performance page, run an ADDM report, and view the findings.

Answer:

1. The ADDM Performance findings are different after the reorganization of the OE.ADDM table. Space management waits should be reduced or eliminated.

Solutions for Practice 14-1: Tuning Database Space Usage (continued)

Locally managed tablespaces should have a much lower wait time for ST waits than a dictionary-managed tablespace with the same workload. Waits on ST enqueue will be reduced. The impact of the finding “Contention on the High Water Mark (HW) enqueue…” may be reduced, but if the change reduces ST waits enough, the HW waits may increase. The finding “Read and write contention on database blocks was consuming significant database time” is not directly affected by the change to a locally managed tablespace. But if there are fewer waits for space management, more time will be spent on other operations. If the CPU has an Impact % of 100, the other impact statements may be skewed because of wait inflation due to the CPU bottleneck. Improvements in any other area may not be shown because the CPU is the bottleneck. Further analysis may be necessary.



Performance Analysis

Task Name **ADDM:4280621777_1_177** Time Range

View Snapshots | View Report

Database Time (minutes) **3.6** Period Start Time **Jan 4, 2007 9:43:21 PM PST** Period Duration (minutes) **4.8**

Task Owner **SYS** Average Active Sessions **0.8**

Impact (%)	Finding	Recommendations
100	SQL statements consuming significant database time were found.	**5** SQL Tuning
39.1	Time spent on the CPU by the instance was responsible for a substantial part of database time.	**2** SQL Tuning
14.5	Waits on event "log file sync" while performing COMMIT and ROLLBACK operations were consuming significant database time.	**1** Host Configuration
6.6	Read and write contention on database blocks was consuming significant database time. However, no single object was the predominant cause for this contention.	
3.7	Contention on the high watermark (HW) enqueue was consuming significant database time.	
3.2	Wait event "enq: TX - contention" in wait class "Other" was consuming significant database time.	**1** Application Analysis
2.8	PL/SQL execution consumed significant database time.	**1** SQL Tuning

2. Run an AWR report and check the segments with the top logical reads.

Segments by Logical Reads

- Total Logical Reads: 2,152,869
- Captured Segments account for 91.3% of Total

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Logical Reads	%Total
OE	USERS	ADDM		TABLE	1,724,896	80.12
SYS	SYSTEM	FILE$		TABLE	47,328	2.20
SYS	SYSTEM	TSQ$		TABLE	47,200	2.19
SYS	SYSTEM	SEG$		TABLE	46,896	2.18
SYS	SYSTEM	I_FILE#_BLOCK#		INDEX	31,696	1.47

7) What conclusions can you draw from this?

Answer:

There is a small but measurable improvement in performance with the simulated workload. More improvements are expected for a high-volume insert load using locally managed tablespaces. The performance difference will be magnified with workloads that perform more extent allocations.

8) Reset your environment; run the `cleanup 14 1` script.

Answer:

```
$ ./cleanup 14 1

Table dropped.


Tablespace dropped.


Tablespace created.

Start the ORCL database and EM

SQL*Plus: Release 10.2.0.1.0 - Production on Thu Jan 25
10:55:34 2007

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> ORACLE instance started.

Total System Global Area  629145600 bytes
Fixed Size                  1220988 bytes
Variable Size             318770820 bytes
Database Buffers          306184192 bytes
```

```
Redo Buffers                2969600 bytes
Database mounted.
Database opened.
SQL> Disconnected from Oracle Database 10g Enterprise
Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
TZ set to US/Pacific
Oracle Enterprise Manager 10g Database Control Release
10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation.  All rights
reserved.
http://edrsr1p1.us.oracle.com:1158/em/console/aboutApplica
tion
Starting Oracle Enterprise Manager 10g Database Control
...................... started.
--------------------------------------------------------------
--------
Logs are generated in directory
/u01/app/oracle/product/10.2.0/db_1/edrsr1p1.us.oracle.com
_orcl/sysman/log
$
```

Practice Solutions for Lesson 15

No practices for this lesson.

Practice Solutions for Lesson 16

Practice 16 is the optional practice for Appendix C.

Solutions for Practice 16-1: *Using Services with a Single-Instance Oracle Database (Optional)*

The first step in creating a service configuration is to plan services for each application or application function that uses your database.

In this practice, you create the following configuration in the `orcl` database:

Service Name	Usage	Response Time (sec)– Warning/Critical
SERV1	Client service	0.4, 1.0

1) Use the `DBMS_SERVICE` package to create a service called `SERV1`. Then, make sure that you add your service name to your `tnsnames.ora` file.

   Answer:

   1. The recommended method of adding a service name to the `tnsnames.ora` file is to use Net Manager. For this exercise, execute the `lab_c_01_01.sh` script. Review the `tnsnames.ora` file at `$ORACLE_HOME/network/admin` to confirm that the following lines are included. Substitute the output of the `hostname` command for <hostname> below.

```
SERV1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)
      (HOST = <hostname>.us.oracle.com)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = SERV1.us.oracle.com)
    )
  )
```

```
$ . oraenv
ORACLE_SID = [prod] ? orcl
$ cd /home/oracle/labs
$ ./lab_c_01_01.sh
edrsr3p1
$
```

   2. Use the `DBMS_SERVICE.CREATE_SERVICE` procedure to create a service.

```
$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Thu Dec 8
00:48:43 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.
```

```
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> EXEC -
DBMS_SERVICE.CREATE_SERVICE('SERV1','SERV1.us.oracle.com')

PL/SQL procedure successfully completed.

SQL> exit;
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
```

2) After you have created your services, try connecting to your database by using your service name. What happens and why?

Answer:

You cannot connect using your service because although it is defined, it is not started on your instance. You can verify this by looking at the `SERVICE_NAME` initialization parameter, and by looking at the services known to the listener.

```
[oracle@edrsr3p1 labss]$ lsnrctl services

LSNRCTL for Linux: Version 10.2.0.1.0 - Production on 08-
DEC-2005 00:54:16

Copyright (c) 1991, 2005, Oracle.  All rights reserved.

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1)))
Services Summary...
Service "PLSExtProc" has 1 instance(s).
  Instance "PLSExtProc", status UNKNOWN, has 1 handler(s)
for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0
         LOCAL SERVER
Service "orcl.oracle.com" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "DEDICATED" established:2 refused:0 state:ready
         LOCAL SERVER
Service "orclXDB.oracle.com" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "D000" established:0 refused:0 current:0 max:1022
state:ready
```

```
         DISPATCHER <machine: edrsr3p1, pid: 8787>

(ADDRESS=(PROTOCOL=tcp)(HOST=edrsr3p1.us.oracle.com)(PORT=
27969))
Service "orcl_XPT.oracle.com" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "DEDICATED" established:2 refused:0 state:ready
         LOCAL SERVER
Service "prod.oracle.com" has 1 instance(s).
  Instance "prod", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "DEDICATED" established:21 refused:0 state:ready
         LOCAL SERVER
Service "prodXDB.oracle.com" has 1 instance(s).
  Instance "prod", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "D000" established:0 refused:0 current:0 max:1022
state:ready
         DISPATCHER <machine: edrsr3p1, pid: 5119>

(ADDRESS=(PROTOCOL=tcp)(HOST=edrsr3p1.us.oracle.com)(PORT=
59406))
Service "prod_XPT.oracle.com" has 1 instance(s).
  Instance "prod", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "DEDICATED" established:21 refused:0 state:ready
         LOCAL SERVER
The command completed successfully

[oracle@edrsr3p1 solutions]$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Thu Dec 8
00:56:30 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> show parameter service

NAME                              TYPE        VALUE
------------------------------------ ----------- ----------------
```

```
service_names                 string      orcl.oracle.com
SQL> connect system/oracle@SERV1
ERROR:
ORA-12514: TNS:listener does not currently know of service
requested in connect
descriptor


Warning: You are no longer connected to ORACLE.
SQL> exit
 $
```

3) How would you make sure that you could connect using your service? Do it, and connect to your instance using your service.

   Answer:

   You must start your service on your instance.

```
 $ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Thu Dec 8
01:01:13 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> show parameter service

NAME                          TYPE        VALUE
----------------------------- ----------- ----------------
service_names                 string

SQL> EXEC DBMS_SERVICE.START_SERVICE('SERV1');

PL/SQL procedure successfully completed.

SQL> show parameter service

NAME                      TYPE        VALUE
------------------------- ----------- --------------------
service_names             string      SERV1.us.oracle.com

SQL> host lsnrctl services

LSNRCTL for Linux: Version 10.2.0.1.0 - Production on 08-
DEC-2005 01:01:51
```

```
Copyright (c) 1991, 2005, Oracle.  All rights reserved.

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1)))
Services Summary...
Service "PLSExtProc" has 1 instance(s).
  Instance "PLSExtProc", status UNKNOWN, has 1 handler(s)
for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0
         LOCAL SERVER
Service "SERV1.us.oracle.com" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "orcl.oracle.com" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "orclXDB.oracle.com" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "D000" established:0 refused:0 current:0 max:1022
state:ready
         DISPATCHER <machine: edrsr3p1, pid: 8787>

(ADDRESS=(PROTOCOL=tcp)(HOST=edrsr3p1.us.oracle.com)(PORT=
27969))
Service "orcl_XPT.oracle.com" has 1 instance(s).
  Instance "orcl", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "prod.oracle.com" has 1 instance(s).
  Instance "prod", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "DEDICATED" established:22 refused:0 state:ready
         LOCAL SERVER
Service "prodXDB.oracle.com" has 1 instance(s).
  Instance "prod", status READY, has 1 handler(s) for this
service...
    Handler(s):
```

```
      "D000" established:0 refused:0 current:0 max:1022
state:ready
         DISPATCHER <machine: edrsr3p1, pid: 5119>

(ADDRESS=(PROTOCOL=tcp)(HOST=edrsr3p1.us.oracle.com)(PORT=
59406))
Service "prod_XPT.oracle.com" has 1 instance(s).
  Instance "prod", status READY, has 1 handler(s) for this
service...
    Handler(s):
      "DEDICATED" established:22 refused:0 state:ready
         LOCAL SERVER
The command completed successfully

SQL> connect system/oracle@SERV1
Connected.
SQL> exit
Disconnected from Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options
```

4) Execute the `lab_c_01_04.sql` script as `sysdba`. This script creates a new user; connect to your instance using this user and the `SERV1` service. The script then executes the following query:

    `SELECT COUNT(*) FROM DBA_OBJECTS,DBA_OBJECTS,DBA_OBJECTS`

    Do not wait for the script to complete continue to next step.

    Answer:

    Execute the `lab_c_01_04.sql` script in the `/home/oracle/labs` directory.

```
$ cd /home/oracle/labs
$ sqlplus / as sysdba

SQL*Plus: Release 10.2.0.1.0 - Production on Thu Dec 8
01:15:02 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0
- Production
With the Partitioning, OLAP and Data Mining options

SQL> @lab_c_01_04.sql
SQL>
SQL> drop user jfv cascade;
```

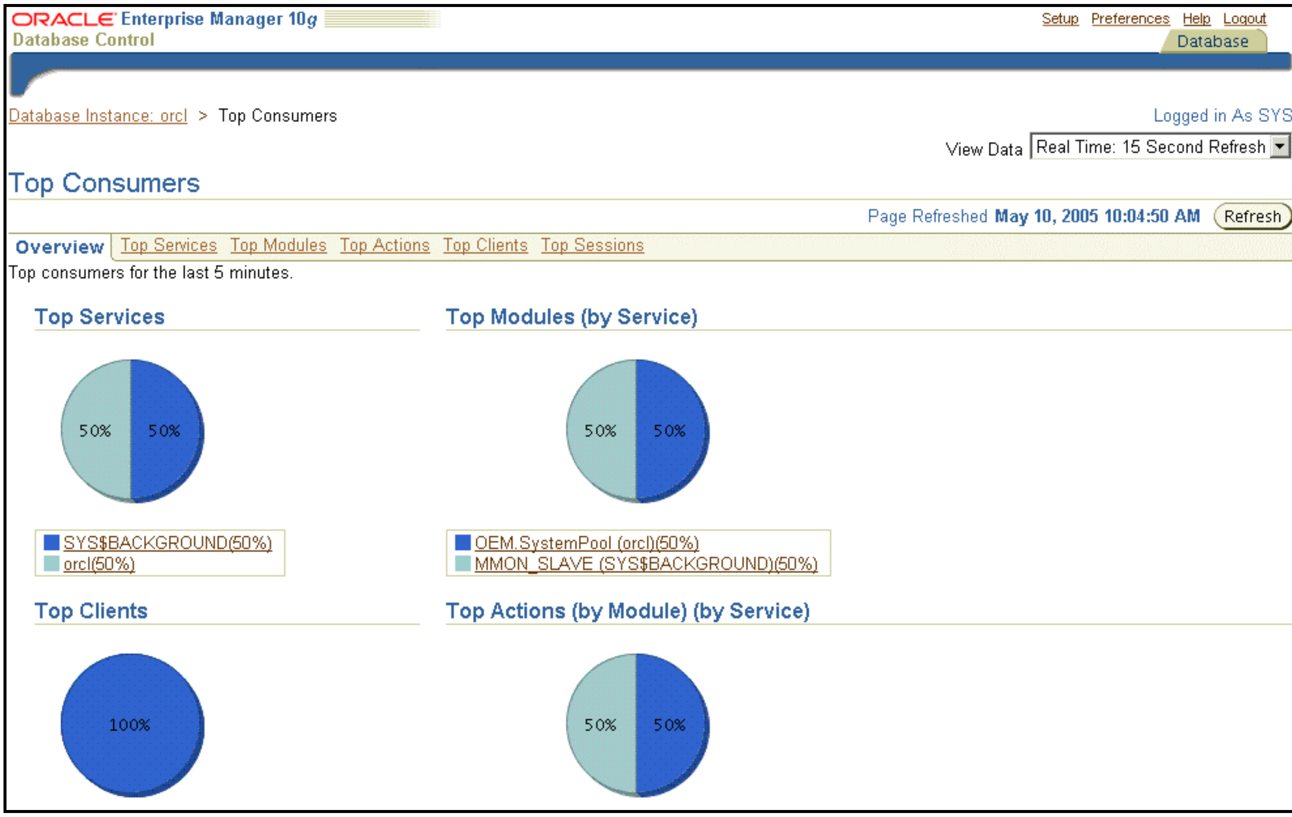
```
User dropped.

SQL>
SQL> create user jfv identified by jfv
  2  default tablespace users
  3  temporary tablespace temp;

User created.

SQL>
SQL> grant connect, resource, dba to jfv;

Grant succeeded.

SQL>
SQL> connect jfv/jfv@SERV1
Connected.
SQL>
SQL> select count(*) from
dba_objects,dba_objects,dba_objects;
```

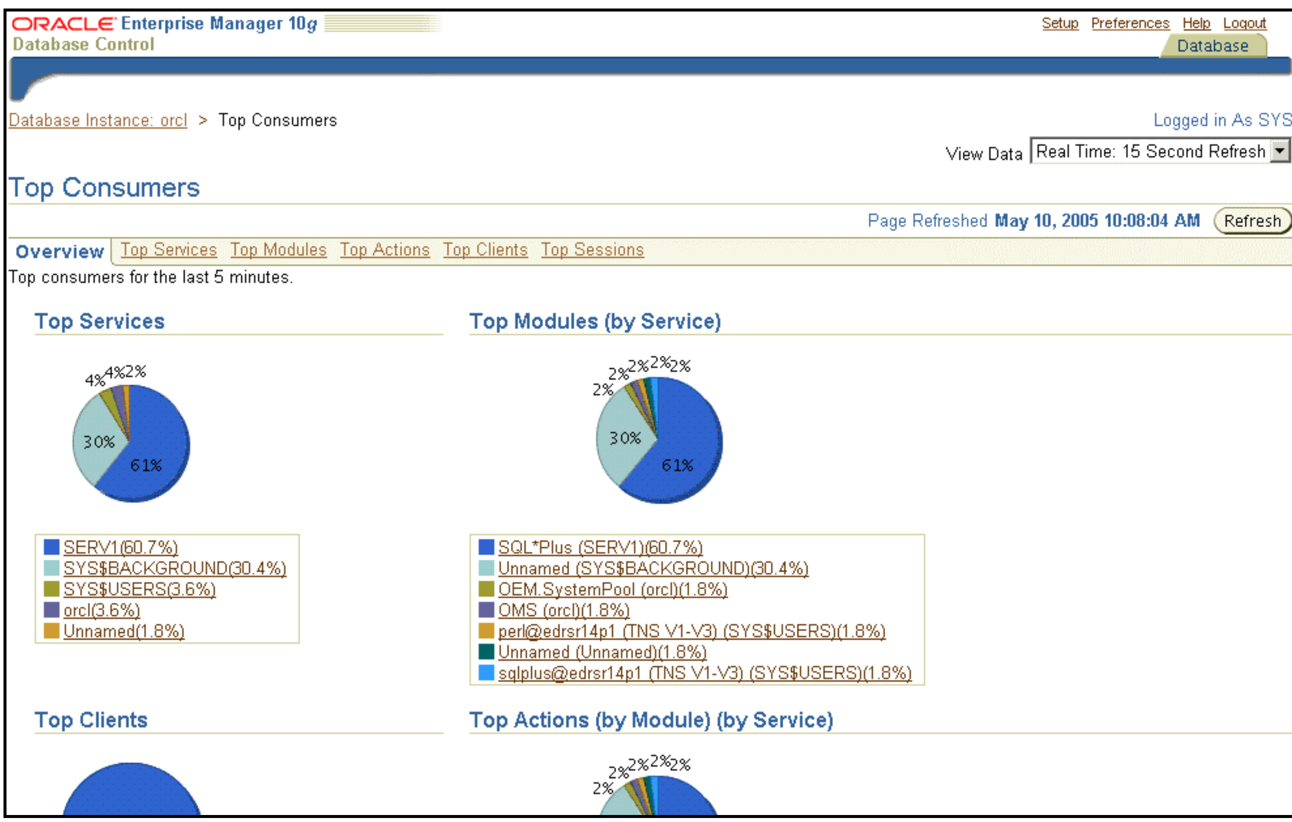
5) After the execution starts, access the EM Top Consumers page from the Performance tabbed page, and check that `SERV1` is using more and more resources. Also, check the statistics on your service with `V$SERVICE_STATS` from a SQL*Plus session connected as `SYSDBA`.

Answer:

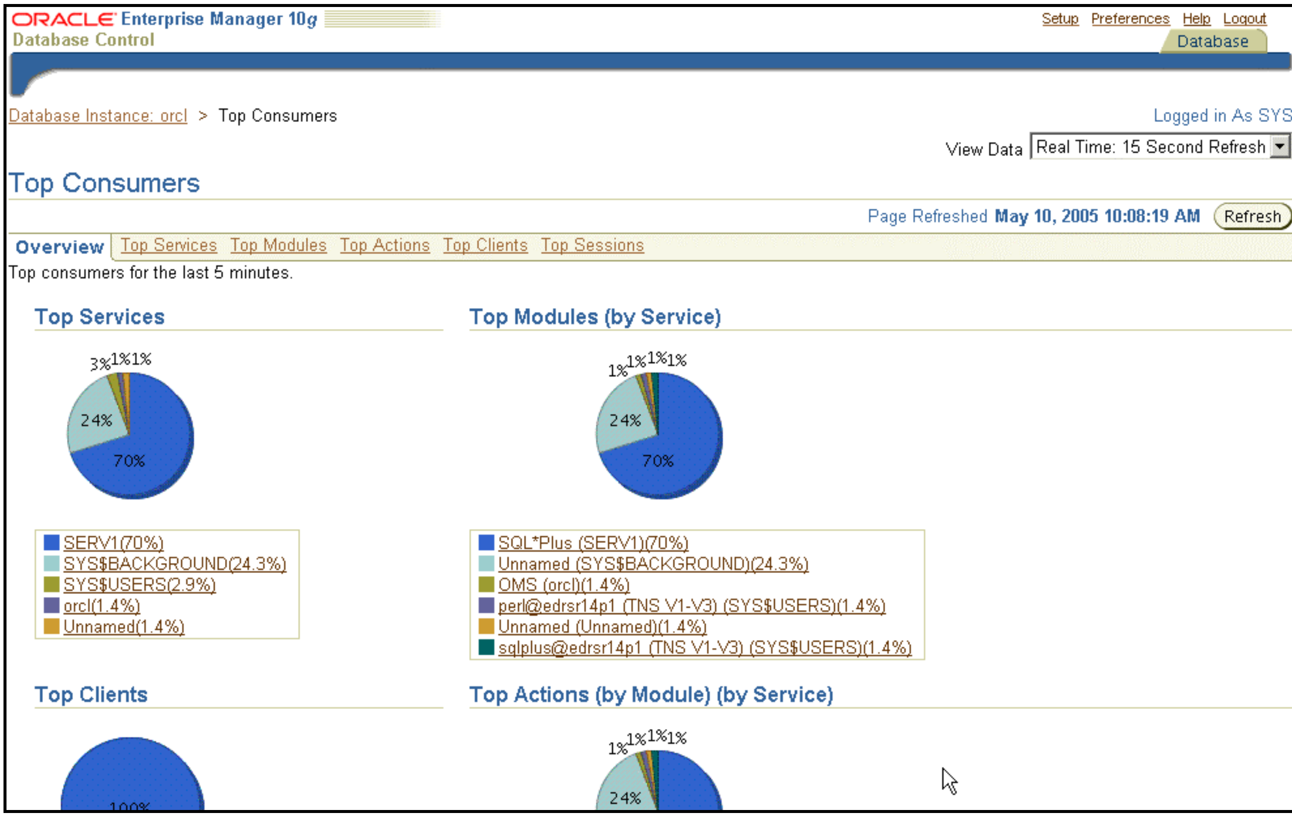
1. On the home page, click the Performance tab. In the middle of the Performance tabbed page, click the Top Consumers link. Refresh the Top Consumers Overview page several times. The names and number of services listed in the Top Services graph depend on the number and type of connections to the database. The network service name of each connection is recorded as a separate sevice. As a result, all the connections made without a service name will be aggregated, as will all the connections made as `SERV1`.

Solutions for Practice 16-1: Using Services with a Single-Instance Oracle Database (Optional) (continued)

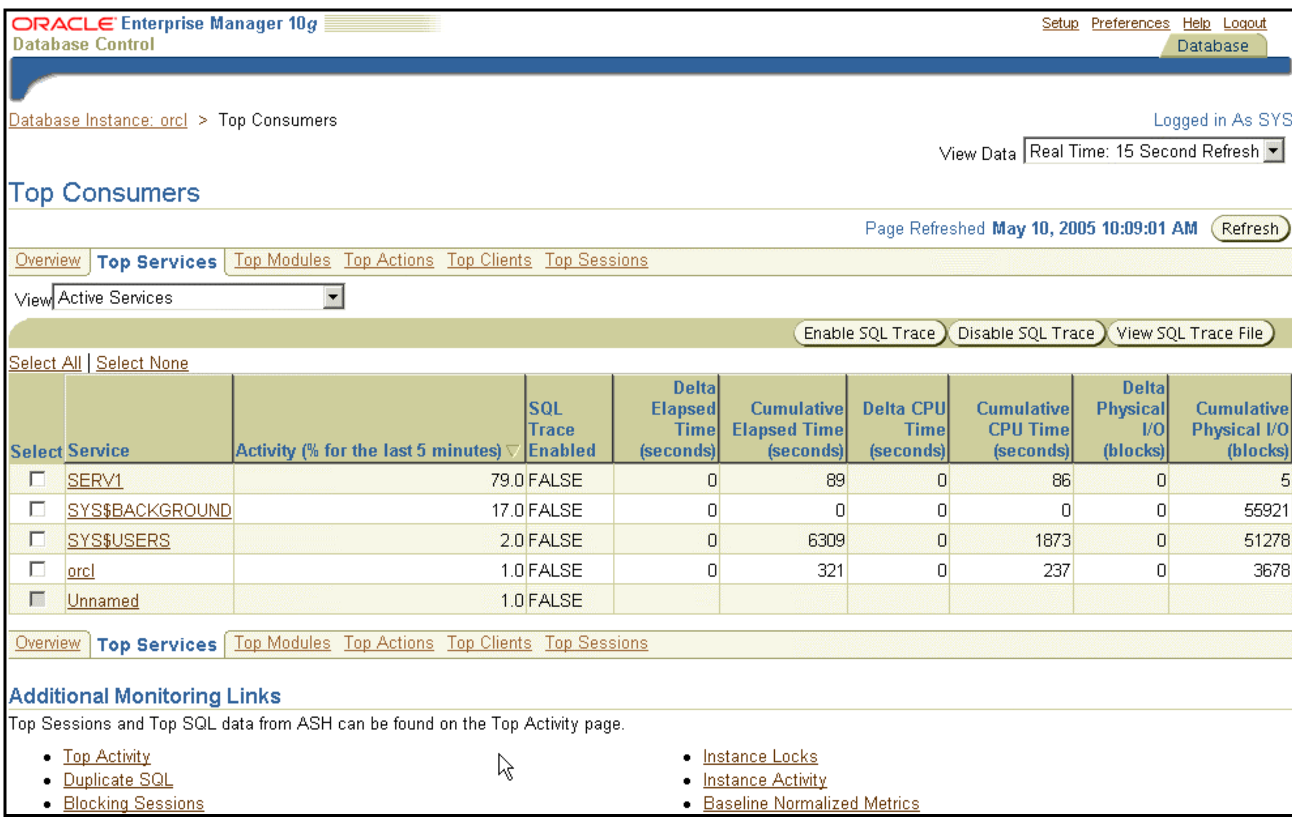




Solutions for Practice 16-1: Using Services with a Single-Instance Oracle Database (Optional) (continued)

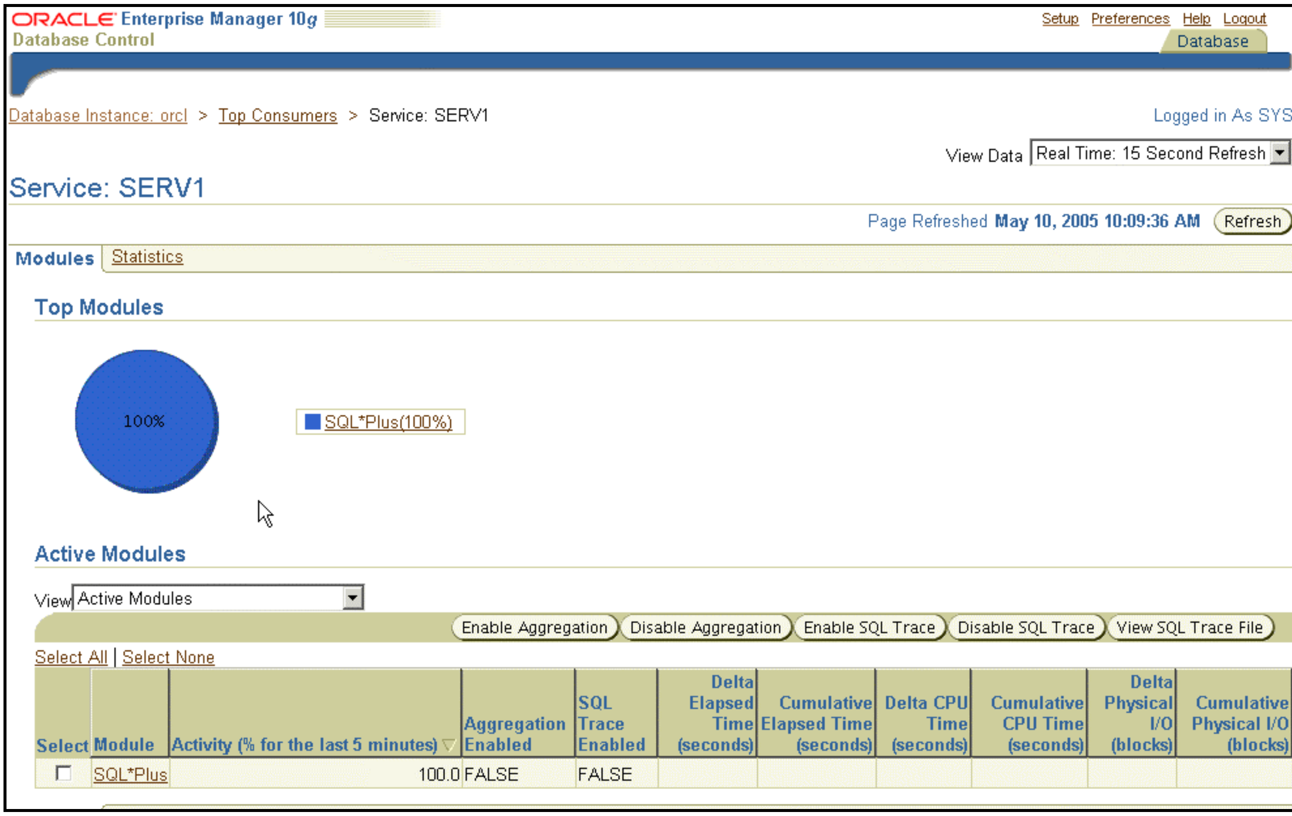


2. You can also see the detailed statistics by clicking the Top Services tab…



Solutions for Practice 16-1: Using Services with a Single-Instance Oracle Database (Optional) (continued)

3. … and then by clicking the SERV1 link…



4. … and then by clicking the Statistics tab.

Solutions for Practice 16-1: Using Services with a Single-Instance Oracle Database (Optional) (continued)

Service: SERV1

Page Refreshed **May 10, 2005 10:09:57 AM** Refresh

Modules | **Statistics**

Previous 1-25 of 28 Next 3

Name	Delta Value	Cumulative Value
logons cumulative	0	2
user calls	0	37
DB time	0	142126880
DB CPU	0	137527712
parse count (total)	0	47
parse time elapsed	0	153454
execute count	0	173
sql execute elapsed time	0	141950848
opened cursors cumulative	0	48
session logical reads	0	11189
physical reads	0	5
physical writes	0	0
redo size	0	0
user commits	0	0
workarea executions - optimal	0	13
workarea executions - onepass	0	0
workarea executions - multipass	0	0
session cursor cache hits	0	0
user rollbacks	0	0
db block changes	0	0
gc cr blocks received	0	0
gc cr block receive time	0	0
gc current blocks received	0	0
gc current block receive time	0	0
cluster wait time	0	0

5. Make sure that you stop your running workload by pressing [Ctrl] + [C] in your terminal window if necessary.

```
select count(*) from dba_objects,dba_objects,dba_objects
                          *
ERROR at line 1:
ORA-01013: user requested cancel of current operation



SQL> exit
```

6) Set alert thresholds for your SERV1 service by using Database Control. Specify the values defined at the beginning of this practice.

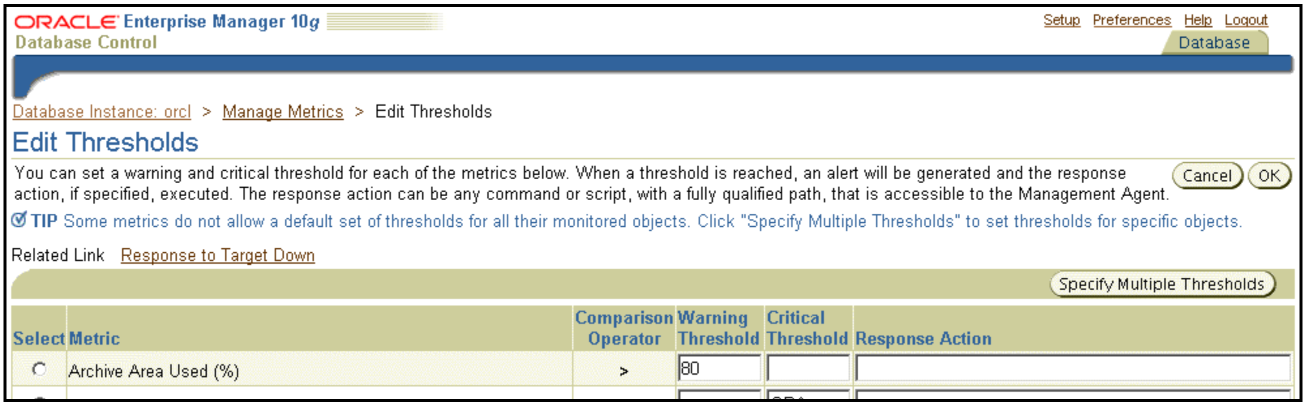
Answer:

1. On the Database Control home page, click the Manage Metrics link in the Related Links section.
2. On the Manage Metrics page, click the Edit Thresholds button.
3. On the Edit Thresholds page, select the metric called Service Response Time (per user call) (microseconds).

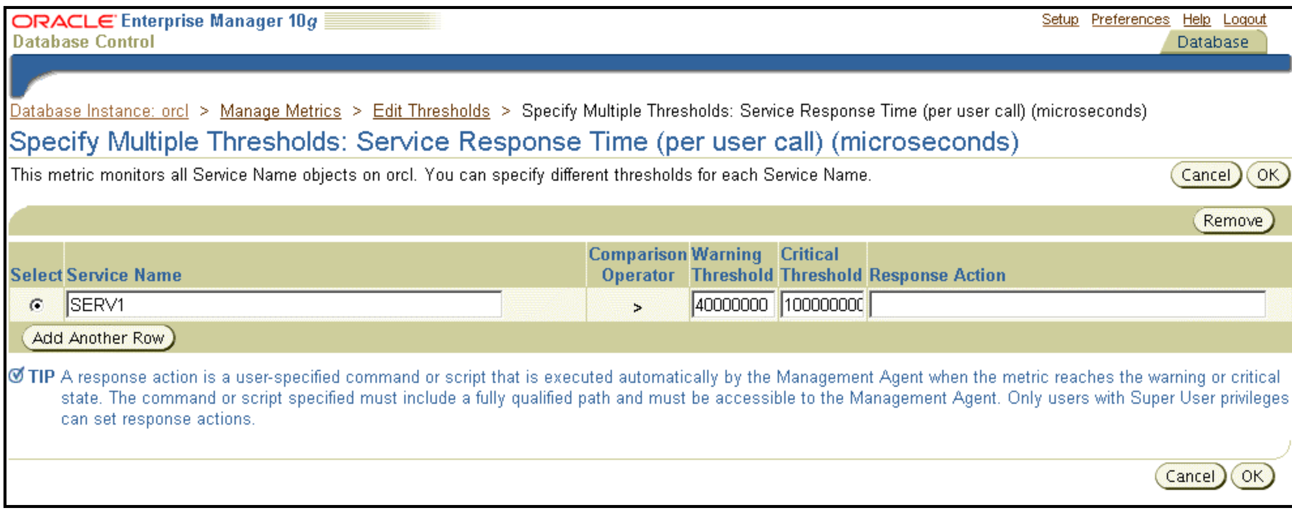
⊙	Service Response Time (per user call) (microseconds)	>			

4. Click the Specify Multiple Thresholds button at the beginning of the page.

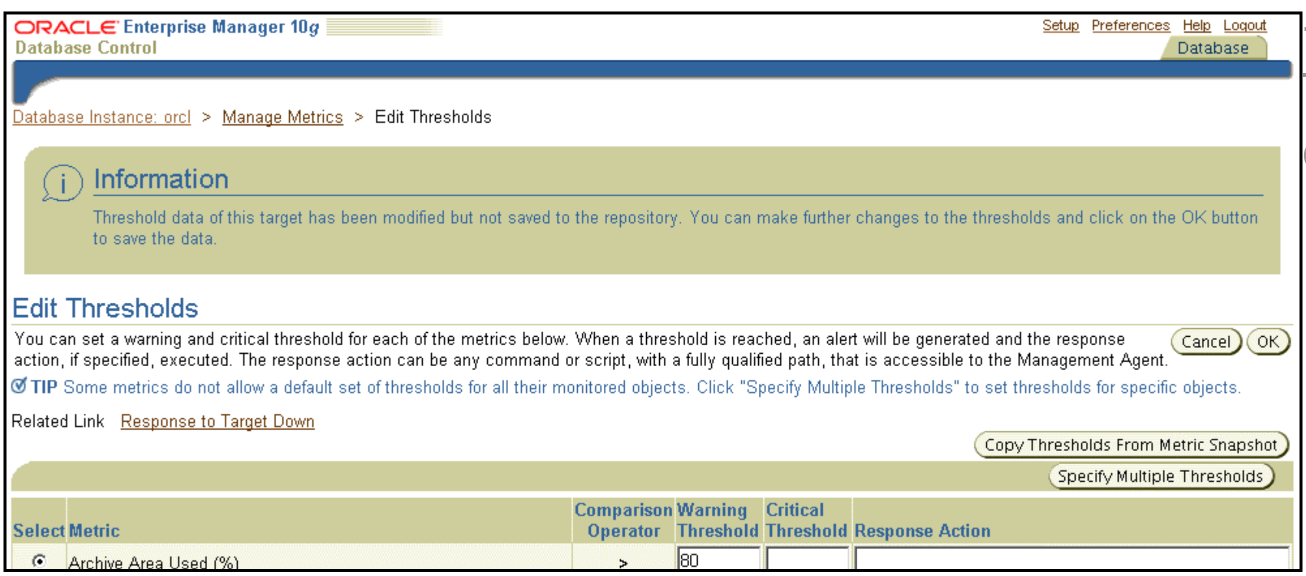
Solutions for Practice 16-1: Using Services with a Single-Instance Oracle Database (Optional) (continued)



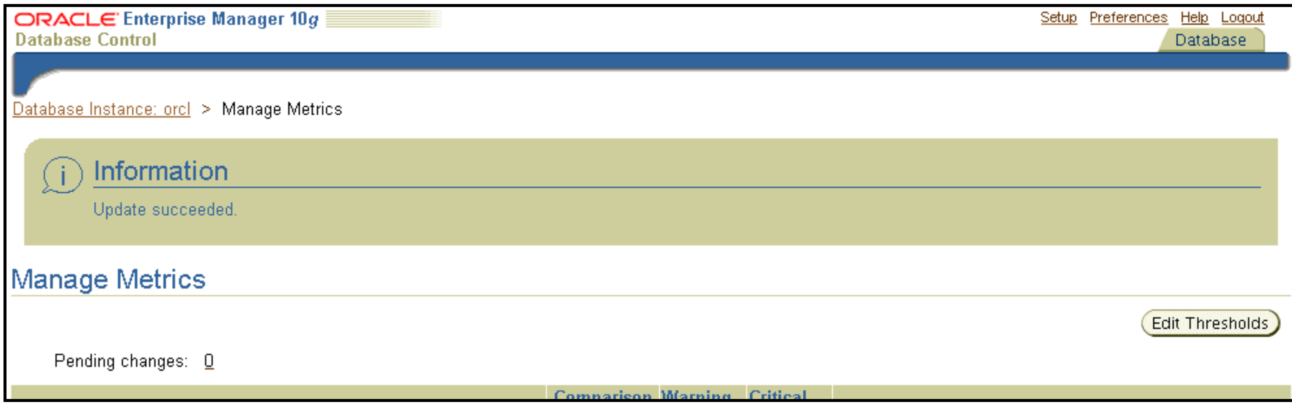
5. On the Specify Multiple Thresholds: Service Response Time (per user call) (microseconds) page, enter `SERV1` in the Service Name field, `40000000` in the Warning Threshold field, and `100000000` in the Critical Threshold field.



6. Click OK.



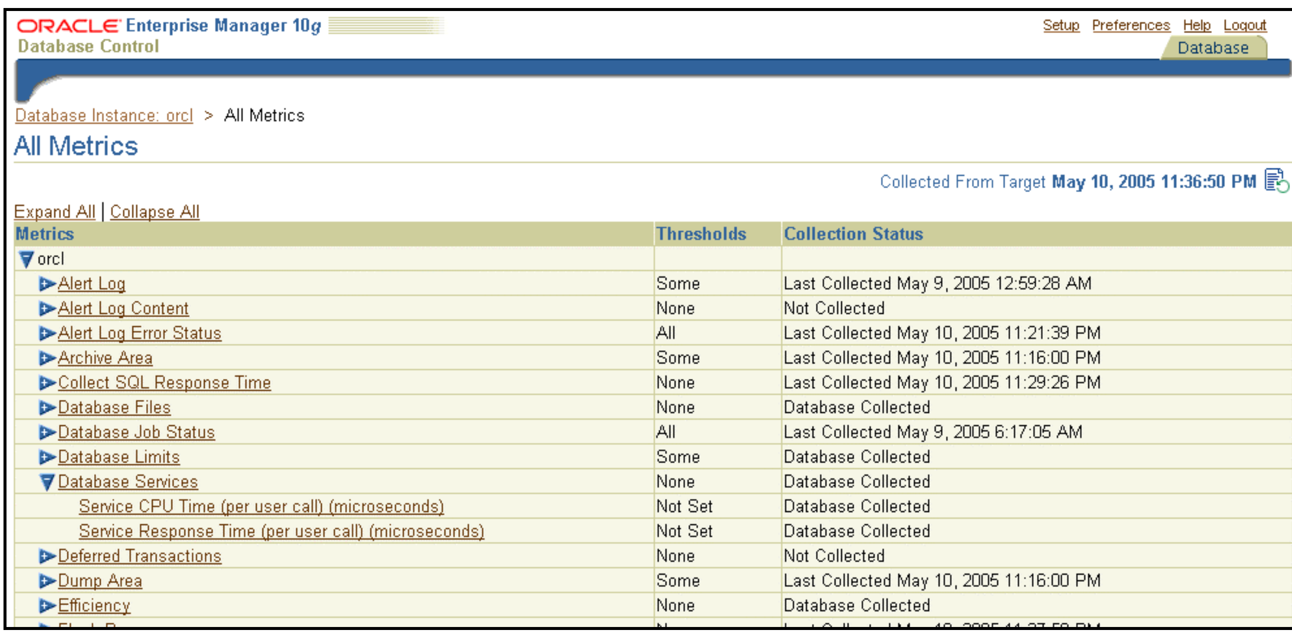
7. Back on the Edit Thresholds page, click OK.



8. On the Manage Metrics page, click the Database tab.

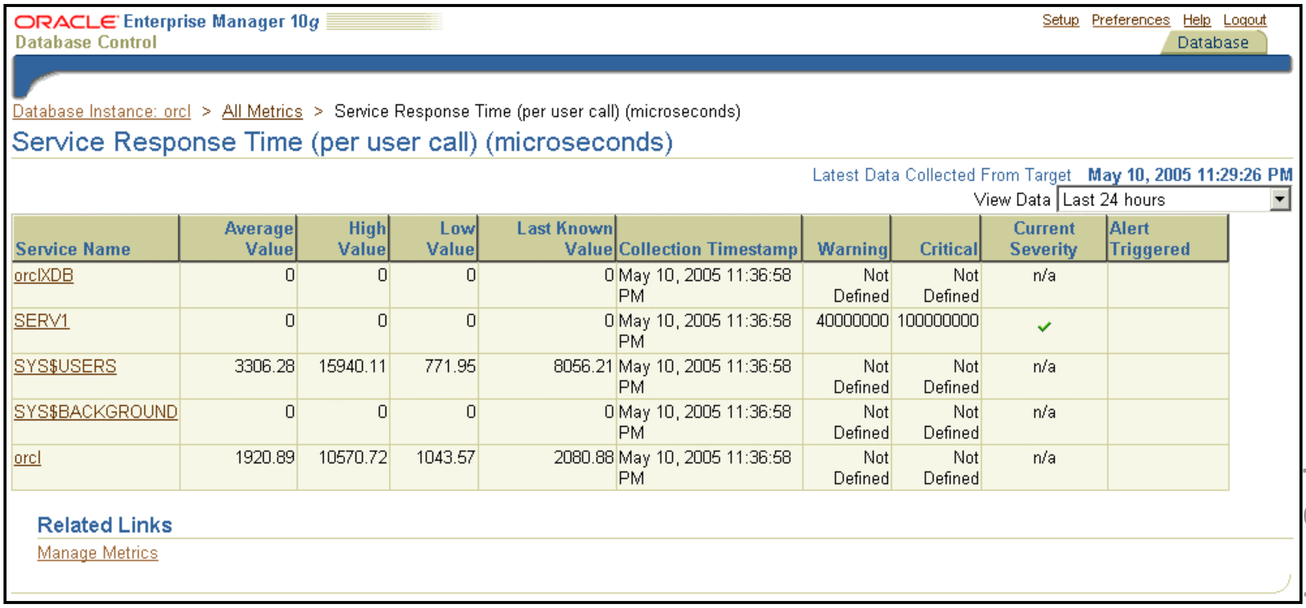
9. Back on the home page, click the All Metrics link in the Related Links section.

10. On the All Metrics page, expand the Database Services link.



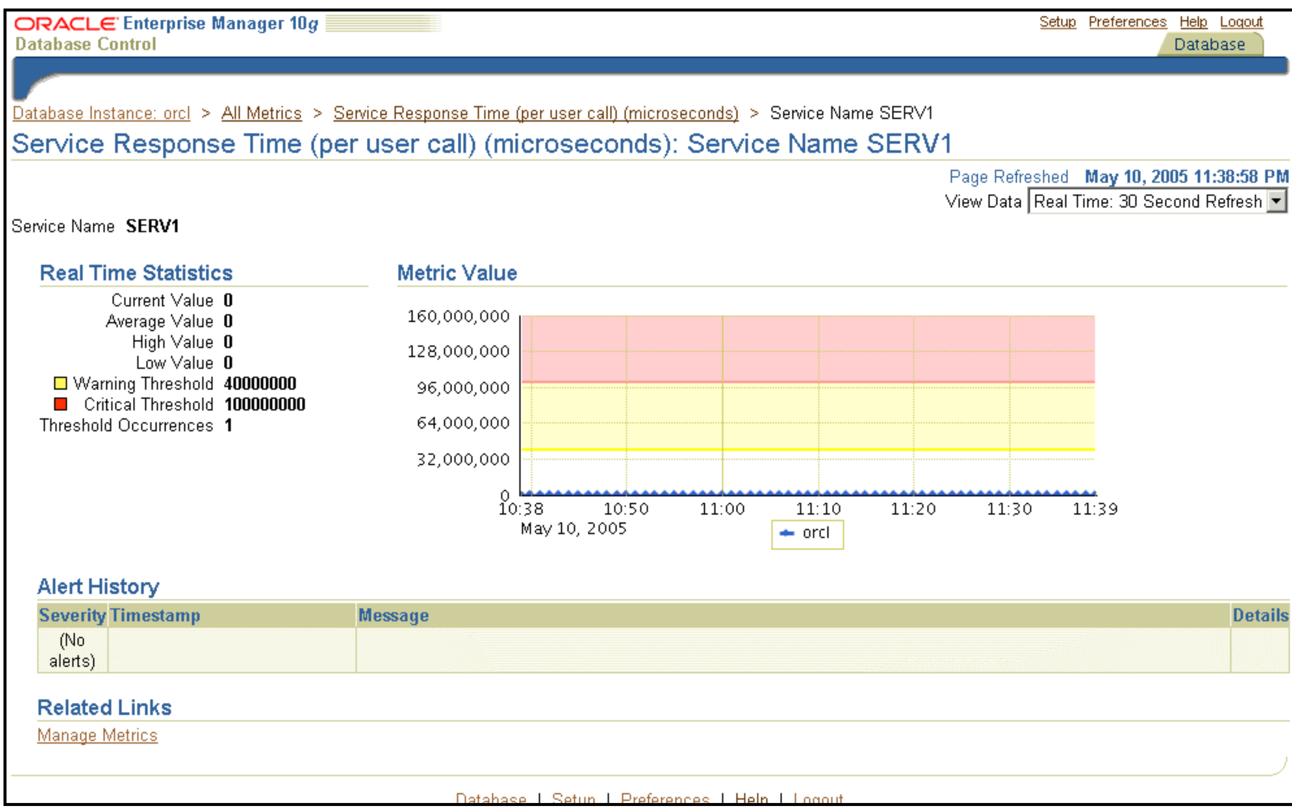
11. On the All Metrics page, click the Service Response Time (per user call) (microseconds) link.

12. On the Service Response Time (per user call) (microseconds) page, click the SERV1 link in the Service Name column.



Service Name	Average Value	High Value	Low Value	Last Known Value	Collection Timestamp	Warning	Critical	Current Severity	Alert Triggered
orclXDB	0	0	0	0	May 10, 2005 11:36:58 PM	Not Defined	Not Defined	n/a	
SERV1	0	0	0	0	May 10, 2005 11:36:58 PM	40000000	100000000	✓	
SYS$USERS	3306.28	15940.11	771.95	8056.21	May 10, 2005 11:36:58 PM	Not Defined	Not Defined	n/a	
SYS$BACKGROUND	0	0	0	0	May 10, 2005 11:36:58 PM	Not Defined	Not Defined	n/a	
orcl	1920.89	10570.72	1043.57	2080.88	May 10, 2005 11:36:58 PM	Not Defined	Not Defined	n/a	

13. On the Service Response Time (per user call) (microseconds): Service Name SERV1 page, select Real Time: 30 Second Refresh from the View Data drop-down list. Do NOT close this window; you will return to it in the next step.



7) From your terminal emulator session, execute the `lab_c_01_07.sql` script. This script executes a workload under the `SERV1` service. Observe the Metric Value graph on the Service Response Time (per user call) (microseconds): Service Name SERV1 page. What is your conclusion?

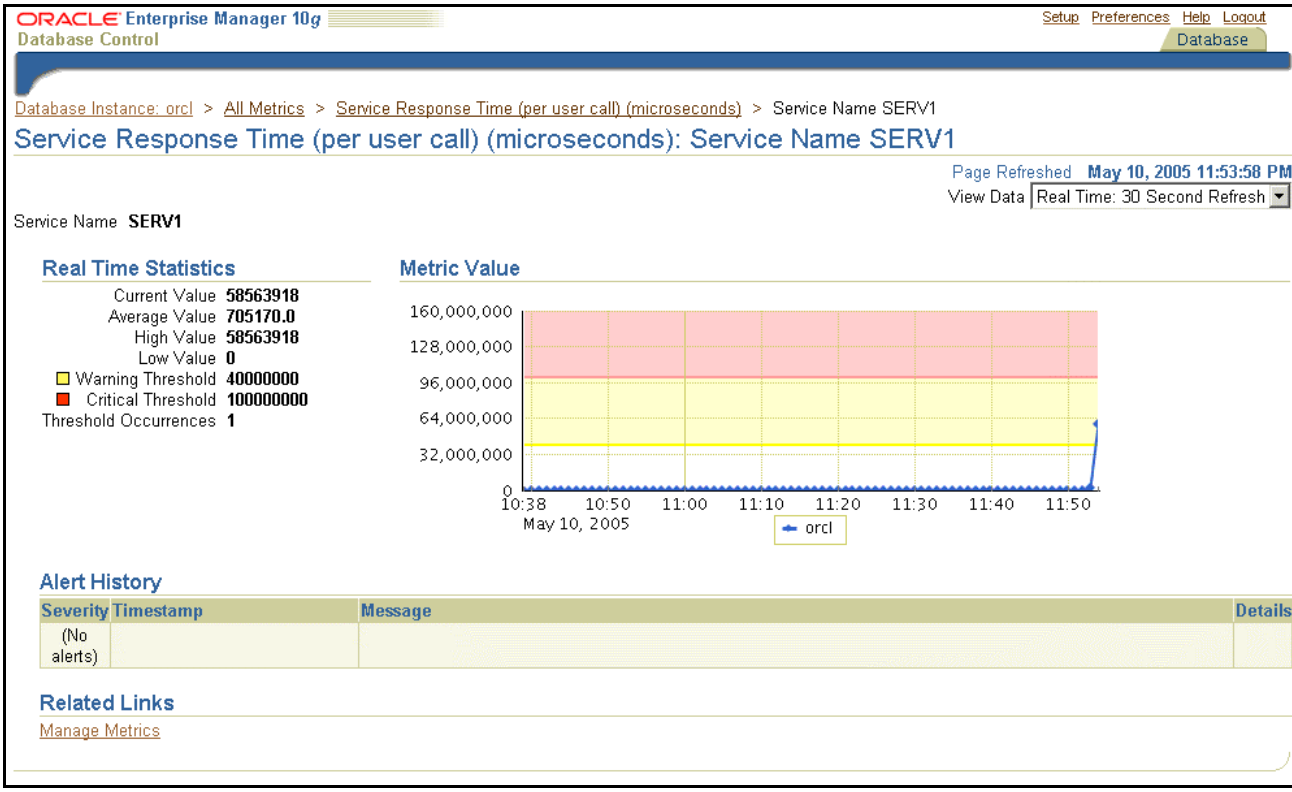
   Answer: You see that the metric passes the warning threshold. Soon after, you see an alert raised on the Database Console.

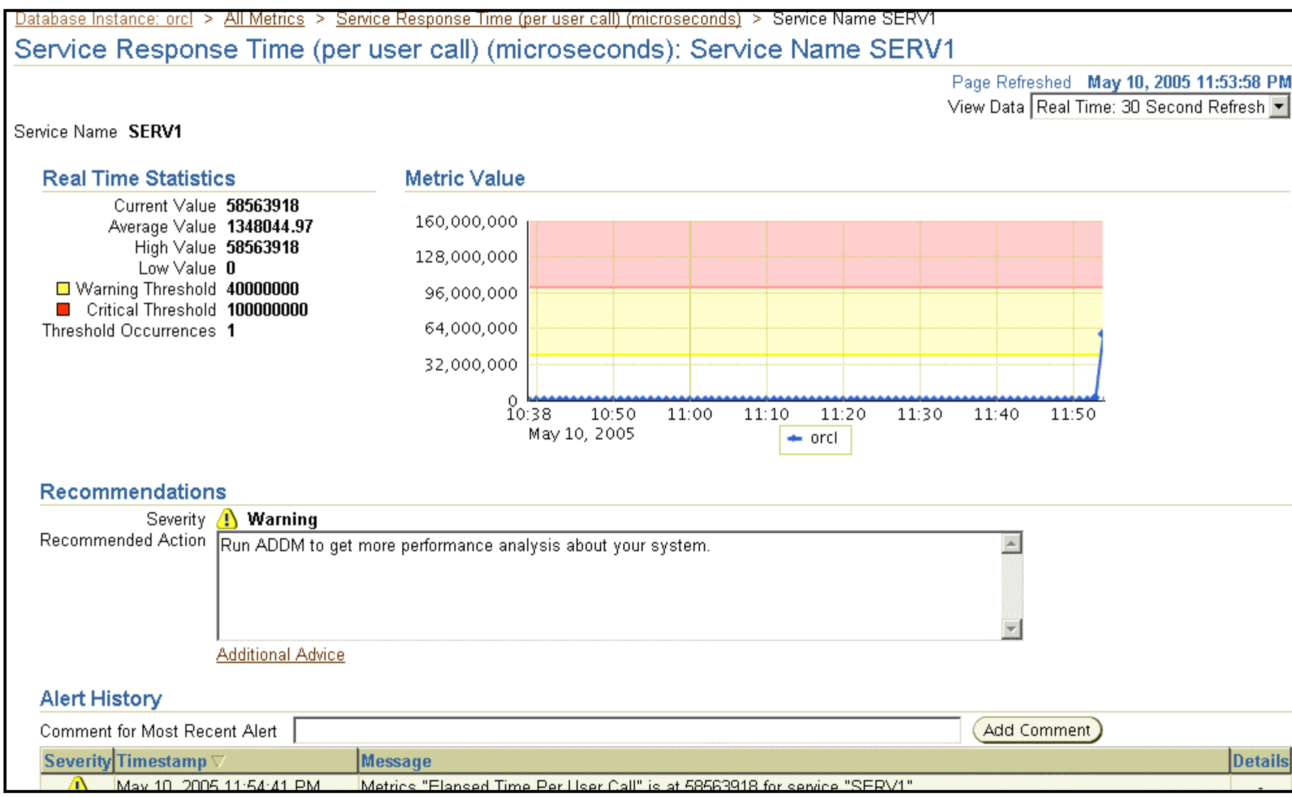
   1. Execute the script. It takes a while to complete. Do not wait until it completes. Look at the corresponding threshold history by using Database Control. Return to the Service Response Time (per user call) (microseconds): Service Name `SERV1` page, from the previous step.

```
 $ sqlplus /nolog

SQL*Plus: Release 10.2.0.1.0 - Production on Thu Dec 8
01:57:39 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

SQL> @lab_c_01_07.sql
SQL>
SQL> connect jfv/jfv@SERV1
Connected.
SQL>
SQL>
SQL> DECLARE
  2  t number;
  3  BEGIN
  4  for i in 1..2000 loop
  5    select count(*) into t from dba_objects;
  6  end loop;
  7  END;
  8  /
```

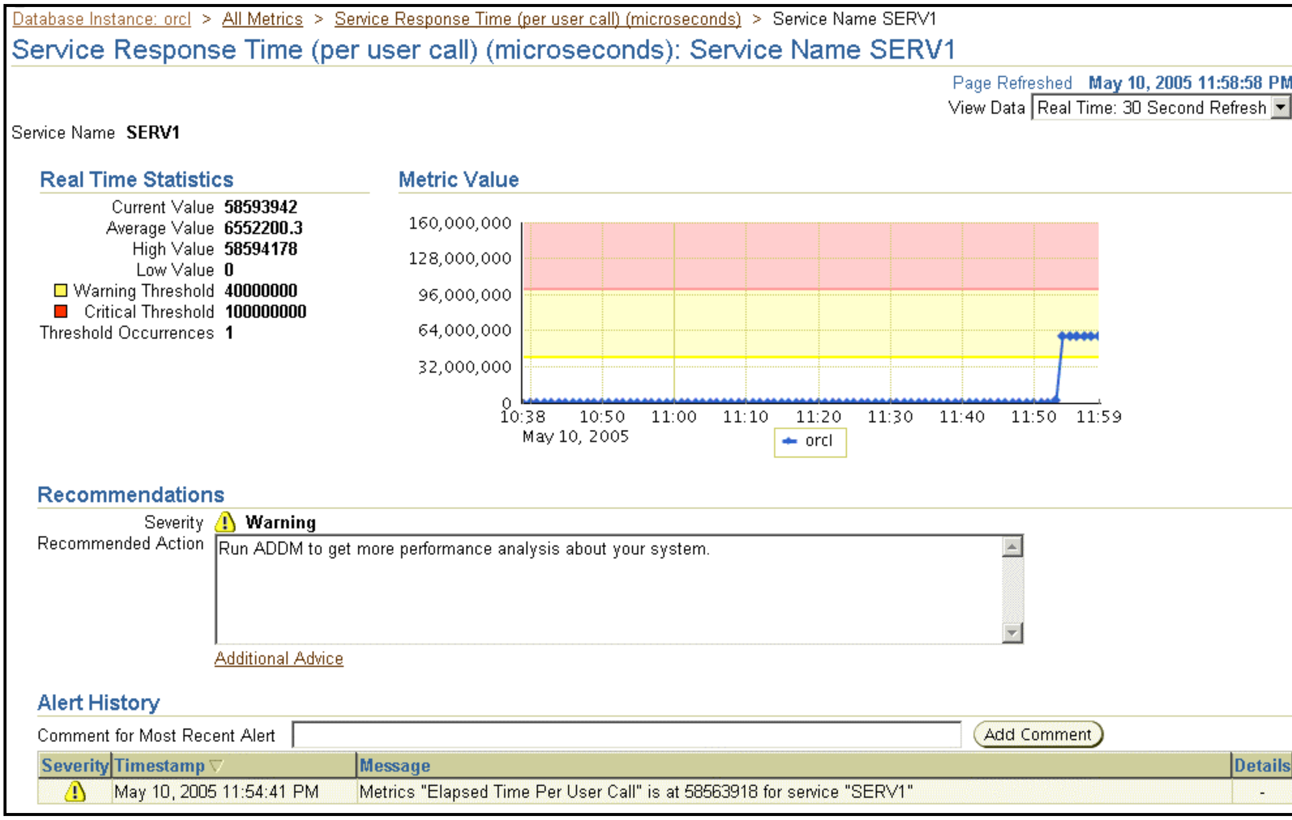
   2. On the Service Response Time (per user call) (microseconds): Service Name SERV1 page, monitor the Metric Value graph. The graph on your machine may vary from the examples.

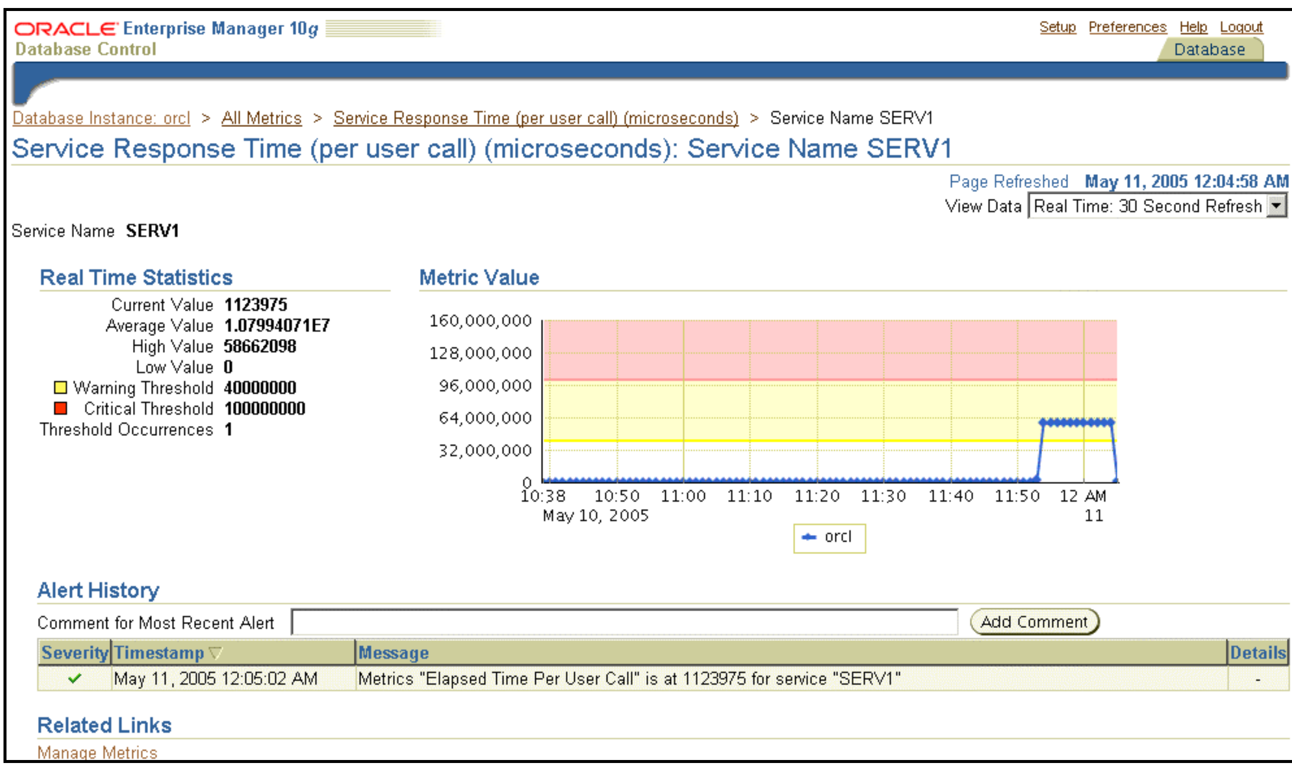
Solutions for Practice 16-1: Using Services with a Single-Instance Oracle Database (Optional) (continued)





Solutions for Practice 16-1: Using Services with a Single-Instance Oracle Database (Optional) (continued)



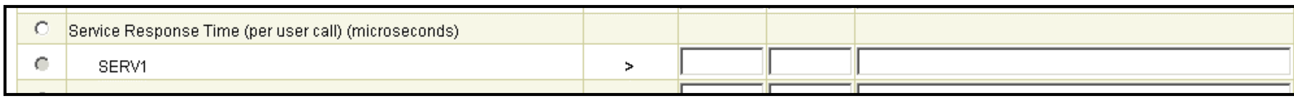


8) Use Database Control to remove the thresholds that you specified during this practice.

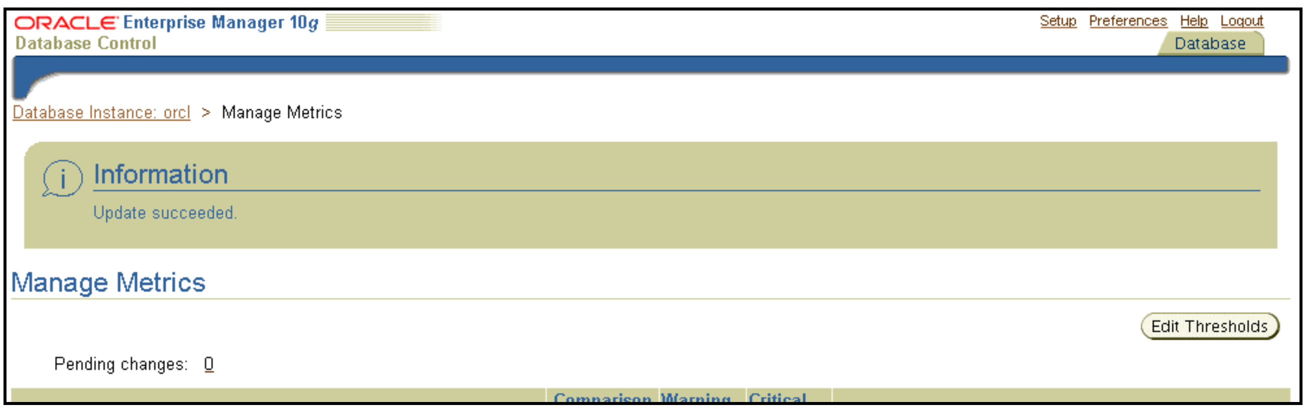
Solutions for Practice 16-1: Using Services with a Single-Instance Oracle Database (Optional) (continued)

Answer:

1. On the Database Control home page, click the Manage Metrics link in the Related Links section.
2. On the Manage Metrics page, click the Edit Thresholds button.
3. On the Edit Thresholds page, remove the thresholds that you previously set for the metric called Service Response Time (per user call) (microseconds).



4. Click OK.



Solutions for Practice 16-2: *Trace Services in a Single-Instance Oracle Environment (Optional)*

Unless specified differently, you should log in as `SYSDBA` under your Database Control Console.

1) Execute the `lab_c_02_01.sh` script to remove all the trace files already generated in your `$ORACLE_BASE/admin/orcl/udump` directory. Then, execute the `start_servwork.sh` script to start four sessions using the `SERV1` service.

   Answer:

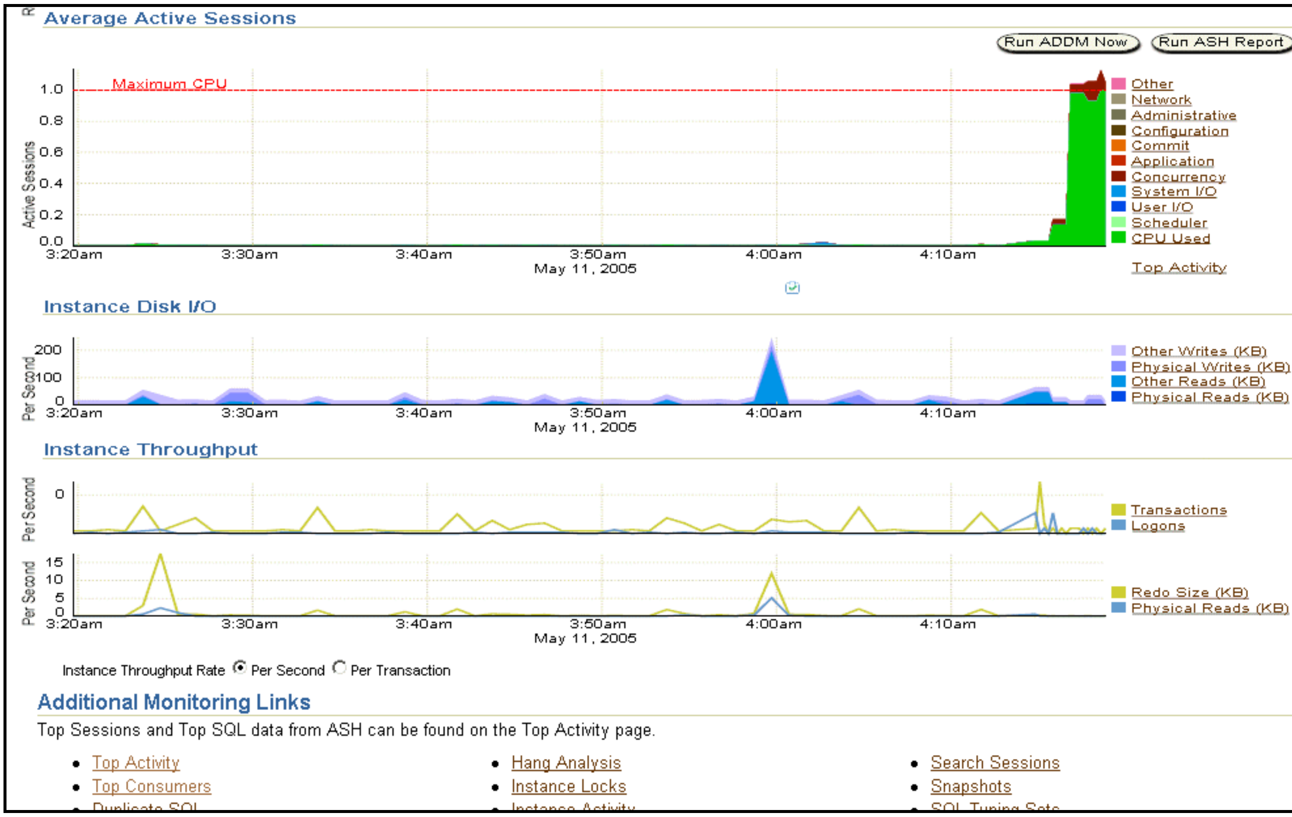
```
$ rm -f $ORACLE_BASE/admin/orcl/udump/*


$ ./start_servwork.sh
Started stream with pid=7904
Started stream with pid=7905
Started stream with pid=7906
Started stream with pid=7907
$
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


$
```

Solutions for Practice 16-2: Trace Services in a Single-Instance Oracle Environment (Optional) (continued)

2) Using Database Control, determine the list of services, modules, and actions that the workload is using.

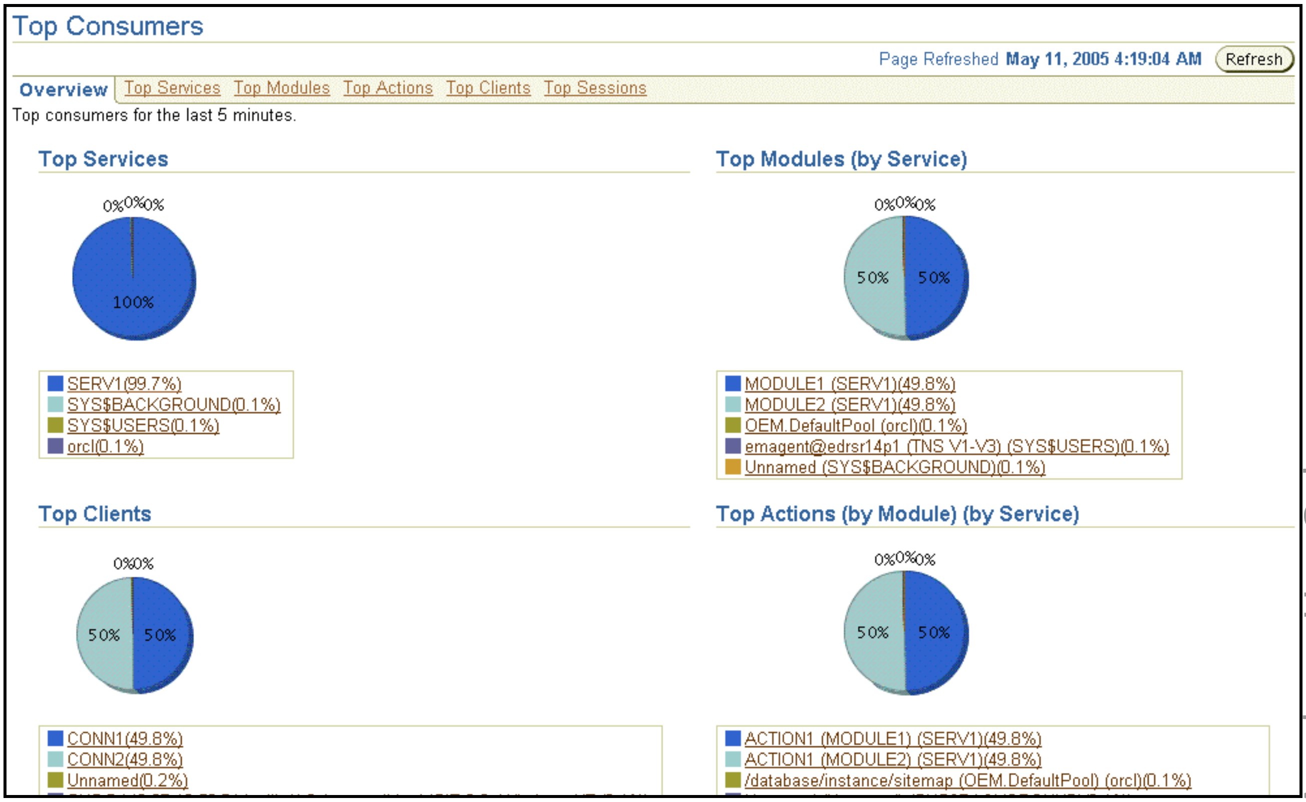
Answer:

1. On your Database Control home page, click the Performance tab.



2. On the Performance tabbed page, click the Top Consumers link. You should see that the current workload uses only the `SERV1` service, and that half of the sessions are using the `SERV1/MODULE1/ACTION1` action, and that the other half is using the `SERV1/MODULE2/ACTION1` action.

Solutions for Practice 16-2: Trace Services in a Single-Instance Oracle Environment (Optional) (continued)

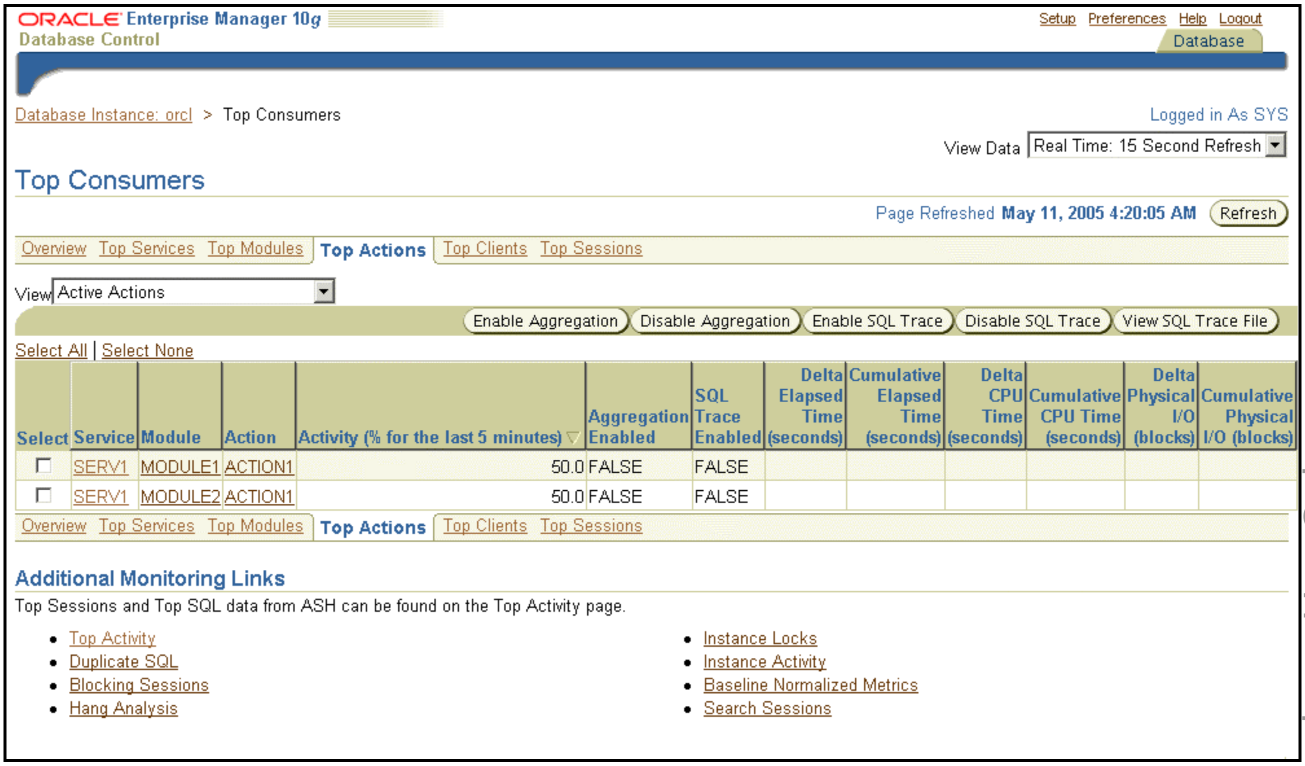


3) Using Database Control, enable statistics aggregation for both `SERV1/MODULE1/ACTION1` and `SERV1/MODULE2/ACTION1`.

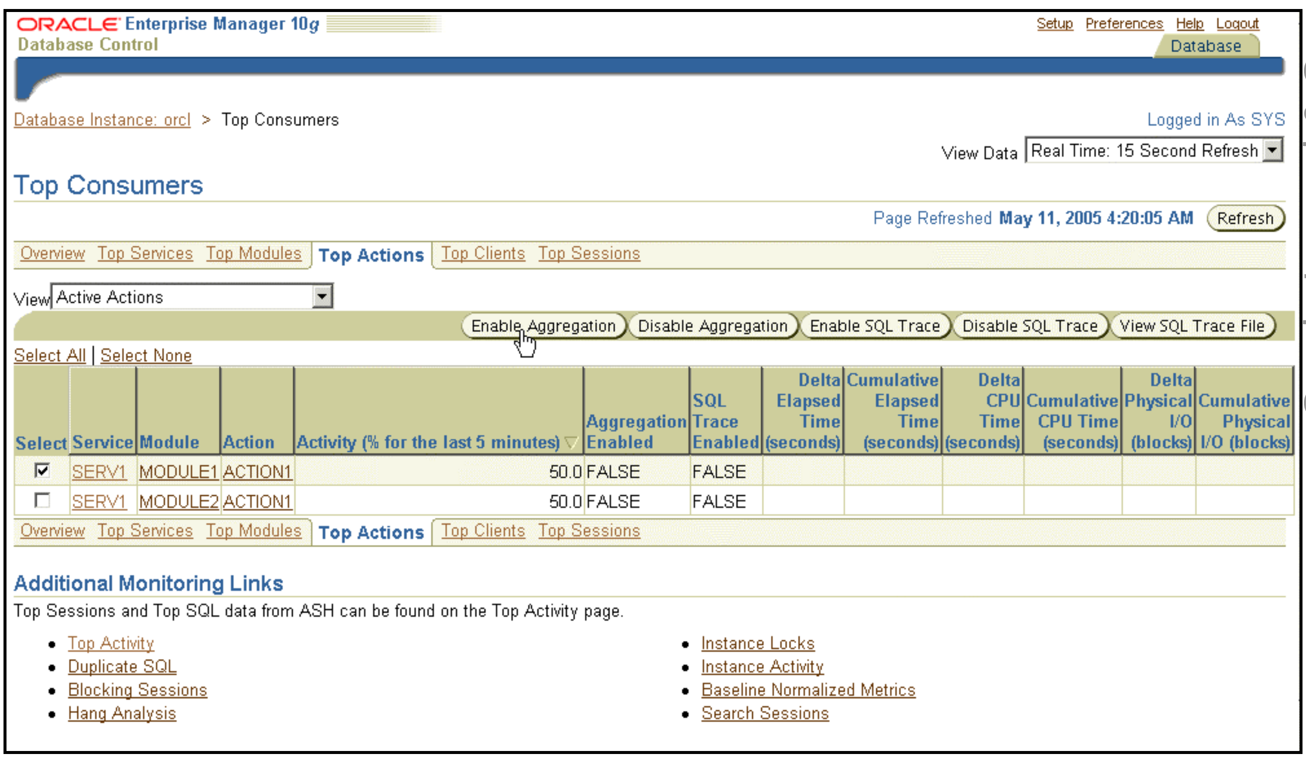
Answer:

1. On the Top Consumers page, click the Top Actions tab.

Solutions for Practice 16-2: Trace Services in a Single-Instance Oracle Environment (Optional) (continued)

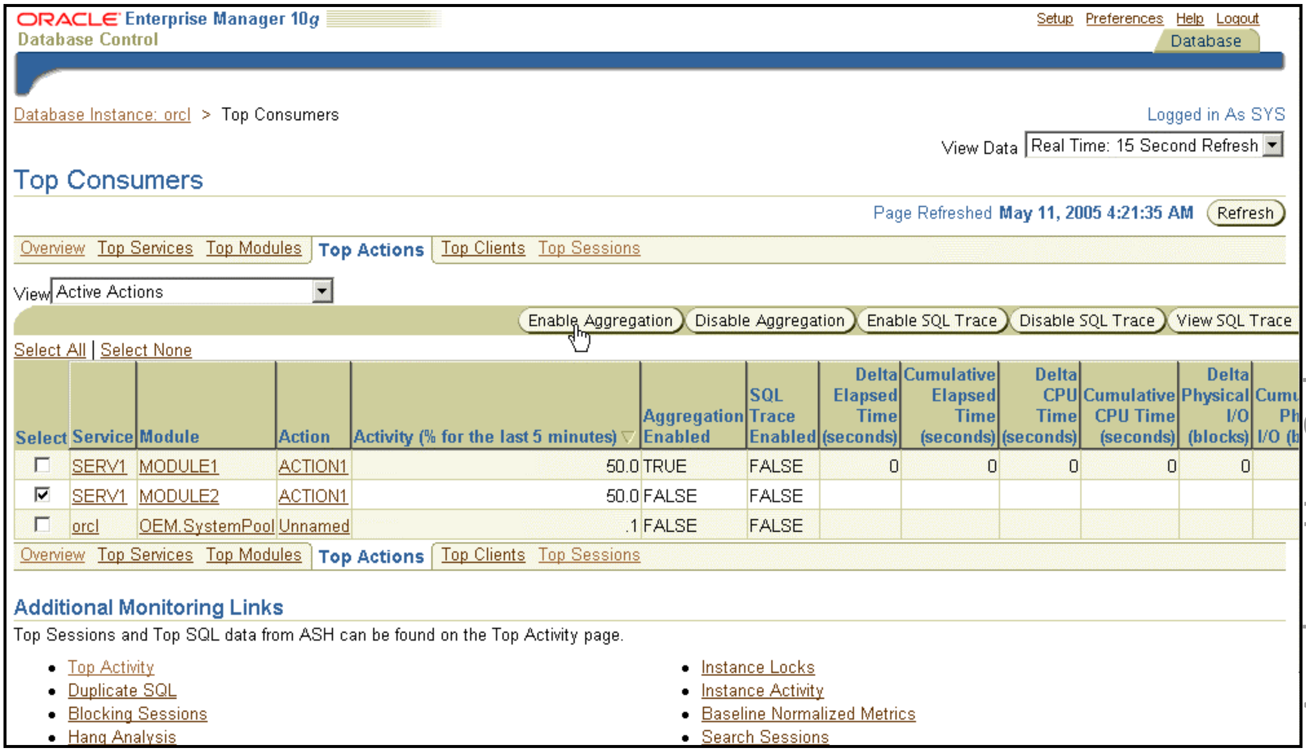


2. From the Top Actions tabbed page, select the SERV1/MODULE1/ACTION1 action, and click the Enable Aggregation button.



Solutions for Practice 16-2: Trace Services in a Single-Instance Oracle Environment (Optional) (continued)

3. After this is done, do the same for the SERV1/MODULE2/ACTION1 action. You should see that additional statistics are now tracked for both actions.

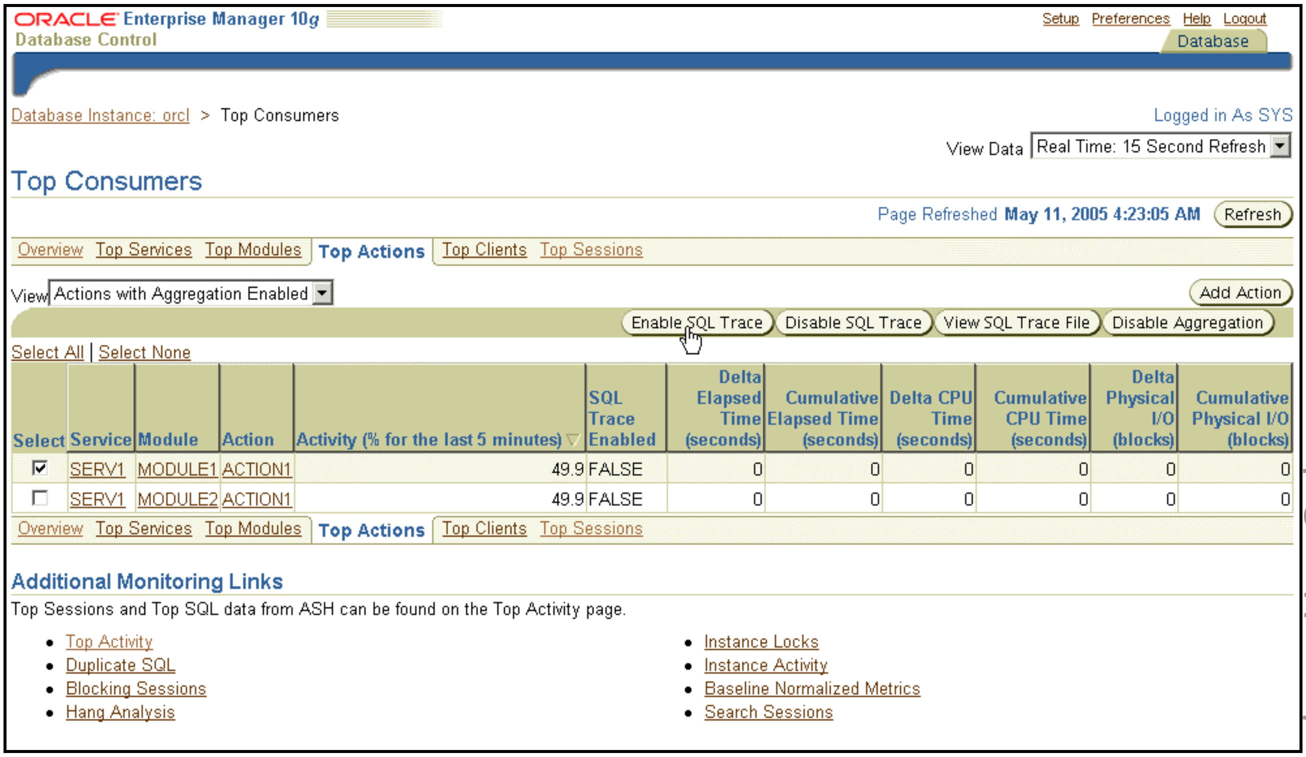


4) Using Database Control, enable tracing for both SERV1/MODULE1/ACTION1 and SERV1/MODULE2/ACTION1. Let the system generate the trace files for one minute, and then disable tracing for both SERV1/MODULE1/ACTION1 and SERV1/MODULE2/ACTION1. After this is done, determine the list of generated trace files.

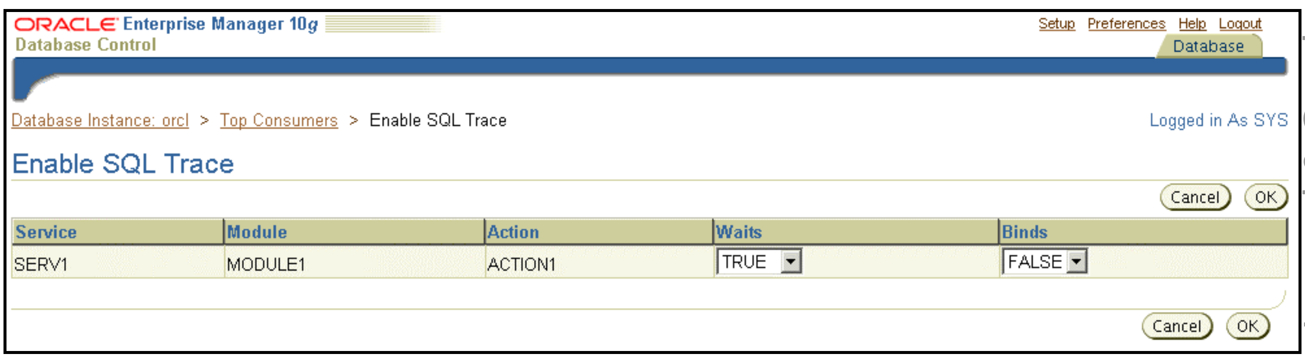
Answer:

1. On the Top Actions page, select "Actions with Aggregation Enable" from the View drop-down list. Then, select the SERV1/MODULE1/ACTION1 action and click the Enable SQL Trace button.

Solutions for Practice 16-2: Trace Services in a Single-Instance Oracle Environment (Optional) (continued)

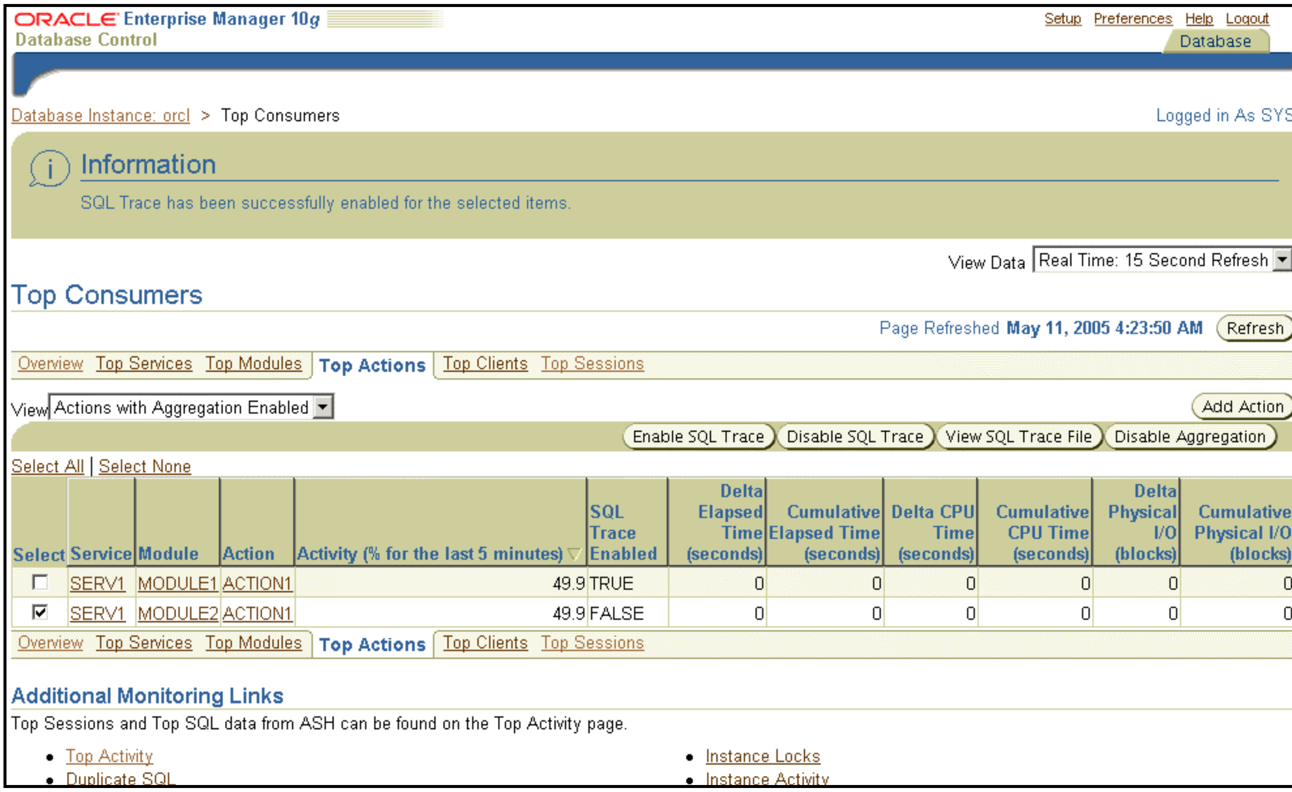


2. On the Enable SQL Trace page, click OK.



3. After this is done, do the same for the `SERV1/MODULE2/ACTION1` action.

Solutions for Practice 16-2: Trace Services in a Single-Instance Oracle Environment (Optional) (continued)




```
$ ls -lt $ORACLE_BASE/admin/orcl/udump
total 14316
-rw-rw----    1 oracle   oinstall  7514908 May 11 04:25 orcl_ora_7909.trc
-rw-rw----    1 oracle   oinstall    34596 May 11 04:25 orcl_ora_7911.trc
-rw-rw----    1 oracle   oinstall    34452 May 11 04:25 orcl_ora_7915.trc
-rw-rw----    1 oracle   oinstall  7041458 May 11 04:25 orcl_ora_7913.trc
$
```

5) Stop your workload by executing the `stop_servwork.sh` script, and then start it up again by executing the `start_servwork.sh` script. When done, check that the statistics for your enabled aggregation actions are increasing.

Answer:

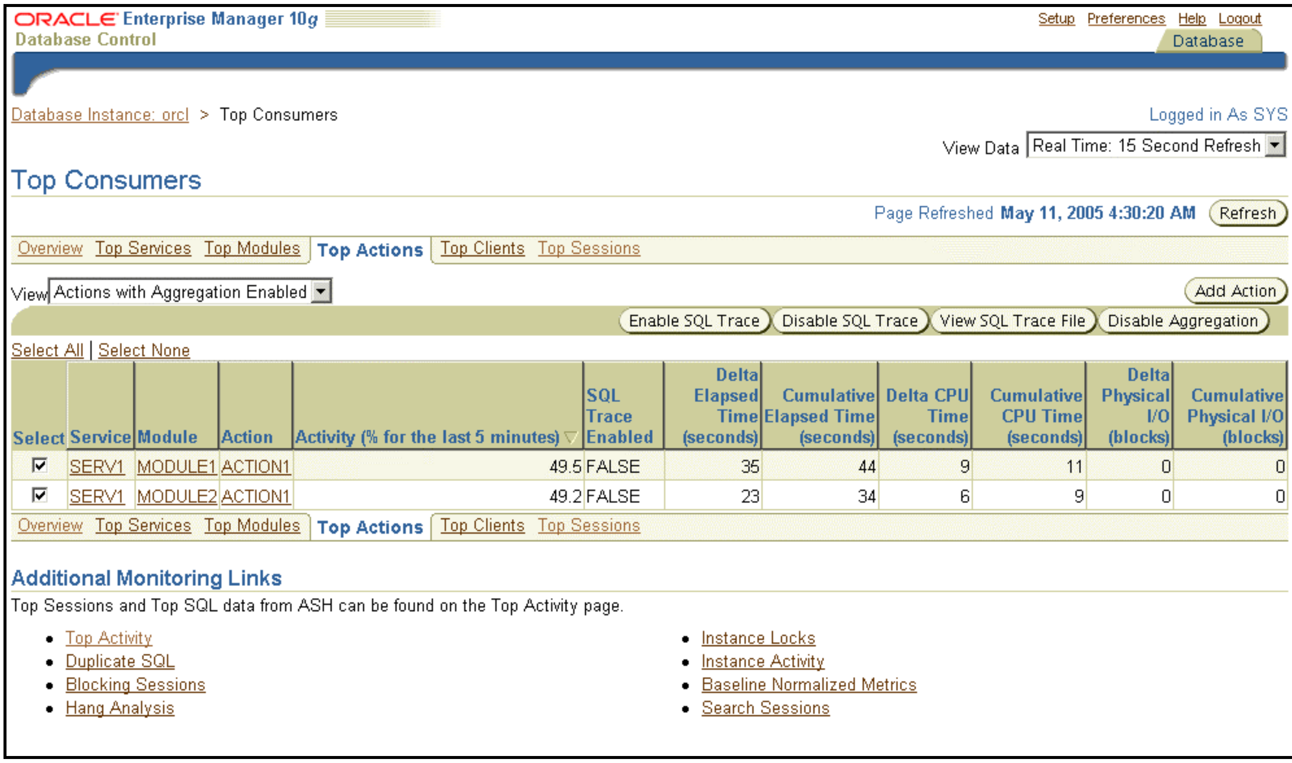
```
$ ./stop_servwork.sh
Killing stream with pid=7907
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed


Killing stream with pid=7906
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed
```

Solutions for Practice 16-2: Trace Services in a Single-Instance Oracle Environment (Optional) (continued)

```
Killing stream with pid=7905
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed


Killing stream with pid=7904
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed


$




$ ./start_servwork.sh
Started stream with pid=8535
Started stream with pid=8536
Started stream with pid=8539
Started stream with pid=8540
$
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

$
```

Solutions for Practice 16-2: Trace Services in a Single-Instance Oracle Environment (Optional) (continued)

6) Look again at your Top Actions page. You should see statistics refreshing for your two actions.

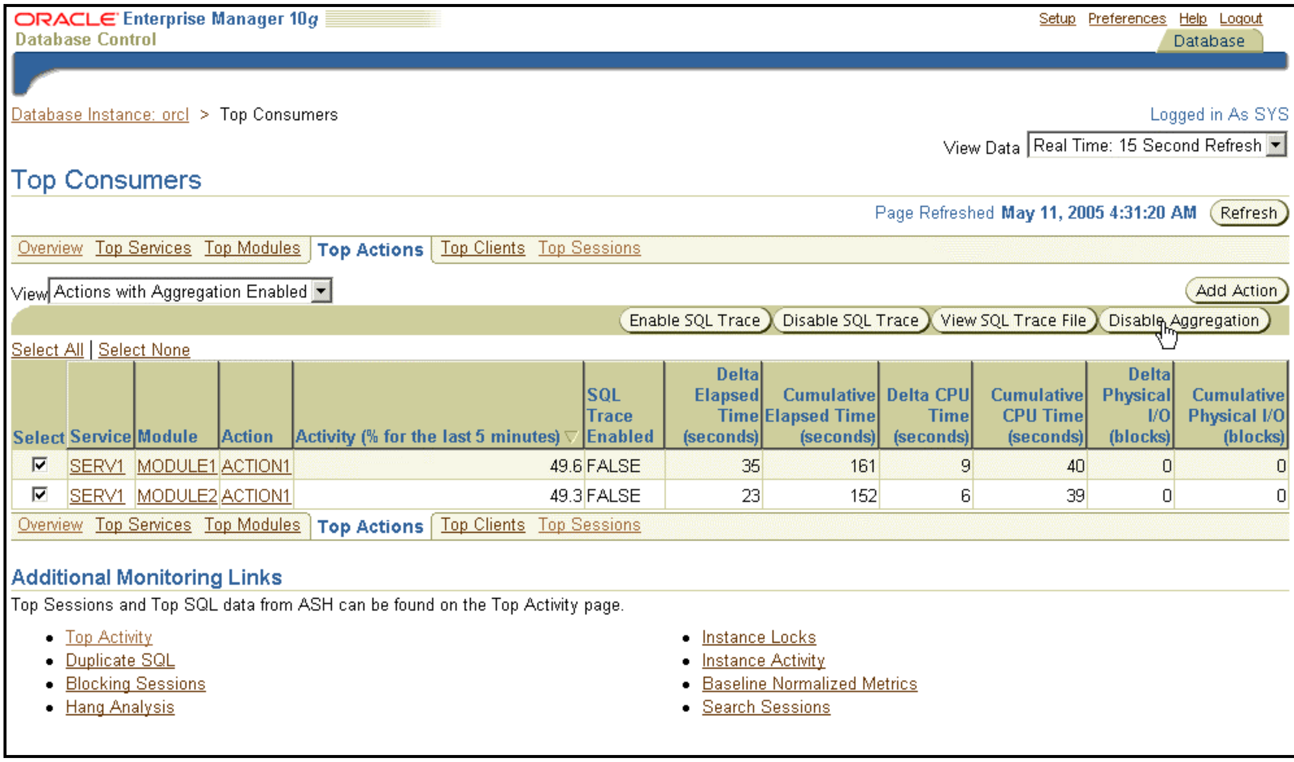


7) Using Database Control, disable statistic aggregations for both the actions.

Answer:

From the Top Actions page, select both the actions, and click the Disable Aggregation button.

Solutions for Practice 16-2: Trace Services in a Single-Instance Oracle Environment (Optional) (continued)



8) Stop the workload by using the `stop_servwork.sh` script. Use the trace session utility and `tkprof` to interpret the generated trace files for the `SERV1` service and the `MODULE1` module.

Answer:

```
$ ./stop_servwork.sh
Killing stream with pid=8540
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed


Killing stream with pid=8539
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed


Killing stream with pid=8536
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed
```

```
Killing stream with pid=8535
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed


$

$ trcsess output=serv1_module1.trc service=SERV1
module=MODULE1 $ORACLE_BASE/admin/orcl/udump/*
$ tkprof serv1_module1.trc results.trc

TKPROF: Release 10.2.0.1.0 - Production on Wed May 11 04:35:28
2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.


$


$ vi results.trc


TKPROF: Release 10.2.0.1.0 - Production on Wed May 11 04:35:28 2005

Copyright (c) 1982, 2005, Oracle.  All rights reserved.

Trace file: serv1_module1.trc
Sort options: default

****************************************************************************
count    = number of times OCI procedure was executed
cpu      = cpu time in seconds executing
elapsed  = elapsed time in seconds executing
disk     = number of physical reads of buffers from disk
query    = number of buffers gotten for consistent read
current  = number of buffers gotten in current mode (usually for update)
rows     = number of rows processed by the fetch or execute call
****************************************************************************

SELECT COUNT(*)
FROM
 DBA_OBJECTS

call     count       cpu    elapsed       disk      query    current    rows
------- ------  -------- ---------- ---------- ---------- ----------  ------
Parse        0      0.00       0.00          0          0          0       0
Execute    395      0.04       0.05          0          0          0       0
Fetch      395     44.37     182.35          0    2176845          0     395
------- ------  -------- ---------- ---------- ---------- ----------  ------
total      790     44.41     182.41          0    2176845          0     395

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 71     (recursive depth: 1)
```

Solutions for Practice 16-2: Trace Services in a Single-Instance Oracle Environment (Optional) (continued)

```
Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait  Total Waited
  ----------------------------------------   Waited  ----------  ------------
  latch: cache buffers chains                    20        0.33          4.40
  latch: library cache                           26        0.29          4.74

…
```

C

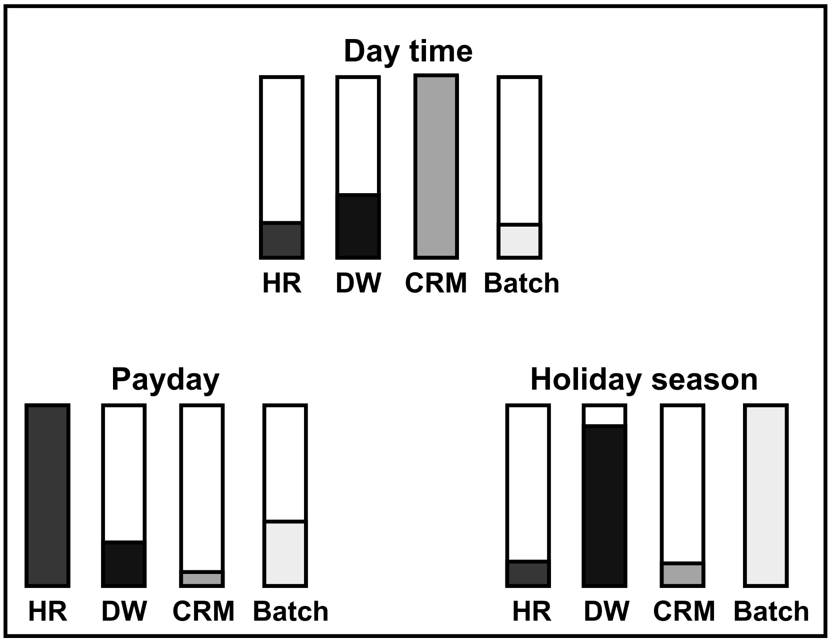
Services

Objectives

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

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

Traditional Workload Dispatching

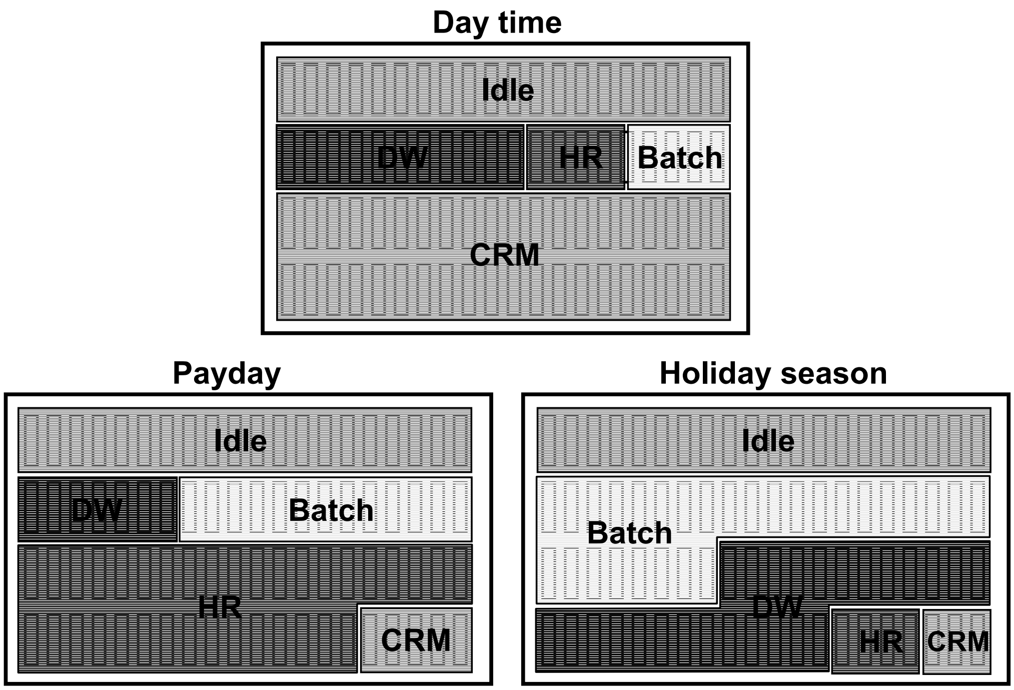


Traditional Workload Dispatching

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

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

Grid Workload Dispatching



Grid Workload Dispatching

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

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

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

What Is a Service?

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

What Is a Service?

Services organize work execution within the database to make that work more manageable, measurable, tunable, and recoverable. A service is a group of related tasks within the database with common functionality, quality expectations, and priority relative to other services. The notion of a service provides a single-system image for managing competing applications running within a single instance and across multiple instances. Using standard interfaces, such as DBCA, Enterprise Manager, and `SRVCTL`, services can be configured, administered, enabled, disabled, and measured as a single entity.

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

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

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

High Availability of Services in RAC

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

ORACLE

High Availability of Services in RAC

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

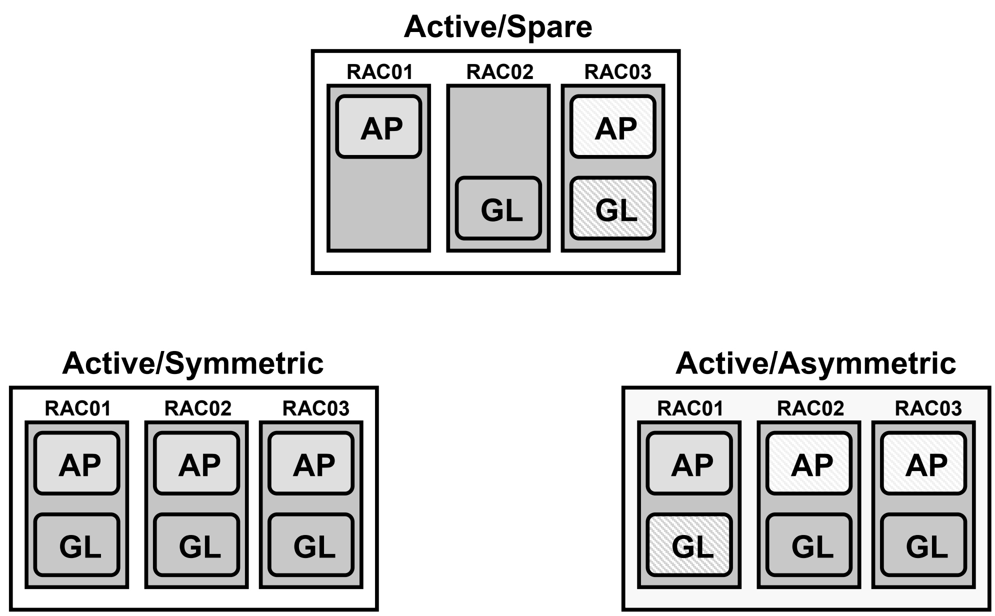
Services must be location independent and the RAC HA framework is used to implement this. Services are made available continuously with load shared across one or more instances in the cluster. Any instance can offer services in response to run-time demands, failures, and planned maintenance.

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

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

An available instance does not initially support a service. If a preferred instance fails, then the service is transparently restored to an available instance defined for the service.

Possible Service Configuration with RAC



Possible Service Configuration with RAC

Active/Spare: With this service configuration, there are N primary RAC instances providing service, and M spare RAC instances available to provide the service. An example of this solution is a three-node configuration in which one instance provides the AP service; the second instance provides the GL service; and the third instance provides service failover capability for both services. The spare node can still be available for other applications during normal operation.

Active/Symmetric: With this service configuration, the same set of services is active on every instance. An example of this is illustrated in the slide, with both AP and GL services being offered on all three instances. Each instance provides service load-sharing and service failover capabilities for the other.

Active/Asymmetric: With this service configuration, services with lower capacity needs can be defined with a single active instance and configured as having all other instances capable of providing the service in the event of failure. The slide shows the AP service running on only one instance, and the GL service running on two instances. The first instance supports the AP services and offers failover for the GL service. Likewise, the second and third instances support the GL service and offer failover for AP. If either the first or third instance dies, then GL and AP are still offered through the second instance.

Service Attributes

- Global unique name
- Network name
- Load Balancing Advisory goal
- Distributed transactions flag
- Advance queuing notification characteristics for OCI and ODP.NET clients
- Failover characteristics
- Connection load-balancing algorithm
- Threshold
- Priority
- High-availability configuration

ORACLE

Service Attributes

When you create new services for your database, you should define each service's workload management characteristics. In addition to the required service name and network name, the characteristics of a service include:

- A service goal that determines whether work requests are made to the service based on best service quality (service response time), or best throughput (how much work is completed in a unit of time), as determined by the Load Balancing Advisory.
- The session failover characteristics when using transparent application failover
- The method for load balancing connections (which you define) for each service:
  - `SHORT`: Using Load Balancing Advisory (open workloads)
  - `LONG`: Using session count by service (closed workload)
- Services metric thresholds for response time and CPU consumption.
- Mapping of services to consumer groups instead of usernames to consumer groups
- How the service is distributed across instances when the system first starts

Note: For more information, refer to the *Oracle Clusterware and Oracle Real Application Clusters Administration and Deployment* guide.

Service Types

- **Application services**
- **Internal services:**
  - `SYS$BACKGROUND`
  - `SYS$USERS`
  - **Cannot be deleted or changed**
- **Limit of 64 services per database:**
  - **62 application services**
  - **2 internal services**

Service Types

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

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

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

Note: Shadow services used by Transparent Application Failover are also included in the application services category. In addition, a service is also created for each Streams Buffered Queue.

Creating Services

- Services are maintained in the data dictionary.
- Use `DBMS_SERVICE.CREATE` to create a service for single-instance database.
- Services are created automatically based on the `SERVICE_NAMES` initialization parameter.
- Create a service in RAC with the following:
  - Database Configuration Assistant (DBCA)
  - `SRVCTL`
  - Enterprise Manager
- High-availability business rules are maintained in OCR and are managed by Oracle Clusterware.

Creating Services

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

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

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

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

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

Managing Services in a Single-Instance Environment

- Create a new service.

```
exec DBMS_SERVICE.CREATE_SERVICE('SERV1','SERV1.oracle.com');
```

- Start a service.

```
exec DBMS_SERVICE.START_SERVICE('SERV1');
```

- Stop a service.

```
exec DBMS_SERVICE.STOP_SERVICE('SERV1');
```

- Delete a service.

```
exec DBMS_SERVICE.DELETE_SERVICE('SERV1');
```

- Disconnect sessions connected under a service.

```
exec DBMS_SERVICE.DISCONNECT_SESSION('SERV1');
```

ORACLE

Managing Services in a Single-Instance Environment

- **CREATE_SERVICE:** This procedure creates a service name in the data dictionary. Services are also created in the data dictionary implicitly when you set the service in the SERVICE_NAMES parameter or by means of the ALTER SYSTEM SET SERVICE_NAMES command. When you create a service, you must specify its name, and its network's name. The network name should then be used in the SERVICE_NAME parameter of your corresponding connect descriptor.
- **START_SERVICE:** This procedure starts a service. In a single-instance environment, this procedure alters SERVICE_NAMES to contain this service name. In RAC, implementing this option acts on the instance specified.
- **STOP_SERVICE:** This procedure stops a service. In single-instance environment, this procedure alters the SERVICE_NAMES to remove this service name. In RAC, it calls out to Oracle Clusterware to stop the service optionally on the instance specified.
- **DELETE_SERVICE:** This procedure deletes a service from the data dictionary. You must stop a service before you can delete it.

Managing Services in a Single-Instance Environment (continued)

- **DISCONNECT_SESSION:** This procedure disconnects sessions with the named service at the current instance. This subprogram does not return until all corresponding sessions are disconnected.

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

Everything Switches to Services

- Data dictionary maintains services.
- AWR measures the performance of services.
- The Database Resource Manager uses service in place of users for priorities.
- Job scheduler, parallel query (PQ), and Streams queues run under services.
- RAC keeps services available within a site.
- Data Guard Broker with RAC keeps primary services available across sites.

ORACLE

Everything Switches to Services

Several database features support services. A session is tracked by the service with which it connects. In addition, performance-related statistics and wait events are also tracked by services.

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

In Oracle Database 10*g*, the Database Resource Manager is capable of managing services for prioritizing application workloads within an instance. In addition, jobs can now run under a service, as opposed to a specific instance. Parallel slave processes inherit the service of their coordinator.

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

Using Services with Client Applications

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

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

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

ORACLE

Using Services with Client Applications

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

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

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

The first example in the slide shows the TNS connect descriptor that can be used to access the ERP service.

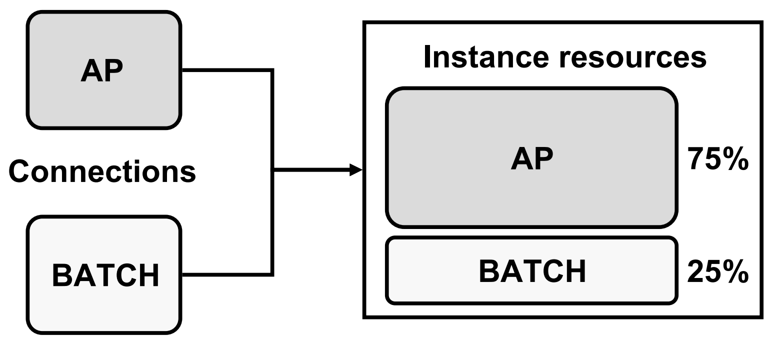
The second example shows the thick JDBC connection description using the previously defined TNS connect descriptor.

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

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

Using Services with the Resource Manager

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



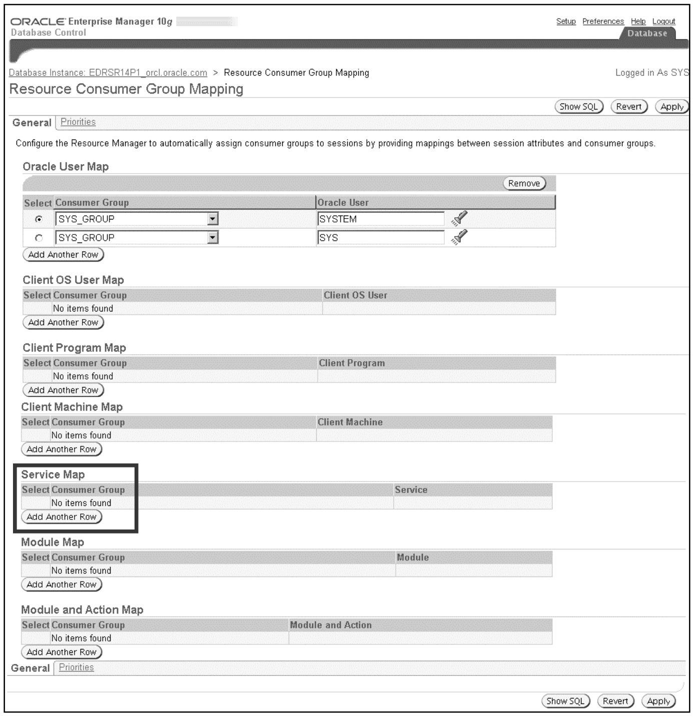
Using Services with the Resource Manager

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

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

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

Services and Resource Manager with EM



Services and Resource Manager with EM

Enterprise Manager (EM) presents a GUI interface through the Resource Consumer Group Mapping page to automatically map sessions to consumer groups. You can access this page by clicking the Resource Consumer Group Mappings link on the Administration page.

Using the General tabbed page of the Resource Consumer Group Mapping page, you can set up a mapping of sessions connecting with a service name to consumer groups. At the bottom of the page, there is an option for a module name and action mapping.

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

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

Services and the Resource Manager: Example

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

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

ORACLE

Services and the Resource Manager: Example

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

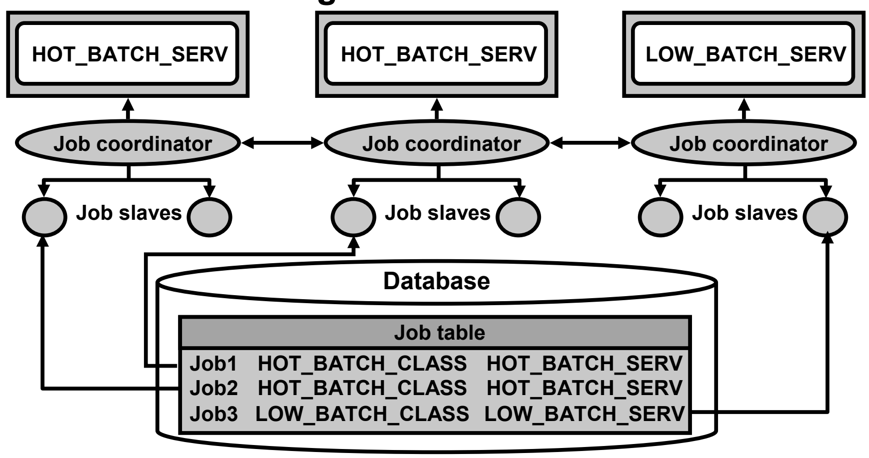
Before mapping services to consumer groups, you must first create the consumer groups and the resource plan for these consumer groups. The resource plan can be priority based or ratio based. The PL/SQL calls shown in the slide are used to create the HIGH_PRIORITY consumer group, and map the AP service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

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

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

Using Services with the Scheduler

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



ORACLE

Using Services with the Scheduler

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

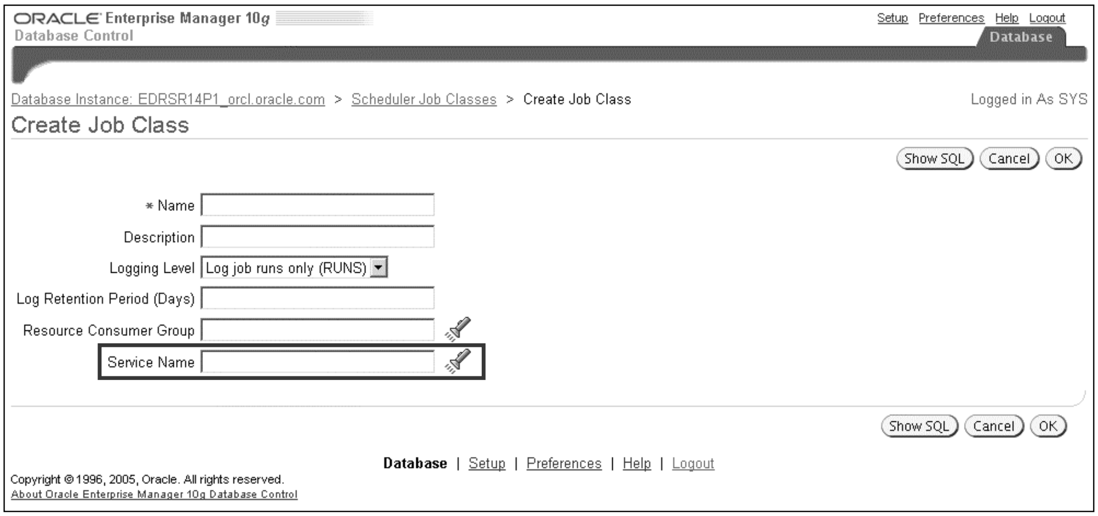
The Scheduler can use services. When you create a job class, you define the service that the job class uses. You assign jobs to job classes and job classes run within services. Using services with job classes ensures that the work of the Scheduler is identified for workload management and performance tuning.

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

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

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

Services and the Scheduler with EM

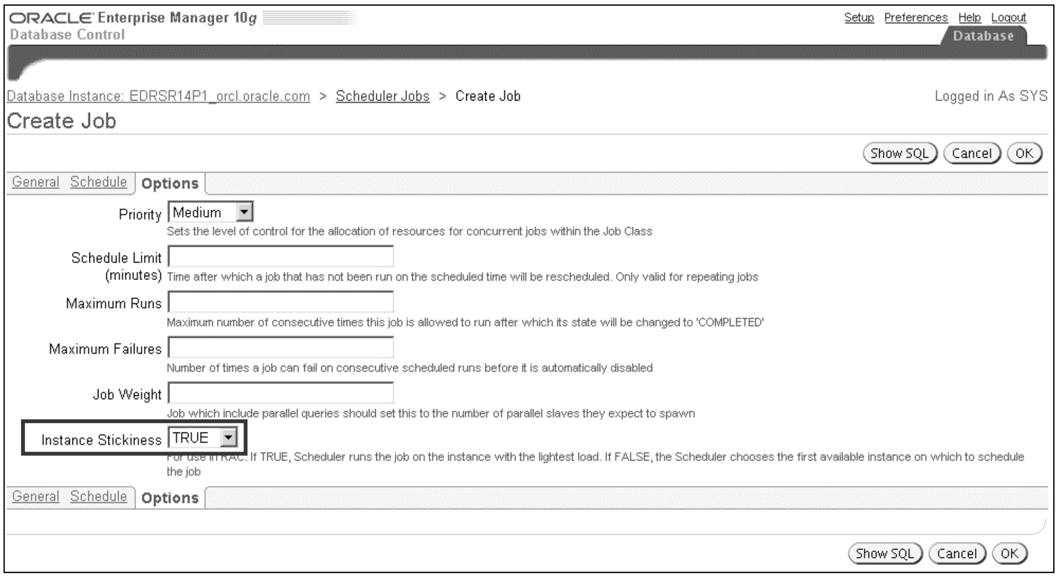


Services and the Scheduler with EM

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

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

Services and the Scheduler with EM



ORACLE

Services and the Scheduler with EM (continued)

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

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

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

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

Services and the Scheduler: Example

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

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


Services and the Scheduler: Example

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

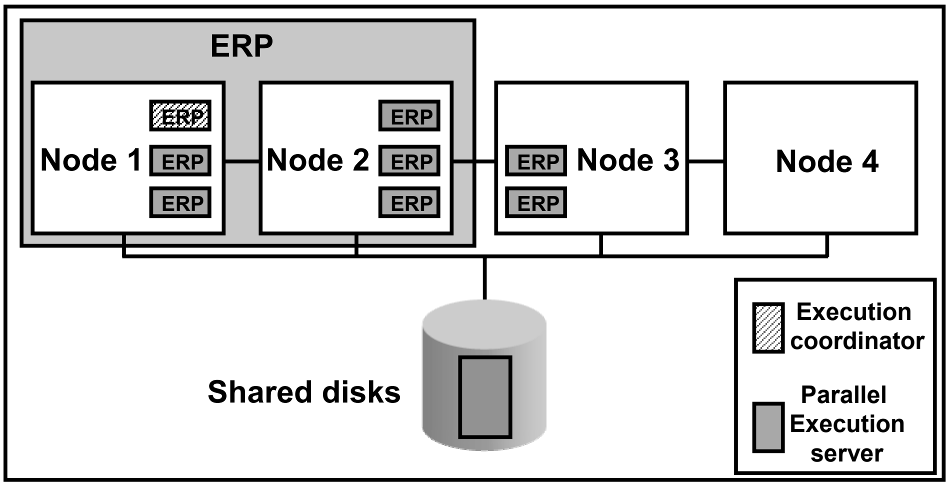
After the class is defined, you can define your job. In this example, the MY_REPORT_JOB job executes in the HOT_BATCH_CLASS job class at instances offering the HOT_BATCH_SERV service.

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

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

Using Services with Parallel Operations

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



Using Services with Parallel Operations

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

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

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

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

Using Services with Metric Thresholds

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

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

ORACLE

Using Services with Metric Thresholds

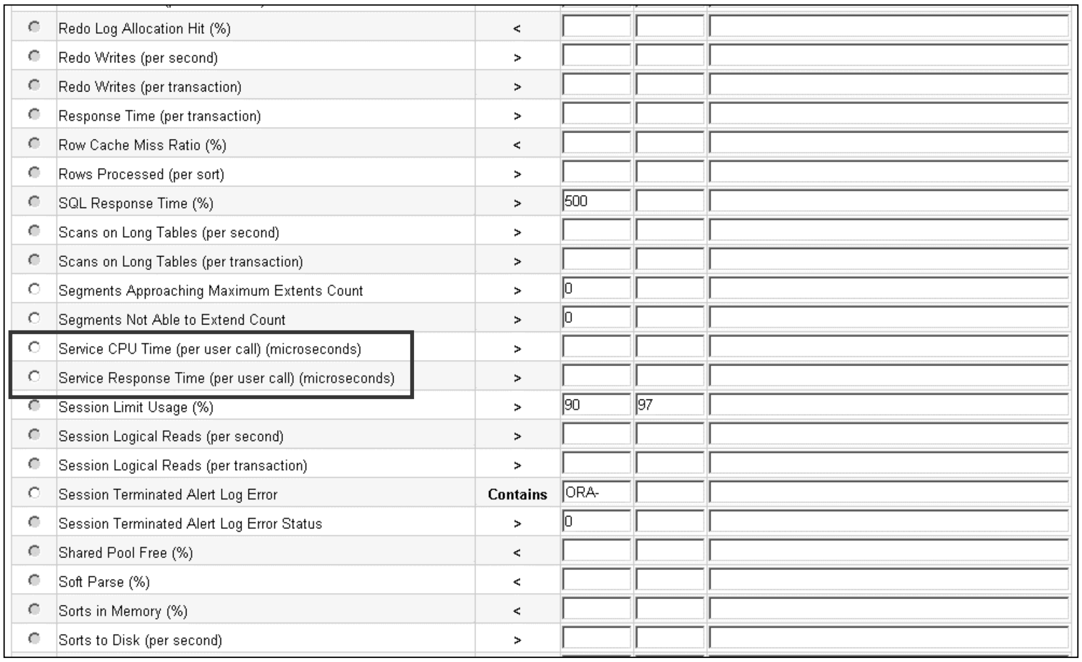
Service-level thresholds permit the comparison of actual service levels against the accepted minimum required level. This provides accountability with respect to delivery or failure to deliver an agreed service level. You can explicitly specify two metric thresholds for each service on a particular instance:

- **The response time for calls:** `ELAPSED_TIME_PER_CALL`. The alert on this threshold is activated when the elapsed time (wall-clock time) exceeds the threshold value. This is a fundamental measure that reflects all delays and faults the call experiences.
- **CPU time for calls:** `CPU_TIME_PER_CALL`

AWR monitors the service time and publishes AWR alerts when the performance exceeds the thresholds. You can respond to these alerts by changing the priority of a job, stopping overloaded processes, or relocating, expanding, shrinking, starting, or stopping a service. You can use automated tasks to respond to these alerts. These alerts enable you to maintain service quality despite changes in demand.

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

Changing Service Thresholds by Using EM



Redo Log Allocation Hit (%)	<			
Redo Writes (per second)	>			
Redo Writes (per transaction)	>			
Response Time (per transaction)	>			
Row Cache Miss Ratio (%)	<			
Rows Processed (per sort)	>			
SQL Response Time (%)	>	500		
Scans on Long Tables (per second)	>			
Scans on Long Tables (per transaction)	>			
Segments Approaching Maximum Extents Count	>	0		
Segments Not Able to Extend Count	>	0		
Service CPU Time (per user call) (microseconds)	>			
Service Response Time (per user call) (microseconds)	>			
Session Limit Usage (%)	>	90	97	
Session Logical Reads (per second)	>			
Session Logical Reads (per transaction)	>			
Session Terminated Alert Log Error	Contains	ORA-		
Session Terminated Alert Log Error Status	>	0		
Shared Pool Free (%)	<			
Soft Parse (%)	<			
Sorts in Memory (%)	<			
Sorts to Disk (per second)	>			

Changing Service Thresholds by Using EM

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

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

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

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

Services and Metric Thresholds: Example

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

Thresholds must be set on each instance supporting the service.

ORACLE

Services and Metric Thresholds: Example

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

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

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

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

Service Aggregation and Tracing

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

ORACLE

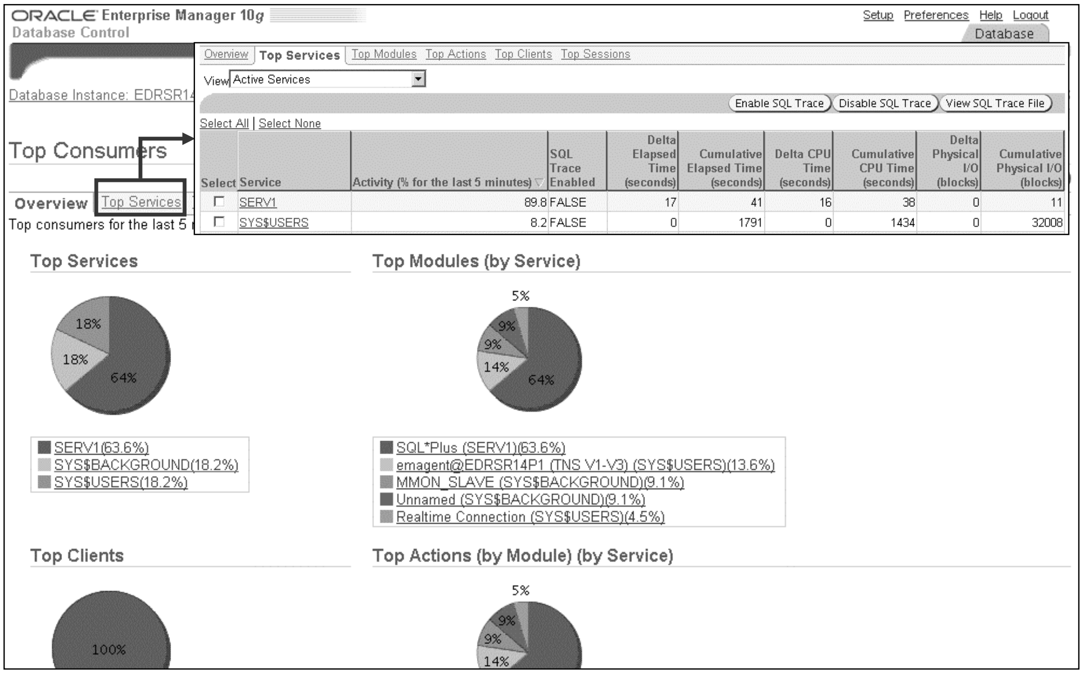
Service Aggregation and Tracing

Important statistics and wait events are collected for the work attributed to every service by default. An application can qualify a service by MODULE and ACTION names to identify the important transactions within the service. This enables you to locate exactly the poorly performing transactions. In systems where the sessions are shared, accountability is difficult. For example, in systems where connection pools or transaction processing monitors are used, the sessions are shared.

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

Workload aggregation also enables tracing by service. The traditional method of tracing each session produces trace files with SQL commands that can span workloads. This results in a hit-or-miss approach to diagnose problematic SQL. You can produce a single output trace file that contains SQL that is relevant to a specific workload by using the SERVICE_NAME, MODULE, or ACTION criteria.

Top Services Performance Page



ORACLE

Top Services Performance Page

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

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

The Top Services tabbed page displays performance-related information for the services that are defined in your database. Using this page, you can enable or disable tracing at the service level, as well as view the resulting SQL trace file.

Service Aggregation Configuration

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

ORACLE

Service Aggregation Configuration

On each instance, important statistics and wait events are automatically aggregated and collected by service. To get this level of aggregation, you only connect with different connect strings using the services you want to connect to. You can achieve a finer level of granularity of statistics collection for services by using `SERV_MOD_ACT_STAT_ENABLE` procedure in the `DBMS_MONITOR` package. This procedure enables statistics gathering for additional hierarchical combinations of `SERVICE_NAME/MODULE` and `SERVICE_NAME/MODULE/ACTION`. The `SERV_MOD_ACT_STAT_DISABLE` procedure stops the statistics gathering that was turned on. The enabling and disabling of statistics aggregation within the service applies to every instance accessing the database. Furthermore, these settings are persistent across instance restarts.

The `SERV_MOD_ACT_TRACE_ENABLE` procedure enables tracing for services with three hierarchical possibilities: `SERVICE_NAME`, `SERVICE_NAME/MODULE`, and `SERVICE_NAME/MODULE/ACTION`. The default is to trace for all instances that access the database. A parameter is provided that restricts tracing to specified instances where poor performance is known to exist. This procedure also gives you the option of capturing relevant waits and bind variable values in the generated trace files.

Service Aggregation Configuration (continued)

SERV_MOD_ACT_TRACE_DISABLE disables the tracing at all enabled instances for a given combination of service, module, and action. Like the statistics gathering mentioned previously, service tracing persists across instance restarts.

Service Aggregation: Example

- **Collect statistics on service and module:**

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

- **Collect statistics on service, module, and action:**

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

- **Trace all sessions of an entire service:**

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

- **Trace on service, module, and action:**

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

ORACLE

Service Aggregation: Example

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

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

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

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

Client Identifier Aggregation and Tracing

- Collect statistics on client identifier:

```
exec DBMS_MONITOR.CLIENT_ID_STAT_ENABLE('HR.HR');
```

- View collected data:

```
SELECT * FROM V$CLIENT_STATS;
```

- Disable statistics collection:

```
exec DBMS_MONITOR.CLIENT_ID_STAT_DISABLE('HR.HR');
```

- Trace client identifiers:

```
exec DBMS_MONITOR.CLIENT_ID_TRACE_ENABLE(-
client_id => 'HR.HR',waits => TRUE, binds => FALSE);
```

ORACLE

Client Identifier Aggregation and Tracing

It is also possible to use the DBMS_MONITOR package to enable and disable statistics aggregation for a particular client identifier. The client identifier is set using the DBMS_SESSION.SET_IDENTIFIER procedure, and is visible through the CLIENT_IDENTIFIER column from V$SESSION.

V$CLIENT_STATS displays the resulting measures for all sessions that are active for the client identifier per instance. Similar to the aggregated statistics available for service aggregation, the statistics published in V$CLIENT_STATS are a subset of those available in V$SESSTAT and V$SESS_TIME_MODEL.

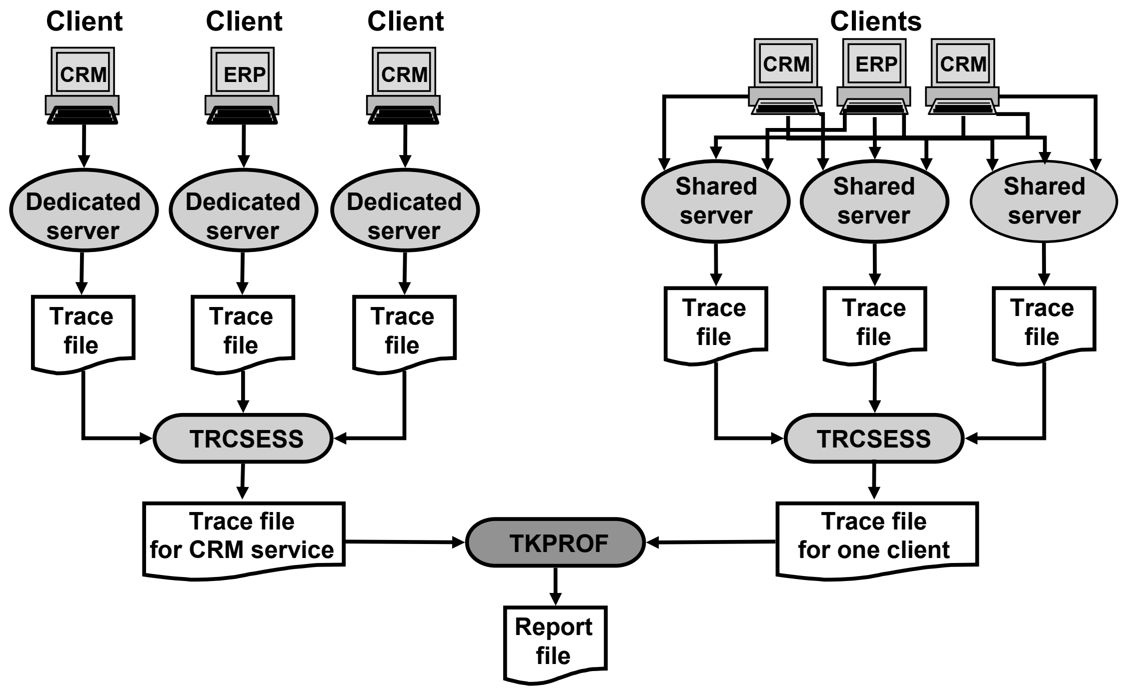
Using the CLIENT_ID_STAT_DISABLE procedure, you can disable accumulation of wait model statistics for the specified client identifier.

The CLIENT_ID_TRACE_ENABLE procedure enables tracing globally for the database for a given client identifier. As you can see from the example, you can also request to dump wait events and bind variables to the trace files.

Use the CLIENT_ID_TRACE_DISABLE procedure to disable the generation of the trace files for a specified client identifier.

Note: You can use the Top Clients page accessible from the Top Consumers page of Enterprise Manager to graphically do the same thing as with the PL/SQL interface.

trcsess Utility



trcsess Utility

The trcsess utility consolidates trace output from selected trace files based on several criteria: Session ID, client ID, service name, action name, and module name. After trcsess merges the trace information into a single output file, the output file can be processed by tkprof. When using the DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE procedure, tracing information is present in multiple trace files and you must use the trcsess tool to collect it into a single file. The trcsess utility is useful for consolidating the tracing of a particular session or service for performance or debugging purposes.

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

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

Service Performance Views

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

Service Performance Views

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

The call times and performance statistics are visible in V$SERVICE_STATS, V$SERVICE_EVENT, V$SERVICE_WAIT_CLASS, V$SERVICEMETRIC, and V$SERVICEMETRIC_HISTORY. These views show the statistics and metrics gathered at the service level. Examples are:

```
SQL> SELECT service_name, stat_name, value
  2  FROM V$SERVICE_STATS
  3  WHERE service_name = 'SERV1'

SERVICE_NAME STAT_NAME                                VALUE
------------ ----------------------------------- ----------
SERV1        user calls                                  37
SERV1        DB time                              225407870
SERV1        DB CPU                               216824843
SERV1        parse count (total)                        134
SERV1        parse time elapsed                      975732
…
```

Service Performance Views (continued)

```
SQL> SELECT service_name, elapsedpercall, cpupercall,
  2        dbtimepercall, dbtimepersec
  3  FROM V$SERVICEMETRIC
  4* WHERE service_name = 'SERV1';
```

```
SERVICE_NAME ELAPSEDPERCALL CPUPERCALL DBTIMEPERCALL DBTIMEPERSEC
------------ -------------- ---------- ------------- ------------
SERV1                     0          0             0            0
SERV1              58593980   55399177      58593980   97.6078294
```

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

Of the over 300 performance-related statistics that are tracked and visible in V$SYSSTAT, 28 statistics are tracked for services. To see the statistics measured for services, run the following query:

```
SELECT DISTINCT stat_name FROM V$SERVICE_STATS
```

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

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

Generalized Trace Enabling

- **For all sessions in the database:**

```
EXEC DBMS_MONITOR.DATABASE_TRACE_ENABLE(TRUE,TRUE);
```

```
EXEC DBMS_MONITOR.DATABASE_TRACE_DISABLE();
```

- **For a particular session:**

```
EXEC DBMS_MONITOR.SESSION_TRACE_ENABLE(session_id =>
27, serial_num => 60, waits => TRUE, binds =>
FALSE);
```

```
EXEC DBMS_MONITOR.SESSION_TRACE_DISABLE(session_id
=> 27, serial_num => 60);
```


Generalized Trace Enabling

You can debug performance problems with tracing. The DBMS_MONITOR package contains trace-enabling procedures. These procedures enable tracing globally for a database.

You can use the DATABASE_TRACE_ENABLE procedure to enable instancewide session-level SQL tracing. The procedure has the following parameters:

- **WAITS:** Specifies whether wait information is to be traced
- **BINDS:** Specifies whether bind information is to be traced
- **INSTANCE_NAME:** Specifies the instance for which tracing is to be enabled. Omitting INSTANCE_NAME enables the session-level tracing for the whole database.

Use the DATABASE_TRACE_DISABLE procedure to disable SQL tracing for the whole database or a specific instance.

Similarly, you can use the SESSION_TRACE_ENABLE procedure to enable tracing for a given database session identifier (SID), on the local instance. The SID and SERIAL# information can be found from V$SESSION.

Use the SESSION_TRACE_DISABLE procedure to disable the trace for a given database session identifier (SID) and serial number.

Trace Your Own Session

- **Enable trace:**

```
EXEC DBMS_SESSION.SESSION_TRACE_ENABLE(waits
=> TRUE, binds => FALSE);
```

- **Disable trace:**

```
EXEC DBMS_SESSION.SESSION_TRACE_DISABLE();
```

ORACLE

Trace Your Own Session

While the DBMS_MONITOR package can be invoked only by a user with the DBA role, any user can enable SQL tracing for his or her own session by using the DBMS_SESSION package. The SESSION_TRACE_ENABLE procedure can be invoked by any user to enable session-level SQL tracing for his or her own session. An example is shown in the slide.

You can then use the DBMS_SESSION.SESSION_TRACE_DISABLE procedure to stop dumping to your trace file.

Note: The SQL_TRACE initialization parameter is deprecated as of Oracle Database 10*g*. You can obtain the complete list of deprecated parameters by using the following statement:

```
SELECT name FROM v$parameter WHERE isdeprecated = 'TRUE'
```

Summary

In this lesson, you should have learned how to:

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

ORACLE

Practice 16 Overview: Using Services with a Single-Instance Oracle Database

This practice covers the following topics:

- Using services in a single-instance environment
- Tracing services in a single-instance environment

ORACLE

Index

A

A (continued)

A (continued)

B

B (continued)

B (continued)

C

D (continued)

D (continued)

E

I

L (continued)

L (continued)

M

M (continued)

N

O

R (continued)

R (continued)

S

S (continued)

S (continued)

S (continued)

S (continued)

T

T (continued)

U

U (continued)

V

V (continued)

V (continued)

V (continued)

W