
Oracle9i Performance Tuning

Student Guide . Volume 2

D11299GC10
Production 1.0
July 2001
D33520

ORACLE®

Authors

Peter Kilpatrick
Shankar Raman
Jim Womack

Technical Contributors and Reviewers

Mirza Ahmad
Harald Van Breederode
Howard Bradley
Howard Ostrow
Alexander Hunold
Joel Goodman
John Watson
Michele Cyran
Pietro Colombo
Ranbir Singh
Ruth Baylis
Sander Rekveld
Tracy Stollberg
Connie Dialeris
Wayne Stokes
Scott Gossett
Sushil Kumar
Benoit Dagerville

Copyright © Oracle Corporation, 2001. All rights reserved.

This documentation contains proprietary information of Oracle Corporation. It is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited. If this documentation is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

Restricted Rights Legend

Use, duplication or disclosure by the Government is subject to restrictions for commercial computer software and shall be deemed to be Restricted Rights software under Federal law, as set forth in subparagraph (c)(1)(ii) of DFARS 252.227-7013, Rights in Technical Data and Computer Software (October 1988).

This material or any portion of it may not be copied in any form or by any means without the express prior written permission of Oracle Corporation. Any other copying is a violation of copyright law and may result in civil and/or criminal penalties.

If this documentation is delivered to a U.S. Government Agency not within the Department of Defense, then it is delivered with "Restricted Rights," as defined in FAR 52.227-14, Rights in Data-General, including Alternate III (June 1987).

The information in this document is subject to change without notice. If you find any problems in the documentation, please report them in writing to Education Products, Oracle Corporation, 500 Oracle Parkway, Box SB-6, Redwood Shores, CA 94065. Oracle Corporation does not warrant that this document is error-free.

Oracle and all references to Oracle Products are trademarks or registered trademarks of Oracle Corporation.

All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

Contents

Preface

1 Overview of Oracle9i Performance Tuning

- Objectives 1-2
- Tuning Questions 1-3
- Tuning Phases 1-5
- Tuning Goals 1-6
- Examples of Measurable Tuning Goals 1-7
- Common Tuning Problems 1-8
- Results of Common Tuning Problems 1-9
- Proactive Tuning Considerations During Development 1-10
- Tuning Steps During Production 1-11
- Performance versus Safety Trade-Offs 1-12
- Summary 1-13

2 Diagnostic and Tuning Tools

- Objectives 2-2
- Maintenance of the Alert Log File 2-3
- Tuning Using the Alert Log File 2-4
- Background Processes Trace Files 2-5
- User Trace Files 2-6
- Views, Utilities, and Tools 2-7
- Dictionary and Special Views 2-9
- Dynamic Troubleshooting and Performance Views 2-10
- Topics for Troubleshooting and Tuning 2-11
- Collecting Systemwide Statistics 2-13
- Collecting Session-Related Statistics 2-16
- STATSPACK 2-18
- STATSPACK Output 2-20
- UTLBSTAT and UTLESTAT Utilities 2-23
- Oracle Wait Events 2-24
- The V\$EVENT_NAME View 2-25
- Statistics Event Views 2-26
- The V\$SYSTEM_EVENT View 2-27
- The V\$SESSION_WAIT View 2-28
- Performance Manager 2-31
- Overview of Oracle Expert Tuning Methodology 2-33
- DBA-Developed Tools 2-34
- Summary 2-35

3 Sizing the Shared Pool

- Objectives 3-2
- The System Global Area 3-3
- The Shared Pool 3-4
- The Library Cache 3-5
- Tuning the Library Cache 3-7
- Terminology 3-9
- Diagnostic Tools for Tuning the Library Cache 3-10
- Are Cursors Being Shared? 3-11
- Guidelines: Library Cache Reloads 3-12
- Invalidations 3-14
- Sizing the Library Cache 3-16
- Cached Execution Plans 3-17
- View to Support Cached Execution Plans 3-18
- Global Space Allocation 3-20
- Large Memory Requirements 3-22
- Tuning the Shared Pool Reserved Space 3-24
- Keeping Large Objects 3-26
- Anonymous PL/SQL Blocks 3-28
- Other Parameters Affecting the Library Cache 3-30
- The Data Dictionary Cache, Terminology, and Tuning 3-32
- Diagnostic Tools for Tuning the Data Dictionary Cache 3-33
- Measuring the Dictionary Cache Statistics 3-34
- Tuning the Data Dictionary Cache 3-36
- Guidelines: Dictionary Cache Misses 3-37
- UGA and Oracle Shared Server 3-38
- Sizing the User Global Area 3-39
- Large Pool 3-40
- Summary 3-41

4 Sizing the Buffer Cache

- Objectives 4-2
- Overview 4-3
- Buffer Cache Sizing Parameters in Oracle9i 4-5
- Dynamic SGA Feature in Oracle9i 4-6
- Unit of Allocation in the Dynamic SGA 4-7
- Granule 4-8
- Allocating Granules at Startup 4-9
- Adding Granules to Components 4-10
- Dynamic Buffer Cache Size Parameters 4-11

Example: Increasing the Size of an SGA Component	4-12
Deprecated Buffer Cache Parameters	4-13
Dynamic Buffer Cache Advisory Parameter	4-14
View to Support Buffer Cache Advisory	4-15
Using V\$DB_CACHE_ADVICE	4-16
Managing the Database Buffer Cache	4-17
Tuning Goals and Techniques	4-20
Diagnostic Tools	4-22
Measuring the Cache Hit Ratio	4-23
Guidelines for Using the Cache Hit Ratio	4-24
Guidelines to Increase the Cache Size	4-25
Using Multiple Buffer Pools	4-27
Defining Multiple Buffer Pools	4-28
Enabling Multiple Buffer Pools	4-30
KEEP Buffer Pool Guidelines	4-31
RECYCLE Buffer Pool Guidelines	4-32
Calculating the Hit Ratio for Multiple Pools	4-35
Identifying Candidate Pool Segments	4-36
Dictionary Views with Buffer Pools	4-37
Caching Tables	4-38
Other Cache Performance Indicators	4-39
Free Lists	4-41
Diagnosing Free List Contention	4-42
Resolving Free List Contention	4-44
Auto-management of Free Space	4-45
Summary	4-46

5 Sizing Other SGA Structures

Objectives	5-2
The Redo Log Buffer	5-3
Sizing the Redo Log Buffer	5-4
Diagnosing Redo Log Buffer Inefficiency	5-5
Using Dynamic Views to Analyze Redo Log Buffer Efficiency	5-6
Redo Log Buffer Tuning Guidelines	5-8
Reducing Redo Operations	5-10
Monitoring Java Pool Memory	5-12
Sizing the SGA for Java	5-13
Multiple I/O Slaves	5-14
Multiple DBWR Processes	5-15
Tuning DBWn I/O	5-16
Summary	5-17

6 Database Configuration and I/O Issues

- Objectives 6-2
- Oracle Processes and Files 6-3
- Performance Guidelines 6-4
- Distributing Files across Devices 6-5
- Tablespace Usage 6-6
- Diagnostic Tools for Checking I/O Statistics 6-7
- I/O Statistics 6-9
- File Striping 6-10
- Tuning Full Table Scan Operations 6-12
- Checkpoints 6-15
- Diagnostic Tools for Tuning Checkpoint Performance 6-16
- Guidelines 6-17
- Defining and Monitoring FASTSTART Checkpointing 6-19
- Redo Log Groups and Members 6-20
- Online Redo Log File Configuration 6-21
- Archive Log File Configuration 6-23
- Diagnostic Tools 6-24
- Summary 6-25

7 Optimizing Sort Operations

- Objectives 7-2
- The Sorting Process 7-3
- Sort Area and Parameters 7-4
- New Sort Area Parameters 7-6
- Sort Area and Parameters 7-7
- Tuning Sorts 7-9
- The Sorting Process and Temporary Space 7-10
- Temporary Space Segments 7-11
- Operations Requiring Sorts 7-12
- Avoiding Sorts 7-14
- Diagnostic Tools 7-16
- Diagnostics and Guidelines 7-18
- Monitoring Temporary Tablespaces 7-19
- Temporary Tablespace Configuration 7-20
- Summary 7-22

8 Diagnosing Contention for Latches

- Objectives 8-2
- Latches: Overview 8-3
- Purpose of Latches 8-4
- Waiting for a Latch 8-5
- Latch Request Types 8-6
- Latch Contention 8-7
- Reducing Contention for Latches 8-9
- Important Latches for the DBA 8-10
- Shared Pool and Library Cache Latches 8-12
- Summary 8-13

9 Tuning Rollback Segments

- Objectives 9-2
- Rollback Segments: Usage 9-3
- Rollback Segment Activity 9-4
- Rollback Segment Header Activity 9-5
- Growth of Rollback Segments 9-6
- Tuning the Manually Managed Rollback Segments 9-7
- Diagnostic Tools 9-8
- Diagnosing Contention for Manual Rollback Segment Header 9-9
- Guidelines: Number of Manual Rollback Segments (RBSs) 9-11
- Guidelines: Sizing Manual Rollback Segments 9-13
- Sizing Transaction Rollback Data 9-14
- Using Less Rollback 9-18
- Possible Problems Caused by Small Rollback Segments 9-20
- Automatic Undo Management in Oracle9i 9-21
- Tablespace for Automatic Undo Management 9-22
- Altering an Undo Tablespace 9-23
- Switching Undo Tablespaces 9-24
- Dropping an Undo Tablespace 9-25
- Parameters for Automatic Undo Management 9-26
- Monitoring Automatic Undo Management 9-28
- Summary 9-29

10 Monitoring and Detecting Lock Contention

- Objectives 10-2
- Locking Mechanism 10-3
- Two Types of Locks 10-6
- DML Locks 10-8
- Table Lock Modes 10-10
- DML Locks in Blocks 10-14
- DDL Locks 10-15

Possible Causes of Lock Contention 10-17
Diagnostics Tools for Monitoring Locking Activity 10-18
Guidelines for Resolving Contention 10-20
Deadlocks 10-22
Summary 10-25

11 Tuning the Oracle Shared Server

Objectives 11-2
Overview 11-3
Oracle Shared Server Characteristics 11-4
Monitoring Dispatchers 11-5
Monitoring Shared Servers 11-7
Monitoring Process Usage 11-9
Shared Servers and Memory Usage 11-10
Troubleshooting 11-11
Obtaining Dictionary Information 11-12
Summary 11-13

12 Application Tuning

Objectives 12-2
The Role of the Database Administrator 12-3
Data Storage Structures 12-4
Selecting the Physical Structure 12-5
Data Access Methods 12-7
Clusters 12-8
Cluster Types 12-9
Situations Where Clusters Are Useful 12-10
B-Tree Indexes 12-11
Compressed Indexes 12-13
Bitmap Indexes 12-14
Creating and Maintaining Bitmap Indexes 12-16
B-Tree Indexes and Bitmap Indexes 12-17
Reverse Key Index 12-18
Creating Reverse Key Indexes 12-19
Index-Organized Tables (IOTs) 12-20
Index-Organized Tables and Heap Tables 12-21
Creating Index-Organized Tables 12-22
IOT Row Overflow 12-23
IOT Dictionary Views 12-24
Using a Mapping Table 12-25
Maintaining a Mapping Table 12-26
Materialized Views 12-27
Creating Materialized Views 12-28
Materialized Views: Manual Refreshing 12-30
Query Rewrites 12-31

- Materialized Views and Query Rewrites: Example 12-33
- Enabling and Controlling Query Rewrites 12-35
- Disabling Query Rewrites: Example 12-37
- OLTP Systems 12-38
- OLTP Requirements 12-39
- OLTP Application Issues 12-41
- Decision Support Systems (Data Warehouses) 12-42
- Data Warehouse Requirements 12-43
- Data Warehouse Application Issues 12-45
- Hybrid Systems 12-46
- Summary 12-47

13 Using Oracle Blocks Efficiently

- Objectives 13-2
- Database Storage Hierarchy 13-3
- Allocation of Extents 13-4
- Avoiding Dynamic Allocation 13-5
- Locally Managed Extents 13-6
- Pros and Cons of Large Extents 13-7
- The High-Water Mark 13-9
- Table Statistics 13-11
- The DBMS_SPACE Package 13-12
- Recovering Space 13-15
- Database Block Size 13-16
- The DB_BLOCK_SIZE Parameter 13-17
- Small Block Size: Pros and Cons 13-18
- Large Block Size: Pros and Cons 13-19
- PCTFREE and PCTUSED 13-20
- Guidelines for PCTFREE and PCTUSED 13-22
- Migration and Chaining 13-23
- Detecting Migration and Chaining 13-24
- Selecting Migrated Rows 13-25
- Eliminating Migrated Rows 13-26
- Index Reorganization 13-28
- Monitoring Index Space 13-29
- Deciding Whether to Rebuild or Coalesce an Index 13-30
- Monitoring Index Usage 13-32
- Identifying Unused Indexes 13-33
- Summary 13-34

14 SQL Statement Tuning

- Objectives 14-2
- Optimizer Modes 14-3
- Setting the Optimizer Mode 14-4
- Optimizer Plan Stability 14-6
- Plan Equivalence 14-7
- Creating Stored Outlines 14-8
- Using Stored Outlines 14-9
- Editing Stored Outlines 14-11
- Maintaining Stored Outlines 14-13
- Using Hints in a SQL Statement 14-14
- Overview of Diagnostic Tools 14-15
- SQL Reports in STATSPACK 14-16
- Explain Plan 14-17
- Using SQL Trace and TKPROF 14-18
- Enabling and Disabling SQL Trace 14-20
- Formatting the Trace File with TKPROF 14-21
- TKPROF Statistics 14-23
- SQL*Plus AUTOTRACE 14-24
- Managing Statistics 14-25
- Table Statistics 14-27
- Index Statistics 14-28
- Column Statistics 14-29
- Histograms 14-30
- Generating Histogram Statistics 14-31
- Copying Statistics Between Databases 14-33
- Example: Copying Statistics 14-34
- Summary 14-37

15 Tuning the Operating System and Using Resource Manager

- Objectives 15-2
- Operating System Tuning 15-3
- System Architectures 15-4
- Virtual and Physical Memory 15-5
- Paging and Swapping 15-6
- Tuning Memory 15-7
- Tuning I/O 15-8
- Understanding Different I/O System Calls 15-9
- CPU Tuning 15-10
- Process versus Thread 15-11
- Overview of Database Resource Manager 15-12
- Database Resource Management Concepts 15-13

- Resource Allocation Methods 15-14
- The Original Plan: SYSTEM_PLAN 15-16
- Using Subplans 15-17
- Administering the Database Resource Manager 15-18
- Assigning the Resource Manager Privilege 15-20
- Creating Database Resource Manager Objects 15-21
- Assigning Users to Consumer Groups 15-25
- Setting the Resource Plan for an Instance 15-26
- Changing a Consumer Group Within a Session 15-27
- Changing Consumer Groups for Sessions 15-28
- Database Resource Manager Information 15-29
- Current Database Resource Manager Settings 15-32
- Guidelines 15-33
- Summary 15-34

16 Workshop Overview

- Objectives 16-2
- Approach to Workshop 16-3
- Company Information 16-4
- Workshop Database Configuration 16-6
- Workshop Procedure 16-7
- Choosing a Scenario 16-8
- Workshop Scenarios 16-9
- Collecting Information 16-10
- Generating a Workshop Load 16-11
- Results 16-12
- Summary 16-13

A Practices Solutions

B Tuning Workshop

C Example of STATSPACK Repot

D Redundant Arrays of Inexpensive Disks Technology (RAID)



Practice Solutions

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Practice 2

The goal of this practice is to familiarize yourself with the different methods of collecting statistical information.

1. Log on as directed by the instructor. If the database is not already started, connect to sqlplus using “system/manager as sysdba” then start it using the STARTUP command. Check that TIMED_STATISTICS has been set to true, if it has not then set it using the init.ora file (located at \$HOME/ADMIN/PFILE), or the alter system statement.
 - SQL> show parameter timed_statistics
 - If a value of true is return, continue to question 2. If a value of false is returned, then either edit the init.ora and add the parameter TIMED_STATISTICS = TRUE
 - Or, set the parameter dynamically using
 - SQL> alter system set timed_statistics = true;
2. Connect to SQLPLUS as user SYSTEM, and issue a command that will create a trace file for this session. Run a query to count the number of rows in the dictionary view DBA_TABLES. In order to locate your new trace file easier, if possible, delete all the trace files in the USER_DUMP_DEST before running the trace. Disable the trace command after running the query. Do not try to interpret the content of the trace file as this is the topic of a later lesson.
 - SQL> ALTER SESSION SET SQL_TRACE = TRUE;
 - SQL> SELECT COUNT(*) FROM DBA_TABLES;
 - SQL> ALTER SESSION SET SQL_TRACE = FALSE;
3. At the operating system level view the resulting trace file located in the directory set by USER_DUMP_DEST
 - \$cd \$HOME/ADMIN/UDUMP
 - \$ls -l
 - -rw-r----- 1 user487 dba 4444 Apr 24 22:28 U487_ora_3270.trc
4. Open two sessions, the first as user HR/HR, and the second as user “sys/oracle as sysdba”. From the second session generate a user trace file for the first session using DBMS_SYSTEM.SET_SQL_TRACE_IN_SESSION procedure.
 - From Session 1
 - \$ sqlplus hr/hr
 - Change to Session 2
 - \$ sqlplus sys/oracle as sysdba
 - SQL> select username, sid, serial# from v\$session where username = ‘HR’;
 - SQL> begin
 - 2> dbms_system.set_sql_trace_in_session
 - 3> (&SID,&SERIALNUM,TRUE);
 - 4> end;
 - 5> /

Practice 2 (continued)

5. Change to Session 1

```
SQL> select * from employees;
```

6. Change to Session 2

```
SQL> begin
```

```
2> dbms_system.set_sql_trace_in_session
```

```
3> (&SID,&SERIALNUM,FALSE);
```

```
4> end;
```

```
5> /
```

7. Confirm that the trace file has been created in the directory set by
USER_DUMP_DEST

```
$cd $HOME/ADMIN/UDUMP
```

```
$ls -l
```

```
-rw-r----- 1 dba01 dba 4444 Apr 24 22:28 dba01_ora_3270.trc
```

```
-rw-r----- 1 dba01 dba 15443 Apr 24 22:42 dba01_ora_3281.trc
```

8. Connect to SQLPLUS using “sys/oracle as sysdba”, and create a new tablespace (TOOLS) to hold the tables and other segments required by STATSPACK. This tablespace needs to be 100M, and be dictionary managed (this will be used later in the course). Name the datafile tools01.dbf and place in the \$HOME/ORADATA/u05 directory.

Note: Dictionary managed is not the default.

```
SQL> connect sys/oracle as sysdba
```

```
SQL> create tablespace TOOLS
```

```
2> datafile '$HOME/ORADATA/u05/tools01.dbf' size 100m
```

```
3> extent management dictionary;
```

9. Confirm, and record, the amount of free space available within the TOOLS tablespace by querying the DBA_FREE_SPACE view.

```
SQL> select tablespace_name, sum (bytes)
```

```
2> from dba_free_space
```

```
3> where tablespace_name = 'TOOLS'
```

```
4> group by tablespace_name;
```

10. Connect using “sys/oracle as sysdba”, then install STATSPACK using the spcreate.sql script located in your \$HOME/STUDENT/LABS directory. Use the following settings when asked by the installation program:

User's Default Tablespace = TOOLS

User's Temporary Tablespace = TEMP

```
SQL > @$HOME/STUDENT/LABS/spcreate.sql
```

Practice 2 (continued)

11. Query DBA_FREE_SPACE to determine the amount of free space left in the TOOLS tablespace. The difference between this value and the one records in question (7) will be the space required for the initial installation of STATSPACK. Note that this amount will increase in proportion to the amount of information stored within the STATSPACK tables, i.e., the number of snapshots.

Subtract the value received now, from the value received in Step 7 to get the amount of space required in order to install STATSPACK

12. Manually collect current statistics using STATSPACK by running the snap.sql script located in \$HOME/STUDENT/LABS. This will return the snap_id for the snapshot just taken, which should be recorded.

```
SQL > @$HOME/STUDENT/LABS/snap.sql
```

13. In order to have STATSPACK automatically collect statistics every 3 minutes execute the spauto.sql script located in your \$HOME/STUDENT/LABS directory. Query the database to confirm that the job has been registered using the user_jobs view..

```
SQL > @$HOME/STUDENT/LABS/spauto.sql
```

```
SQL > select job, next_date, next_sec, last_sec
```

```
2 > from user_jobs;
```

14. After waiting for a period in excess of 3 minutes query the stats\$snapshot view in order to list what snapshots have been collected. There must be at least two snapshots before moving to the next step.

```
SQL> select snap_id, to_char(startup_time,'dd Mon "at" HH24:mi:ss') instart_fm,
```

```
2> to_char(snap_time,'dd Mon YYYY HH24:mi') snapdat, snap_level "level"
```

```
3> from stats$snapshot
```

```
4> order by snap_id;
```

15. Once there are at least two snapshots, start to generate a report. This is performed using the spreport.sql script found in the \$HOME/STUDENT/LABS directory. The script lists the snapshot options available, and then requests the beginning snap id and the end snap id. The user is then requested to give a filename for the report. It is often best left to the default.

```
SQL> @$HOME/STUDENT/LABS/spreport.sql
```

16. Locate the report file in the user's current directory, then using any text editor, open and examine the report. The first page shows a collection of the most queried statistics.

```
$ vi sp_X_Y.lst
```

where X is the starting snapshot, and Y is the ending snapshot (this is true if the default report filename was used).

Practice 2 (continued)

Dynamic Performance Views

17. Connect to the database as a system administrator “sys/oracle as sysdba”.

```
SQL> Connect sys/oracle as sysdba
```

18. Query the database in order to determine what system wait events have been registered since startup using v\$system_event.

```
SQL> select event, total_waits, time_waited  
2> from v$system_event;
```

19. Determine if there are any sessions actually waiting for resources, using v\$session_wait.

```
SQL> select sid, event, p1text, wait_time, state  
2 > from v$session_wait;
```

20. Stop the automatic collection of statistics by removing the job. This is performed by connecting as user perfstat/perfstat and querying the view user_jobs to get the job number. Then execute DBMS_JOB.REMOVE (<job number>);

```
SQL> connect perfstat/perfstat
```

```
SQL> select job, log_user
```

```
2> from user_jobs;
```

```
SQL> execute dbms_job.remove ($job);
```

Practice 3

The objective of this practice is to use diagnostic tools to monitor and tune the shared pool.

1. Connect using 'sys/oracle as sysdba' and check the size of the shared pool.

```
SQL> show parameter shared_pool
NAME                                TYPE                                VALUE
-----                                -
shared_pool_reserved_size          big integer 2516582
shared_pool_size                    big integer 50331648
```

2. Connect as perfstat/perfstat user, execute the \$HOME/STUDENT/LABS/snap.sql script to collect initial snapshot of statistics, and note the snapshot number by

```
SQL> connect perfstat/perfstat
SQL> @$HOME/STUDENT/LABS/snap.sql
```

3. To simulate user activity against the database open two operating system sessions. In session 1 connect as hr/hr and run the \$HOME/STUDENT/LABS/lab03_03_1.sql script. In the second session connect as hr/hr and run the script \$HOME/STUDENT/LABS/lab03_03_2.sql.

In session 1

```
SQL> connect hr/hr
SQL> @$HOME/STUDENT/LABS/lab03_03_1.sql
```

In session 2

```
SQL> connect hr/hr
SQL> @$HOME/STUDENT/LABS/lab03_03_2.sql
```

4. Connect as "system/manager" and measure the pin-to-reload ratio for the library cache by querying v\$librarycache. Determine if it is a good ratio or not.

Using the dynamic view:

```
SQL> connect system/manager
SQL> select sum(pins), sum(reloads),
2> sum(pins) * 100 / sum(pins+reloads) "Pin Hit%"
3> from v$librarycache;
```

Practice 3 (continued)

5. Connect as “system/manager” and measure the get-hit ratio for the data dictionary cache by querying v\$rowcache. Determine if it is a good ratio or not using the dynamic view.

```
SQL> connect system/manager
SQL> SELECT SUM(getmisses), SUM(gets),
2      SUM(getmisses )*100/SUM(gets) "MISS %"
3      FROM v$rowcache;
```

If GETMISSES are lower than 15% of the GETS, it is a good ratio.

6. Connect as perfstat/perfstat and run the script \$HOME/STUDENT/LABS/snap.sql to collect a statistic snap shot and obtain the snapshot number. Record this number.

```
SQL> connect perfstat/perfstat
SQL> @$HOME/STUDENT/LABS/snap.sql
```

7. As user perfstat/perfstat obtain the statistics report between the two recorded snapshot_ids (from questions 2 and 6) by running the script \$HOME/STUDENT/LABS/spreport.sql.

```
SQL> connect perfstat/perfstat
SQL> @$HOME/STUDENT/LABS/spreport.sql
```

The following is an example of using a beginning snapshot_id of 3, and an ending snapshot_id of 5.

Specify the Begin and End Snapshot Ids

~~~~~

Enter value for begin\_snap: 3

Enter value for end\_snap: 5

End Snapshot Id specified: 5

Specify the Report Name

~~~~~

The default report file name is sp_3_5. To use this name, press <return> to continue, otherwise enter an alternative.

Enter value for report_name:

Practice 3 (continued)

8. Analyze the generated report in the current directory (named sp_3_5.lst in the previous example). What would you consider to address if the library hit ratio (found under the heading “Instance Efficiency Percentages) is less than 98%?

Increase the SHARED_POOL_SIZE parameter.

9. Determine which packages, procedures, and triggers are pinned in the shared pool by querying v\$db_object_cache.

```
SQL> select name,type,kept
2> from v$db_object_cache
3> where type IN
4> ( 'PACKAGE', 'PROCEDURE', 'TRIGGER', 'PACKAGE
BODY' );
```

10. Connect using “sys/oracle as sysdba” and pin one of the Oracle supplied packages that needs to be kept in memory, such as SYS.STANDARD using the procedure DBMS_SHARED_POOL.KEEP, that is created by running the script \$ORACLE_HOME/rdbms/admin/dbmspool.sql.

```
SQL> CONNECT sys/oracle AS SYSDBA
SQL> @?/rdbms/admin/dbmspool
SQL> execute DBMS_SHARED_POOL.KEEP( 'SYS.STANDARD' );
SQL> select distinct owner, name
2> from v$db_object_cache
3> where kept='YES'
4> and name like '%STAND%';
```

11. Determine the amount of session memory used in the shared pool for your session by querying the v\$mystat view. Limit the output by including the clause where name = 'session uga memory'

```
SQL> select a.name, b.value
2> from v$statname a, v$mystat b
3> where a.statistic# = b.statistic#
4> and name = 'session uga memory';
```

12. Determine the amount of session memory used in the shared pool for all sessions, using V\$SESSTAT and V\$STATNAME views:

```
SQL> select SUM(value)||' bytes' "Total session memory"
2 from v$sesstat, v$statname
3 where name = 'session uga memory'
4 and v$sesstat.statistic# = v$statname.statistic#;
```

Practice 4

The objective of this practice is to use available diagnostic tools to monitor and tune the database buffer cache.

1. Connect as perfstat/perfstat user, and run a statistic snapshot, and note the snapshot number.

```
SQL> connect perfstat/perfstat
SQL> @$HOME/STUDENT/LABS/snap.sql
```

2. To simulate user activity against the database, connect as hr/hr user and run the lab04_02.sql script.

```
SQL> connect hr/hr
SQL> @$HOME/STUDENT/LABS/lab04_02.sql
```

3. Connect as system/manager and measure the hit ratio for the database buffer cache using the v\$sysstat view. Determine if it is a good ratio or not.

To query the V\$SYSSTAT view:

```
SQL> SELECT 1 - (phy.value - lob.value - dir.value)
2>           / ses.value "CACHE HIT RATIO"
3> FROM   v$sysstat ses, v$sysstat lob,
4>        v$sysstat dir, v$sysstat phy
5> WHERE  ses.name = 'session logical reads'
6> AND    dir.name = 'physical reads direct'
7> AND    lob.name = 'physical reads direct (lob)'
8> AND    phy.name = 'physical reads';
```

4. Connect as perfstat/perfstat, and run a statistic snapshot, and note the snapshot number. This can be performed by running the script file \$HOME/STUDENT/LABS/snap.sql.

```
SQL> connect perfstat/perfstat
SQL> @$HOME/STUDENT/LABS/snap.sql
```

5. Use the report from Statspack between the last two snapshots to check the buffer cache hit ratio, using the script \$HOME/STUDENT/LABS/spreport.sql. Then analyze the buffer hit % in the “Instance Efficiency Percentages” section.

```
SQL> @$HOME/STUDENT/LABS/spreport.sql
```

Note: On a production database if the ratio is bad, add new buffers, run steps 2 to 5, and examine the new ratio to verify that the ratio has improved. If the ratio is good, remove buffers, run steps 2 to 5, and verify if the ratio is still good.

Practice 4 (continued)

6. Connect as system/manager and determine the size of the table temp_emps in the hr schema that you want to place in the KEEP buffer pool. Do this by using the ANALYZE command, then query the BLOCKS column of the DBA_TABLES view for the table temp_emps.

```
SQL> connect system/manager
SQL> analyze table hr.temp_emps compute statistics;
SQL> select table_name , blocks
       2  from dba_tables
       3  where table_name in ('TEMP_EMPS');
```

7. We intend to keep TEMP_EMPS in the KEEP pool. Use the “alter system” command to set DB_KEEP_CACHE_SIZE to 4M for the KEEP pool. This will generate an error due to insufficient memory in the SGA.

```
SQL> alter system set db_keep_cache_size=4M;
```

8. To resolve the memory problem reduce the size of the shared pool by 8M, using the “alter system” command to set the value of SHARED_POOL_SIZE. Then reissue the command to size the db_keep_cache_size to 4M.

Note: In a production environment check if you have sufficient SGA size to grow, and if any other component could be reduced in size without adversely affecting performance.

```
SQL> alter system set shared_pool_size=40M;
SQL> alter system set db_keep_cache_size=4M;
```

9. Connect as system/manager and enable the TEMP_EMPS table in the hr schema for caching in the keep pool, using the storage clause of the “alter table” command..

```
SQL> connect system/manager
SQL> alter table hr.temp_emps
       2> storage (buffer_pool keep);
SQL> select table_name, buffer_pool
       2> FROM dba_tables
       3> WHERE buffer_pool = 'KEEP';
```

10. Connect as hr/hr, and run the script \$HOME/STUDENT/LABS/lab04_10.sql. This will execute a query against the temp_emps table in the hr schema.

```
SQL> connect hr/hr
SQL> @$HOME/STUDENT/LABS/lab04_10.sql
```

11. Connect using sys/oracle as sysdba and check for the hit ratio in different buffer pools, using the V\$BUFFER_POOL_STATISTICS view.

```
SQL> select name, physical_reads, db_block_gets,
       2> consistent_gets, 1 - (physical_reads
       3> / (db_block_gets + consistent_gets)) "hits"
       4> from v$buffer_pool_statistics;
```

Practice 5

1. Connect as perfstat/perfstat and collect a snapshot of the current statistics by running the script \$HOME/STUDENT/LABS/snap.sql. Record the snapshot id for later use.

```
SQL> connect perfstat/perfstat
```

```
SQL> @$HOME/STUDENT/LABS/snap;
```

2. Connect as user SH/SH and run the \$HOME/STUDENT/LABS/lab05_02.sql script in the STUDENT/LABS directory in order to have a workload.

```
SQL> connect sh/sh
```

```
SQL> @$HOME/STUDENT/LABS/lab05_02.sql
```

3. Connect as system/manager and query the V\$SYSSTAT view to determine if there are space requests for the redo log buffer.

```
SQL> SELECT  RBAR.name, RBAR.value, RE.name, RE.value
```

```
2> FROM      v$sysstat RBAR, v$sysstat RE
```

```
3> WHERE     RBAR.name = 'redo buffer allocation  
retries'
```

```
4> AND       RE.name    = 'redo entries';
```

4. Connect as perfstat/perfstat and collect another set of statistics using the \$HOME/STUDENT/LABS/snap.sql script. Then use \$HOME/STUDENT/LABS/spreport.sql to generate a report using the two snapshot ids that you have collected. From the report determine log buffer statistics. View the generated file using an editor, and locate the “log buffer space” statistic.

```
SQL> @$HOME/STUDENT/LABS/snap.sql;
```

```
SQL> @$HOME/STUDENT/LABS/spreport.sql
```

- From the list of snapshots select a beginning and end value. The beginning should be the value recorded in step 1, and the end value step 4.
- Note the name of the report file being generated.

5. Increase the size of the redo log buffer in the init.ora file located in the \$HOME/ADMIN/PFILE directory.

This is performed by increasing the value of LOG_BUFFER.

Practice 6

1. Connect as system/manager and diagnose database file configuration by querying the V\$DATAFILE, V\$LOGFILE and V\$CONTROLFILE dynamic performance views.

```
SQL> connect system/manager
SQL> SELECT name FROM v$datafile
2> UNION
3> SELECT member FROM v$logfile
4> UNION
5> SELECT name FROM v$controlfile
6> UNION
7> SELECT value from v$parameter
8> WHERE (name like 'log_archive_dest%'
9> and name NOT like 'log_archive_dest_state%')
10> or name in
11> ('log_archive_dest', 'log_archive_duplex_dest');
```

2. Diagnose database file usage by querying the V\$FILESTAT dynamic performance view, combine with V\$DATAFILE in order to get the datafile names

```
SQL> SELECT phyhdrs,phywrts,d.name
2> FROM v$datafile d, v$filestat f
3> WHERE d.file#=f.file#;
```

3. Determine if there are waits for redo log files by querying the V\$SYSTEM_EVENT dynamic performance view, where the waiting event is 'log file sync' or 'log file parallel write'

```
SQL> SELECT event, total_waits,time_waited, average_wait
2> from v$system_event
3> where event = 'log file sync'
4> or event = 'log file parallel write';
```

4. Waits for: log file sync are indicative of slow disks that store the online logs, or unbatched commits.

Log file parallel write is much less useful. The reason it is less useful is that this event only shows how often LGWR waits, not how often server processes wait. If LGWR waits without impacting user processes, there is no performance problem. If LGWR waits, it is likely that the log file sync event (mentioned above) will also be evident.

Practice 6 (continued)

5. Connect as perfstat/perfstat and diagnose file usage from STATSPACK
 - a. Generate a Statspack report using \$HOME/STUDENT/LABS/spreport.sql.
 - b. Locate and open the report file.
 - c. Examine the report, and search for the string “File IO Stats”

Note: On a production database care would be taken in monitoring the disk, and controller usage by balancing the workload across all devices. If your examination shows a distinct over utilization of a particular datafile, consider resolving the cause of the amount of I/O, for example investigate the number of full table scans, clustering of files on a specific device, and under utilization of indexes. If after this the problem remains then look at placing the datafile on a low utilization device.

6. Connect as system/manager and enable checkpoints to be logged in the alert file by setting the value of the parameter log_checkpoints_to_alert to true using “alter system set” command.

```
SQL> alter system set log_checkpoints_to_alert = true;
```

7. Connect as sh/sh and execute the \$HOME/STUDENT/LABS/lab06_06.sql script to provide a workload against the database.

```
SQL> connect sh/sh
```

```
SQL> @$HOME/STUDENT/LABS/lab06_06.sql
```

8. At the operation system level use the editor to open the alert log file (located in the directory specified by BACKGROUND_DUMP_DEST). Then determine the checkpoint frequency for your instance by searching for messages containing the phrase “Completed Checkpoint”. The time difference between two consecutive messages is the checkpoint interval

9. Open the alert log file using an editor, and search for the line: Completed checkpoint

The line before this will be the time at which the checkpoint occurred. Search for the following checkpoint time, and then subtract to get the time between checkpoints.

Note: On a production system the checkpoint interval should range between 15 minutes to 2 hours. The actual interval is dependant on the type of application, and the amount of DML activity.

Practice 7

1. Connect as system/manager and query the V\$SYSSTAT view, and record the value for sorts (memory) and sorts (disk). If the ratio of Disk to Memory sorts is greater than 5% then increase the sort area available.

```
SQL> connect system/manager
SQL> select name, value
2> from v$sysstat
3> where name like 'sorts%';
```

Note: The statistics collected from v\$sysstat are collected from startup. If you need to get accurate statistics per statement, you must record statistics from before the statement has run and again afterwards. Subtracting two values will give the statistics for the statement.

2. Connect as user sh/sh. In order to ensure that some sorts go to disk run the command “alter session set sort_area_size = 512;”. Then execute the SQL script (\$HOME/STUDENT/LABS/lab07_02.sql) that will force sorts to disk.

```
SQL> connect sh/sh
SQL> alter session set sort_area_size = 512;
SQL> @$HOME/STUDENT/LABS/lab07_02.sql;
```

Note: If this script fails due to a lack of free space in the TEMP tablespace. Resize the temporary tablespace.

```
SQL> alter database tempfile
2> '$HOME/ORADATA/u02/temp01.dbf' resize 200m;
```

3. Connect as system/manager and query the columns TABLESPACE_NAME, CURRENT_USERS, USED_EXTENTS and FREE_EXTENTS from the V\$SORT_SEGMENT view. The columns USED_EXTENTS, and FREE_EXTENTS are useful in monitoring the usage of the TEMPORARY tablespace.

```
SQL> select tablespace_name, current_users, used_extents, free_extents
2> from v$sort_segment;
```

Note: If this statement returns no rows, it means that all sort operations since startup have completed in memory.

4. To decrease the sorts number of sorts going to a temporary tablespace, increase the value of the parameter SORT_AREA_SIZE to 512000 using the “alter session” command.

```
SQL> Alter session set Sort_area_size = 512000;
```

5. Connect as system/manager and configure the new parameters for PGA memory allocation using the “alter system” command. Use the values AUTO for WORKAREA_SIZE_POLICY and 10M for PGA_AGGREGATE_TARGET)

```
SQL> connect system/manager
SQL> alter system set PGA_AGGREGATE_TARGET = 10M;
SQL> alter system set WORKAREA_SIZE_POLICY = AUTO;
```

Practice 9

The objective of this practice is to use available diagnostic tools to monitor and tune the rollback segments. This would require setting the database to Manual Undo Management mode.

1. Set the database in Manual Undo Mode.
 - a. Connect sys/oracle as SYSDBA and use shutdown immediate to close the database.
 - b. Edit the initialization parameter file locate \$HOME/ADMIN/PFILE and comment out the following lines
undo_management = AUTO
undo_tablespace = UNDOTBS
 - c. Save the modifications and startup the database. Confirm that the UNDO_MANAGEMENT is MANUAL, and UNDO_TABLESPACE is null
2. Connect sys/oracle as SYSDBA and create a new rollback segment tablespace RBS_TEST that is 2 MB in size using the CREATE UNDO TABLESPACE command. Name the datafile rbs_test.dbf and place it in the \$HOME/ORADATA/u03 directory.

```
SQL> connect sys/oracle as sysdba
SQL> CREATE UNDO TABLESPACE rbs_test
      2> DATAFILE '$HOME/ORADATA/u03/rbs_test.dbf' size 2M;
```

3. For the purposes of this practice, create a new rollback segment called RBSX in the RBS_TEST tablespace. For the storage parameters, use 10 KB for the INITIAL and NEXT extent sizes with MINEXTENTS value set to 20. Set the OPTIMAL value so that the segment shrinks back to 200 KB automatically.

```
SQL> create rollback segment rbsx
      2> tablespace rbs_test
      3> storage (initial 10K next 10K minextents 20
      4> optimal 200K);
```

4. Bring the rbsx rollback segment online and ensure that any others (except the SYSTEM rollback segment) are offline. Query the DBA_ROLLBACK_SEGS view to get the segment_name and status of the rollback segments to be taken offline using the ALTER ROLLBACK SEGMENT command.

```
SQL> alter rollback segment rbsx online;
SQL> select segment_id,segment_name,status
      2> from dba_rollback_segs;
```

5. Before executing a new transaction, find the number of bytes written so far in the RBSX rollback segment, using the writes column of v\$rollstat

```
SQL> select usn, writes
      2> from v$rollstat
      3> where usn>0;
```

Practice 9 (continued)

6. Open two sessions. In session 1 connect as hr/hr, and run the script \$HOME/STUDENT/LABS/ins_temps.sql. The script inserts 100 new rows into the TEMP_EMPS table – DO NOT COMMIT this transaction. In the second session, log in as system/manager, determine how many rollback segment blocks or bytes the transaction is using? To do this query the writes column of V\$ROLLSTAT to get the number of bytes written in the RBSX rollback segment so far. Record this value.

In session 1:

```
SQL> connect hr/hr
```

```
SQL> @$HOME/STUDENT/LABS/ins_temps.sql
```

In Session 2:

```
SQL> connect system/manager
```

```
SQL> select usn, writes
```

```
2> from v$rollstat
```

```
3> where usn>0;
```

NOTE: The number of writes in the rollback segment between questions 5 and 6 is the difference in the value of the writes column at the respective times.

7. Join the V\$TRANSACTION and V\$SESSION views to find, in the USED_UBLK column, how many blocks the ins_temps transaction is using.

```
SQL> SELECT s.username, t.used_ublk, t.start_time
```

```
2> FROM v$transaction t, v$session s
```

```
3> WHERE t.addr = s.taddr;
```

8. Return to the hr session (the first session) and commit the insert. Run the \$HOME/STUDENT/LABS/del_temps.sql script – DO NOT COMMIT. The script deletes the hundred rows you have just inserted. As user system (in the second session), check the amount of rollback space used, using the writes column of v\$rollstat . Note the difference between the return value, and that found in question 6.

In session 1:

```
SQL> commit;
```

```
SQL> @$HOME/STUDENT/LABS/del_temps.sql
```

In session 2:

```
SQL> select usn, writes
```

```
2> from v$rollstat
```

```
3> where usn>0;
```

Practice 9 (continued)

9. Connect as system/manager and find out if you have had any rollback segment contention since startup, using the waits and gets columns in the v\$rollstat view.

```
SQL> select sum(waits)/sum(gets) "Ratio",  
2> sum(waits) "Waits", sum(gets) "Gets"  
3> from v$rollstat;
```

Note: In a production environment a better source of information would be "Rollback Segment" section in the STATSPACK report.

10. Does the V\$SYSTEM_EVENT view show any waits related to rollback segments? Query in V\$SYSTEM_EVENT view for the "undo segment tx slot" entry.

```
SQL> select * from v$system_event  
2> where event = 'undo segment tx slot';
```

11. Connect as hr/hr and run the \$HOME/STUDENT/LABS/ins_temps.sql script again, allocating the transaction to a specific rollback segment RBSX, using the set transaction use rollback segment command. To check that the transaction is using the defined rollback segment join the V\$ROLLSTAT, V\$SESSION, and V\$TRANSACTION views.

```
SQL> set transaction use rollback segment rbsx;  
SQL> @$HOME/STUDENT/LABS/ins_temps.sql  
SQL> select s.username, rn.name  
2> from v$session s, v$transaction t,  
3> v$rollstat r, v$rollname rn  
4> where s.saddr = t.ses_addr  
5> and t.xidusn = r.usn  
6> and r.usn = rn.usn;
```

12. Set the database in Auto Undo Mode.

- Connect sys/oracle as SYSDBA and use shutdown immediate to close the database.
- Edit the initialization parameter file locate \$HOME/ADMIN/PFILE and uncomment the following lines
undo_management = AUTO
undo_tablespace = UNDOTBS
- Save the modifications and startup the database. Confirm that the UNDO_MANAGEMENT is AUTO, and UNDO_TABLESPACE is UNDOTBS

Practice 10

The objective of this practice is to use available diagnostic tools to monitor lock contention. You will need to start three sessions, in separate windows. Log in as hr/hr in two separate sessions (sessions 1 and 2) and as sys/oracle as sysdba in a third session (session 3).

1. In session 1 (user hr/hr), update the salary by 10% for all employee with a salary < 15000 in the temp_emps table. DO NOT COMMIT.

```
SQL> connect hr/hr
SQL> UPDATE TEMP_EMPS
2> SET SALARY = SALARY * 1.1
3> where salary <15000;
```

2. In session 3 (sys/oracle as sysdba) check to see if any locks are being held by querying the V\$LOCK.

```
SQL> connect sys/oracle as sysdba
SQL> select SID, TYPE, ID1, LMODE, REQUEST
2> from v$lock
3> where type in ('TX','TM');
```

3. In session 2 (user hr/hr – the session not yet used) drop the TEMP_EMPS table. Does it work?

```
SQL> connect hr/hr
SQL> DROP TABLE hr.temp_emps;
```

Note: The DDL statement requires an exclusive table lock. It cannot obtain it, because session 1 already holds a row exclusive table lock on the TEMP_EMPS table.

4. In session 2 (hr/hr), update the salary by 5% for all employee with a salary > 15000 in the temp_emps table. DO NOT COMMIT.

```
SQL> connect hr/hr
SQL> UPDATE TEMP_EMPS
2> SET SALARY = SALARY * 1.05
3> where salary > 15000;
```

5. In session 3, check to see what kind of locks are being held on the TEMP_EMPS table, using the V\$LOCK view.

```
SQL> SELECT sid, type, id1, id2, lmode, request
2> FROM v$lock
3> WHERE id1 =
4> (SELECT object_id FROM dba_objects
5> WHERE object_name = 'TEMP_EMPS'
6> AND object_type = 'TABLE');
```

Practice 10 (continued)

6. Roll back the changes you made in the second session (using hr/hr), and set the manager_id column to 10 for all employees who have a salary < 15000.

In session 2:

```
SQL> rollback;

SQL> UPDATE hr.temp_emps
2  SET MANAGER_id = 10
3  WHERE salary < 15000;
(This session will be hanging).
```

7. In session 3, check to see what kind of locks are being held on the TEMP_EMPS table, using the V\$LOCK view.

```
SQL> SELECT sid, type, id1, id2, lmode, request
FROM v$lock
WHERE id1 = (SELECT object_id FROM dba_objects
              WHERE object_name = 'TEMP_EMPS'
              AND object_type = 'TABLE');
```

8. In session 3, run the script \$ORACLE_HOME/rdbms/admin/catblock.sql. The script will create a view DBA_WAITERS, that gives information regarding sessions holding or waiting on a lock. Use this view to determine the session id for the session that is holding locks. Use this value to query v\$session to get the serial# for the session holding the lock. Then issue the alter system kill session command in order to release the session holding the lock.

```
SQL>
```

```
SQL> @$ORACLE_HOME/rdbms/admin/catblock.sql
```

```
SQL> select waiting_session, holding_session
```

```
2> from dba_waiters;
```

```
SQL> select SID,serial#,username
```

```
2> from v$session
```

```
3> where SID = '<SID>';
```

```
SQL> ALTER SYSTEM KILL SESSION '<SID>,<SERIAL#>';
```

Note: The second session would now show the message.

Practice 12

In this practice you will make use of the AUTOTRACE feature, and create the table plan_table. These are covered in detail in the chapter titled “SQL Statement Tuning”

1. Connect as hr/hr and create an IOT called ‘New_employees’ in the ‘HR’ schema. Give the table the same columns as the hr.employees table. Make the column employee_id the primary key, and name the primary key index new_employees_employee_id_pk

```
SQL> connect hr/hr
```

```
SQL> create table new_employees
```

```
2> (employee_id    number(6),
```

```
3> first_name      varchar2(20),
```

```
4> last_name       varchar2(25),
```

```
5> email           varchar2(25),
```

```
6> phone_number    varchar2(20),
```

```
7> hire_date       date,
```

```
8> job_id           varchar2(10),
```

```
9> salary           number(8,2),
```

```
10> commission_pct number (2,2),
```

```
11> manager_id     number(6),
```

```
12> department_id          number(4),
```

```
13> constraint new_employees_employee_id_pk
```

```
14> primary key (employee_id))
```

```
15> organization index;
```

2. Confirm the creation of the table by querying the user_tables, and the user_indexes views

- The IOT is a table, and so will be found in the user_tables view.

```
SQL> select table_name, iot_name, iot_type
```

```
2> from user_tables
```

```
3> where table_name like 'NEW_EMPLOYEES%';
```

- The IOT is an index, and so will be found in the user_indexes view.

```
SQL> select index_name, index_type
```

```
2> from user_indexes
```

```
3> where table_name like 'NEW_EMPLOYEES%';
```

3. Populate the new_employees table with the rows from the hr.employees table

```
SQL> insert into new_employees
```

```
2> select * from employees;
```


Practice 12 (continued)

4. Create a secondary B-Tree index on the column last_name of the new_employees table. Place the index in the INDX tablespace. Name the index last_name_new_employees_idx. Use the Analyze Index command to collect the statistics for the secondary index.

```
SQL> create index last_name_new_employees_idx
      2> on new_employees(last_name)
      3> tablespace indx;
SQL> analyze index last_name_new_employees_idx
      2> compute statistics;
```

5. Confirm the creation of the index by using the user_indexes view in the data dictionary. Query the index_name, index_type, blevel, and leaf_blocks.

```
SQL> select index_name, index_type, blevel, leaf_blocks
      2> from user_indexes
      3> where index_name = 'LAST_NAME_NEW_EMPLOYEES_IDX';
```

Note: If the values for blevel and leaf blocks are null then there were no statistics collected.

Confirm that the value of index_type is normal.

6. Create a reverse key index on the department_id of the employees_hist table. Place the index in the INDX tablespace. Name the index emp_hist_dept_id_idx.

```
SQL> create index emp_hist_dept_id_idx
      2> on employees_hist (department_id)
      3> tablespace indx
      4> reverse;
```

7. Confirm the creation of the index, and that it is a reverse key index, by querying the user_indexes view in the data dictionary. Query the index_name, index_type, blevel, and leaf_blocks.

```
SQL> select index_name, index_type, blevel, leaf_blocks
      2> from user_indexes
      3> where index_name = 'EMP_HIST_DEPT_ID_IDX';
```

Note: this time the values of blevel and leaf blocks should be null as you did not collect statistics for this index while creating it. Also the value for index type should now be normal/reverse.

8. Create a bitmap index on the job_id column of the employees_hist table. Place the index in the INDX tablespace. Name the index bitmap_emp_hist_idx.

```
SQL> create bitmap index bitmap_emp_hist_idx
      2> on employees_hist (job_id)
      3> tablespace indx;
```

Practice 12 (continued)

9. Confirm the creation of the index, and that it is a bitmapped index , by querying the user_indexes view in the data dictionary. Query the index_name, index_type, blevel, and leaf_blocks.

```
SQL> select index_name, index_type
2> from user_indexes
3> where index_name = 'BITMAP_EMP_HIST_IDX';
```

10. Connect as sh/sh, and confirm that the PLAN_TABLE exists. If the table does exist then truncate it, otherwise create the PLAN_TABLE using \$ORACLE_HOME/rdbms/admin/utlxplan.sql.

```
SQL> connect sh/sh
SQL> desc PLAN_TABLE
```

If the table is found:

```
SQL> truncate table plan_table;
```

If the table is not found then:

```
SQL> @$ORACLE_HOME/rdbms/admin/utlxplan
```

11. Create a Materialized View cust_sales having two columns, cust_last_name and the total sales for that customer. This will mean joining the sales and customers tables using cust_id, and grouping the results by cust_last_name. Make sure that query rewrite is enabled on the view.

```
SQL> connect sh/sh
SQL> create materialized view cust_sales
2> enable query rewrite as
3> select c.cust_last_name, sum(s.amount_sold)
4> from sales s, customers c
5> where c.cust_id = s.cust_id
6> group by c.cust_last_name;
```

12. Confirm the creation of the Materialized View cust_sales by querying the data dictionary view USER_MVIEWS, selecting the columns mview_name, rewrite_enabled and query.

```
SQL> select mview_name, rewrite_enabled, query
2> from user_mviews;
```

Note: Rewrite_enabled must be set to Y in order for the practice on query rewrite to work.

Practice 12 (continued)

13. Set AUTOTRACE to traceonly explain, to generate the explain plan for the query \$HOME/STUDENT/LABS/lab12_13.sql

```
SQL> set autotrace traceonly explain
```

```
SQL> @$HOME/STUDENT/LABS/lab12_13.sql
```

14. Set the query_rewrite_enabled parameter to true for the session and run the same query, \$HOME/STUDENT/LABS/lab12_13.sql, as in the previous practice. Note the change in the explain plan due to the query rewrite. Set AUTOTRACE off and disable query rewrite after the script has completed running.

```
SQL> alter session set query_rewrite_enabled = true;
```

```
SQL> @$HOME/STUDENT/LABS/lab12_13.sql
```

```
SQL> set autotrace off
```

```
SQL> alter session set query_rewrite_enabled = false;
```

Practice 13

1. Connect using sys/oracle as sysdba and query the tablespace_name and extent_management columns of DBA_TABLESPACES to determine which tablespaces are locally managed, and which are dictionary managed. Record which tablespaces are dictionary managed.

```
SQL> connect / as sysdba
```

```
SQL> select tablespace_name, extent_management
```

```
2> from dba_tablespaces;
```

2. Alter user HR to have the TOOLS tablespace as the default.

```
SQL> alter user hr default tablespace tools;
```

3. Examine the v\$system_event view and note the total_waits for the statistic enqueue.

```
SQL> select event, total_waits
```

```
2> from v$system_event
```

```
3> where event = 'enqueue';
```

Note: On a production system you would be more likely to pickup the contention through the STATSPACK report.

4. Also examine the view v\$enqueue_stat for eq_type 'ST' in order to determine the total_wait# for the ST enqueue, which is the space management enqueue.

```
SQL> select * from v$enqueue_stat
```

```
2> where eq_type = 'ST';
```

5. Exit out of the SQLPlus session and change directory to \$HOME/STUDENT/LABS. Run the script lab13_04.sh from the operating system prompt. This script will log five users onto the database simultaneously and then each user creates, then drops, tables. The tables each have many extents. The script must be run from the \$HOME/STUDENT/LABS directory, or it will fail.

```
$ cd $HOME/STUDENT/LABS
```

```
$ ./lab13_04.sh
```

6. Connect as system/manager and again examine the view v\$enqueue_stat for eq_type 'ST' in order to determine if the value of total_wait# for the ST enqueue, which is the space management enqueue.

```
$ sqlplus system/manager
```

```
SQL> select * from v$enqueue_stat
```

```
2> where eq_type = 'ST';
```

Note: Record the difference in the number of waits for the ST enqueue for extent management using a dictionary managed tablespace. This value is found by subtracting the first wait value (from practice 13-04) from the second wait value (from practice 13-06).

Practice 13 (continued)

7. Create a new locally managed tablespace TEST, name the datafile test01.dbf, and place it in the directory \$HOME/ORADATA/u06. Set the size to 120M, and a uniform extent size of 20k.

```
SQL> create tablespace test
```

```
2> datafile '$HOME/ORADATA/u06/test01.dbf' size 120M
```

```
3> uniform size 20k;
```

8. Alter the default tablespace of user hr to test.

```
SQL> alter user hr default tablespace test;
```

Note: The same steps are covered again. This time you are looking for the number of waits for the ST enqueue caused by locally managed tablespaces.

9. Examine, and record the initial total_wait# for 'ST' in the v\$enqueue_stat view.

```
SQL> select * from v$enqueue_stat
```

```
2> where eq_type = 'ST';
```

10. Exit out of the SQLPlus session and change directory to \$HOME/STUDENT/LABS.

Run the script lab13_04.sh from the operating system prompt. This script will log five users onto the database simultaneously and then each user creates, then drops, tables. The tables each have many extents. The script must be run from the \$HOME/STUDENT/LABS directory, or it will fail.

```
$ cd $HOME/STUDENT/LABS
```

```
$ ./lab13_04.sh
```

11. Again examine, and record the final total_wait# for 'ST' in the v\$enqueue_stat view.

```
SQL> select * from v$enqueue_stat
```

```
2> where eq_type = 'ST';
```

Note: Record the difference in the total_wait# for the ST enqueue for extent management using a locally managed tablespace. This value is found by subtracting the first wait value (from practice 13-09) from the second wait value (from practice 13-11). Compare the two results for the different tablespaces. The locally managed tablespace would be far less contentious with extent management since it is managing the space within the tablespace itself.

12. Connect as user hr/hr and run the script \$HOME/STUDENT/LABS/lab13_12.sql. This will create a similar table (NEW_EMP) as the employees table but with PCTFREE = 0. The table is then populated with data from the employees table.

```
– SQL> connect hr/hr
```

```
– SQL> @$HOME/STUDENT/LABS/lab13_12.sql;
```

13. Run analyze on the new_emp table and query the DBA_TABLES view to determine the value of chain_cnt for the new_emp table. Record this value.

```
– SQL> analyze table new_emp compute statistics;
```

```
– SQL> select table_name, chain_cnt
```

```
– 2> from user_tables
```

```
– 3> where table_name = 'NEW_EMP';
```

Practice 13 (continued)

14. Create an index `new_emp_name_idx` on the `last_name` column of the `new_emp` table. Place the index in the tablespace `INDX`. Then confirm the index's status in `user_indexes`.

```
SQL> create index new_emp_name_idx on new_emp(last_name)
      2> tablespace indx;
SQL> select index_name, status
      2> from user_indexes
      3> where index_name = 'NEW_EMP_NAME_IDX';
```

15. Run the script `$HOME/STUDENT/LABS/lab13_15.sql` which will update the rows of the `new_emp` table. Analyze the `new_emp` table again and query the `USER_TABLES` view to get the new value of `chain_cnt`. Record this value. Also check the status of the index `new_emp_name_idx`.

```
SQL> @$HOME/STUDENT/LABS/lab13_15.sql
SQL> analyze table new_emp compute statistics;
SQL> select table_name, chain_cnt
      2> from user_tables
      3> where table_name = 'NEW_EMP';
SQL> select index_name, status
      2> from user_indexes
      3> where index_name = 'NEW_EMP_NAME_IDX';
```

16. Resolve the migration caused by the previous update, by using the `alter table move` command. This will cause the index to become unusable, and should be rebuilt using the `alter index rebuild` command before re-analyzing the table `new_emp`. Confirm that the migration has been resolved by querying `chain_cnt` column in `user_tables`, and confirm that the index is valid by querying `user_indexes`.

```
SQL> alter table new_emp move
      2> tablespace users;
SQL> alter index new_emp_name_idx rebuild;
SQL> analyze table new_emp compute statistics;
SQL> select table_name, blocks, empty_blocks, chain_cnt
      2> from user_tables
      3> where table_name = 'NEW_EMP';
SQL> select index_name, status
      2> from user_indexes
      3> where index_name = 'NEW_EMP_NAME_IDX';
```

Practice 14

The objective of this practice is to familiarize you with SQL statement execution plans and to interpret the formatted output of a trace file generated using SQL Trace and the formatted output generated by TKPROF.

1. Connect as hr/hr, and create the PLAN_TABLE under the HR schema, if it is not already created, by running \$ORACLE_HOME/rdbms/admin/utlxplan.sql

```
SQL> connect hr/hr
```

```
SQL> @$ORACLE_HOME/rdbms/admin/utlxplan.sql
```

Note: If plan_table already exists and holds rows truncate the table.

2. Set the Optimizer_goal to rule based using the alter session command, and generate the explain plan for the statement \$HOME/STUDENT/LABS/lab14_02.sql. View the generated plan by querying object_name, operation, optimizer from PLAN_TABLE.

```
SQL> alter session set optimizer_goal = rule;
```

```
SQL> explain plan for
```

```
2> @$HOME/STUDENT/LABS/lab14_02.sql
```

```
SQL> select object_name, operation, optimizer
```

```
2> from plan_table;
```

3. Truncate the PLAN_TABLE. Change the optimizer_goal to cost based by setting the value to ALL_ROWS, and rerun the explain plan for \$HOME/STUDENT/LABS/lab14_02.sql. Notice that the Optimizer mode, and the explain plan have changed.

```
SQL> alter session set optimizer_goal = all_rows;
```

```
SQL> explain plan for
```

```
2> @$HOME/STUDENT/LABS/lab14_02.sql
```

```
SQL> select object_name, operation, optimizer
```

```
2> from plan_table;
```

Note: Although exactly the same scripts are being run, due to the different optimizer settings, different explain paths are found. With rule based, one of the rules is to use any index that is on the columns in the where clause. By using cost based optimizer mode, the server has been able to determine that it will be faster to just perform a full table scan, due to the number of rows being returned by the script.

4. Truncate the PLAN_TABLE, and set the optimizer goal to rule by using the alter session command. This time generate the explain plan for the script \$HOME/STUDENT/LABS/lab14_04.sql. Examine the script which is a copy of \$HOME/STUDENT/LABS/lab14_02.sql except it changes the line “select *” to include a hint /*+ all_rows*/ for the optimizer. View the generated execution plan by querying object_name, operation, optimizer from PLAN_TABLE.

```
SQL> truncate table plan_table;
```

```
SQL> alter session set optimizer_goal = rule;
```

Practice 14 (continued)

```
SQL> explain plan for
```

```
2> @$HOME/STUDENT/LABS/lab14_04.sql
```

```
SQL> select object_name, operation, optimizer
```

```
2> from plan_table;
```

5. Exit out of SQLPLUS, change the directory to \$HOME/ADMIN/UDUMP and delete all the trace files already generated.

```
SQL> exit
```

```
$ cd $HOME/ADMIN/UDUMP
```

```
$ rm *.trc
```

6. Connect as sh/sh and enable SQL TRACE, using the alter session command, to collect statistics for the script, \$HOME/STUDENT/LABS/lab14_05.sql. Run the script. After the script has completed, disable the SQL_TRACE. and then format your trace file using TKPROF. Use the options SYS=NO and EXPLAIN= sh/sh. Name the file myfile.txt

```
$ sqlplus sh/sh
```

```
SQL> alter session set sql_trace = true;
```

```
SQL> @$HOME/STUDENT/LABS/lab14_05.sql
```

```
SQL> alter session set sql_trace = false;
```

```
$ cd $HOME/ADMIN/UDUMP
```

```
$ ls -l
```

```
-rw-r----- 1 user457 dba 2180 May 4 00:27 user457_ora_10424.trc
```

```
$ tkprof user457_ora_10424.trc myfile.txt explain=sh/sh sys=no
```

7. View the output file myfile.txt, and note the CPU, current, and query figures for the fetch phase. Do not spend time analyzing the contents of this file as the only objective here is to become familiar and comfortable with running TKPROF and SQL Trace.

```
$ more myfile.txt
```


Practice 14 (continued)

8. Connect hr/hr and gather statistics for all objects under the HR schema using the DBMS_STATS package, while saving the current statistics then restore the original statistics.

- a. Connect as HR and create a table to hold statistics in that schema.

```
$ sqlplus hr/hr
SQL> execute -
dbms_stats.create_stat_table('HR','MY_STATS');
```

- b. Save the current schema statistics into your local statistics table.

```
SQL> execute -
dbms_stats.export_schema_stats('HR','MY_STATS');
```

- c. Analyze all objects under the HR schema.

```
SQL> execute dbms_stats.gather_schema_stats('HR');
```

- d. Remove all schema statistics from the dictionary and restore the original statistics you saved in step b.

```
SQL> execute dbms_stats.delete_schema_stats('HR');
SQL> execute -
dbms_stats.import_schema_stats('HR','MY_STATS');
```


B Tuning Workshop

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

Workshop Scenarios

- **Small shared pool**
- **Small database buffer cache**
- **Small redo log buffer cache**
- **Missing indexes**
- **Rollback segments and undo tablespace**
- **Sort Area Size incorrectly set**
- **Reading STATSPACK report**

ORACLE

Appendix B-2

Copyright © Oracle Corporation, 2001. All rights reserved.

Workshop Scenario

At the beginning of each scenario, the database is shut down and then restarted. Perform the following steps:

Make sure that the job scheduler is set to collect statistics every 10 minutes.

Start the workload generator and note the time. If the workload generator is still running against the database, stop the generator and restart it.

Allow time for some statistics to be generated (at least 20 minutes). Shorter time periods make it more difficult to determine where problems exist.

After an adequate period of time, run the script spreport.sql. Choose a start and end time that falls between the period that the workload generator was running. Name the report in a manner associated with the scenario, for example, for scenario 1 use reportscn1.txt, for Scenario 5 use reportscn5.txt, and so on.

Look for the generated report in the directory from which SQLPLUS was executed.

When resizing memory components, do not consume more memory than is actually required in order to meet the requirements of the objects given. For example, there is little point in resizing the shared pool to 500MB. The purpose of the workshop is to be as realistic as possible.

Workshop Scenario (continued)

The company concerned has at present two OLTP users and two DSS users. The system was set up by a trainee DBA, and though it works, the performance is very slow. You have been invited in to get the present system working for 20 users. The company is about to expand to 10 of each type of user (twenty users in all). At the same time the company is unwilling to spend extra on new hardware components.

Therefore, the management has imposed a limit of 20 MB for the entire SGA.

Workshop Scenario 1 (Shared Pool)

- **Collect a snapshot.**
- **Provide a workload by executing the workload generator.**
- **Collect a snapshot.**
- **Generate a report.**

ORACLE

Appendix B-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Workshop Scenario 1

Waits recorded on the latch “Shared pool, library cache” could be indicative of a small shared pool. However, before rushing out and increasing the `SHARED_POOL_SIZE`, it would be advisable to determine why the pool is too small. Some reasons are listed below:

- Many SQL statements stored in the SQL area that are only executed 1, and they differ only in literals in the where clause. A case could be made here for using bind variables, or setting `CURSOR_SHARING`. In order to determine these SQL statements examine the report created by `STATSPACK`, or query `v$sql` using the like option of the where clause to collect information regarding similar SQL statements.
- Examine what packages are loaded using the query shown below:

```
select * from v$db_object_cache
where sharable_mem > 10000
and (type='PACKAGE' or type='PACKAGE BODY' or
     type='FUNCTION' or type='PROCEDURE')
and KEPT='NO';
```

Workshop Scenario 1 (continued)

If these packages are used frequently, pin them in memory using the `Dbms_shared_pool.keep('Package_name');` package. Another area to examine would be reducing the size of the package. Determine if there are large portions of the package that are not commonly used. If possible, separate procedures into packages that will have their contents utilized.

Examine SQL statements that are executed often using the following statement:

```
select sum(sharable_mem)
from V$SQLAREA where executions > 5;
```

With this information determine if the SQL statement can be converted into a procedure and stored as a package; this can assist users in sharing the same cursor.

After you have reduced the number of SQL statements as much as possible, run the following query:

```
select sum(pins) "Executions", sum(reloads)
       "Cache Misses", sum(reloads)/sum(pins)
from v$sqllibrarycache;
```

Increase the shared pool in order to reduce *cache misses*. Record the increase received for each increase in the shared pool in order to determine if the extra memory is worth the increase received.

Data Dictionary Cache

The data dictionary cache cannot be independently resized. The Oracle server automatically assigns space from the `Shared_Pool_Size` parameter for the shared SQL area, and the data dictionary cache. Most of the increase to `SHARED_POOL_SIZE` is allocated to the shared SQL area.

In order to determine the hit ratio of the data dictionary cache, run the query:

```
select parameter, gets, getmisses
from v$rowcache;
```

The result of this query is a row for each segment of the data dictionary cache. Each area then can be checked for usage. For example, if there is a large number of gets on `dc_sequences`, this is probably due to sequence numbers not being cached. To reduce the number of gets on `dc_sequences`, examine increasing the number of sequence numbers cached.

Workload Scenario 2 (Buffer Cache)

- **Collect a snapshot.**
- **Provide a workload by executing the workload generator.**
- **Collect a snapshot.**
- **Generate a report.**

ORACLE

Appendix B-6

Copyright © Oracle Corporation, 2001. All rights reserved.

Workshop Scenario 2

The first indication that the buffer cache is too small is waits on free buffer waits event. The cache buffers LRU chain latch might also indicate that the buffer cache is too small. Waits on the latch may also signify that the DBWR process is not able to keep up with the work load.

In order to determine which problem is causing the latch contention, examine the number of writes in the file statistics found in the STATSPACK report.

On the front page of the STATSPACK report, the section named “Instance Efficiency Percentages” lists the important ratios of the instance. For this scenario, the values of “Buffer Hit%:” are of interest.

Values for the “Buffer Hit%:” depend on individual systems. Ideally, this value should be close to 100 percent; however, there might be several reasons why this goal cannot be realized. A low percentage indicates that there are a lot of buffers being read into the database buffer cache.

Before increasing the size of the database buffer cache, you should examine what SQL statements are being run against the database. You are looking for statements that cause a high number of *buffer gets* and how many times these statements are executed. If a statement is executed only a few times, it may not be worth the effort of tuning. However, a statement that is executed many times, and has a high number of *buffer gets* is an ideal candidate for SQL tuning.

Workshop Scenario 2 (continued)

STATSPACK lists the SQL statements that have been executed on the database. The first such listing orders the statements according to the number of gets. Examine the top statements, keeping in mind that packages are also listed here, not just the SQL statements. When a candidate statement is found, examine the SQL statement to determine if:

- There are indexes that could be created to assist in using less blocks
- There is a better way to write the statement in order to return the same data, but use less blocks
- Investigate if the statement has to be executed so often. Can output be generated and then shared between users?

After you have examined the SQL statements, and exhausted all means to reduce the number of buffers, then consider changing the size of the buffer cache. You must determine two items before changing the buffer cache size:

1. What is the current size?

Query the data dictionary in order to determine the current size of the buffer cache. This can be performed by:

Show parameter

```
Select * from v$sgastat where name = 'db_block_buffers';
```

Front page of the STATSPACK report

2. What is an appropriate size for the buffer cache?

Determine the new value of the buffer cache by using the `db_cache_advice` parameter. This parameter can have one of three values: OFF, ON and READY. Setting the value to ON will start collecting the required statistics. The value can be changed by either:

- Editing the `init.ora` file and bouncing the database
- Using the `alter system set db_cache_advice = on;` command

When Values has been set to ON, let the system run the required scripts in order to collect information regarding buffer usage.

After the database executes a typical load, query the `$db_cache_advice` view and set a new value for the `db_cache_size` parameter. When a new size has been determined, use the following command to dynamically change the size of the cache.

```
alter system set db_block_buffers = new_value;
```

Run a test load with this new value. Collect the statistics again. If the increase has resolved the problem then change the value in the `init.ora` file.

Workshop Scenario 3 (Redo Log Buffer)

- **Collect a snapshot.**
- **Provide a workload by executing the workload generator.**
- **Collect a snapshot.**
- **Generate a report.**

ORACLE

Appendix B-8

Copyright © Oracle Corporation, 2001. All rights reserved.

Workshop Scenario 3

Waits on the event *log buffer space* is an indication that your log buffer is too small.

On the first page of the STATSPACK report there is section named “Instance Efficiency Percentages.” Note the value of the statistic `redo no wait`. While this statistic’s ideal value of 100 percent is seldom achieved, any lesser value indicates that the Redo Log Buffer is not correctly sized. If no memory is available, consider reducing the amount of redo created by the use of `no logging` in appropriate statements.

Query the data dictionary in order to determine the current size of the Redo Log Buffer.

In order to determine an estimate on the increase required, examine the amount of redo that is generated; this is found on the first page of the STATSPACK report, under the heading “Load Profile.”

Edit the `init.ora` file to set a new size for the redo log buffer.

Bounce the database.

Workshop Scenario 3 (continued)

Rerun the workload generator and collect the statistics again. If the increase has resolved the problem; and then, if the change was vast, repeat the process with a larger redo log buffer.

Note: The redo log buffer does not always have the size stipulated in the parameter file. This is due to minimum size, and rounding upwards to the nearest Oracle block. To confirm the actual redo log buffer size use the `v$sgastat` view.

Workshop Scenario 4 (Indexes)

- **All the indexes have been eliminated.**
 - Result full table scans instead of using indexes
- **Run workload generator**
- **Examine the SQL statements executed**
- **Determine which indexes should be recreated**

ORACLE

Appendix B-10

Copyright © Oracle Corporation, 2001. All rights reserved.

Workshop Scenario 4

Several indexes have been deleted and performance has decreased. The results are seen in the STATSPACK report where there are many indications that *untuned* SQL is running. For example:

- The buffer cache hit ratio is lower. This can be seen on the first page of the STATSPACK report in the Load Profile section.
- There are more waits on the free buffer waits event.
- There are more full table scans occurring.

This indicates that, because of incorrectly written SQL scripts, or missing or incorrect indexes, the database is performing too many full table scans.

In order to resolve the problem, you must determine which SQL statements are run on the database during a normal workload. Either use the SQL_Trace utility or examine the top resource users in the STATSPACK report to collect a representation of these SQL statements.

After the appropriate statements are collected, examine the WHERE clauses. Any columns referenced in the WHERE clause are good candidates for indexes.

Workshop Scenario 4 (continued)

In order to determine if an index would be used by the optimizer, you must look at the expected number of rows to be returned. The more rows to be returned, the less likely an index will improve performance.

Confirm that the required indexes are present and enabled. The index might have been disabled for some reason. If the index is not present, then create the index.

Workshop Scenario 5 (Rollback Segments)

Create a workload by running:

- **A daytime workload with the workload generator**
- **A nighttime workload with the `wk3.sh` script**

You may not use more disk space than is already used.

ORACLE

Appendix B-12

Copyright © Oracle Corporation, 2001. All rights reserved.

Workshop Scenario 5

The company has 20 users that log on to the database during daytime hours and perform OLTP transactions. To simulate this workload use the workload generator and set both order entry and shipping to OLTP1. During the nighttime, there are five users that log on and perform refreshes of certain tables. Currently, the system is set up with one very large rollback segment.

For the STATSPACK report collected between normal work hours (that is, when the workload generator is running), there is a section named “Buffer wait Statistics”. In this section should be found a statistic “undo header” which indicates that there is contention for the rollback segment header blocks.

In resolving this problem, do not forget that the nighttime run has to use the large rollback segment.

Use the two methods to resolve this problem.

Pre-Oracle9i Method

In order to resolve contention on the rollback segment header, more rollback segments have to be created. In order to do this without using more disk space requires some planning. For the nighttime run there should be few very large rollback segments, and smaller rollback segments during the daytime run. The scenario is further complicated because the nighttime run must not have access to the smaller rollback segments in case Oracle server allocates a small rollback segment to a large transaction, in which case the large transaction is likely to fail.

Workshop Scenario 5 (continued)

Oracle9i Method

In Oracle9i, the problems around the number and size of rollback segments has been eliminated. Instead of creating rollback segments, the DBA only has to create an undo tablespace that is large enough.

Workload Scenario 6 (Sorting)

- **Create a data warehouse type workload by running the workload generator. Have both sets of users using query1.**
- **Make sure that STATSPACK is collecting statistics.**
- **After statistics have been collected, generate a report.**
- **Examine the report.**

ORACLE

Appendix B-14

Copyright © Oracle Corporation, 2001. All rights reserved.

Workshop Scenario 6

During the evening hours, the company has twenty batch programs that log on and create a series of reports for management. This requires a lot of sorting. Currently, the scripts are completing. However, management would appreciate a more rapid completion.

The first step is to run the scripts and collect the STATSPACK report. Also make a note of how many transactions are completed during your running period (should be at least ten minutes).

Because we are concerned about sorts, the tendency is to jump straight to the Instance Activity stats and look for the values of Sorts (Disk), Sorts (Memory), and Sorts (Rows). However, doing so ignores some good information found on the front page.

On the front page, look at the Buffer Hit ratio. In a data warehouse environment we would expect this ratio to drop; however, combining this information with the high number of Buffer Busy Waits indicates that the buffer cache is too small for the number of sorts taking place. So you would likely wish to increase the size of the buffer cache.

Moving to the Instance Activity report, you find that a large number of sorts are having to go to disk. Ideally, no sorts should go to disk; however, accomplishing this has a high cost in memory. The ratio of sorts (Disk) to Sorts (Memory) should be less than 5 percent.

Increasing the value of `SORT_AREA_SIZE` will resolve this issue, and more sorts will be performed in memory. The disadvantage of increasing the `SORT_AREA_SIZE` parameter is that more memory is consumed, and after completing the sort run, this memory is not released.

Workshop Scenario 6 (continued)

back to the operation system. It is released, but only to the sessions UGA. Therefore many users performing large sorts will consume memory.

Try increasing the value of the `SORT_AREA_SIZE` parameter by a factor of 10, and see what the effect is on the number of sorts (Disk).

In releases prior to Oracle9i, there was nothing a DBA could do in order to resolve this problem. A choice had to be made between increasing memory consumption (by changing the value of `SORT_AREA_SIZE`), or live with sorts going to disk.

In Oracle9i it is possible to change the manner in which sorts are handled in memory. Instead of allocating memory in terms of `SORT_AREA_SIZE` (which would then be locked onto one session), Oracle9i allows one memory allocation to be used for all sessions, thereby enabling the sharing of memory after the sort has completed.

The new parameter is `PGA_AGGREGATE_TARGET` and ranges in size from 10Mb to 4000Gb. When setting the value for this parameter, keep in mind that this is the total sort area for all sessions performing sorts.

In order to enable use of `PGA_AGGREGATE_TARGET`, the `WORKAREA_SIZE_POLICY` to `AUTO` parameter must be set.

Workload Scenario 7 (Reading STATSPACK Report)

- **This is an actual STATSPACK report.**
- **Examine the report in Appendix C.**
- **Make a list of problems and solutions.**

ORACLE

Appendix B-16

Copyright © Oracle Corporation, 2001. All rights reserved.

Workshop Scenario 7

The report in Appendix C is an actual production STATSPACK report. You are going to use this report to determine what should be looked at on the system in order to increase performance. For this exercise there will not be any actions processed on the database.

When reading the STATSPACK report and deciding upon an action, you must consider all the information given. Never just look at the front page, and discard all the rest. Among the many items on the front page of the STATSPACK that you should take note of are:

1. The length of the period over which the report is taken. This figure gets used when examining other statistics, for example, when looking at how many times a SQL statement was run during the collection period.
In the example, the period of collection is 239.93 minutes (roughly 4 hrs).
2. Examine the load profile statistics. It would be useful to compare this report with a previous report (a benchmark). Doing so could provide important information on how the load has changed, which would then point the way to where a bottleneck is occurring.
3. Take note of the percentage for “Rollback per transaction.” A value of 12.43% indicates that roughly 1 transaction in 8 gets rolled back. Rolling back increases the workload of the database.

Workshop Scenario 7 (continued)

3. Examine why there is such a high amount of rollback occurring. Check that users have received the correct training on the system that they are using. Confirm that the code itself is not making use of unnecessary rollback. Often, educating the users and developers on the effects of rollback drastically reduces the number of rollbacks performed. Reducing the amount of rollbacks performed will also reduce the amount of redo generated.
4. Examine the “Instance Efficiency Percentages.” The target is to have all of these figures at 100 percent; however, it is uncommon to achieve this target for a production database. Again it would be very useful to compare these figures against the benchmark report. Statistics that have altered dramatically could provide an indication of how the system has changed, and therefore where the bottlenecks might be occurring.
5. The last area to examine on the front page is the “Top 5 Wait Events.” The report is a summary of the wait events view from the next page. The purpose of highlighting the top five wait events is to reinforce the importance of resolving the areas of high total wait time. Both the front page Top 5 report, and the complete wait event report on the following page, sort the background process wait events, such as pmon timer, to the bottom of the list.
6. Examining the report will show that there is a total wait time of 1439745.1 seconds for the enqueue event. This should be investigated further by examining the `v$enqueue_stat` view. From this view you can determine where the bottleneck is occurring, and thereby attempt to eliminate the waits. For example, if there are waits on the ‘ST’ enqueue (which is the space management enqueue), the likely cause is dynamic extent management, and you need to investigate using locally managed tablespaces, or pre-allocate the extents.
After examining the first page of the report, the next section of interest contains the SQL statements executed on the database. The first of these reports sorts the SQL statements in accordance with buffer gets. The first statement listed is the statement consuming the most buffer gets.
7. In the example, the top statement has 174,229,988 buffer gets during the four hour monitoring period. The statement is executed 12,762 times, meaning that there are 13,652 buffer gets per execution. This presents a good candidate for some tuning. Examine the following:
 - a. Do indexes exist on the appropriate columns?
 - b. Are the indexes valid?
 - c. Are the indexes being used? (Check the explain plan for this statement.)Due to the number of executions, and the high number of buffer gets that this statement generates, it is highly likely to give the biggest gain for time spent in tuning.
8. Another part of the SQL statement report to examine are the SQL statements sorted by physical reads. The first few statements are the ones that generate the best time returned on tuning.

Workshop Scenario 7 (continued)

9. Another good use for the SQL statement section of the STATSPACK report is to examine what statements are being run on the database. In order to increase the benefits of searching through the statements, a DBA should understand what the application does, and why. However, even without that level of understanding, some statements will stand out.

For example, in our report in the section that sorts the SQL statements by executions is found the statements:

- `SELECT SYSDATE FROM SYS.DUAL` which is executed 652,994 times.
- `SELECT SYSDATE FROM DUAL` which is executed 532,476 times.

The first item of concern is why are two cursors being used, instead of sharing cursors by making both statements the same.

The second item of concern is why, in a 4-hour period (or 240 minutes, or 14,400 seconds), there is a total of 1,185,470 queries regarding the system date. This works out to be roughly 82 queries per second and would warrant an investigation as to why these are occurring. However, because neither of these statements are featured on the other SQL statement lists, the performance impact is minor and not worthy of a prolonged investigation.



Example of STATSPACK Report

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

STATSPACK report for

DB Name	DB Id	Instance	Inst Num	Release	OPS	Host
dba01	123456789	dba01		# 8.1.7.0.0	NO	dba01

	Snap Id	Snap Time	Sessions
Begin Snap:	2	02-May-01 12:00:10	2,519
End Snap:	4	02-May-01 16:00:06	2,519
Elapsed:		239.93 (mins)	

Cache Sizes

~~~~~

|                   |       |                   |           |
|-------------------|-------|-------------------|-----------|
| db_block_buffers: | 90000 | log_buffer:       | 524288    |
| db_block_size:    | 8192  | shared_pool_size: | 262144000 |

Load Profile

~~~~~

	Per Second	Per Transaction
Redo size:	132,801.80	11,445.59
Logical reads:	43,217.78	3,724.75
Block changes:	710.62	61.25
Physical reads:	2,649.21	228.32
Physical writes:	181.81	15.67
User calls:	1,545.14	133.17
Parses:	630.82	54.37
Hard parses:	1.84	0.16
Sorts:	289.81	24.98
Logons:	0.63	0.05
Executes:	752.20	64.83
Transactions:	11.60	
% Blocks changed per Read:	1.64	Recursive Call %: 11.57
Rollback per transaction %:	12.43	Rows per Sort: 9.08

Instance Efficiency Percentages (Target 100%)

~~~~~

|                               |       |                   |        |
|-------------------------------|-------|-------------------|--------|
| Buffer Nowait %:              | 99.97 | Redo NoWait %:    | 100.00 |
| Buffer Hit %:                 | 93.87 | In-memory Sort %: | 100.00 |
| Library Hit %:                | 99.79 | Soft Parse %:     | 99.71  |
| Execute to Parse %:           | 16.14 | Latch Hit %:      | 97.52  |
| Parse CPU to Parse Elapsed %: | 71.99 | % Non-Parse CPU:  | 100.00 |

Shared Pool Statistics

|                            | Begin | End   |
|----------------------------|-------|-------|
| Memory Usage %:            | 89.60 | 89.68 |
| % SQL with executions>1:   | 59.55 | 78.54 |
| % Memory for SQL w/exec>1: | 44.22 | 65.22 |

Top 5 Wait Events

~~~~~

Event	Waits	Wait Time (cs)	% Total Wt Time
enqueue	472,825	143,974,510	82.43
db file sequential read	15,863,265	15,686,892	8.98
PX Deq: Execution Msg	33,383	5,719,203	3.27
latch free	1,618,619	2,300,843	1.32
db file scattered read	717,629	1,364,228	.78

Wait Events for DB: dba01 Instance: dba01 Snaps: 2 -4

-> cs - centisecond - 100th of a second

-> ms - millisecond - 1000th of a second

-> ordered by wait time desc, waits desc (idle events last)

Event	Waits	Timeouts	Total Wait Time (cs)	Avg wait (ms)	Waits /txn
enqueue	472,825	471,414	#####	3045	2.8
db file sequential read	15,863,265	0	15,686,892	10	95.0
PX Deq: Execution Msg	33,383	27,499	5,719,203	1713	0.2
latch free	1,618,619	1,515,932	2,300,843	14	9.7
db file scattered read	717,629	0	1,364,228	19	4.3
PX Deq: Table Q Sample	5,406	5,129	1,040,147	1924	0.0
Replication Dequeue	92,127	2,617	929,538	101	0.6
PX Deq Credit: send blkd	4,556	4,156	857,664	1882	0.0
PX Deq: Execute Reply	5,636	1,474	535,227	950	0.0
PX Deque wait	91,970	2,211	468,056	51	0.6
global cache lock open x	101	78	453,806	44931	0.0
PX Deq: Table Q Get Keys	2,196	2,089	436,612	1988	0.0
PX Deq: Table Q Normal	4,042	1,215	354,350	877	0.0
buffer busy waits	160,335	978	286,225	18	1.0
log file parallel write	319,798	0	88,161	3	1.9
file open	78,950	0	58,851	7	0.5
log file sync	107,835	109	58,767	5	0.6
SQL*Net more data to client	2,331,201	0	22,557	0	14.0
PX qref latch	151	129	13,621	902	0.0
SQL*Net message from dblink	19,350	0	8,834	5	0.1
log file sequential read	32,674	0	3,435	1	0.2
db file parallel read	369	0	2,842	77	0.0
control file parallel write	5,073	0	2,814	6	0.0
control file sequential read	9,687	0	665	1	0.1
log file switch completion	76	0	578	76	0.0
process startup	86	0	559	65	0.0
db file parallel write	368,945	0	507	0	2.2
library cache pin	271	0	501	18	0.0
PX Deq: Signal ACK	178	36	328	18	0.0
LGWR wait for redo copy	5,140	65	315	1	0.0
PX Deq: Parse Reply	389	0	191	5	0.0
refresh controlfile command	801	0	173	2	0.0
PX Deq: Join ACK	436	0	156	4	0.0
local write wait	24	0	140	58	0.0
SQL*Net break/reset to clien	3,539	0	102	0	0.0
log file single write	34	0	29	9	0.0
SQL*Net message to dblink	19,355	0	19	0	0.1
file identify	153	0	19	1	0.0
PX Deq: Msg Fragment	298	0	18	1	0.0
direct path read	3,110	0	5	0	0.0
direct path write	1,049	0	4	0	0.0
PX Deq: Table Q qref	74	0	2	0	0.0

Event	Waits	Timeouts	Total Wait Time (cs)	Avg wait (ms)	Waits /txn
SQL*Net break/reset to dblin	10	0	1	1	0.0
single-task message	1	0	1	10	0.0
PX Deq Credit: need buffer	19	0	0	0	0.0
buffer deadlock	6	6	0	0	0.0
SQL*Net message from client	22,245,825	0	#####	1478	133.2
PX Idle Wait	31,957	31,475	6,474,543	2026	0.2
SQL*Net more data from clien	391,640	0	1,910,922	49	2.3
SQL*Net message to client	22,245,863	0	19,322	0	133.2

Background Wait Events for DB: dba01 Instance: dba01 Snaps: 2 -4
-> ordered by wait time desc, waits desc (idle events last)

Event	Waits	Timeouts	Total Wait Time (cs)	Avg wait (ms)	Waits /txn
log file parallel write	319,796	0	88,155	3	1.9
latch free	10,909	10,905	17,404	16	0.1
log file sequential read	32,674	0	3,435	1	0.2
control file parallel write	5,017	0	2,784	6	0.0
db file sequential read	1,464	0	1,106	8	0.0
file open	4,422	0	922	2	0.0
db file scattered read	333	0	513	15	0.0
db file parallel write	368,945	0	507	0	2.2
LGWR wait for redo copy	5,140	65	315	1	0.0
enqueue	3	0	258	860	0.0
control file sequential read	3,336	0	187	1	0.0
log file single write	34	0	29	9	0.0
file identify	85	0	17	2	0.0
direct path write	969	0	3	0	0.0
direct path read	1,258	0	2	0	0.0
rdbms ipc message	717,126	38,588	15,344,729	214	4.3
pmon timer	4,709	4,646	1,437,589	3053	0.0
smon timer	47	46	1,414,668	#####	0.0

SQL ordered by Gets for DB: dba01 Instance: dba01 Snaps: 2 -4

-> End Buffer Gets Threshold: 10000

-> Note that resources reported for PL/SQL includes the resources used by all SQL statements called within the PL/SQL code. As individual SQL statements are also reported, it is possible and valid for the summed total % to exceed 100

Buffer Gets	Executions	Gets per Exec	% Total	Hash Value
174,229,988	12,762	13,652.2	28.0	4180317975
SELECT "EXT_ACCOUNT_LINK"."COMPASS_ID", "EXT_ACCOUNT_LINK"."EXTACCT_IDENTIFIER", "EXT_ACCOUNT_LINK"."EXTSYS_ID" FROM "EXT_ACCOUNT_LINK" WHERE ("EXT_ACCOUNT_LINK"."COMPASS_ID" = :as_compass_id) AND ("EXT_ACCOUNT_LINK"."EXTSYS_ID" = :as_extsys_id) AND				
21,475,679	1,253	17,139.4	3.5	3671942761
select /*+ INDEX(CA compass_account_ndx6) */ ROWIDTOCHAR(CA.rowid) ,ML.MARKET_ID ,NVL(RTRIM(AD.ADDRESS_APT_DESIGNATOR),' ') ,NVL(RTRIM(AD.ADDRESS_APT_NUM),' ') ,NVL(RTRIM(AD.ADDRESS_ATTENTION),' ') ,NVL(RTRIM(AD.ADDRESS_CITY),' ') ,NVL(RTRIM(AD.ADDRESS_COUNTRY),' ') ,NVL(RTRIM(AD.ADDRESS_FORMAT),' ') ,NVL(RTRIM(AD				
17,919,523	365	49,094.6	2.9	405379982
(SELECT "INV_MOVEMENT_SUBLN_A"."MVMNT_LOCATION_ID", "INV_MOVEMENT_SUBLN_A"."ACTIVITY_TYPE", "INV_MOVEMENT_SUBLN_A"."MVMNT_OID", "INV_MOVEMENT_SUBLN_A"."MVMNT_LINE_SEQ", "INV_MOVEMENT_SUBLN_A"."MVMNT_SUBLINE_SEQ", "INV_MOVEMENT_SUBLN_A"."SYS_CREATION_DATE", "INV_MOVEMENT_SUBLN_A"."SYS_UPDATE_DATE", "INV_MOVEMENT_SUBLN				
12,562,692	3,703,535	3.4	2.0	3533983708
SELECT "SECURITY_INFO"."WINDOW", "SECURITY_INFO"."CONTROL", "SECURITY_INFO"."STATUS", "SECURITY_USERS"."PRIORITY" FROM "SECURITY_INFO", "SECURITY_USERS" WHERE ("SECURITY_USERS"."NAME" = "SECURITY_INFO"."USER_NAME") and ("SECURITY_INFO"."WINDOW" = :winna				
12,469,136	4	3,117,284.0	2.0	3966365613
SELECT Distinct("INV_PURCHASE_ORDER"."PURO_OID"), "INV_PURCHASE_ORDER"."STATUS", "INV_PURCHASE_ORDER"."SYS_CREATION_DATE", "INV_PURCHASE_ORDER"."OPERATOR_ID", "INV_PURCHASE_ORDER"."LOCATION_ID", "INV_PURCHASE_ORDER"."PURO_NEEDBY_DATE", "INV_PURCHASE				
11,632,584	10,460	1,112.1	1.9	3903562577
SELECT "SALES_REPRESENTATIVE"."SALESREP_ID", "SALES_REPRESENTATIVE"."SALESREP_NAME" FROM "LOCATION_SALESREP_LINK", "SALES_REPRESENTATIVE" WHERE ("LOCATION_SALESREP_LINK"."SALESREP_ID" = "SALES_REPRESENTATIVE"."SALESREP_ID") and ("LOCATION_SALESREP_LINK"."LOCATION_ID" = :as_l				

```

SQL ordered by Gets for DB: dba01 Instance: dba01 Snaps: 2 -4
-> End Buffer Gets Threshold: 10000
-> Note that resources reported for PL/SQL includes the resources used by
    all SQL statements called within the PL/SQL code. As individual SQL
    statements are also reported, it is possible and valid for the summed
    total % to exceed 100
    Buffer Gets      Executions  Gets per Exec  % Total  Hash Value
-----
6,235,002          2    3,117,501.0      1.0    1491625703
    SELECT Distinct("INV_PURCHASE_ORDER"."PURO_OID"),      "INV_PUR
CHASE_ORDER"."STATUS",      "INV_PURCHASE_ORDER"."SYS_CR
EATION_DATE",      "INV_PURCHASE_ORDER"."OPERATOR_ID",
      "INV_PURCHASE_ORDER"."LOCATION_ID",      "IN
V_PURCHASE_ORDER"."PURO_NEEDBY_DATE",      "INV_PURCHASE
5,768,648          47,284      122.0      0.9    3627000592
    SELECT "MARKET"."MARKET_ID" ,      "MARKET"."MARKET_NAME
"      FROM "MARKET" ,      "MARKET_POLICY"      WHERE ( "MAR
KET"."MARKET_ID" = "MARKET_POLICY"."MARKET_ID" ) and      (
( "MARKET"."EXTSYS_ID" = DECODE( :as_extsys_id, '*', market.exts
ys_id, :as_extsys_id ) ) and      ( "MARKET_POLICY"."MKTPOL_
5,432,674          1,278      4,250.9      0.9    1349835727
    SELECT distinct "SECURITY_USERS"."NAME", "SECURITY_USERS"."DESC
RIPTION"      FROM "LOCATION", "SECURITY_USERS", "WORKSTATION
_SETTINGS", "USER_MARKET_LINK"      WHERE ( "WORKSTATION_SETT
INGS"."WS_HOSTNAME" = :as_hostname )      AND ( "WORKSTATION_SETT
INGS"."LOCATION_ID" = "LOCATION"."LOCATION_ID" )      AND ( "LOCA
4,069,186          13,253      307.0      0.7    2947143525
    SELECT "ITEM_DEFINITION"."ITEM_ID" ,      "ITEM_DEFINITI
ON"."SYS_CREATION_DATE" ,      "ITEM_DEFINITION"."SYS_UPDAT
E_DATE" ,      "ITEM_DEFINITION"."OPERATOR_ID" ,
"ITEM_DEFINITION"."APPLICATION_ID" ,      "ITEM_DEFINITION"
."DL_SERVICE_CODE" ,      "ITEM_DEFINITION"."DL_UPDATE_STAM
4,056,847          23      176,384.7      0.7    2734622026
DECLARE job BINARY_INTEGER := :job; next_date DATE := :mydate;
broken BOOLEAN := FALSE; BEGIN declare rc binary_integer; begin
rc := sys.dbms_defer_sys.push(destination=>'dba01.WORLD', stop
_on_error=>TRUE, execution_seconds=>600, delay_seconds=>180, par
allelism=>1); end; :mydate := next_date; IF broken THEN :b := 1;
3,873,264          28      138,330.9      0.6    3293673928
    SELECT DISTINCT "VENDOR"."VENDOR_ID", "VENDOR"."VENDOR_NAME", "V
ENDOR"."VENDOR_SDESC", "VENDOR"."VENDOR_TEL_NO", "VENDOR"."VENDO
R_FAX_NO", "VENDOR"."VENDOR_CONTACT_NAME" FROM "VENDOR", "VENDOR
_ITEM" WHERE ( "VENDOR"."VENDOR_ID" = "VENDOR_ITEM"."VENDOR_ID"
) ORDER BY "VENDOR"."VENDOR_ID" ASC, "VENDOR"."VENDOR_NAME" ASC
3,845,686          451,430      8.5      0.6    3709651884
insert into SYSTEM.DEF$_AQCALL (q_name, msgid, corrid, priority
, state, delay, expiration, time_manager_info, local_order_no,
chain_no, enq_time, step_no, enq_uid, enq_tid, retry_count, e
xception_qschema, exception_queue, recipient_key, dequeue_msg
-----

```

SQL ordered by Reads for DB: dba01 Instance: dba01 Snaps: 2 -4

-> End Disk Reads Threshold: 1000

Physical Reads	Executions	Reads per Exec	% Total	Hash Value
8,087,240	1,253	6,454.3	21.2	3671942761
<pre>select /*+ INDEX(CA compass_account_ndx6) */ ROWIDTOCHAR(CA. rowid) ,ML.MARKET_ID ,NVL(RTRIM(AD.ADDRESS_APT_DESIGNATOR),' ') ,NVL(RTRIM(AD.ADDRESS_APT_NUM),' ') ,NVL(RTRIM(AD.ADDRESS_ATTEN TION),' ') ,NVL(RTRIM(AD.ADDRESS_CITY),' ') ,NVL(RTRIM(AD.ADDRES S_COUNTRY),' ') ,NVL(RTRIM(AD.ADDRESS_FORMAT),' ') ,NVL(RTRIM(AD</pre>				
1,171,800	241	4,862.2	3.1	2416933180
<pre>select ROWIDTOCHAR(CA.rowid) ,ML.MARKET_ID ,NVL(RTRIM(AD.ADDRES S_APT_DESIGNATOR),' ') ,NVL(RTRIM(AD.ADDRESS_APT_NUM),' ') ,NVL(RTRIM(AD.ADDRESS_ATTENTION),' ') ,NVL(RTRIM(AD.ADDRESS_CITY),' ') ,NVL(RTRIM(AD.ADDRESS_COUNTRY),' ') ,NVL(RTRIM(AD.ADDRESS_FORM AT),' ') ,NVL(RTRIM(AD.ADDRESS_PRIMARY_LN),' ') ,NVL(RTRIM(AD.AD</pre>				
797,761	364	2,191.7	2.1	2188104695
<pre>SELECT "CAS_RESULT"."CAS_COMPASS_ID" FROM "CAS_RESULT" WHERE "CAS_RESULT"."CAS_COMPASS_ID" = :as_CompassID</pre>				
719,730	36	19,992.5	1.9	596848251
<pre>SELECT "INV_MOVEMENT"."MVMNT_OID" , "INV_MOVEMENT"."MVMNT_S TATUS" , "INV_MOVEMENT"."MVMNT_LOCATION_ID" '-' "INV_MOVEMENT"."ACTIVITY_TYPE" '-' lpad("INV_MOVEMENT"."M VMNT_OID" ,6 , '0') trans_id, "INV_MOVEMENT"."TRANS</pre>				
458,074	3,445	133.0	1.2	4195953210
<pre>SELECT "POS_INVOICE_LINE"."LOCATION_ID" , "POS_INV OICE_LINE"."POSINV_ACTIVITY" , "POS_INVOICE_LINE"."P OSINV_ID" , "POS_INVOICE_LINE"."ACTIVITY_ID" , "POS_INVOICE_LINE"."POSINVLN_ID" , "POS_INVOI CE_LINE"."SYS_CREATION_DATE" , "POS_INVOICE_LINE"."S</pre>				
435,057	206	2,111.9	1.1	1396905682
<pre>SELECT TO_CHAR(max(CAS_CREDIT_DATE), 'MM/DD/YYYY HH24:MI:SS') FROM CAS_RESULT WHERE CAS_COMPASS_ID = :as_CompassID</pre>				
432,358	206	2,098.8	1.1	464556795
<pre>SELECT "CAS_RESULT"."CAS_ORDER_NUM" , "CAS_RESULT" ."SYS_CREATION_DATE" , "CAS_RESULT"."SYS_UPDATE_DATE " , "CAS_RESULT"."OPERATOR_ID" , "CAS_RE SULT"."APPLICATION_ID" , "CAS_RESULT"."DL_SERVICE_CO DE" , "CAS_RESULT"."DL_UPDATE_STAMP" , "</pre>				
348,312	1	348,312.0	0.9	3411194506
<pre>SELECT DISTINCT "POS_TRANSACTION"."LOCATION_ID" , "POS_TRANSACTION"."POSTRANS_PAYMENT_TYPE" , "POS_TRA NSACTION"."POSTRANS_ID" , "POS_TRANSACTION"."COMPASS _ID" , "POS_TRANSACTION"."POSTRANS_EXTACCT_NUM" , "POS_TRANSACTION"."POSTRANS_DATE" , "POS_T</pre>				
333,311	22	15,150.5	0.9	3863558257
<pre>SELECT DISTINCT "POS_ORDER"."LOCATION_ID" , "POS_O RDER"."ORDER_ACTIVITY" , "POS_ORDER"."ORDER_ID" , "POS_ORDER"."COMPASS_ID" , "ORDER_FULFILLM ENT"."ORDFILL_ID" , "ORDER_FULFILLMENT"."SYS_CREATIO</pre>				

SQL ordered by Reads for DB: dba01 Instance: dba01 Snaps: 2 -4
 -> End Disk Reads Threshold: 1000

Physical Reads	Executions	Reads per Exec	% Total	Hash Value
8,087,240	1,253	6,454.3	21.2	3671942761
<pre>select /*+ INDEX(CA compass_account_ndx6) */ ROWIDTOCHAR(CA. rowid) ,ML.MARKET_ID ,NVL(RTRIM(AD.ADDRESS_APT_DESIGNATOR),' ') ,NVL(RTRIM(AD.ADDRESS_APT_NUM),' ') ,NVL(RTRIM(AD.ADDRESS_ATTEN TION),' ') ,NVL(RTRIM(AD.ADDRESS_CITY),' ') ,NVL(RTRIM(AD.ADDRES S_COUNTRY),' ') ,NVL(RTRIM(AD.ADDRESS_FORMAT),' ') ,NVL(RTRIM(AD</pre>				
1,171,800	241	4,862.2	3.1	2416933180
<pre>select ROWIDTOCHAR(CA.rowid) ,ML.MARKET_ID ,NVL(RTRIM(AD.ADDRES S_APT_DESIGNATOR),' ') ,NVL(RTRIM(AD.ADDRESS_APT_NUM),' ') ,NVL(RTRIM(AD.ADDRESS_ATTENTION),' ') ,NVL(RTRIM(AD.ADDRESS_CITY),' ') ,NVL(RTRIM(AD.ADDRESS_COUNTRY),' ') ,NVL(RTRIM(AD.ADDRESS_FORM AT),' ') ,NVL(RTRIM(AD.ADDRESS_PRIMARY_LN),' ') ,NVL(RTRIM(AD.AD</pre>				
797,761	364	2,191.7	2.1	2188104695
<pre>SELECT "CAS_RESULT"."CAS_COMPASS_ID" FROM "CAS_RESULT" WHERE "CAS_RESULT"."CAS_COMPASS_ID" = :as_CompassID</pre>				
719,730	36	19,992.5	1.9	596848251
<pre>SELECT "INV_MOVEMENT"."MVMNT_OID" , "INV_MOVEMENT"."MVMNT_S TATUS" , "INV_MOVEMENT"."MVMNT_LOCATION_ID" '-' "INV_MOVEMENT"."ACTIVITY_TYPE" '-' lpad("INV_MOVEMENT"."M VMNT_OID" ,6 , '0') trans_id, "INV_MOVEMENT"."TRANS _OUT_STS" , "INV_MOVEMENT"."SYS_CREATION_DATE" ,</pre>				
458,074	3,445	133.0	1.2	4195953210
<pre>SELECT "POS_INVOICE_LINE"."LOCATION_ID" , "POS_INV OICE_LINE"."POSINV_ACTIVITY" , "POS_INVOICE_LINE"."P OSINV_ID" , "POS_INVOICE_LINE"."ACTIVITY_ID" , "POS_INVOICE_LINE"."POSINVLN_ID" , "POS_INVOI CE_LINE"."SYS_CREATION_DATE" , "POS_INVOICE_LINE"."S</pre>				
435,057	206	2,111.9	1.1	1396905682
<pre>SELECT TO_CHAR(max(CAS_CREDIT_DATE), 'MM/DD/YYYY HH24:MI:SS') FROM CAS_RESULT WHERE CAS_COMPASS_ID = :as_CompassID</pre>				
432,358	206	2,098.8	1.1	464556795
<pre>SELECT "CAS_RESULT"."CAS_ORDER_NUM" , "CAS_RESULT" ."SYS_CREATION_DATE" , "CAS_RESULT"."SYS_UPDATE_DATE " , "CAS_RESULT"."OPERATOR_ID" , "CAS_RE SULT"."APPLICATION_ID" , "CAS_RESULT"."DL_SERVICE_CO DE" , "CAS_RESULT"."DL_UPDATE_STAMP" , "</pre>				
348,312	1	348,312.0	0.9	3411194506
<pre>SELECT DISTINCT "POS_TRANSACTION"."LOCATION_ID" , "POS_TRANSACTION"."POSTRANS_PAYMENT_TYPE" , "POS_TRA NSACTION"."POSTRANS_ID" , "POS_TRANSACTION"."COMPASS _ID" , "POS_TRANSACTION"."POSTRANS_EXTACCT_NUM" , "POS_TRANSACTION"."POSTRANS_DATE" , "POS_T</pre>				
333,311	22	15,150.5	0.9	3863558257
<pre>SELECT DISTINCT "POS_ORDER"."LOCATION_ID" , "POS_O RDER"."ORDER_ACTIVITY" , "POS_ORDER"."ORDER_ID" , "POS_ORDER"."COMPASS_ID" , "ORDER_FULFILLM ENT"."ORDFILL_ID" , "ORDER_FULFILLMENT"."SYS_CREATIO</pre>				

SQL ordered by Reads for DB: dba01 Instance: dba01 Snaps: 2 -4
 -> End Disk Reads Threshold: 1000

Physical Reads	Executions	Reads per Exec	% Total	Hash Value

N_DATE",	"ORDER_FULFILLMENT".	ORDFILL_TRACKING",		
327,408	145	2,258.0	0.9	1606276826
SELECT "CAS_RESULT"."CAS_COMPASS_ID" , "CAS_RESULT"				
."CAS_RESULT_TYPE" FROM "CAS_RESULT" WHERE (:as_Compas				
SID = "CAS_RESULT"."CAS_COMPASS_ID")				
320,452	7,303	43.9	0.8	2440585122
SELECT "CHECK_INOUT_LINE"."LOCATION_ID" , "CHECK_IN				
OUT_LINE"."CHKINOUT_ACTIVITY" , "CHECK_INOUT_LINE"."CH				
KINOUT_ID" , "CHECK_INOUT_LINE"."CHKINOUTL_ID" ,				
"CHECK_INOUT_LINE"."SYS_CREATION_DATE" , "CHECK_I				
NOUT_LINE"."SALESREP_ID" , "CHECK_INOUT_LINE"."ITEM_ID				
281,487	782	360.0	0.7	1321409306
SELECT "POS_ORDER_SUBLINE"."LOCATION_ID" , "POS_ORD				
ER_SUBLINE"."ORDER_ACTIVITY" , "POS_ORDER_SUBLINE"."OR				
DER_ID" , "POS_ORDER_SUBLINE"."ORDERLN_ID" ,				
"POS_ORDER_SUBLINE"."ORDERSLN_ID" , "POS_ORDER_SUBLIN				
E"."SYS_CREATION_DATE" , "POS_ORDER_SUBLINE"."SERIAL_I				
280,075	18	15,559.7	0.7	3733389583
SELECT "ORDER_FULFILLMENT"."LOCATION_ID" , "ORDER_F				
ULFILLMENT"."ORDFILL_ID" , "ORDER_FULFILLMENT"."ORDFIL				
L_STATUS" , "ORDER_FULFILLMENT_SUBLINE"."SERIAL_ID" ,				
"ORDER_FULFILLMENT"."ORDFILL_LOCATIONID" , "				
ORDER_FULFILLMENT"."ORDFILL_ORDERID" , "ORDER_FULFILLM				
259,743	11	23,613.0	0.7	2742230160
SELECT /*+ ORDERED NO_EXPAND USE_NL(A2) */ A1.C1 C0,NVL(A1.C5,'				
') C1,NVL(A1.C2,' ') C2,NVL(A1.C3,' ') C3,NVL(A1.C4,' ') C4,NVL(
A1.C6,' ') C5,NVL(A1.C10,0) C6 FROM :Q2112000 A1,(SELECT /*+ NO_				
EXPAND ROWID(A3) */ A3."COMPASS_ID" C0,A3."POSINV_STATUS" C1,A3.				
"POSINV_REMAINING_AR_AMT" C2,A3."MARKET_ID" C3 FROM "DEVOWNER".				
192,141	13,253	14.5	0.5	2947143525
SELECT "ITEM_DEFINITION"."ITEM_ID" , "ITEM_DEFINITI				
ON"."SYS_CREATION_DATE" , "ITEM_DEFINITION"."SYS_UPDAT				

SQL ordered by Executions for DB: dba01 Instance: dba01 Snaps: 2 -4
 -> End Executions Threshold: 100

Executions	Rows Processed	Rows per Exec	Hash Value
3,703,535	370,614	0.1	3533983708
SELECT "SECURITY_INFO"."WINDOW" , "SECURITY_INFO"." CONTROL" , "SECURITY_INFO"."STATUS" , "SECUR ITY_USERS"."PRIORITY" FROM "SECURITY_INFO" , "SECU RITY_USERS" WHERE ("SECURITY_USERS"."NAME" = "SECURITY_INFO ". "USER_NAME") and ("SECURITY_INFO"."WINDOW" = :winna 873,612 869,514 1.0 203525044			
873,612	869,514	1.0	203525044
SELECT "LOCATION"."MARKET_ID" FROM "LOCATION" WHERE "LOCATION". " LOCATION_ID" =:1 652,994 652,993 1.0 2182723903			
652,994	652,993	1.0	2182723903
SELECT SYSDATE FROM SYS.DUAL 536,404 536,360 1.0 3258376580			
536,404	536,360	1.0	3258376580
SELECT "MARKET"."MARKET_ID" , "MARKET"."SYS_CREATIO N_DATE" , "MARKET"."SYS_UPDATE_DATE" , "MARK ET"."OPERATOR_ID" , "MARKET"."APPLICATION_ID" , "MARKET"."DL_SERVICE_CODE" , "MARKET"."DL_UPDATE_S TAMP" , "MARKET"."COMPANY_ID" , "MARKET"."MA 532,476 532,473 1.0 1645188330			
532,476	532,473	1.0	1645188330
SELECT SYSDATE FROM DUAL 451,430 451,430 1.0 3709651884			
451,430	451,430	1.0	3709651884
insert into SYSTEM.DEF\$AQCALL (q_name, msgid, corrid, priority , state, delay, expiration, time_manager_info, local_order_no, chain_no, enq_time, step_no, enq_uid, enq_tid, retry_count, e xception_qschema, exception_queue, recipient_key, dequeue_msggi d, user_data) values (:1, :2, :3, :4, :5, :6, :7, :8, :9, :10, 298,348 10,263 0.0 2207347125			
298,348	10,263	0.0	2207347125
SELECT "SECURITY_INFO"."WINDOW" , "SECURITY_INFO"." CONTROL" , "SECURITY_INFO"."STATUS" , "SECUR ITY_USERS"."PRIORITY" FROM "SECURITY_INFO" , "SECU RITY_USERS" WHERE ("SECURITY_USERS"."NAME" = "SECURITY_INFO ". "USER_NAME") and (("SECURITY_INFO"."WINDOW" = :as_ 230,434 38,386 0.2 3175436978			
230,434	38,386	0.2	3175436978
SELECT "TAX_ASSOCIATIONS"."STATE_CODE" , "TAX_ASSOC IATIONS"."MARKET_ID" , "TAX_ASSOCIATIONS"."LOCATION_ID " , "TAX_ASSOCIATIONS"."FUNCTION_ID" , "TAX_ ASSOCIATIONS"."SYS_CREATION_DATE" , "TAX_ASSOCIATIONS" ". "SYS_UPDATE_DATE" , "TAX_ASSOCIATIONS"."OPERATOR_ID" 99,743 99,732 1.0 3174277601			
99,743	99,732	1.0	3174277601
SELECT "ITEM_POLICY"."ITEM_ID" , "ITEM_POLICY"."EF FECTIVE_DATE" , "ITEM_POLICY"."SYS_CREATION_DATE" , "ITEM_POLICY"."SYS_UPDATE_DATE" , "ITEM_ POLICY"."OPERATOR_ID" , "ITEM_POLICY"."APPLICATION_I D" , "ITEM_POLICY"."DL_SERVICE_CODE" , " 98,607 98,605 1.0 1661068441			
98,607	98,605	1.0	1661068441
SELECT "WHSE_ITEM"."ITEM_ID" , "WHSE_ITEM"."WHSE_ID " , "WHSE_ITEM"."TRACKING_IND" FROM "WHSE_ITEM"			

SQL ordered by Executions for DB: dba01 Instance: dba01 Snaps: 2 -4
 -> End Executions Threshold: 100

Executions	Rows Processed	Rows per Exec	Hash Value
WHERE ("WHSE_ITEM"."ITEM_ID" = :as_item) and ("WHSE_ITEM"."WHSE_ID" = :as_warehouse)			
98,215	96,727	1.0	3239320604
SELECT "ITEM_WAC_ACCUMULATOR"."WAC_LOCATION_ID" , "ITEM_WAC_ACCUMULATOR"."ITEM_ID" , "ITEM_WAC_ACCUMULATOR"."LAST_UPDATE_DATE" , "ITEM_WAC_ACCUMULATOR"."SYS_CREATION_DATE" , "ITEM_WAC_ACCUMULATOR"."SYS_UPDATE_DATE" , "ITEM_WAC_ACCUMULATOR"."OPERATOR_ID" ,			
95,244	90,281	0.9	1418052366
SELECT "ITEM_DEFINITION"."ITEM_ID" , "ITEM_DEFINITION"."SYS_CREATION_DATE" , "ITEM_DEFINITION"."SYS_UPDATE_DATE" , "ITEM_DEFINITION"."OPERATOR_ID" , "ITEM_DEFINITION"."APPLICATION_ID" , "ITEM_DEFINITION"."DL_SERVICE_CODE" , "ITEM_DEFINITION"."DL_UPDATE_STAMP"			
87,594	87,594	1.0	254026706
update SYSTEM.DEF\$_AQCALL set dscn = :1, cscn = :2 where rowid = :3			
87,590	448,047	5.1	568505420
SELECT /*+ ORDERED USE_NL(P) */ aq.step_no, P.sname, P.ename, aq. chain_no, aq.user_data FROM system.def\$aqcall aq, system.repca t\$_repprop P WHERE aq.enq_tid = :tid AND P.recipient_key = aq.r ecipient_key AND P.how = 1 AND P.dblink = :dest ORDER BY aq.enq tid, aq.step_no			
86,861	75,704	0.9	1084722568
SELECT "SERIAL_ITEM_INV"."SERIAL_NUMBER" , "SERIAL_ITEM_INV"."APPLICATION_ID" , "SERIAL_ITEM_INV"."DL_SERVICE_CODE" , "SERIAL_ITEM_INV"."DL_UPDATE_STAMP" , "SERIAL_ITEM_INV"."ACT_ISSUE_DATE" , "SERIAL_ITEM_INV"."ITEM_ID" , "SERIAL_ITEM_INV"."POOL" ,			
86,579	0	0.0	256517786
delete from system.def\$_lob where (enq_tid = :1)			
86,579	0	0.0	368735379
delete from system.def\$aqcall where (enq_tid = :1)			
86,579	0	0.0	1565159147
delete from system.def\$_calldest where (enq_tid = :1)			
75,981	75,978	1.0	3356946589
SELECT "MARKET_POLICY"."MARKET_ID" , "MARKET_POLICY"			

SQL ordered by Version Count for DB: dba01 Instance: dba01 Snaps: 2 -4
 -> End Version Count Threshold: 20

Version	Count	Executions	Hash Value
31	1,155	1668540021	
select i.obj# from ind\$ i, obj\$ o where i.obj# = o.obj# and i.bo# = :1 and o.name = :2			

Instance Activity Stats for DB: dba01 Instance: dba01 Snaps: 2 -4

Statistic	Total	per Second	per Trans
CPU used by this session	93,508,307	6,495.4	559.8
CPU used when call started	5,274,242	366.4	31.6
CR blocks created	213,017	14.8	1.3
DBWR buffers scanned	12,552,443	871.9	75.2
DBWR checkpoint buffers written	2,260,416	157.0	13.5
DBWR checkpoints	50	0.0	0.0
DBWR free buffers found	12,423,954	863.0	74.4
DBWR lru scans	132,887	9.2	0.8
DBWR make free requests	146,146	10.2	0.9
DBWR revisited being-written buff	7	0.0	0.0
DBWR summed scan depth	12,552,443	871.9	75.2
DBWR transaction table writes	128,583	8.9	0.8
DBWR undo block writes	299,657	20.8	1.8
DFO trees parallelized	31	0.0	0.0
PX local messages rcv'd	192,660	13.4	1.2
PX local messages sent	192,431	13.4	1.2
Parallel operations downgraded 1	0	0.0	0.0
Parallel operations downgraded 25	6	0.0	0.0
Parallel operations downgraded 50	0	0.0	0.0
Parallel operations downgraded to	2	0.0	0.0
Parallel operations not downgrade	25	0.0	0.0
SQL*Net roundtrips to/from client	22,221,687	1,543.6	133.0
SQL*Net roundtrips to/from dblink	19,350	1.3	0.1
background checkpoints completed	17	0.0	0.0
background checkpoints started	17	0.0	0.0
background timeouts	45,484	3.2	0.3
branch node splits	16	0.0	0.0
buffer is not pinned count	448,781,851	31,174.1	2,686.8
buffer is pinned count	7,069,302,643	491,060.2	42,322.3
bytes received via SQL*Net from c	7,237,120,160	502,717.4	43,327.0
bytes received via SQL*Net from d	1,112,038	77.3	6.7
bytes sent via SQL*Net to client	6,529,057,411	453,532.8	39,088.0
bytes sent via SQL*Net to dblink	243,692,274	16,927.8	1,458.9
calls to get snapshot scn: kcmgss	12,967,019	900.7	77.6
calls to kcmgas	162,786	11.3	1.0
calls to kcmgcs	237,905	16.5	1.4
change write time	34,321	2.4	0.2
cleanouts and rollbacks - consist	193,015	13.4	1.2
cleanouts only - consistent read	119,182	8.3	0.7
cluster key scan block gets	1,203,291	83.6	7.2
cluster key scans	458,032	31.8	2.7
commit cleanout failures: block 1	103,076	7.2	0.6
commit cleanout failures: buffer	48,540	3.4	0.3
commit cleanout failures: callbac	1,958	0.1	0.0
commit cleanout failures: cannot	1,564	0.1	0.0
commit cleanouts	1,252,949	87.0	7.5
commit cleanouts successfully com	1,097,811	76.3	6.6

Instance Activity Stats for DB: dba01 Instance: dba01 Snaps: 2 -4

Statistic	Total	per Second	per Trans
consistent changes	704,266	48.9	4.2
consistent gets	612,237,120	42,528.3	3,665.3
current blocks converted for CR			
cursor authentications	9,490	0.7	0.1
data blocks consistent reads - un	685,485	47.6	4.1
db block changes	10,230,141	710.6	61.3
db block gets	17,972,677	1,248.5	107.6
deferred (CURRENT) block cleanout	251,171	17.5	1.5
dirty buffers inspected	99,512	6.9	0.6
enqueue conversions	20,795	1.4	0.1
enqueue releases	676,178	47.0	4.1
enqueue requests	690,640	48.0	4.1
enqueue timeouts	14,029	1.0	0.1
enqueue waits	14,033	1.0	0.1
exchange deadlocks	6	0.0	0.0
execute count	10,828,737	752.2	64.8
free buffer inspected	133,072	9.2	0.8
free buffer requested	30,749,604	2,136.0	184.1
hot buffers moved to head of LRU	3,656,215	254.0	21.9
immediate (CR) block cleanout app	312,198	21.7	1.9
immediate (CURRENT) block cleanou	77,797	5.4	0.5
index fast full scans (direct rea	0	0.0	0.0
index fast full scans (full)	1,310	0.1	0.0
index fast full scans (rowid rang	0	0.0	0.0
leaf node splits	5,576	0.4	0.0
logons cumulative	9,049	0.6	0.1
messages received	1,077,361	74.8	6.5
messages sent	1,077,363	74.8	6.5
no buffer to keep pinned count	106,061,351	7,367.4	635.0
no work - consistent read gets	279,634,470	19,424.5	1,674.1
opened cursors cumulative	226,594	15.7	1.4
opened cursors current			
parse count (hard)	26,557	1.8	0.2
parse count (total)	9,081,240	630.8	54.4
parse time cpu	110,501	7.7	0.7
parse time elapsed	153,495	10.7	0.9
physical reads	38,138,047	2,649.2	228.3
physical reads direct	8,131,985	564.9	48.7
physical writes	2,617,275	181.8	15.7
physical writes direct	130,381	9.1	0.8
physical writes non checkpoint	1,051,637	73.1	6.3
pinned buffers inspected	1,108	0.1	0.0
prefetched blocks	13,426,209	932.6	80.4
prefetched blocks aged out before	3,043	0.2	0.0
process last non-idle time	#####	#####	#####
queries parallelized	31	0.0	0.0
recursive calls	2,909,676	202.1	17.4
recursive cpu usage	142,030	9.9	0.9

Instance Activity Stats for DB: dba01 Instance: dba01 Snaps: 2 -4

Statistic	Total	per Second	per Trans
redo blocks written	2,054,773	142.7	12.3
redo buffer allocation retries	71	0.0	0.0
redo entries	5,288,608	367.4	31.7
redo log space requests	76	0.0	0.0
redo log space wait time	578	0.0	0.0
redo ordering marks	0	0.0	0.0
redo size	1,911,814,688	132,801.8	11,445.6
redo synch time	62,483	4.3	0.4
redo synch writes	148,848	10.3	0.9
redo wastage	159,124,280	11,053.4	952.6
redo write time	88,292	6.1	0.5
redo writer latching time	327	0.0	0.0
redo writes	319,838	22.2	1.9
rollback changes - undo records a	19,210	1.3	0.1
rollbacks only - consistent read	24,812	1.7	0.2
rows fetched via callback	101,187,960	7,028.9	605.8
session connect time	#####	#####	#####
session cursor cache count	2,727	0.2	0.0
session cursor cache hits	8,331,445	578.7	49.9
session logical reads	622,163,159	43,217.8	3,724.8
session pga memory	10,422,061,992	723,955.4	62,394.5
session pga memory max	9,945,785,568	690,871.5	59,543.1
session uga memory	66,411,464	4,613.2	397.6
session uga memory max	1,202,141,544	83,505.3	7,196.9
sorts (disk)	35	0.0	0.0
sorts (memory)	4,172,017	289.8	25.0
sorts (rows)	37,870,908	2,630.7	226.7
summed dirty queue length	18,049	1.3	0.1
switch current to new buffer			
table fetch by rowid	3,742,857,902	259,992.9	22,407.6
table fetch continued row	14,854	1.0	0.1
table scan blocks gotten	21,913,768	1,522.2	131.2
table scan rows gotten	972,327,781	67,541.5	5,821.1
table scans (direct read)	2,881	0.2	0.0
table scans (long tables)	3,986	0.3	0.0
table scans (rowid ranges)	#####	#####	#####
table scans (short tables)	1,460,315	101.4	8.7
total file opens	78,945	5.5	0.5
transaction rollbacks	1,609	0.1	0.0
transaction tables consistent rea	60	0.0	0.0
transaction tables consistent rea	20,137	1.4	0.1
user calls	22,243,790	1,545.1	133.2
user commits	146,266	10.2	0.9
user rollbacks	20,769	1.4	0.1
write clones created in backgroun	2,222	0.2	0.0
write clones created in foregroun	206,401	14.3	1.2

Tablespace IO Stats for DB: dba01 Instance: dba01 Snaps: 2 -4
 ->ordered by IOs (Reads + Writes) desc

Tablespace

	Av Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Av Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
TBSL05	4,585,443	319	2.2	1.0	62,533	4	66,891	6.4
TBSL04	3,403,037	236	11.9	1.0	61,022	4	34,308	6.2
TBSM51	1,671,136	116	5.5	6.0	116,667	8	23,664	5.7
TBSL03	1,513,824	105	12.4	6.3	27,934	2	12,410	12.0
TBS0A	420,331	29	7.6	1.0	774,704	54	10,771	9.3
TBSL06	1,048,150	73	13.7	1.0	25,361	2	20	10.5
TBSM02	1,022,425	71	6.8	2.7	29,989	2	3,148	6.3
TBSM01	911,925	63	5.6	2.1	33,137	2	1,751	3.1
TBSM53	483,197	34	10.4	1.0	399,383	28	204	14.3
TBSS01	610,371	42	73.1	4.3	105,236	7	5,701	316.6
TBSS51	368,345	26	12.4	1.0	34,239	2	814	8.5
TBSL51	327,978	23	5.2	1.0	56,849	4	67	15.5
TBSL53	142,554	10	14.0	1.0	150,217	10	85	11.6
RBS1	27,336	2	7.2	1.0	209,775	15	224	1.4
TBSL52	135,013	9	23.6	1.0	98,892	7	1	0.0
RBS2	21,384	1	7.0	1.0	212,179	15	213	1.4
TBSL02	75,604	5	10.3	1.0	42,365	3	8	2.5
TBSM52	23,793	2	10.8	1.0	18,339	1	1	0.0
TBSL01	16,006	1	13.5	1.0	14,247	1	3	16.7
SYSTEM	17,443	1	8.6	2.7	6,757	0	30	1.0
TOOLS	11,865	1	2.4	1.3	1,499	0	1	0.0
TEMP1	3,770	0	0.0	22.3	5,267	0	0	0.0

Tablespace IO Stats for DB: dba01 Instance: dba01 Snaps: 2 -4
 ->ordered by IOs (Reads + Writes) desc

Tablespace

	Av Reads	Av Reads/s	Av Rd(ms)	Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
RBSLARGE2	428	0	6.4	1.0	3,445	0	6	1.7
RBSLARGE1	402	0	6.8	1.0	2,898	0	3	0.0
TBSSNP	291	0	11.6	3.8	17	0	0	0.0
USERS	104	0	8.2	1.0	38	0	0	0.0
TBS_BACK	86	0	16.2	1.0	17	0	0	0.0
PRECISE_TBS	17	0	0.0	1.0	17	0	0	0.0

File IO Stats for DB: dba01 Instance: dba01 Snaps: 2 -4

->ordered by Tablespace, File

Tablespace		Filename							
		Av	Av	Av	Av		Buffer	Av	Buf
Reads		Reads/s	Rd(ms)	Blks/Rd	Writes	Writes/s	Waits	Wt(ms)	
PRECISE_TBS			/dba01/ORADATA/dbf/precise0.dbf						
	17	0	0.0	1.0	17	0	0		
RBS1			/dba01/ORADATA/dbf/rbs1dba01_1.dbf						
	27,336	2	7.2	1.0	209,775	15	224	1.4	
RBS2			/dba01/ORADATA/dbf/rbs2dba01_1.dbf						
	21,384	1	7.0	1.0	212,179	15	213	1.4	
RBSLARGE1			/dba01/ORADATA/dbf/rbslarge1dba01_1.dbf						
	402	0	6.8	1.0	2,898	0	3	0.0	
RBSLARGE2			/dba01/ORADATA/dbf/rbslarge2dba01_1.dbf						
	428	0	6.4	1.0	3,445	0	6	1.7	
SYSTEM			/dba01/ORADATA/dbf/systemdba01_1.dbf						
	12,030	1	8.3	2.6	5,906	0	30	1.0	
			/dba01/ORADATA/dbf/systemdba01_2.dbf						
	5,413	0	9.2	3.0	851	0	0		
TBS0A			/dba01/ORADATA/dbf/tbs0adba01_1.dbf						
	27,610	2	8.3	1.0	62,721	4	253	4.2	
			/dba01/ORADATA/dbf/tbs0adba01_2.dbf						
	320,242	22	7.3	1.0	563,844	39	10,301	9.6	
			/dba01/ORADATA/dbf/tbs0adba01_3.dbf						
	46,608	3	8.6	1.0	97,013	7	62	2.1	
			/dba01/ORADATA/dbf/tbs0adba01_4.dbf						
	25,871	2	8.7	1.0	51,126	4	155	5.4	
TBSL01			/dba01/ORADATA/dbf/tbsl01dba01_2.dbf						
	3,959	0	13.7	1.0	17	0	0		
			/dba01/ORADATA/dbf/tbsl01dba01_3.dbf						
	3,209	0	13.9	1.0	792	0	0		
			/dba01/ORADATA/dbf/tbsl01dba01_4.dbf						
	4,981	0	11.4	1.0	13,421	1	3	16.7	
			/dba01/ORADATA/dbf/tbsl01dba01_1.dbf						
	3,857	0	15.9	1.0	17	0	0		
TBSL02			/dba01/ORADATA/dbf/tbsl02dba01_1.dbf						
	41,395	3	10.2	1.0	30,308	2	8	2.5	
			/dba01/ORADATA/dbf/tbsl02dba01_2.dbf						
	23,648	2	10.3	1.0	5,922	0	0		
			/dba01/ORADATA/dbf/tbsl02dba01_3.dbf						
	10,561	1	10.9	1.0	6,135	0	0		
TBSL03			/dba01/ORADATA/dbf/tbsl03dba01_1.dbf						
	577,583	40	15.2	6.5	933	0	2,479	13.6	
			/dba01/ORADATA/dbf/tbsl03dba01_2.dbf						
	571,901	40	13.8	6.7	26	0	3,297	14.0	
			/dba01/ORADATA/dbf/tbsl03dba01_3.dbf						
	364,340	25	5.8	5.4	26,975	2	6,634	10.4	

File IO Stats for DB: dba01 Instance: dba01 Snaps: 2 -4

->ordered by Tablespace, File

Tablespace			Filename							
			Av	Av	Av	Av		Buffer	Av	Buf
			Reads	Reads/s	Rd(ms)	Blks/Rd	Writes	Writes/s	Waits	Wt(ms)
TBSL04					/dba01/ORADATA/dbf/tbsl04dba01_1.dbf					
	854,941	59	21.3	1.0	833	0		3,095		6.2
					/dba01/ORADATA/dbf/tbsl04dba01_3.dbf					
	1,125,051	78	4.7	1.0	59,924	4		7,362		3.7
					/dba01/ORADATA/dbf/tbsl04dba01_2.dbf					
	1,423,045	99	11.9	1.0	265	0		23,851		6.9
TBSL05					/dba01/ORADATA/dbf/tbsl05dba01_3.dbf					
	1,543,543	107	1.8	1.0	61,292	4		32		5.0
					/dba01/ORADATA/dbf/tbsl05dba01_1.dbf					
	1,924,576	134	2.1	1.0	345	0		24,043		6.0
					/dba01/ORADATA/dbf/tbsl05dba01_2.dbf					
	1,117,324	78	2.8	1.0	896	0		42,816		6.6
TBSL06					/dba01/ORADATA/dbf/tbsl06dba01_2.dbf					
	121,166	8	10.6	1.0	25,107	2		13		6.2
					/dba01/ORADATA/dbf/tbsl06dba01_1.dbf					
	926,984	64	14.1	1.0	254	0		7		18.6
TBSL51					/dba01/ORADATA/dbf/tbsl512dba01_2.dbf					
	91,297	6	8.0	1.0	42,160	3		47		15.1
					/dba01/ORADATA/dbf/tbsl512dba01_3.dbf					
	150,685	10	3.0	1.0	13,326	1		18		15.6
					/dba01/ORADATA/dbf/tbsl512dba01_1.dbf					
	85,996	6	5.8	1.0	1,363	0		2		25.0
TBSL52					/dba01/ORADATA/dbf/tbsl52dba01_2.dbf					
	47,152	3	36.7	1.0	28,202	2		0		
					/dba01/ORADATA/dbf/tbsl52dba01_1.dbf					
	87,861	6	16.5	1.0	70,690	5		1		0.0
TBSL53					/dba01/ORADATA/dbf/tbsl53dba01_1.dbf					
	46,492	3	11.5	1.0	31,670	2		12		10.8
					/dba01/ORADATA/dbf/tbsl53dba01_2.dbf					
	31,676	2	10.8	1.0	19,058	1		2		15.0
					/dba01/ORADATA/dbf/tbsl53dba01_3.dbf					
	64,386	4	17.4	1.0	99,489	7		71		11.7
TBSM01					/dba01/ORADATA/dbf/tbsm01dba01_2.dbf					
	43,083	3	7.9	2.4	12,864	1		19		71.6
					/dba01/ORADATA/dbf/tbsm01dba01_1.dbf					
	868,842	60	5.5	2.1	20,273	1		1,732		2.4
TBSM02					/dba01/ORADATA/dbf/tbsm02dba01_1.dbf					
	1,022,425	71	6.8	2.7	29,989	2		3,148		6.3
TBSM51					/dba01/ORADATA/dbf/tbsm51dba01_2.dbf					
	889,427	62	5.1	5.9	60,741	4		11,273		5.6
					/dba01/ORADATA/dbf/tbsm51dba01_1.dbf					
	781,709	54	5.9	6.1	55,926	4		12,391		5.7

File IO Stats for DB: dba01 Instance: dba01 Snaps: 2 -4

->ordered by Tablespace, File

Tablespace	Filename							

	Av	Av	Av		Av	Buffer	Av	Buf
	Reads	Reads/s	Rd(ms)	Blks/Rd	Writes	Writes/s	Waits	Wt(ms)

TBSM52			/dba01/ORADATA/dbf/tbsm52dba01_1.dbf					
	16,649	1	10.1	1.0	11,222	1	1	0.0
TBSM52			/dba01/ORADATA/dbf/tbsm52dba01_2.dbf					
	7,144	0	12.4	1.0	7,117	0	0	
TBSM53			/dba01/ORADATA/dbf/tbsm53dba01_3.dbf					
	240,303	17	8.8	1.0	338,846	24	67	14.6
			/dba01/ORADATA/dbf/tbsm53dba01_1.dbf					
	128,858	9	12.0	1.0	28,166	2	111	12.9
			/dba01/ORADATA/dbf/tbsm53dba01_2.dbf					
	114,036	8	11.9	1.0	32,371	2	26	19.6
TBSS01			/dba01/ORADATA/dbf/tbss01dba01_1.dbf					
	610,371	42	73.1	4.3	105,236	7	5,701	316.6
TBSS51			/dba01/ORADATA/dbf/tbss51dba01_1.dbf					
	368,345	26	12.4	1.0	34,239	2	814	8.5
TBSSNP			/dba01/ORADATA/dbf/tbssnp_1.dbf					
	291	0	11.6	3.8	17	0	0	
TBS_BACK			/dba01/ORADATA/dbf/tbs_backdba01_1.dbf					
	86	0	16.2	1.0	17	0	0	
TEMP1			/dba01/ORADATA/dbf/temp1dba01_1.dbf					
	34	0	0.0	1.0	17	0	0	
			/dba01/ORADATA/dbf/temp1dba01_2.dbf					
	3,736	0	0.0	22.5	5,250	0	0	
TOOLS			/dba01/ORADATA/dbf/toolsdba01_1.dbf					
	11,207	1	2.2	1.3	1,137	0	1	0.0
			/dba01/ORADATA/dbf/toolsdba01_2.dbf					
	658	0	4.4	1.0	362	0	0	
USERS			/dba01/ORADATA/dbf/usersdba01_1.dbf					
	87	0	9.8	1.0	21	0	0	
			/dba01/ORADATA/dbf/usersdba01_2.dbf					
	17	0	0.0	1.0	17	0	0	

Buffer Pool Statistics for DB: dba01 Instance: dba01 Snaps: 2 -4
 -> Pools D: default pool, K: keep pool, R: recycle pool

	Buffer	Consistent	Physical	Physical	Free	Write	Buffer
P	Gets	Gets	Reads	Writes	Buffer	Complete	Busy
					Waits	Waits	Waits
D	30,754,181	0	30,010,736	2,486,800	0	0	160,335

Buffer wait Statistics for DB: dba01 Instance: dba01 Snaps: 2 -4
 -> ordered by wait time desc, waits desc

Class	Waits	Tot Wait Time (cs)	Avg Time (cs)
data block	159,656	286,451	2
segment header	226	100	0
undo header	326	40	0
undo block	120	22	0

Enqueue activity for DB: dba01 Instance: dba01 Snaps: 2 -4
 -> ordered by waits desc, gets desc

Enqueue	Gets	Waits
TX	177,029	14,031

Rollback Segment Stats for DB: dba01 Instance: dba01 Snaps: 2 -4

->A high value for "Pct Waits" suggests more rollback segments may be required

RBS No	Trans Table Gets	Pct Waits	Undo Bytes Written	Wraps	Shrinks	Extends
0	100.0	0.00	0	0	0	0
76	4,009.0	0.00	3,011,846	0	0	0
77	4,274.0	0.00	4,036,866	0	0	0
79	5,010.0	0.00	4,279,146	3	0	0
80	4,730.0	0.00	4,180,974	2	0	0
81	3,418.0	0.00	3,087,764	2	0	0
82	7,077.0	0.00	6,523,904	4	0	0
83	4,798.0	0.00	4,159,058	2	0	0
84	4,757.0	0.02	3,809,838	2	0	0
85	5,005.0	0.00	4,020,058	2	0	0
86	4,460.0	0.02	3,950,066	2	0	0
87	5,175.0	0.04	5,554,790	3	0	0
88	5,032.0	0.02	5,355,284	3	0	0
89	5,244.0	0.00	4,220,486	2	0	0
90	4,350.0	0.02	3,973,104	2	0	0
91	9,388.0	0.00	31,638,132	15	0	0
92	5,800.0	0.00	5,369,852	2	0	0
93	5,743.0	0.00	5,332,320	2	0	0
94	5,333.0	0.00	15,988,656	8	0	0
95	5,063.0	0.00	3,859,788	2	0	0
96	3,559.0	0.00	3,028,792	2	0	0
97	5,482.0	0.02	5,431,612	3	0	0
98	4,020.0	0.00	2,925,590	2	0	0
99	4,072.0	0.00	2,928,528	1	0	0
100	2,941.0	0.00	1,801,368	1	0	0
101	7,341.0	0.03	7,861,032	4	0	0
102	4,651.0	0.02	4,961,230	2	0	0
103	5,536.0	0.00	4,666,606	2	0	0
104	4,048.0	0.02	3,630,112	1	0	0
105	4,602.0	0.00	5,101,026	2	0	0
106	12,514.0	0.01	69,910,272	34	0	15
107	3,982.0	0.00	2,917,216	2	0	0
108	4,548.0	0.02	3,689,526	2	0	0
109	4,450.0	0.00	3,408,124	1	0	0
110	3,694.0	0.00	3,103,236	1	0	0
111	10,235.0	0.02	32,542,386	16	0	0
112	4,149.0	0.00	3,026,554	1	0	0
113	4,771.0	0.02	3,423,674	2	0	0
114	4,917.0	0.00	3,505,344	2	0	0
115	4,063.0	0.00	4,003,340	2	0	0
116	5,211.0	0.00	4,240,924	2	0	0
117	5,580.0	0.02	4,637,814	3	0	0
118	4,654.0	0.00	3,164,486	1	0	0
119	4,769.0	0.00	3,884,436	2	0	0

Rollback Segment Stats for DB: dba01 Instance: dba01 Snaps: 2 -4

->A high value for "Pct Waits" suggests more rollback segments may be required

RBS No	Trans Table Gets	Pct Waits	Undo Bytes Written	Wraps	Shrinks	Extends
120	4,021.0	0.00	3,034,092	1	0	0
121	4,039.0	0.00	3,191,548	2	0	0
122	3,870.0	0.00	3,778,458	2	0	0
123	3,644.0	0.00	3,018,332	2	0	0
124	3,191.0	0.00	2,697,214	2	0	0
125	2,967.0	0.00	2,088,206	1	0	0
126	4,149.0	0.00	3,454,876	2	0	0
127	3,975.0	0.00	3,879,908	2	0	0
128	5,836.0	0.00	4,917,940	2	0	0
129	4,245.0	0.00	3,963,266	2	0	0
130	6,379.0	0.00	6,483,112	3	0	0
132	12,307.0	0.02	69,432,430	34	0	15
133	14,715.0	0.02	76,919,156	38	0	17
134	3,828.0	0.00	3,213,020	2	0	0
135	5,614.0	0.04	4,922,670	2	0	0
136	5,114.0	0.04	3,676,814	2	0	0
137	3,419.0	0.03	2,525,042	2	0	0
138	6,949.0	0.01	6,341,890	4	0	0
139	4,328.0	0.02	4,086,654	2	0	0
140	4,767.0	0.00	4,266,816	2	0	0
141	3,859.0	0.03	3,044,358	1	0	0
142	3,712.0	0.00	3,077,798	1	0	0
143	5,535.0	0.02	16,545,146	8	0	0
144	4,025.0	0.02	3,053,298	1	0	0
145	5,772.0	0.00	5,768,094	3	0	0
146	4,597.0	0.00	5,101,434	3	0	0
147	5,234.0	0.02	4,554,324	2	0	0
148	5,064.0	0.00	4,216,938	3	0	0
149	5,836.0	0.00	6,692,288	4	0	0
150	4,388.0	0.00	4,262,966	3	0	0
151	3,653.0	0.00	2,727,238	2	0	0
152	4,589.0	0.00	5,336,590	2	0	0
153	5,316.0	0.00	4,030,252	2	0	0
154	4,843.0	0.00	4,235,564	2	0	0
155	5,598.0	0.00	16,840,224	9	0	0
156	5,245.0	0.02	15,931,224	7	0	2
157	3,502.0	0.03	2,713,450	1	0	0
158	3,909.0	0.00	2,782,836	2	0	0
159	4,628.0	0.00	3,758,808	2	0	0
160	7,436.0	0.00	27,300,394	13	0	0
161	4,897.0	0.00	11,305,832	5	0	0
162	4,899.0	0.00	3,819,822	2	0	0
163	4,417.0	0.00	3,056,898	2	0	0
164	4,052.0	0.02	3,256,102	2	0	0
165	3,972.0	0.00	3,645,832	2	0	0

Rollback Segment Stats for DB: dba01 Instance: dba01 Snaps: 2 -4

->A high value for "Pct Waits" suggests more rollback segments may be required

RBS No	Trans Table Gets	Pct Waits	Undo Bytes Written	Wraps	Shrinks	Extends
166	5,505.0	0.00	5,354,352	3	0	0
167	5,949.0	0.00	16,781,928	8	0	0
168	3,713.0	0.00	3,255,552	1	0	0
169	5,009.0	0.02	4,150,532	2	0	0
170	5,466.0	0.02	4,449,518	2	0	0
171	3,958.0	0.00	3,185,946	1	0	0
172	3,355.0	0.00	2,636,056	2	0	0
173	4,568.0	0.00	4,095,850	2	0	0
174	3,830.0	0.00	4,194,192	2	0	0
175	3,895.0	0.00	3,489,200	2	0	0
176	5,780.0	0.00	16,842,136	8	0	0
177	3,969.0	0.00	3,334,172	2	0	0
178	4,636.0	0.00	3,859,346	2	0	0
179	4,290.0	0.00	3,641,736	2	0	0

Rollback Segment Storage for DB: dba01 Instance: dba01 Snaps: 2 -4
 ->Optimal Size should be larger than Avg Active

RBS No	Segment Size	Avg Active	Optimal Size	Maximum Size
0	1,138,688	7,372		1,138,688
76	419,422,208	49,131,373	419,430,400	419,422,208
77	419,422,208	0	419,430,400	419,422,208
79	42,590,208	4,297,880	41,943,040	42,590,208
80	42,590,208	5,130,462	41,943,040	42,590,208
81	42,590,208	5,508,272	41,943,040	42,590,208
82	42,590,208	5,646,238	41,943,040	42,590,208
83	42,590,208	3,806,628	41,943,040	42,590,208
84	42,590,208	3,065,805	41,943,040	42,590,208
85	42,590,208	5,637,723	41,943,040	42,590,208
86	42,590,208	3,437,246	41,943,040	42,590,208
87	42,590,208	2,635,135	41,943,040	42,590,208
88	42,590,208	2,241,796	41,943,040	42,590,208
89	42,590,208	1,993,453	41,943,040	42,590,208
90	42,590,208	1,884,079	41,943,040	42,590,208
91	42,590,208	13,249,096	41,943,040	42,590,208
92	42,590,208	2,496,391	41,943,040	42,590,208
93	42,590,208	4,960,473	41,943,040	42,590,208
94	42,590,208	7,398,537	41,943,040	42,590,208
95	42,590,208	2,788,505	41,943,040	42,590,208
96	42,590,208	3,681,341	41,943,040	42,590,208
97	42,590,208	3,620,442	41,943,040	42,590,208
98	42,590,208	2,747,634	41,943,040	42,590,208
99	42,590,208	2,922,087	41,943,040	42,590,208
100	42,590,208	4,692,974	41,943,040	42,590,208
101	42,590,208	4,897,030	41,943,040	42,590,208
102	42,590,208	2,708,323	41,943,040	42,590,208
103	42,590,208	2,083,221	41,943,040	42,590,208
104	42,590,208	8,305,261	41,943,040	42,590,208
105	42,590,208	5,268,467	41,943,040	42,590,208
106	74,539,008	53,892,383	41,943,040	74,539,008
107	42,590,208	7,586,007	41,943,040	42,590,208
108	42,590,208	6,615,188	41,943,040	42,590,208
109	42,590,208	6,571,361	41,943,040	42,590,208
110	42,590,208	6,016,008	41,943,040	42,590,208
111	42,590,208	17,454,199	41,943,040	42,590,208
112	42,590,208	2,285,771	41,943,040	42,590,208
113	42,590,208	3,052,768	41,943,040	42,590,208
114	42,590,208	1,858,952	41,943,040	42,590,208
115	42,590,208	9,023,487	41,943,040	42,590,208
116	42,590,208	4,732,644	41,943,040	42,590,208
117	42,590,208	6,813,383	41,943,040	42,590,208
118	42,590,208	8,513,858	41,943,040	42,590,208
119	42,590,208	4,735,765	41,943,040	42,590,208
120	42,590,208	21,716,974	41,943,040	51,109,888
121	42,590,208	1,964,246	41,943,040	42,590,208
122	42,590,208	217,693,625	41,943,040	511,172,608

Rollback Segment Storage for DB: dba01 Instance: dba01 Snaps: 2 -4
 ->Optimal Size should be larger than Avg Active

RBS No	Segment Size	Avg Active	Optimal Size	Maximum Size
123	42,590,208	5,718,693	41,943,040	42,590,208
124	42,590,208	4,443,753	41,943,040	42,590,208
125	42,590,208	2,435,736	41,943,040	42,590,208
126	42,590,208	2,643,884	41,943,040	42,590,208
127	42,590,208	7,561,782	41,943,040	46,850,048
128	42,590,208	2,465,094	41,943,040	42,590,208
129	42,590,208	2,566,604	41,943,040	42,590,208
130	42,590,208	3,693,499	41,943,040	42,590,208
132	74,539,008	53,827,217	41,943,040	74,539,008
133	78,798,848	59,264,629	41,943,040	78,798,848
134	42,590,208	5,467,496	41,943,040	51,109,888
135	42,590,208	13,483,036	41,943,040	46,850,048
136	42,590,208	12,899,828	41,943,040	42,590,208
137	42,590,208	4,609,633	41,943,040	42,590,208
138	42,590,208	3,637,713	41,943,040	42,590,208
139	42,590,208	2,491,798	41,943,040	42,590,208
140	42,590,208	7,321,185	41,943,040	42,590,208
141	42,590,208	4,909,855	41,943,040	42,590,208
142	42,590,208	13,213,714	41,943,040	48,979,968
143	42,590,208	9,388,604	41,943,040	42,590,208
144	42,590,208	9,073,568	41,943,040	42,590,208
145	42,590,208	6,221,551	41,943,040	42,590,208
146	42,590,208	4,588,554	41,943,040	42,590,208
147	42,590,208	2,342,772	41,943,040	42,590,208
148	42,590,208	4,365,806	41,943,040	42,590,208
149	42,590,208	5,018,033	41,943,040	42,590,208
150	42,590,208	4,579,270	41,943,040	42,590,208
151	42,590,208	7,870,376	41,943,040	46,850,048
152	42,590,208	6,982,647	41,943,040	42,590,208
153	42,590,208	19,944,820	41,943,040	68,149,248
154	42,590,208	9,829,923	41,943,040	42,590,208
155	42,590,208	5,274,796	41,943,040	42,590,208
156	46,850,048	27,978,008	41,943,040	46,850,048
157	42,590,208	11,577,743	41,943,040	42,590,208
158	42,590,208	5,274,989	41,943,040	42,590,208
159	42,590,208	4,877,205	41,943,040	42,590,208
160	42,590,208	10,702,935	41,943,040	42,590,208
161	42,590,208	3,769,022	41,943,040	42,590,208
162	42,590,208	8,625,142	41,943,040	51,109,888
163	42,590,208	4,548,788	41,943,040	42,590,208
164	42,590,208	6,881,776	41,943,040	42,590,208
165	42,590,208	4,090,556	41,943,040	42,590,208
166	42,590,208	8,519,558	41,943,040	42,590,208
167	42,590,208	9,969,199	41,943,040	42,590,208
168	42,590,208	2,372,896	41,943,040	42,590,208
169	42,590,208	2,259,582	41,943,040	42,590,208
170	42,590,208	8,825,690	41,943,040	48,979,968

Rollback Segment Storage for DB: dba01 Instance: dba01 Snaps: 2 -4
 ->Optimal Size should be larger than Avg Active

RBS No	Segment Size	Avg Active	Optimal Size	Maximum Size
171	42,590,208	6,191,750	41,943,040	42,590,208
172	42,590,208	11,510,328	41,943,040	42,590,208
173	42,590,208	7,715,760	41,943,040	42,590,208
174	42,590,208	2,461,638	41,943,040	42,590,208
175	42,590,208	6,116,715	41,943,040	42,590,208
176	42,590,208	6,379,282	41,943,040	42,590,208
177	42,590,208	15,872,040	41,943,040	63,889,408
178	42,590,208	5,381,336	41,943,040	42,590,208
179	42,590,208	7,941,825	41,943,040	42,590,208

Latch Activity for DB: dba01 Instance: dba01 Snaps: 2 -4

->"Get Requests", "Pct Get Miss" and "Avg Slps/Miss" are statistics for willing-to-wait latch get requests

->"NoWait Requests", "Pct NoWait Miss" are for no-wait latch get requests

->"Pct Misses" for both should be very close to 0.0

Latch Name	Get Requests	Pct Get Miss	Avg Slps /Miss	NoWait Requests	Pct NoWait Miss
Token Manager	78,427	0.0	0.3	2,596	0.0
X\$KSFP	8	0.0		0	
active checkpoint queue latch	791,889	0.1	0.3	0	
archive control	274	0.0		0	
archive process latch	34	0.0		0	
begin backup scn array	10	0.0		0	
cache buffer handles	592,279,666	7.1	0.0	0	
cache buffers chains	984,933,131	0.1	0.3	44,678,716	0.1
cache buffers lru chain	9,553,732	0.2	0.5	30,330,326	0.2
channel handle pool latch	11,379	0.0		11,562	0.0
channel operations parent lat	19,532	0.0		11,562	0.0
checkpoint queue latch	29,410,803	0.1	0.1	0	
dictionary lookup	63	0.0		0	
dml lock allocation	753,125	0.0	0.1	0	
enqueue hash chains	1,875,817	0.1	0.7	0	
enqueuees	1,425,986	0.0	0.1	0	
error message lists	1,031	3.9	0.1	0	
event group latch	8,153	0.0		0	
file number translation table	33	0.0		0	
global transaction	53,851	0.0		0	
global tx free list	2,532	0.0		0	
global tx hash mapping	19,572	0.0		0	
job_queue_processes parameter	250	0.0		0	
ktm global data	48	0.0		0	
latch wait list	141,415	0.6	0.3	141,300	0.2
library cache	68,488,577	0.5	0.6	80,409	1.3
library cache load lock	4,216	0.0		0	
list of block allocation	386,860	0.0	0.4	0	
loader state object freelist	6,126	0.1	0.0	0	
longop free list	1,328,432	0.6	0.0	0	
messages	3,778,456	0.3	0.1	0	
mostly latch-free SCN	490,890	0.0	0.0	0	
multiblock read objects	1,708,086	0.0	0.1	0	
ncodef allocation latch	250	0.0		0	
parallel query alloc buffer	14,696	20.2	0.3	0	
parallel query stats	632	29.3	1.8	0	
process allocation	8,153	0.1	1.0	8,153	0.0
process group creation	16,403	0.0		0	
process queue	5,099	1.1	0.3	0	
process queue reference	4,020,210	0.0	0.1	196,962	7.9
query server freelists	9,226	5.1	0.1	0	
query server process	86	0.0		86	0.0

Latch Activity for DB: dba01 Instance: dba01 Snaps: 2 -4

->"Get Requests", "Pct Get Miss" and "Avg Slps/Miss" are statistics for willing-to-wait latch get requests

->"NoWait Requests", "Pct NoWait Miss" are for no-wait latch get requests

->"Pct Misses" for both should be very close to 0.0

		Pct	Avg		Pct
	Get	Get	Slps	NoWait	NoWait
Latch Name	Requests	Miss	/Miss	Requests	Miss
-----	-----	-----	-----	-----	-----
redo allocation	5,928,960	0.1	0.1	0	
redo writing	2,092,841	0.4	0.4	0	
row cache objects	8,936,427	0.0	0.3	2,590	0.2
sequence cache	68,942	0.0	1.0	0	
session allocation	480,695	0.1	0.8	0	
session idle bit	44,701,432	0.0	0.1	0	
session switching	265	0.0		0	
shared pool	5,458,933	0.3	0.6	0	
sort extent pool	257	0.0		0	
transaction allocation	1,191,981	0.0	0.1	0	
transaction branch allocation	19,941	0.0		0	
undo global data	2,376,711	0.1	0.6	0	
user lock	33,572	0.0	0.5	0	

Latch Sleep breakdown for DB: dba01 Instance: dba01 Snaps: 2 -4
-> ordered by misses desc

Latch Name	Get Requests	Misses	Sleeps	Spin & Sleeps 1->4
cache buffer handles	592,279,666	42,157,280	1,013,749	41179902/942 959/32745/16 74/0
cache buffers chains	984,933,131	1,331,118	384,653	1013733/2580 73/52615/669 7/0
library cache	68,488,577	341,875	192,024	215804/69669 /48519/7883/ 0
checkpoint queue latch	29,410,803	16,889	1,747	15149/1733/7 /0/0
cache buffers lru chain	9,553,732	14,343	7,191	7195/7107/40 /1/0
shared pool	5,458,933	14,057	8,674	8554/2865/24 29/209/0
messages	3,778,456	9,750	1,134	8646/1074/30 /0/0
longop free list	1,328,432	8,521	250	8273/246/2/0 /0
redo writing	2,092,841	7,903	3,199	4770/3075/54 /4/0
redo allocation	5,928,960	6,633	817	5852/745/36/ 0/0
parallel query alloc buffe	14,696	2,969	960	2198/606/142 /23/0
undo global data	2,376,711	2,827	1,643	1234/1543/50 /0/0
session idle bit	44,701,432	2,166	238	1937/220/9/0 /0
enqueue hash chains	1,875,817	1,021	694	336/678/6/1/ 0
row cache objects	8,936,427	849	283	570/275/4/0/ 0
active checkpoint queue la	791,889	836	267	579/247/10/0 /0
latch wait list	141,415	802	220	658/70/72/2/ 0
enqueuees	1,425,986	632	87	551/75/6/0/0
multiblock read objects	1,708,086	540	59	481/59/0/0/0
query server freelists	9,226	475	57	423/48/3/1/0
session allocation	480,695	442	345	198/180/38/2 6/0
transaction allocation	1,191,981	400	34	372/23/4/1/0
process queue reference	4,020,210	359	41	320/37/2/0/0

Latch Sleep breakdown for DB: dba01 Instance: dba01 Snaps: 2 -4
 -> ordered by misses desc

Latch Name	Get Requests	Misses	Sleeps	Spin & Sleeps 1->4
parallel query stats	632	185	328	18/78/43/46/ 0
dml lock allocation	753,125	124	16	109/14/1/0/0
process queue	5,099	57	17	45/7/5/0/0
error message lists	1,031	40	3	37/3/0/0/0
mostly latch-free SCN	490,890	34	1	33/1/0/0/0
list of block allocation	386,860	31	11	21/9/1/0/0
process allocation	8,153	12	12	0/12/0/0/0
Token Manager	78,427	4	1	3/1/0/0/0
sequence cache	68,942	2	2	0/2/0/0/0
user lock	33,572	2	1	1/1/0/0/0

Latch Miss Sources for DB: dba01 Instance: dba01 Snaps: 2 -4

-> only latches with sleeps are shown

-> ordered by name, sleeps desc

Latch Name	Where	NoWait Misses	Sleeps	Waiter Sleeps
Token Manager	kgklookup	0	1	1
active checkpoint queue	kcbbacq: scan active check	0	266	267
active checkpoint queue	kcbstcl: Start a new check	0	1	0
cache buffer handles	kcbzgs	0	691,673	#####
cache buffer handles	kcbzfs	0	321,247	#####
cache buffers chains	kcbgtcr: kslbegin	0	356,732	#####
cache buffers chains	kcbzib: multi-block read:	0	7,768	0
cache buffers chains	kcbgcur: kslbegin	0	4,399	4,049
cache buffers chains	kcbrls: kslbegin	0	3,449	36,008
cache buffers chains	kcbget: pin buffer	0	3,040	2,723
cache buffers chains	kcbchg: kslbegin: bufs not	0	1,836	2,403
cache buffers chains	kcbgtcr	0	1,768	49
cache buffers chains	kcbbxsv	0	1,435	1,940
cache buffers chains	kcbbwbl	0	1,270	2,634
cache buffers chains	kcbzwb	0	1,133	628
cache buffers chains	kcbzgb: scan from tail. no	0	785	0
cache buffers chains	kcbnlc	0	515	661
cache buffers chains	kcbzib: finish free bufs	0	239	9,539
cache buffers chains	kcbchg: kslbegin: call CR	0	147	943
cache buffers chains	kcbzsc	0	61	8
cache buffers chains	kcbget: exchange rls	0	23	123
cache buffers chains	kcbget: exchange	0	21	295
cache buffers chains	kcbzib: exchange rls	0	8	41
cache buffers chains	kcbbwdb	0	7	1,208
cache buffers chains	kcbchg: no fast path	0	4	17
cache buffers chains	kcbesc: escalate	0	3	0
cache buffers chains	kcbcge	0	1	27
cache buffers chains	kcbso1: set no access	0	1	15
cache buffers lru chain	kcbzgb: multiple sets nowa	0	6,359	0
cache buffers lru chain	kcbbiop: lru scan	0	545	84
cache buffers lru chain	kcbzgb: posted for free bu	0	212	433
cache buffers lru chain	kcbzar: KSLNBEGIN	0	56	4,892
cache buffers lru chain	kcbzgm	0	9	365
cache buffers lru chain	kcbbioc: age write clones	0	5	167
cache buffers lru chain	kcbbioc	0	3	1,025
cache buffers lru chain	kcbbxsv: move to being wri	0	1	109
cache buffers lru chain	kcbgtcr:CR Scan:KCBRSKIP	0	1	51
checkpoint queue latch	kcbkllrba: compute lowest	0	728	1,430
checkpoint queue latch	kcbk0rrd: update recovery	0	348	3
checkpoint queue latch	kcbklbc: Link buffer into	0	310	144
checkpoint queue latch	kcbbxsv: move to being wri	0	142	29
checkpoint queue latch	kcbbwthc: thread checkpoin	0	111	39
checkpoint queue latch	kcbbcrcv: check recovery q	0	74	71
checkpoint queue latch	kcbnlc: Link buffers into	0	22	8
checkpoint queue latch	kcbswcu: Switch buffers	0	11	23
checkpoint queue latch	kcbwtsc: file queues	0	1	0

Latch Miss Sources for DB: dba01 Instance: dba01 Snaps: 2 -4

-> only latches with sleeps are shown

-> ordered by name, sleeps desc

Latch Name	Where	NoWait Misses	Sleeps	Waiter Sleeps
cost function	kzulrl	0	1	1
dml lock allocation	ktaiam	0	10	10
dml lock allocation	ktaidm	0	6	6
enqueue hash chains	ksqcmi: get hash chain lat	0	486	288
enqueue hash chains	ksqgtl3	0	184	245
enqueue hash chains	ksqrcl	0	23	157
enqueue hash chains	ksqcnl	0	1	2
enqueuees	ksqgtl2	0	38	21
enqueuees	ksqgel: create enqueue	0	20	21
enqueuees	ksqdel	0	12	3
enqueuees	ksqrcl	0	10	21
enqueuees	ksqies	0	4	18
enqueuees	ksqgel: failed to get enqu	0	3	3
error message lists	kxfpqidqr2: KSLBEGIN	0	2	1
error message lists	kxfpqsnd	0	1	2
latch wait list	kslfre	90	159	220
latch wait list	kslges	134	61	0
library cache	kglpna1: child: alloc spac	0	68,765	50,847
library cache	kglhdgn: child:	0	64,595	11,806
library cache	kglupc: child	0	15,536	57,068
library cache	kglpna1: child: before pro	0	13,866	16,187
library cache	kglhdgc: child:	0	8,352	5,193
library cache	kglpnc: child	0	6,325	14,381
library cache	kglldl: child: cleanup	0	4,892	5,716
library cache	kglidp: parent	0	1,791	3
library cache	kglpnp: child	0	1,450	9,315
library cache	kglldl: child: free pin	0	1,141	11,290
library cache	kglic	0	869	72
library cache	kglget: child: KGLDSBYD	0	532	5,168
library cache	kglpin	0	391	1,738
library cache	kgltdi: 2child	0	356	295
library cache	kglget: child: KGLDSBRD	0	241	34
library cache	kglobpn: child:	0	135	473
library cache	kglpna1: child: check gran	0	111	100
library cache	kglati	0	91	13
library cache	kglpnd1: parent: purge	0	23	4
library cache	kglobld: child:	0	18	193
library cache	kglrdp: parent	0	12	1
library cache	kglde: child 0	0	8	683
library cache	kglndp: child	0	7	64
library cache	kglpna1: parent held, no p	0	7	0
library cache	kglde: child	0	2	1
library cache	kgltdld: 2child	0	1	0
checkpoint queue latch	kcbwcu: Switch buffers	0	11	23
checkpoint queue latch	kcbwtsc: file queues	0	1	0
cost function	kzulrl	0	1	1

Latch Miss Sources for DB: dba01 Instance: dba01 Snaps: 2 -4

-> only latches with sleeps are shown

-> ordered by name, sleeps desc

Latch Name	Where	NoWait Misses	Sleeps	Waiter Sleeps
dml lock allocation	ktaiam	0	10	10
dml lock allocation	ktaidm	0	6	6
enqueue hash chains	ksqcmi: get hash chain lat	0	486	288
enqueue hash chains	ksqgtl3	0	184	245
enqueue hash chains	ksqrcl	0	23	157
enqueue hash chains	ksqcnl	0	1	2
enqueuees	ksqgtl2	0	38	21
enqueuees	ksqgel: create enqueue	0	20	21
enqueuees	ksqdel	0	12	3
enqueuees	ksqrcl	0	10	21
enqueuees	ksqies	0	4	18
enqueuees	ksqgel: failed to get enqu	0	3	3
error message lists	kxfpqidqr2: KSLBEGIN	0	2	1
error message lists	kxfpqsnd	0	1	2
latch wait list	kslfre	90	159	220
latch wait list	kslges	134	61	0
library cache	kglpna1: child: alloc spac	0	68,765	50,847
library cache	kglhdgn: child:	0	64,595	11,806
library cache	kglupc: child	0	15,536	57,068
library cache	kglpna1: child: before pro	0	13,866	16,187
library cache	kglhdgc: child:	0	8,352	5,193
library cache	kglpnc: child	0	6,325	14,381
library cache	kglldk1: child: cleanup	0	4,892	5,716
library cache	kglidp: parent	0	1,791	3
library cache	kglpnp: child	0	1,450	9,315
library cache	kglldk1: child: free pin	0	1,141	11,290
library cache	kglic	0	869	72
library cache	kglget: child: KGLDSBYD	0	532	5,168
library cache	kglpin	0	391	1,738
library cache	kgltdi: 2child	0	356	295
library cache	kglget: child: KGLDSBRD	0	241	34
library cache	kglobpn: child:	0	135	473
library cache	kglpna1: child: check gran	0	111	100
library cache	kglati	0	91	13
library cache	kglpnd1: parent: purge	0	23	4
library cache	kglobld: child:	0	18	193
library cache	kglrdp: parent	0	12	1
library cache	kglkte: child 0	0	8	683
library cache	kglndp: child	0	7	64
library cache	kglpna1: parent held, no p	0	7	0
library cache	kglkte: child	0	2	1
library cache	kgltdld: 2child	0	1	0
library cache	kglpin: child: KGLMX	0	1	0
list of block allocation	ktlabl	0	7	5
list of block allocation	ktlbb1	0	4	6
longop free list	ksuloget	0	250	250

Latch Miss Sources for DB: dba01 Instance: dba01 Snaps: 2 -4

-> only latches with sleeps are shown

-> ordered by name, sleeps desc

Latch Name	Where	NoWait Misses	Sleeps	Waiter Sleeps
messages	ksaamb: after wakeup	0	569	811
messages	ksarcv: after wait	0	389	193
messages	ksarcv	0	175	125
messages	ksaclr	0	1	3
mostly latch-free SCN	kcs02	0	1	1
multiblock read objects	kcbzib: MBRGET	0	33	35
multiblock read objects	kcbzib: MBRFRE	0	26	24
parallel query alloc buf	kxfpbalo	0	565	559
parallel query alloc buf	kxfpbfre	0	394	401
parallel query stats	kxfprst: KSLBEGIN	0	328	328
process allocation	ksuapc	0	12	12
query server freelists	kxfpqrsnd	0	41	0
query server freelists	kxfpobadf	0	37	37
query server freelists	kxfpobrmf	0	20	20
query server freelists	kxfpqsnd: KSLBEGIN	0	11	13
query server freelists	kxfpqidqr1: KSLBEGIN	0	6	4
redo allocation	kcrfwr: redo allocation	0	541	712
redo allocation	kcrfwi: before write	0	233	50
redo allocation	kcrfwi: more space	0	43	55
redo writing	kcrfsr	0	2,872	11
redo writing	kcrfwi: after write	0	163	44
redo writing	kcrfwcr	0	88	813
redo writing	kcrfss	0	76	2,330
row cache objects	kqrpre: find obj	0	252	161
row cache objects	kqreqd: rel enqueue	0	13	30
row cache objects	kqrso	0	6	1
row cache objects	kqreqd	0	4	86
row cache objects	kqrpsc: incr stat	0	2	0
sequence cache	kdnnext: cached seq	0	2	0
session allocation	ksuxds: KSUSFCLC not set	0	260	173
session allocation	kxfpqidqr	0	50	38
session allocation	ksucri	0	23	47
session allocation	ksursi	0	4	45
session allocation	ksusin	0	4	2
session allocation	ksuxds: not user session	0	4	25
session idle bit	ksupuc: clear busy	0	118	78
session idle bit	ksupuc: set busy	0	114	159
session idle bit	ksuxds	0	6	1
shared pool	kghfrunp: alloc: clatch no	0	3,149	0
shared pool	kghalo	0	2,626	1,620
shared pool	kghfrunp: clatch: nowait	0	2,282	0
shared pool	kghupr1	0	1,214	6,125
shared pool	kghfre	0	975	644
shared pool	kghfrunp: alloc: wait	0	339	38
shared pool	kghfrunp: clatch: wait	0	220	1,372
shared pool	kghfnd: min scan	0	176	0

Latch Miss Sources for DB: dba01 Instance: dba01 Snaps: 2 -4

-> only latches with sleeps are shown

-> ordered by name, sleeps desc

Latch Name	Where	NoWait Misses	Sleeps	Waiter Sleeps
shared pool	kghfen: not perm alloc cla	0	52	94
shared pool	kghfnd: req scan	0	49	0
shared pool	kghalp	0	41	145
shared pool	kghfnd: get next extent	0	36	0
shared pool	kghfrunp: no latch	0	10	0
shared pool	kghfru	0	7	8
transaction allocation	ktcxba	0	30	20
transaction allocation	ktcdso	0	4	14
undo global data	ktubnd	0	1,247	152
undo global data	ktudba: KSLBEGIN	0	323	1,302
undo global data	ktudnx: KSLBEGIN	0	70	186
undo global data	ktucof: at start	0	3	0

Dictionary Cache Stats for DB: dba01 Instance: dba01 Snaps: 2 -4

->"Pct Misses" should be very low (< 2% in most cases)

->"Cache Usage" is the number of cache entries being used

->"Pct SGA" is the ratio of usage to allocated size for that cache

Cache	Get Requests	Pct Miss	Scan Requests	Pct Miss	Mod Req	Final Usage	Pct SGA
dc_constraints	9	33.3	0		9	184	93
dc_database_links	2,331	0.0	0		0	20	95
dc_files	172	0.0	0		0	1	5
dc_free_extents	148	39.9	87	0.0	128	341	29
dc_global_oids	0		0		0	2	9
dc_histogram_data	0		0		0	0	0
dc_histogram_data_valu	0		0		0	0	0
dc_histogram_defs	831,084	0.0	0		14	702	97
dc_object_ids	1,301,711	0.0	0		29	1,517	100
dc_objects	139,492	0.1	0		92	2,037	100
dc_outlines	0		0		0	0	0
dc_profiles	7,783	0.0	0		0	1	14
dc_rollback_segments	44,335	0.0	0		0	181	99
dc_segments	345,098	0.0	0		195	1,249	99
dc_sequence_grants	0		0		0	0	0
dc_sequences	5,942	0.0	0		5,939	8	50
dc_synonyms	61,318	0.0	0		0	205	100
dc_tablespace_quotas	3	0.0	0		3	9	17
dc_tablespaces	19,905	0.0	0		0	43	77
dc_used_extents	122	71.3	0		122	298	100
dc_user_grants	84,772	0.0	0		0	40	98
dc_usernames	80,477	0.0	0		0	26	60
dc_users	114,086	0.0	0		0	42	86
ifs_acl_cache_entries	0		0		0	0	0

Library Cache Activity for DB: dba01 Instance: dba01 Snaps: 2 -4
 ->"Pct Misses" should be very low

Namespace	Get Requests	Pct Miss	Pin Requests	Pct Miss	Reloads	Invali- dations
BODY	359,086	0.0	359,084	0.0	0	0
CLUSTER	1,575	0.3	1,130	0.9	0	0
INDEX	0		0		0	0
OBJECT	0		0		0	0
PIPE	0		0		0	0
SQL AREA	738,856	3.1	20,928,868	0.2	3,122	1,434
TABLE/PROCEDURE	146,977	0.2	4,040,503	0.0	1,154	0
TRIGGER	374,089	0.0	374,091	0.0	73	0

SGA Memory Summary for DB: dba01 Instance: dba01 Snaps: 2 -4

SGA regions	Size in Bytes
Database Buffers	737,280,000
Fixed Size	76,820
Redo Buffers	532,480
Variable Size	379,277,312
sum	1,117,166,612

SGA breakdown difference for DB: dba01 Instance: dba01 Snaps: 2 -4

Pool	Name	Begin value	End value	Difference
java pool	free memory	20,684,800	20,684,800	0
java pool	memory in use	286,720	286,720	0
large pool	free memory	25,000,000	25,000,000	0
shared pool	DML locks	3,487,200	3,487,200	0
shared pool	KGFF heap	39,392	39,392	0
shared pool	KGK heap	5,848	5,848	0
shared pool	KQLS heap	2,937,448	2,278,992	-658,456
shared pool	PL/SQL DIANA	2,983,520	336,624	-2,646,896
shared pool	PL/SQL MPCODE	1,697,264	1,318,112	-379,152
shared pool	PLS non-lib hp	2,104	2,104	0
shared pool	PX msg pool	1,100,752	1,100,752	0
shared pool	PX subheap	137,848	137,848	0
shared pool	State objects	6,890,864	6,890,864	0
shared pool	db_block_buffers	12,240,000	12,240,000	0
shared pool	db_handles	3,000,000	3,000,000	0
shared pool	dictionary cache	2,846,608	2,953,016	106,408
shared pool	enqueue_resources	2,109,600	2,109,600	0
shared pool	event statistics per ses	22,932,560	22,932,560	0
shared pool	fixed allocation callbac	3,600	3,600	0
shared pool	free memory	34,661,656	34,407,664	-253,992
shared pool	joxs heap init	4,152	4,152	0
shared pool	ktlbk state objects	3,109,424	3,109,424	0
shared pool	library cache	88,638,712	91,396,248	2,757,536
shared pool	miscellaneous	19,692,288	19,220,056	-472,232
shared pool	processes	4,776,000	4,776,000	0
shared pool	sessions	14,240,384	14,240,384	0
shared pool	sql area	99,182,568	100,743,304	1,560,736
shared pool	table columns	31,144	18,592	-12,552
shared pool	table definiti	25,616	20,104	-5,512
shared pool	transactions	6,480,384	6,480,384	0
shared pool	trigger defini	21,368	29,848	8,480
shared pool	trigger inform	1,120	1,768	648
	db_block_buffers	737,280,000	737,280,000	0
	fixed_sga	76,820	76,820	0
	log_buffer	524,288	524,288	0

init.ora Parameters for DB: dba01 Instance: dba01 Snaps: 2 -4

Parameter Name	Begin value	End value (if different)
-----	-----	-----
_db_handles_cached	0	
_trace_files_public	TRUE	
audit_file_dest	/logs/oracle/dba01/adump	
audit_trail	FALSE	
background_dump_dest	/dba01/ADMIN/CDUMP	
compatible	8.1.7	
control_files	/dba01/ORADATA/ctl/cntrlP	
core_dump_dest	/dba01/ADMIN/CDUMP	
db_block_buffers	90000	
db_block_lru_latches	24	
db_block_size	8192	
db_file_multiblock_read_count	32	
db_files	220	
db_name	dba01	
db_writer_processes	8	
distributed_transactions	350	
global_names	TRUE	
java_pool_size	20971520	
job_queue_processes	3	
large_pool_size	25000000, B	
log_archive_dest	/dba01/BACKUP/LOG/arc	
log_archive_format	%S_%t.arc	
log_archive_start	TRUE	
log_buffer	524288	
log_checkpoint_interval	250000	
max_dump_file_size	512	
max_enabled_roles	30	
max_rollback_segments	200	
nls_date_format	DD-MON-RR	
open_cursors	300	
open_links	16	
optimizer_mode	RULE	
parallel_max_servers	25	
parallel_min_servers	4	
processes	6000	
remote_os_authent	FALSE	
remote_os_roles	TRUE	
resource_limit	FALSE	
rollback_segments	rbs1_01, rbs1_02, rbs1_03, rbs1_0	
session_cached_cursors	90	
shared_pool_reserved_size	12582912	
shared_pool_size	262144000	
sort_area_retained_size	4194304	
sort_area_size	4194304	
sql_trace	FALSE	
timed_statistics	TRUE	
transactions_per_rollback_seg	50	
user_dump_dest	/dba01/ADMIN/UDUMP/dba01/udump	
utl_file_dir	/cmp/work/UTL_FILE	

End of Report

```
transactions_per_rollback_seg 50
user_dump_dest                /dba01/ADMIN/UDUMP/dba01/udump
utl_file_dir                  /cmp/work/UTL_FILE
-----
End of Report
```




Redundant Arrays of Inexpensive Disks Technology (RAID)

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

System Hardware Configuration

Storage Subsystem Detail

Storage subsystem performance is one of the most important aspects of tuning an Oracle database server for optimal performance. The architecture and design of the storage subsystem must therefore be considered early in the system design process. In performing the storage subsystem design, system requirements such as the required volume of online transaction data, peak transactions per second load, and system availability are transformed into specific storage subsystem design requirements for storage capacity, peak sustainable I/Os per second, and fault tolerance. Values for design parameters in the selected technology are then chosen to meet these specific requirements.

Modern storage systems offer great flexibility in meeting a wide range of design criteria; technologies such as striping, mirroring, and other fault-tolerant RAID configurations provide the ability to meet these design requirements. Matching the right technology with application I/O characteristics is key to achieving the promised performance and fault tolerance levels; conversely, using the wrong technology for a specific I/O characteristic can lead to I/O bottlenecks and degraded response times. In this section, key parameters in the design of the storage subsystem are described, as well as how they relate to the performance of an Oracle database.

Storage Subsystem Design Parameters and Oracle

When designing a storage subsystem, the available design parameters are weighed against each other until a design solution is achieved that meets or exceeds all design requirements. In the context of an Oracle database server, certain measures are available to specify these requirements. These measures can be categorized under performance, availability, and cost.

Performance

- Random read performance: Important for Oracle indexed or hash-based queries and rollback segment reads
- Random write performance: Important for Oracle DBWn writes; heavy in an OLTP environment, light in a data warehouse
- Sequential read performance: Backups, Oracle full table scans, index creations, parallel queries, temporary segment reads, and recovery from archived redo log files
- Sequential write performance: Oracle LGWR writes, temporary segment writes, direct-path loader writes, tablespace creations
- Impact of concurrency

Storage Subsystem Design Parameters and Oracle (continued)

Availability

- Outage frequency: Expected number of occurrences of a possible outage per unit of time specified in mean time to failure (MTTF)
- Outage duration: The mean time to repair (MTTR) for a given outage event
- Performance degradation during outage: Whether a disk configuration provides service during a fault, and if so, at what level

Cost

- Acquisition cost: The cost of purchasing, installing, and configuring the storage subsystem
- Operational cost: The cost of running and maintaining the system to meet the system availability and service level requirements

The Redundant Arrays of Inexpensive Disks (RAID) technologies have been developed for nonmainframe, open system solutions. RAID provides low-cost fault tolerance and improved performance. Several levels of RAID are available and may be mixed within one storage subsystem design. Each level of RAID can be categorized against the measures listed above and its impact on Oracle database performance. Some levels of RAID configurations have been available for many years under different names. Key parameters when configuring a RAID system are:

- Array size: The number of drives in the array
- Disk size: The size of each disk
- Stripe size: The size of an I/O chunk, written to or read from a contiguous location on a disk (Striping allows data files to be interleaved and spread across the disks in an array in an attempt to parallelize file I/O.)

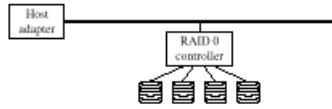
Storage Subsystem Design Parameters and Oracle (continued)

Cost (continued)

The throughput of a storage subsystem, expressed in I/Os per second, determines how many transactions can be processed by the subsystem before queuing delays begin to occur. If the I/Os-per-second requirement of the application is known, broken down into reads per transaction, writes per transaction, and transactions per second, you can determine whether a particular RAID configuration will support the transaction rate applied by an application. To calculate throughput, or total sustainable I/Os-per-second load that a RAID array can support, simply multiply the number of drives in the array times the sustainable I/Os per second of one drive, currently in the approximate range of 35 to 50 I/Os per second. Different RAID configurations, however, add to the I/O load on a disk array applied by the application in order to provide the fault tolerance function. Once the total I/O load on the array is known, the number of drives required to support the load can be found. Simply divide the total I/Os per second load by the random I/O throughput rating for the selected drive.

The sections below briefly describe each RAID level and some of its characteristics. For each level, an equation is provided to calculate the total I/Os per second load on a RAID array, to illustrate the impact of the RAID configuration on the array's capacity. Also provided is an equation for calculating the size of the disk drives required for an array.

RAID Level 0, Nonredundant Striping



RAID 0 refers to simple data striping of multiple disks into a single logical volume, and has no fault tolerance. When properly configured, it provides excellent response times for high concurrency random I/O and excellent throughput for low concurrency sequential I/O. Selection of the array and stripe sizes requires careful consideration in order to achieve the promised throughput. For RAID 0, the total I/Os-per-second load generated against the array is calculated directly from the application load, because there is no fault tolerance in this configuration:

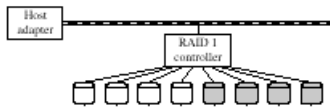
$$\text{total I/O per second load on array} = (\text{reads/transaction} + \text{writes/transaction}) * \text{transactions/second}$$

The size of each drive in the array can be calculated from the online volume requirements as follows:

$$\text{drive size} = [\text{total space required by application} / \text{number of drives in array}] \text{ Rounded up to next drive size.}$$

- Below is a summary of RAID 0 characteristics:
- Random read performance: Excellent under all concurrency levels if each I/O request fits within a single striping segment
- Random write performance: Same as random read performance
- Sequential read performance: Excellent with fine-grained striping at low concurrency levels
- Sequential write performance: Same as sequential read performance
- Outage frequency: Poor; any single disk failure will cause application outage
- Outage duration: Poor; the duration of a RAID 0 outage is the time required to detect the failure, replace the disk drive, and perform Oracle media recovery
- Performance degradation during outage: Poor; any disk failure causes all applications requiring use of the array to crash
- Acquisition cost: Excellent, because there is no redundancy; you buy only enough for storage and I/Os per second requirements
- Operational cost: Fair to poor; frequent media recoveries increase operational costs and may outweigh the acquisition cost advantage

RAID Level 1, Mirroring



RAID Level 1, or disk mirroring, provides the best fault tolerance of any of the RAID configurations. Each disk drive is backed up by an exact copy of itself on an identical drive. A storage subsystem of mirrored drives can continue at full performance with a multiple disk failure as long as no two drives in a mirrored pair have failed. The total I/Os per load applied to a mirrored pair is calculated as follows:

$$\text{total I/O per second load on array} = (\text{reads/transaction} + 2 * \text{writes/transaction}) * \text{transactions/second}$$

Note the two multiplier of the writes/transaction factor. This is due to the fact that each write request by an application to a mirrored pair actually results in two writes, one to the primary disk and one to the backup disk. The size of the drive required is:

$$\text{drive size} = [\text{total space required by application} / \text{number of drives in array} / 2] \text{ Rounded up to next drive size.}$$

In the simplest RAID 1 configuration, the number of drives in the array is two: the primary drive and its backup. The definition of RAID 1, however, includes the ability to expand the array in units of two drives to achieve a striped and mirrored configuration. Striping occurs in an array of four or more disks. Some industry literature (for example, Millsap, 1996) refers to striped and mirrored configurations as RAID 0 + 1. The Compaq hardware used as an example configuration in this document supports both configurations. Compaq uses only the RAID 1 term to describe all 100% mirrored configurations in arrays of even-numbered disks. Because the performance of a simple two-drive RAID 1 pair is somewhat different from a striped and mirrored array, the figures for striped and mirrored are presented separately under the RAID 0 + 1 section.

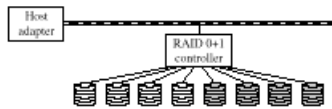
Below is a summary of characteristics of the two-disk array RAID configuration:

- Random read performance: Good; if the implementation uses read-optimized RAID 1 controllers, which read from the drive with the smallest I/O setup cost, then slightly better than an independent disk
- Random write performance: Good (Application write requests are multiplied by two, because the data must be written to two disks. Thus, some of the I/Os-per-second capacity of the two drives is used up by the mirroring function.)

RAID Level 1, Mirroring (continued)

- Sequential read performance: Fair; throughput is limited to the speed of one disk
- Sequential write performance: Fair; same factors as are influencing the random write performance
- Outage frequency: Excellent
- Outage duration: Excellent; for “hot swapable” drives, no application outage is encountered by a single failure
- Performance degradation during outage: Excellent; there is no degradation during a disk outage (After replacing of the failed drive, the resilvering operation that takes place when the failed disk is replaced will consume some of the available I/Os per second capacity.)
- Acquisition cost: Poor; each RAID 1 pair requires two drives to achieve the storage capacity of one
- Operational cost: Fair; increased complexity of the configuration leads to higher training costs and costs to develop custom software to integrate the mirroring procedures into scheduled maintenance operations

RAID Level 0+1, Striping and Mirroring

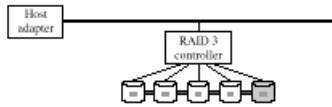


As noted in the previous section, the striped and mirrored configuration is an expansion of the RAID 1 configuration from a simple mirrored pair to an array of even-numbered drives. This configuration offers the performance benefits of RAID 0 striping and the fault tolerance of simple RAID 1 mirroring. The striped and mirrored configuration is especially valuable for Oracle data files holding files with high write rates, such as table data files and online and archived redo log files. Unfortunately, it also presents the high costs of simple RAID 1. The equations for the total I/Os per second and disk drive size calculations for RAID 0 + 1 are identical to those presented for RAID 1 above. The Compaq SMART array controller used in the example configuration supports RAID 0 + 1 (RAID 1 in Compaq terminology) in arrays up to 14 drives, providing the effective storage of 7 drives.

Below is a summary of characteristics of RAID 0 + 1 storage arrays:

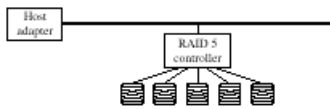
- Random read performance: Excellent under all concurrency levels if each I/O request fits within a single striping segment (Using a stripe size that is too small can cause dramatic performance breakdown at high concurrency levels.)
- Random write performance: Good (Application write requests are multiplied by two because the data must be written to two disks. Thus, some of the I/Os-per-second capacity of the two drives is used up by the mirroring function.)
- Sequential read performance: Excellent under all concurrency levels if each I/O request fits within a single striping segment
- Sequential write performance: Good
- Outage frequency: Excellent; same as RAID 1
- Outage duration: Excellent; same as RAID 1
- Performance degradation during outage: Excellent; there is no degradation during a disk outage (The resilvering operation that takes place when the failed disk is replaced will consume a significant amount of the available I/Os-per-second capacity.)
- Acquisition cost: Poor; same as RAID 1
- Operational cost: Fair; same as RAID 1

RAID Level 3, Bit Interleaved Parity



In the RAID 3 configuration, disks are organized into arrays in which one disk is dedicated to storage of parity data for the other drives in the array. The stripe size in RAID 3 is 1 bit. This enables recovery time to be minimized, because data can be reconstructed with a simple exclusive-OR operation. However, using a stripe size of 1 bit reduces I/O performance. RAID 3 is not recommended for storing any Oracle database files. Also, RAID 3 is not supported by the Compaq SMART array controllers.

RAID Level 5, Block-Interleaved with Distributed Parity



RAID 5 is similar to RAID 3, except that RAID 5 striping segment sizes are configurable, and RAID 5 distributes parity across all the disks in an array. A RAID 5 striping segment contains either data or parity.

Battery-backed cache greatly reduces the impact of this overhead for write calls, but its effectiveness is implementation-dependent. Large write-intensive batch jobs generally fill the cache quickly, reducing its ability to offset the write-performance penalty inherent in the RAID 5 definition.

The total I/Os per second load applied to a RAID 5 array is calculated as follows:

$$\text{total I/O per second load on array} = (\text{reads/transaction} + 4 * \text{writes/transaction}) * \text{transactions/second}$$

The writes/transaction figure is multiplied by four because the parity data must be written in a six-step process:

1. Read the data drive containing the old value of the data to be overwritten. This requires one I/O.
2. Read the parity drive. This requires one I/O.
3. Subtract the contribution of the old data from the parity value.
4. Add the contribution of the new data to the parity value.
5. Write the new value of the parity requiring one I/O.
6. Write the new data value to the data drive. This requires one I/O.

Summing up all I/Os in this process yields four I/Os required for each write requested by the application. This is the main reason that RAID 5 is not recommended for storing files with a high I/O performance requirement; the 4 multiplier reduces the effective I/Os-per-second capacity of the array.

The size of the drive required is:

$$\text{drive size} = [\text{total space required by application} / (\text{total number drives} - \text{number of arrays})] \text{ Rounded up to next drive size.}$$

RAID Level 5, Block-Interleaved with Distributed Parity (continued)

Note that the figure “number of arrays” is used to account for the space of one drive per array consumed by the parity data. If it is necessary to exceed the maximum recommended array size to meet the I/Os-per-second performance requirement, then multiple arrays are required.

Below is a summary of the characteristics of RAID 5 storage arrays:

- Random read performance: Excellent under all concurrency levels if each I/O request fits within a single striping segment (Using a stripe size that is too small can cause dramatic performance breakdown at high concurrency levels.)
- Random write performance: Poor; worst at high concurrency levels (The read-modify-write cycle requirement of RAID 5 parity implementation reduces the effective throughput or I/Os-per-second capacity of the array, especially for heavy write I/O files. It should be noted, however, that under light load, response time is not degraded from that provided by a faster array configuration such as RAID 0. This is due to the asynchronous write capabilities provided by most array controllers. With asynchronous write, the application does not have to wait until the storage subsystem has completed the read-modify-write cycle before continuing. Instead, the controller buffers the data in battery-backed RAM, signals the application that the write has completed, then completes the write to disk. (This buffering does nothing to increase throughput, however.)
- Sequential read performance: Excellent under high concurrency levels if each I/O request fits within a single striping segment; also excellent with fine grain striping under low concurrency levels
- Sequential write performance: Fair for low concurrency levels, poor for high concurrency levels (See random write performance.)
- Outage frequency: Good; can withstand the loss of any single disk in a given array without incurring an application outage (Multiple simultaneous disk failures causes application outage. The possibility of multiple simultaneous failures increase as the size of the array increases.)
- Outage duration: Good; a single disk failure causes no application outage
- Performance degradation during outage: Fair; there is no degradation for reads and writes to or from surviving drives in the array (Reads and writes to a failed drive incur a high performance penalty, requiring data from all surviving drives in the array to be read. Reconstruction of the failed drive’s data also degrades performance.)
- Acquisition cost: Fair (If storage capacity were the only factor, the cost would be $g/(g-1)$ times the cost of the equivalent RAID 0 capacity, where g is the number of disks in the array. However, when factoring I/Os-per-second performance requirement, the cost can meet or exceed the cost of a RAID 0 + 1 implementation.)
- Operational cost: Fair (Training is required to configure striped disk arrays for optimal performance.)

Ranking of RAID Levels Against Oracle File Types

The following table provides relative rankings for RAID configurations for specific Oracle file types. The rankings range from 1 (Best) to 5 (Worst). Adapted from Millsap, 1996, page 13.

Query	None	0	1	0+1	3	5
Control file performance	2	1	2	1	5	3
Redo log file performance	4	1	5	1	2	3
System tablespace performance	2	1	2	1	5	3
Sort segment performance	4	1	5	1	2	3
Rollback segment performance	2	1	2	1	5	5
Indexed read-only datafiles	2	1	2	1	5	1
Sequential read-only datafiles	4	1	5	1	2	3
DBWn intensive data files	1	1	2	1	5	5
Direct load-intensive data files	4	5	1	1	2	2
Data protection	4	5	1	1	2	2
Acquisition and operating costs	1	1	5	5	3	3