Session 4

Monitoring and Diagnostics

Advanced RAC
Auckland - May 2008

© 2008 Julian Dyke

juliandyke.com
Agenda

- Enterprise Manager
- Automatic Workload Repository (AWR)
- Active Session History (ASH)
- Dynamic Performance Views
- Server Alerts
- Trace and Diagnostics
Enterprise Manager Overview

- In Oracle 10.1 and above
  - Database Control
    - Can only manage one database
    - Manages database, instance, listener and nodes
  - Grid Control
    - Can manage multiple databases
    - Manages database, instance listener and nodes
    - Also manages
      - Application servers
      - HTTP servers
      - Web Applications
      - Data Guard Standby databases

In Oracle 10.1 and above, EM is an HTML-based tool with RAC-specific administration and performance monitoring features. The EM Console provides a GUI-based central point of control for the entire Oracle environment.

There are two versions of EM in Oracle 10.1 and above: Database Control and Grid Control. Database Control allows you to manage a single RAC database and its associated instances, listeners, and nodes; Grid Control allows you to manage multiple RAC databases and instances, listeners, nodes, application servers, HTTP servers, and web applications. It also allows you to create Data Guard standby databases.

EM enables you to start, stop, and monitor databases, instances, and listeners. It also allows you to create and assign resource plans; administer storage, such as undo tablespaces and redo logs; manage archive logging; administer ASM; schedule backup and recovery jobs; modify parameters; set the alert threshold; and manage the database scheduler, schemas, security, and storage. EM can also be used to display current host configuration, including memory, CPU, device I/O, network interfaces, and the operating system version. It can be used to apply Oracle patches.
Enterprise Manager
Database Control

- Database Control is
  - Optionally installed by DBCA during database creation
  - Repository is stored within managed database
  - Can only manage a single RAC database
  - Uses EM Agent on each node to communicate with managed objects
    - One EM Agent daemon for each database on each node

- The first database in the cluster uses port 1158 for EM Agent
- Subsequent databases use ports 5500, 5501

- For example, EM Database Control for first database is accessed using:
  
  http://london1:1158/em

If you use the Database Creation Assistant (DBCA) to create your database, then EM Database Control will be automatically configured for your RAC database and an EM Agent will be configured for each node in the cluster to perform database and instance discovery.

You can run EM for the RAC database in a browser using the URL returned by DBCA at the end of the database creation procedure, for example, http://london1:1158/em.

If the database is currently open, this URL causes the Login to Database page to be displayed.
EM Database Control uses the EM Agent to communicate with the database, instances and other processes. You can check if the EM Agent is currently running using the following command:

```
[oracle@london1 oracle]$ emctl status dbconsole
```

If the Agent is running, this command will return output similar to the following:

```
Oracle Enterprise Manager 10g Database Control Release 10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation. All rights reserved.
http://london1:1158/em/console/aboutApplication
Oracle Enterprise Manager 10g is running.
```

Logs are generated in directory
```
/u01/app/oracle/product/10.2.0/db_1/london1_RAC1/sysman/log
```

If the agent is not running, you can start it as follows:

```
[oracle@london1 oracle]$ emctl start dbconsole
```

The agent can be stopped again using the following:

```
[oracle@london1 oracle]$ emctl stop dbconsole
```
While Database Control is installed automatically by DBCA, Grid Control must be manually installed separately. Grid Control has a repository that can be stored in an existing database or a stand-alone database.
A number of optional management packs can be purchased for Enterprise Manager. These increase the amount of available functionality.

Optional management packs include:

- Oracle Database Change Management Pack
- Oracle Database Configuration Management Pack
- Oracle Database Diagnostic Pack
- Oracle Database Tuning Pack
- Oracle Database Provisioning Pack (Oracle 10.2 and above)

The Change Management pack must be installed with the Java Console version of OEM; the remaining packs run with both Enterprise Manager Database Control and Enterprise Manager Grid Control.

Optional management packs cannot be used on databases with the Oracle Database Standard Edition i.e. you must have Oracle Database Enterprise Edition licenses.
The diagnostic pack license is also required to query
• V$ACTIVE_SESSION_HISTORY dynamic performance view
• X$ASM fixed view
• DBA_HIST% data dictionary views
• DBA_ADVISOR% data dictionary views if queries to these views return rows with the value ADDR in the ADVISOR_NAME column or a value of ADDM in the TASK_NAME column or the corresponding TASK_ID

You also need the Diagnostic Pack license to run the following scripts in $ORACLE_HOME/rdbms/admin
• admrpt.sql, admrpti.sql
• awrrpt.sql, awrrpti.sql
• ashrpt.sql, ashrpti.sql
• awrddrpt.sql, awrddrpti.sql
• awrsq rpt.sql, awrsq rpti.sql
Enterprise Manager
Tuning Pack

◆ Licensed cost option

◆ Features include:
  ◆ SQL Access Advisor
  ◆ SQL Tuning Advisor
  ◆ SQL Tuning Sets
  ◆ Object reorganization
Enterprise Manager
Performance Page

Cluster Database: RAC

Cluster Host Load Average

Global Cache Block Access Latency (Current Up Instances: 2/2)

Average Active Sessions (Current Up Instances: 2/2)
Enterprise Manager Performance Page

Continued

Database Throughput (Current Up Instances: 2/2)

Additional Monitoring Links
- Database Links
  - Top Sessions
  - Cluster Cache Granularity
- Top Consumers
- Top Segments

Additional Instance Monitoring Links
- Top Activity
- Duplicate SQL
- Baseline Normalized Metrics
- Search Sessions

© 2008 Julian Dyke
juliandyke.com
Enterprise Manager
Cluster Cache Coherency Page

Cluster Database IAC vs Cluster Cache Coherency

Cluster Cache Coherency
Logged In As SY1

Graphical Chart: Tabular Format
Charts below show time spent by all instances waiting to receive database blocks, global cache transfer rate, and disk & network I/O activity in comparison to logical reads and DB operation delays.

Global Cache Block Access Latency

Global Cache Block Transfer Rate

Global Cache Block Transfers and Physical Reads (vs. Logical Reads)

Note: The latency for IAC operations should be monitored over time, and significant increases in the values should be investigated. The typical average receive time for a DB block is 5 milliseconds, and the upper bound is 10; the typical average receive time for a current block is 5 milliseconds, and the upper bound is 10.
Automatic Workload Repository

Introduction

- Introduced in Oracle 10.1
  - Requires Enterprise Manager Diagnostics Pack

- Can create
  - snapshots
  - baselines

- A baseline is the difference between two snapshots

- By default snapshots are taken every 60 minutes

- DBMS_WORKLOAD_REPOSITORY package includes subroutines to:
  - manage AWR contents
  - generate AWR reports

AWR content management subroutines

CREATE_BASELINE / DROP_BASELINE
CREATE_SNAPSHOT / DROP_SNAPSHOT_RANGE
MODIFY_SNAPSHOT_SETTINGS

AWR report generation subroutines

ASH_REPORT_HTML / ASH_REPORT_TEXT
AWR_DIFF_REPORT_HTML / AWR_DIFF_REPORT_TEXT
AWR_REPORT_HTML / AWR_REPORT_TEXT
AWR_SQL_REPORT_HTML / AWR_SQL_REPORT_TEXT
AWR Parameters

- Reported in `DBA_HIST_WR_CONTROL`
- Configured using `DBMS_WORKLOAD_REPOSITORY`
- Include:
  - Snap Interval (default 60 minutes)
  - Retention (default 7 days)
  - Top SQL (default dependent on `STATISTICS_LEVEL`)
    - `TYPICAL` - 30
    - `ALL` - 100

```sql
SELECT * FROM dba_hist_wr_control;
```

<table>
<thead>
<tr>
<th>DBID</th>
<th>SNAP_INTERVAL</th>
<th>RETENTION</th>
<th>TOPNSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2467148022</td>
<td>+000000</td>
<td>+00007</td>
<td>DEFAULT</td>
</tr>
</tbody>
</table>
AWR Parameters

- **SNAP INTERVAL**
  - Interval between each snapshot specified in minutes
  - Minimum interval is 10 minutes
  - Maximum interval is 52,560,000 minutes

- If zero is specified, automatic and manual snapshots will be disabled

- Default value is 60 minutes

- To set interval to 10 minutes (minimum interval)

  ```sql
  DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS -
  (interval=>10);
  ```
AWR Parameters

- RETENTION
  - Retention time specified in minutes
  - Minimum retention is 1 day (1440 minutes)
  - Maximum retention is 100 years

  - If zero is specified, snapshots will be retained forever

  - Default value is 7 days (10080 minutes)

  - To set retention time to 30 days (43200 minutes)

```
DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS -
(retention=>43200);
```
TOPNSQL

Values are DEFAULT, MAXIMUM or <n>
If <n> is NUMBER e.g. 30
• Minimum value is 30
• Maximum value is 100,000,000
Value unaffected by STATISTICS_LEVEL parameter
If <n> is VARCHAR2 e.g. '30'
• Default value depends on STATISTICS_LEVEL parameter
If STATISTICS_LEVEL = TYPICAL then default value is 30
If STATISTICS_LEVEL = ALL then default value is 100
Snapshots are reported in the DBA_HIST_SNAPSHOT data dictionary view.

DBA_HIST_SNAPSHOT is based on the SYS.WRM$_SNAPSHOT table.

WRM$_SNAPSHOT.SNAP_FLAG describes how snapshot was taken:
- 0 - automatic
- 1 - manual
AWR Baselines

- A baseline represents the difference (delta) between two snapshots
- Can be stored for future comparison
- Reported in DBA_HIST_BASELINE
- For example to create a baseline called BASELINE1 between snapshots 1740 and 1750 use:

  ```sql
  DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE(
    start_snap_id => 1740,
    end_snap_id => 1750,
    baseline_name => 'BASELINE1'
  );
  ```

When baseline is created from two snapshots
- Both snapshots are copied from the WRH$_% table to the equivalent WRH$_%_BL table
- For example if a baseline is created for snapshots 1740 to 1750 then for the system statistics the following SQL will be executed

  ```sql
  INSERT INTO wrh$_sysstat_bl
  SELECT * FROM wrh$_sysstat tab
  WHERE :beg_snap = tab.snap_id
  AND tab.snap_id <= :end_snap
  AND tab.dbid = :dbid
  AND EXISTS
  (  SELECT 1 FROM wrm$_snapshot s
  WHERE s.dbid = tab.dbid
  AND s.snap_id = tab.snap_id
  AND s.instance_number = tab.instance_number
  AND s.status = 0
  AND s.bl_moved = 0
  );
  ```
AWR
Dynamic Performance Views

- **DBA_HIST views**

<table>
<thead>
<tr>
<th>View</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBA_HIST_ACTIVE_SESSION_HISTORY</td>
<td>DBA_HIST_LATCH</td>
</tr>
<tr>
<td>DBA_HIST_BASELINE</td>
<td>DBA_HIST_LATCH_CHILDREN</td>
</tr>
<tr>
<td>DBA_HIST_BG_EVENT_SUMMARY</td>
<td>DBA_HIST_LATCH_MISSES_SUMMARY</td>
</tr>
<tr>
<td>DBA_HIST_BUFFERED_QUEUES</td>
<td>DBA_HIST_LATCH_NAME</td>
</tr>
<tr>
<td>DBA_HIST_BUFFERED_SUBSCRIBERS</td>
<td>DBA_HIST_LATCH_PARENT</td>
</tr>
<tr>
<td>DBA_HIST_BUFFER_POOL_STAT</td>
<td>DBA_HIST_LIBRARYCACHE</td>
</tr>
<tr>
<td>DBA_HIST_COMP_IOSSTAT</td>
<td>DBA_HIST_LOG</td>
</tr>
<tr>
<td>DBA_HIST_CR_BLOCK_SERVER</td>
<td>DBA_HIST_METRIC_NAME</td>
</tr>
<tr>
<td>DBA_HIST_CURRENT_BLOCK_SERVER</td>
<td>DBA_HIST_MTTR_TARGET_ADVICE</td>
</tr>
<tr>
<td>DBA_HIST_DATABASE_INSTANCE</td>
<td>DBA_HIST_OPTIMIZER_ENV</td>
</tr>
<tr>
<td>DBA_HIST_DATAFILE</td>
<td>DBA_HIST_OSTAT</td>
</tr>
<tr>
<td>DBA_HIST_DB_CACHE_ADVICE</td>
<td>DBA_HIST_OSTAT_NAME</td>
</tr>
<tr>
<td>DBA_HIST_DLM_MISC</td>
<td>DBA_HIST_PARAMETER</td>
</tr>
<tr>
<td>DBA_HIST_ENQUEUE_STAT</td>
<td>DBA_HIST_PARAMETER_NAME</td>
</tr>
<tr>
<td>DBA_HIST_EVENT_NAME</td>
<td>DBA_HIST_PGASTAT</td>
</tr>
<tr>
<td>DBA_HIST_FILEMETRIC_HISTORY</td>
<td>DBA_HIST_PGA_TARGET_ADVICE</td>
</tr>
<tr>
<td>DBA_HIST_FILESTATXS</td>
<td>DBA_HIST_PROCESS_MEM_SUMMARY</td>
</tr>
<tr>
<td>DBA_HIST_INSTANCE_RECOVERY</td>
<td>DBA_HIST_RESOURCE_LIMIT</td>
</tr>
<tr>
<td>DBA_HIST_INST_CACHE_TRANSFER</td>
<td>DBA_HIST_ROWCACHE_SUMMARY</td>
</tr>
<tr>
<td>DBA_HIST_JAVA_POOL_ADVICE</td>
<td>DBA_HIST_RULE_SET</td>
</tr>
<tr>
<td>DBA_HIST_LATCH</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_LATCH_CHILDREN</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_LATCH_MISSES_SUMMARY</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_LATCH_NAME</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_LATCH_PARENT</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_LIBRARYCACHE</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_LOG</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_METRIC_NAME</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_MTTR_TARGET_ADVICE</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_OSTAT_NAME</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_OSTAT</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_PARAMETER</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_PARAMETER_NAME</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_PGASTAT</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_PGA_TARGET_ADVICE</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_PROCESS_MEM_SUMMARY</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_RESOURCE_LIMIT</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_ROWCACHE_SUMMARY</td>
<td></td>
</tr>
<tr>
<td>DBA_HIST_RULE_SET</td>
<td></td>
</tr>
</tbody>
</table>

For each monitored dynamic performance view there are two tables and one data dictionary view.

For example for V$SYSSTAT:

- `WRH$_SYSSTAT` snapshot table
- `WRH$_SYSSTAT_BL` baseline table
- `DBA_HIST_SYSSTAT` data dictionary view
### AWR Dynamic Performance Views

#### DBA_HIST views continued

<table>
<thead>
<tr>
<th>Table Name</th>
<th>View Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBA_HIST_SEG_STAT</td>
<td>DBA_HIST_SQL_WORKAREA_HSTGRM</td>
</tr>
<tr>
<td>DBA_HIST_SEG_STAT_OBJ</td>
<td>DBA_HIST_STAT_NAME</td>
</tr>
<tr>
<td>DBA_HIST_SERVICE_NAME</td>
<td>DBA_HIST_STREAMS_APPLY_SUM</td>
</tr>
<tr>
<td>DBA_HIST_SERVICE_STAT</td>
<td>DBA_HIST_STREAMS_CAPTURE</td>
</tr>
<tr>
<td>DBA_HIST_SERVICE_WAIT_CLASS</td>
<td>DBA_HIST_STREAMS_POOL_ADVICE</td>
</tr>
<tr>
<td>DBA_HIST_SESSMETRIC_HISTORY</td>
<td>DBA_HIST_SYMMETRIC_HISTORY</td>
</tr>
<tr>
<td>DBA_HIST_SESS_TIME_STATS</td>
<td>DBA_HIST_SYMMETRIC_SUMMARY</td>
</tr>
<tr>
<td>DBA_HIST_SGA</td>
<td>DBA_HIST_SYSSTAT</td>
</tr>
<tr>
<td>DBA_HIST_SGASTAT</td>
<td>DBA_HIST_SYSTEM_EVENT</td>
</tr>
<tr>
<td>DBA_HIST_SGA_TARGET_ADVICE</td>
<td>DBA_HIST_SYS_TIME_MODEL</td>
</tr>
<tr>
<td>DBA_HIST_SHARED_POOL_ADVICE</td>
<td>DBA_HIST_TABLESPACE_STAT</td>
</tr>
<tr>
<td>DBA_HIST_SNAPSHOT</td>
<td>DBA_HIST_TBSPC_SPACE_USAGE</td>
</tr>
<tr>
<td>DBA_HIST_SNAP_ERROR</td>
<td>DBA_HIST_TEMPFILE</td>
</tr>
<tr>
<td>DBA_HIST_SQLBIND</td>
<td>DBA_HIST_TEMPSTATXS</td>
</tr>
<tr>
<td>DBA_HIST_SQLSTAT</td>
<td>DBA_HIST_THREAD</td>
</tr>
<tr>
<td>DBA_HIST_SQLTEXT</td>
<td>DBA_HIST_UNDOSTAT</td>
</tr>
<tr>
<td>DBA_HIST_SQL_BIND_METADATA</td>
<td>DBA_HIST_WAITCLASSMET_HISTORY</td>
</tr>
<tr>
<td>DBA_HIST_SQL_PLAN</td>
<td>DBA_HIST_WAITSTAT</td>
</tr>
<tr>
<td>DBA_HIST_SQL_SUMMARY</td>
<td>DBA_HIST_WR_CONTROL</td>
</tr>
</tbody>
</table>

There are also five AWR control tables in the SYS schema. Each control table has a WRMS$ prefix.

<table>
<thead>
<tr>
<th>Table Name</th>
<th>View Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRMS_BASELINE</td>
<td>DBA_HIST_BASELINE</td>
</tr>
<tr>
<td>WRMS_DATABASE_INSTANCE</td>
<td>DBA_HIST_DATABASE_INSTANCE</td>
</tr>
<tr>
<td>WRMS_SNAPSHOT</td>
<td>DBA_HIST_SNAPSHOT</td>
</tr>
<tr>
<td>WRMS_SNAP_ERROR</td>
<td>DBA_HIST_SNAP_ERROR</td>
</tr>
<tr>
<td>WRMS_WR_CONTROL</td>
<td>DBA_HIST_WR_CONTROL</td>
</tr>
</tbody>
</table>

Note that DBA_HIST_BASELINE is a join between WRMS$_BASELINE and two instances of WRMS$_SNAPSHOT.
AWR Report

AWR reports can be generated using:
- Enterprise Manager Grid Control
- the DBMS_WORKLOAD_REPOSITORY package
- the $ORACLE_HOME/rdbms/admin/awrrpt.sql script

- By default the AWR report will be generated for the current instance. To specify another instance use
  - $ORACLE_HOME/rdbms/admin/awrrpti.sql

- AWR reports can be generated in
  - HTML format
  - text format

AWR reports can be generated using DBMS_WORKLOAD_REPOSITORY

For example to generate an AWR report in text format for

<table>
<thead>
<tr>
<th>Database ID</th>
<th>2467148022</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance Number</td>
<td>1</td>
</tr>
<tr>
<td>Start snapshot</td>
<td>1736</td>
</tr>
<tr>
<td>End snapshot</td>
<td>1737</td>
</tr>
</tbody>
</table>

SET PAGESIZE 0
SET LINESIZE 200
SET TRIMSPOOL ON
SET HEADING OFF
SPOOL awrrpt.lst
SELECT output FROM TABLE (DBMS_WORKLOAD_REPOSITORY.AWR_REPORT_TEXT (2467148022, 1, 1736, 1737));
SPOOL OFF
The next two slides show the sections for a typical Oracle 10.2.0.3 AWR report. Some of the sections are optional e.g. the SQL and segment statistics sections.

Some sections only appear in AWR reports for RAC instances. These are shown in blue.
### Oracle 10.2 sections (continued)

- Shared Pool Advisory
- SGA Target Advisory
- Streams Pool Advisory
- Java Pool Advisory
- Enqueue Activity
- Undo Segment Summary
- Undo Segment Statistics
- Latch Activity
- Latch Sleep Breakdown
- Latch Miss Sources
- Segments by Logical Reads
- Segments by Physical Reads
- Segments by Row Lock Waits
- Segments by ITL Waits
- Segments by Buffer Busy Waits
- Dictionary Cache Statistics
- Library Cache Activity
- Process Memory Summary
- SGA Memory Summary
- SGA Breakdown Difference
- Streams CPU/IO Usage
- Streams Capture
- Streams Apply
- Buffered Queues
- Buffered Subscribers
- Rule Set
- Resource Limit Statistics
- Initialization Parameters
- Global Enqueue Statistics
- Global CR Served Statistics
- Global Current Served Statistics
- Global Cache Transfer Statistics

Additional ADDM sections can be included in the AWR report by specifying a value of 8 for the L_OPTION parameter.

For example:

```sql
SELECT output FROM TABLE (DBMS_WORKLOAD_REPOSITORY.AWR_REPORT_TEXT (2467148022, 1, 1736, 1737, 8 ));
```

Includes the following additional sections:

- Buffer Cache Advisory
- Shared Pool Advisory
- PGA Target Advisory
- Shared Pool Advisory
- SGA Target Advisory
### AWR Report Summary

**WORKLOAD REPOSITORY report for**

<table>
<thead>
<tr>
<th>DB Name</th>
<th>DB Id</th>
<th>Instance</th>
<th>Inst Num</th>
<th>Release</th>
<th>RAC Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROD</td>
<td>2149809470</td>
<td>PROD1</td>
<td>1</td>
<td>10.2.0.3.0</td>
<td>YES server3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Snap Id</th>
<th>Snap Time</th>
<th>Sessions</th>
<th>Curs/Sess</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin</td>
<td>1768</td>
<td>10-Aug-07 21:00:28</td>
<td>55 #########</td>
</tr>
<tr>
<td>End</td>
<td>1769</td>
<td>10-Aug-07 22:00:40</td>
<td>57 #########</td>
</tr>
<tr>
<td>Elapsed</td>
<td></td>
<td>60.19 (mins)</td>
<td></td>
</tr>
<tr>
<td>DB Time</td>
<td></td>
<td>7.72 (mins)</td>
<td></td>
</tr>
</tbody>
</table>

**Cache Sizes**

<table>
<thead>
<tr>
<th></th>
<th>Begin</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer Cache:</td>
<td>1.232M</td>
<td>1.232M</td>
</tr>
<tr>
<td>Shared Pool Size:</td>
<td>224M</td>
<td>224M</td>
</tr>
<tr>
<td>Std Block Size:</td>
<td>8K</td>
<td>8K</td>
</tr>
<tr>
<td>Log Buffer:</td>
<td>15,192K</td>
<td></td>
</tr>
</tbody>
</table>

This section of an AWR report describes the database, instance, host and Oracle version on which the report was run. It also shows the start and stop snapshots from which the report was derived together with the elapsed time between them.

Use this information as a sanity check that you are looking at the right report.

If you are comparing against a baseline, check that both reports have been run against the same version of Oracle. If you are experiencing performance problems and the Oracle releases differ between the baseline and the current reports, then you may want to review the values of all parameters including hidden parameters and also to examine the top SQL statements. In this case it is useful to have a baseline which has been run at snapshot level 6 or above so that you can compare the execution plans.

The second section of the report details the cache sizes at the times of the begin and end snapshots.

This is particularly important if you are using Automatic Memory Management as the cache sizes may vary between your baseline and your current report as Oracle attempts to fine-tune the memory structures.
This section of the report shows the load profile for the interval covered by the report. This gives some important basic information. If you have a reasonably predictable workload, you can use "Redo Size" to estimate roughly how much DML activity is occurring on the instance.

The "Logical Reads" statistic gives an indication of how much work your system needed to perform during the period; the "Physical Reads" statistics tells you how much of this work required disk I/Os. Remember that in most databases reducing the overall number of logical reads yields greater performance improvements than concentrating on reducing the number of physical reads.

Compare the number of hard parses to the total number of parses. This gives an indication of how many times sessions were able to reuse an existing cursor in the library cache. Sometimes it is possible to reduce the number of hard parses by setting database parameters or modifying the application; at other times the number of hard parses cannot be reduced as it is dependent on the application. For example, in a data warehouse environment where users issue ad-hoc queries, every statement may require a hard parse.

Also compare the overall number of parses to the number of executions. This gives an indication of how many times a session executed the same statement without needing to reparse it. If this ratio starts to decrease as new users are introduced then you may need to increase the amount of memory available in the SGA if you are using Automatic Memory Management, or to increase the SHARED_POOL_SIZE parameter otherwise.
### AWR

#### Instance Efficiency Percentages

<table>
<thead>
<tr>
<th>Instance Efficiency Percentages (Target 100%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer NoWait %: 99.98</td>
</tr>
<tr>
<td>Buffer Hit %: 99.90</td>
</tr>
<tr>
<td>Library Hit %: 99.99</td>
</tr>
<tr>
<td>Execute to Parse %: 4.66</td>
</tr>
<tr>
<td>Parse CPU to Parse Elapsed %: 104.83</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shared Pool Statistics</th>
<th>Begin</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Usage %:</td>
<td>86.09</td>
<td>86.11</td>
</tr>
<tr>
<td>% SQL with executions&gt;1:</td>
<td>85.59</td>
<td>86.53</td>
</tr>
<tr>
<td>% Memory for SQL w/exec&gt;1:</td>
<td>75.22</td>
<td>76.49</td>
</tr>
</tbody>
</table>

This section shows Instance Efficiency Percentages. A general rule when dealing with ratios is that a low ratio may be indicative of a performance problem; however, a high ratio does not guarantee that your system is performing well.

The next section shows statistics for overall memory usage for the begin and end snapshots. It shows the percentage of the SGA currently in use by the shared pool and also the percentage used for SQL statements and their execution plans.
This section shows the top 5 timed events in the system which is one of the most important sections of the AWR report. I recommend that you make a habit of checking this section on every report. Timed events show where time has been spent by Oracle, either working or waiting to work.

The Top 5 timed Events are also listed later in the report in the Wait Events section. However, it is normally the top five events that have the most significant impact and you should concentrate your tuning efforts on these.

On a well-tuned system, you would expect one of the top waits to be "log file sync", "db file sequential read" or "db file scattered read". These waits indicate that the system has hit an I/O bottleneck. If you need to improve performance you will need to investigate methods of reducing the impact of I/O on system performance. There are a number of techniques for achieving this including distributing the I/O across storage more evenly, eliminating unnecessary statements, reducing the amount of redo generated, modifying the physical database design or tuning application statements.

On a RAC instance, you should expect to see a large number of waits for Global Cache Services. These all have the prefix "gc". Depending on your availability requirements and application design these may be unavoidable. If the number of waits or the time waited is excessive then you may be able to reduce the amount of global cache traffic in a number of ways including eliminating necessary statements, removing redundant indexes or eliminating block contention by reducing the number of rows on a block or by using smaller block sizes.
### Global Cache Statistics and Ratios

#### RAC Statistics
**DB/Inst:** PROD/PROD1  **Snaps:** 1768-1769

| Number of Instances: | 2 | 2 |

#### Global Cache Load Profile

<table>
<thead>
<tr>
<th></th>
<th>Per Second</th>
<th>Per Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Cache blocks received:</td>
<td>21.11</td>
<td>3.25</td>
</tr>
<tr>
<td>Global Cache blocks served:</td>
<td>21.11</td>
<td>3.31</td>
</tr>
<tr>
<td>GCS/GES messages received:</td>
<td>49.84</td>
<td>7.67</td>
</tr>
<tr>
<td>GCS/GES messages sent:</td>
<td>49.84</td>
<td>7.58</td>
</tr>
<tr>
<td>DBWR Fusion writes:</td>
<td>0.88</td>
<td>0.14</td>
</tr>
<tr>
<td>Estd Interconnect traffic (KB)</td>
<td>360.13</td>
<td></td>
</tr>
</tbody>
</table>

#### Global Cache Efficiency Percentages (Target local+remote 100%)

| Buffer access - local cache %: | 95.99 |
| Buffer access - remote cache %:  | 3.92  |
| Buffer access - disk %:          | 0.10  |

This section is RAC specific and is not included for single-instances.

The Global Cache Load Profile presents a summary of the traffic across the interconnect in terms of blocks exchanged by Global Cache Services (GCS) and messages exchanged by both GCS and Global Enqueue Services (GES).

The Global Cache Efficiency Percentages section reports the percentage of blocks accessed from local cache, remote cache and from disk. In an optimum system, the percentage of local cache accesses should approach 100% while the percentage of remote cache accesses and disk accesses should both approach 0%. It is usually more expensive to read a block locally from disk than it is to obtain it from a remote cache, therefore you should concentrate on reducing the amount of disk I/O first.
### Global Cache and Enqueue Services - Workload Characteristics

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg global enqueue get time (ms):</td>
<td>3.8</td>
</tr>
<tr>
<td>Avg global cache cr block receive time (ms):</td>
<td>0.7</td>
</tr>
<tr>
<td>Avg global cache current block receive time (ms):</td>
<td>0.5</td>
</tr>
<tr>
<td>Avg global cache cr block build time (ms):</td>
<td>0.0</td>
</tr>
<tr>
<td>Avg global cache cr block send time (ms):</td>
<td>0.0</td>
</tr>
<tr>
<td>Avg global cache cr block flush time (ms):</td>
<td>2.2</td>
</tr>
<tr>
<td>Avg global cache current block pin time (ms):</td>
<td>0.1</td>
</tr>
<tr>
<td>Avg global cache current block send time (ms):</td>
<td>0.0</td>
</tr>
<tr>
<td>Global cache log flushes for cr blocks served %:</td>
<td>16.8</td>
</tr>
<tr>
<td>Global cache log flushes for current blocks served %:</td>
<td>0.8</td>
</tr>
<tr>
<td>Avg global cache current block flush time (ms):</td>
<td>0.5</td>
</tr>
</tbody>
</table>

The Global Cache and Enqueue Services - Workload Characteristics section describes the average times required to perform various GCS and GES tasks. Of these statistics, the most significant are the enqueue get time which should ideally be below 1ms and the global cache cr and current block receive times which should be less than 1ms.
The Global Cache and Enqueue Services - Messaging Statistics section describes average time to exchange different categories of inter-instance messages.
The Time Model Statistics section was introduced in Oracle 10.1. It is included in AWR reports for both single-instance and RAC databases.

Time Model Statistics provide a breakdown of the CPU time in relation to overall database time. The total amount of database time over the snapshot period is reported by the DB Time statistic. All other statistics are reported both in absolute terms and as a percentage of DB time.

In the above example, SQL execution time represents 92.5% of total database time and is therefore the most significant component. In a well-tuned system, SQL execution time should always be the most significant component of DB time. However, you cannot deduce from this section whether the SQL execution is efficient or otherwise. It is necessary to inspect other sections of the AWR report to determine this.
AWR
Top SQL Statements

- This section lists SQL statements currently in the library cache that exceed specified thresholds

- The statements are reported in a series of tables ordered on different criteria including:
  - By Elapsed Time
  - By CPU Time
  - By Gets
  - By Reads
  - By Executions
  - By Parse Calls
  - By Sharable Memory
  - By Version Count
  - By Cluster Wait Time

SQL Ordered by Elapsed Time - This section reports the statements consuming the largest amount of elapsed time. For each statement AWR reports the number of executions, elapsed time per execution, CPU time and number of physical reads.

SQL Ordered By CPU Time - This section reports the statements consuming the largest amount of CPU time. For each statement AWR reports the number of executions, CPU per execution, elapsed time and number of buffer gets.

SQL Ordered By Buffer Gets - For each SQL statement AWR reports the number of logical reads, executions, logical reads per execution, and the percentage that the logical reads for the statement represents of the total number of logical reads during the snapshot period. The total CPU time and total elapsed time consumed by the statement are also reported.

SQL Ordered by Physical Reads - For each SQL statement AWR reports the number of physical reads, executions, physical reads per execution and the percentage that the number of physical reads for the statement represents of the total number of physical reads during the snapshot period. The total CPU time and total elapsed time consumed by the statement are also reported.

SQL Ordered by Executions - For each SQL statement AWR reports the number of executions, rows processed, rows processed per execution, CPU time per execution and elapsed time per execution.
### AWR Top SQL Statements

#### SQL ordered by Elapsed Time

<table>
<thead>
<tr>
<th>Elapsed Time (s)</th>
<th>CPU Time (s)</th>
<th>% Total Elapsed Time</th>
<th>Executions</th>
<th>Exec % Total Time</th>
<th>DB Time</th>
<th>SQL Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>289</td>
<td>0</td>
<td>16</td>
<td>18.0</td>
<td>62.3</td>
<td>4fpxtnab3p56m</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>4</td>
<td>6,064</td>
<td>0.0</td>
<td>5.8</td>
<td>32qvqpcm517s2</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>9</td>
<td>3,417</td>
<td>0.0</td>
<td>4.5</td>
<td>4t30063mp992</td>
<td></td>
</tr>
</tbody>
</table>

**For UPDATE**

```sql
SELECT WL0.NAME, WL0.SCHEDULE_ID FROM ORDER_ITEMS WL0 WHERE (WL0.NAME = :1 )
```

```sql
SELECT WL0.KEY, WL0.CURRENCY, WL0.ENTRY_DATE, WL0.TOKEN FROM CUSTOMER WL0 WHERE ( WL0.KEY = :1 ) FOR UPDATE
```

```sql
SELECT WL0.order_id, WL0.user_id, WL0.orders_placed, WL0.order_amount, WL0.discount, WL0.name, WL0.first_order_date, WL0.cost_centre_id FROM VF_ORDERS WL0 WHERE ( ( WL0.order_id = :1 ) AND ( WL0.user_id= :2 ) ) FOR UPDATE
```

### SQL Ordered by Parse Calls
- For each SQL statement, AWR reports the number of parse calls, the number of execution, and the number of parse calls per execution.

### SQL Ordered by Version Count
- AWR reports statements with more than child cursors than the threshold value which defaults to 20.

### SQL Ordered by Cluster Wait Time
- For each SQL statement, AWR reports the amount of time the statement was involved in waits for cluster resources. The cluster wait time is reported as a percentage of total elapsed time for the statement together with the elapsed time, CPU time and number of executions. As you would expect, this table only appears in AWR reports for RAC instances.

Additionally for each statement, AWR reports the SQL_ID of the statement within the library cache. You can use this value to find the execution plan and other information about the statement if it has not be invalidated or aged out of the cache.

Both SQL statements and PL/SQL statements are reported by STATSPACK. If a PL/SQL block executes SQL statements then the resources consumed will appear for both the PL/SQL block and the SQL statement. Take care not to double count these.
This section reports on activity in the dictionary or row cache. The dictionary cache is divided into a number of subcaches each of which contains one type of dictionary object.

The first subsection lists the number of gets and the percentage of misses for each subcache. This is based on information from the V$ROWCACHE dynamic performance view. When the instance is started, the percentage of misses will initially be relatively high. Thereafter the percentage of misses should decrease as the contents of the cache become less volatile. I have deleted a handful of entries from this section to improve clarity of the slide.

The second subsection is only included for RAC instances and reports the number of GES requests, conflicts and releases for each subcache. Watch for excessive numbers of GES conflicts.
AWR
Library Cache

This section reports on activity in the library cache. The library cache is divided into a number of namespaces. In Oracle 10.1 there are can be up to 64 namespaces. However, only the first eight are reported in this section.

The first subsection is based on the V$LIBRARYCACHE dynamic performance view and lists the number of get requests, the percentage of missed gets, the number of pin requests and the percentage of missed pins, the number of reloads and the number of invalidations. The percentage of misses for gets and pins should be as low as possible for all objects.

The second subsection is only included for RAC instances and reports the number of GES lock requests, pin requests, pin releases, invalidation requests and the number of invalidations. No tuning suggestions can be derived directly from this information.
### Global Enqueue Statistics

**Global Enqueue Statistics**

**DB/Inst:** PROD/PROD1  **Snaps:** 1768-1769

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Total</th>
<th>per Second</th>
<th>per Trans</th>
</tr>
</thead>
<tbody>
<tr>
<td>acks for commit broadcast(actual)</td>
<td>31,344</td>
<td>8.7</td>
<td>1.3</td>
</tr>
<tr>
<td>acks for commit broadcast(logical)</td>
<td>32,230</td>
<td>8.9</td>
<td>1.4</td>
</tr>
<tr>
<td>broadcast msgs on commit(actual)</td>
<td>31,548</td>
<td>8.7</td>
<td>1.3</td>
</tr>
<tr>
<td>broadcast msgs on commit(logical)</td>
<td>31,593</td>
<td>8.7</td>
<td>1.3</td>
</tr>
<tr>
<td>broadcast msgs on commit(wasted)</td>
<td>240</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>dynamically allocated gcs resourc</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>dynamically allocated gcs shadows</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>false posts waiting for scn acks</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>flow control messages received</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>flow control messages sent</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>gcs assume cvt</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>gcs assume no cvt</td>
<td>14,094</td>
<td>3.9</td>
<td>0.6</td>
</tr>
<tr>
<td>gcs ast xid</td>
<td>2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>gcs blocked converts</td>
<td>28,591</td>
<td>7.9</td>
<td>1.2</td>
</tr>
<tr>
<td>gcs blocked cr converts</td>
<td>41,882</td>
<td>11.6</td>
<td>1.8</td>
</tr>
<tr>
<td>gcs compatible basts (global)</td>
<td>22</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>gcs compatible cr basts (global)</td>
<td>3,538</td>
<td>1.0</td>
<td>0.2</td>
</tr>
<tr>
<td>gcs compatible cr basts (local)</td>
<td>1,224</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td>gcs cr basts to PIs</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>gcs cr serve without current lock</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>gcs dbwr flush pi msgs</td>
<td>2,875</td>
<td>0.8</td>
<td>0.1</td>
</tr>
<tr>
<td>gcs dbwr write request msgs</td>
<td>2,012</td>
<td>0.6</td>
<td>0.1</td>
</tr>
<tr>
<td><strong>etc.....</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This section is only included for RAC instances. It reports the statistics for various GES activities together and calculates the number per second and per transaction. It is highly likely that any serious GES problems will appear in the Top 5 Timed Events or the Wait Events sections described above, so this section should be used for further investigation.

The output on the above slide has been truncated. In Oracle 10.2 a total of 75 statistics are reported in this table, shadowing the statistics externalized in V$GES_STATISTICS (formerly V$DLM_MISC)
## AWR Global CR Served Statistics

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR Block Requests</td>
<td>36,733</td>
</tr>
<tr>
<td>CURRENT Block Requests</td>
<td>4,290</td>
</tr>
<tr>
<td>Data Block Requests</td>
<td>36,733</td>
</tr>
<tr>
<td>Undo Block Requests</td>
<td>633</td>
</tr>
<tr>
<td>TX Block Requests</td>
<td>3,657</td>
</tr>
<tr>
<td>Current Results</td>
<td>35,618</td>
</tr>
<tr>
<td>Private results</td>
<td>2,496</td>
</tr>
<tr>
<td>Zero Results</td>
<td>2,909</td>
</tr>
<tr>
<td>Disk Read Results</td>
<td>0</td>
</tr>
<tr>
<td>Fail Results</td>
<td>0</td>
</tr>
<tr>
<td>Fairness Down Converts</td>
<td>4,892</td>
</tr>
<tr>
<td>Fairness Clears</td>
<td>0</td>
</tr>
<tr>
<td>Free GC Elements</td>
<td>0</td>
</tr>
<tr>
<td>Flushes</td>
<td>6,889</td>
</tr>
<tr>
<td>Flushes Queued</td>
<td>0</td>
</tr>
<tr>
<td>Flush Queue Full</td>
<td>0</td>
</tr>
<tr>
<td>Flush Max Time (us)</td>
<td>0</td>
</tr>
<tr>
<td>Light Works</td>
<td>2,554</td>
</tr>
<tr>
<td>Errors</td>
<td>0</td>
</tr>
</tbody>
</table>

This section is only included for RAC instances. It contains various GCS statistics for consistent read block served by the local instance to remote instances and gives a good overview of GCS activity during the reporting period.
### AWR

**Global CURRENT Served Statistics**

Global CURRENT Served Stats  DB/Inst: PROD/PROD1  Snaps: 1768-1769

- Pins = CURRENT Block Pin Operations
- Flushes = Redo Flush before CURRENT Block Served Operations
- Writes = CURRENT Block Fusion Write Operations

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Total</th>
<th>% &lt;1ms</th>
<th>% &lt;10ms</th>
<th>% &lt;100ms</th>
<th>% &lt;1s</th>
<th>% &lt;10s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pins</td>
<td>35,207</td>
<td>99.83</td>
<td>0.01</td>
<td>0.17</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Flushes</td>
<td>289</td>
<td>95.85</td>
<td>4.15</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Writes</td>
<td>3,184</td>
<td>50.85</td>
<td>49.15</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

This section is only included for RAC instances. It contains histograms for GCS operations required to serve current block requests from remote instances including block pinning, flushing redo to disk and write operations. Check this view if you believe that Cache Fusion is causing a bottleneck on systems with high levels of DML activity.
AWR

Global Cache Transfer Statistics

Global Cache Transfer Stats
DB/Inst: PROD/PROD1 Snaps: 1768-1769
-> Immediate (Immed) - Block Transfer NOT impacted by Remote Processing Delays
-> Busy (Busy) - Block Transfer impacted by Remote Contention
-> Congested (Congst) - Block Transfer impacted by Remote System Load
-> ordered by CR + Current Blocks Received desc

<table>
<thead>
<tr>
<th>Inst No</th>
<th>Block</th>
<th>Class</th>
<th>Blocks Received</th>
<th>Immed</th>
<th>Busy</th>
<th>Congst</th>
<th>Blocks Received</th>
<th>Immed</th>
<th>Busy</th>
<th>Congst</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>data block</td>
<td>39,346</td>
<td>82.9</td>
<td>17.1</td>
<td>0</td>
<td>33,999</td>
<td>98.9</td>
<td>1.1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>undo header</td>
<td>3,167</td>
<td>99.4</td>
<td>0.6</td>
<td>0</td>
<td>65</td>
<td>100.0</td>
<td>0.0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>undo block</td>
<td>579</td>
<td>95.3</td>
<td>4.7</td>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>others</td>
<td>99</td>
<td>99.0</td>
<td>1.0</td>
<td>0</td>
<td>368</td>
<td>100.0</td>
<td>0.0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

This section was introduced in Oracle 10.2 and gives an overview of Global Cache transfer activity. The output contains a summary by block class of consistent read blocks and current blocks received from other instances.
Active Session History

Introduction

- Introduced in Oracle 10.1
- Samples every session once a second
  - Records information about any sessions that are currently waiting
- Reported in `V$ACTIVE_SESSION_HISTORY`
- Stored in Automatic Workload Repository (AWR)
  - Samples flushed to `WRH$ACTIVE_SESSION_HISTORY`

Note that in order to use Active Session History, you must have the Enterprise Manager Diagnostics Pack license. This license is required to select from the `V$ACTIVE_SESSION_HISTORY` dynamic performance view and the underlying `X$ASH` fixed view.

ASH data collection is configured using `_ash_enable` parameter. The default value is `FALSE`.

The ASH sample interval is configured using `_ash_sampling_interval` parameter. The default value is 1000 milliseconds (1 second).

The ratio of active samples written to disk is determined by `_asm_disk_filter_ratio` parameter. The default value is 10 in which case one in ten active samples is written to disk.

The size of the ASH buffer is specified by `_ash_size` parameter. The default value is 1048576. However, the minimum size is 2097152.
ASH reports can also be generated using the DBMS_WORKLOAD_REPOSITORY package.

For example to generate an ASH report in text format for

- Database ID: 2467148022
- Instance Number: 1
- Start time: 03-APR-07 16:00
- End time: 03-APR-07 17:00

use the following SQL*Plus script:

```sql
SET PAGESIZE 0
SET LINESIZE 200
SET TRIMSPOOL ON
SET HEADING OFF
SPOOL ashrpt.lst
SELECT output FROM TABLE (DBMS_WORKLOAD_REPOSITORY.ASH_REPORT_TEXT (2467148022, 1, TO_TIMESTAMP ('03-APR-07 16:00:00', 'DD-Mon-YY HH24:MI:SS'), TO_TIMESTAMP ('03-APR-07 17:00:00', 'DD-Mon-YY HH24:MI:SS')));
SPOOL OFF
```
ASH Report

-The default ASH report contains the following sections:

- Summary
- Top User Events
- Top Background Events
- Top Event P1/P2/P3 Values
- Top Service/Module
- Top Client IDs
- Top SQL Command Types
- Top SQL using literals
- Top Blocking Sessions
- Top DB Objects
- Top DB Files
- Top Latches
- Activity Over Time
Dynamic Performance Views
Statistics

- In Oracle 10.2.0.1 there are 347 statistics reported in
  - V$SYSSTAT
  - V$SESSTAT
  - V$MYSTAT

- Statistic names not reported in V$SESSTAT or V$MYSTAT
  - Join these views to V$STATNAME on STATISTIC#

- In Oracle 10.2.0.1 a subset of 28 statistics is reported in
  - V$SERVICE_STATS
  - V$CLIENT_STATS
  - V$SERV_MOD_ACT_STATS

In Oracle 10.2.0.1 there are 347 statistics reported in
- V$STATNAME - reports list of statistic names
- V$SYSSTAT - system statistics
- V$SESSTAT - statistics for all sessions
- V$MYSTAT - statistics for current session

It is not a good idea to specify the statistic# column explicitly in scripts as
the statistic# can change between releases and ports. It is better to specify
the statistic name and then to lookup the statistic# in V$STATNAME before
joining to V$SESSTAT or V$MYSTAT.

In Oracle 10.2 and above limited statistics (28) are reported in the following
dynamic performance views
- V$SERVICE_STATS - service statistics
- V$CLIENT_STATS - statistics by client identifier
- V$SERV_MOD_ACT_STATS - statistics for modules / actions
The following RAC specific wait events are reported in V$SERVICE_STATS, V$CLIENT_STATS and V$SERV_MOD_ACT_STATS:

- cluster wait time
- gc cr block receive time
- gc cr blocks received
- gc current block receive time
- gc current blocks received

Note that only receive times are reported. If you have more than two instances in your cluster and an asymmetric workload, this information is not sufficient to determine which node is performing most work.
Dynamic Performance Views
Wait Events

- Wait-related dynamic performance views include:
  - V$WAITSTAT
  - V$EVENT_NAME
  - V$SYSTEM_EVENT
  - V$SESSION_EVENT
  - V$SESSION_WAIT
  - V$SYSTEM_WAIT_CLASS
  - V$SERVICE_WAIT_CLASS
  - V$SESSION_WAIT_CLASS
  - V$SESSION_WAIT_HISTORY

The following dynamic performance views report information about wait events:

- V$WAITSTAT summarizes wait events by class
- V$EVENT_NAME returns a static list of all wait events together with parameters. In Oracle 10.1 and above each wait event is assigned to a wait class.
- V$SYSTEM_EVENT summarizes wait events at system level
- V$SESSION_EVENT summarizes wait events at session level
- V$SESSION_WAIT reports the current or last wait event for each session.

In Oracle 10.1 and above:
- V$SYSTEM_WAIT_CLASS summarizes wait events by wait class at system level.
- V$SERVICE_WAIT_CLASS summarizes wait events by wait class at service level
- V$SESSION_WAIT_CLASS summarizes wait events by wait class at system level
- V$SESSION_WAIT_HISTORY reports the last 10 waits for each session.

Prior to Oracle 10.1 there were two ways to capture session wait parameter values:

- Querying V$SESSION_WAIT. In earlier versions parameter values are only set while wait in progress
- Enabling event 10046 level 8 trace.
The following table shows the number of waits in each class in Oracle 10.2.0.3:

<table>
<thead>
<tr>
<th>Wait Class</th>
<th># Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrative</td>
<td>46</td>
</tr>
<tr>
<td>Application</td>
<td>12</td>
</tr>
<tr>
<td>Cluster</td>
<td>47</td>
</tr>
<tr>
<td>Commit</td>
<td>1</td>
</tr>
<tr>
<td>Concurrency</td>
<td>25</td>
</tr>
<tr>
<td>Configuration</td>
<td>23</td>
</tr>
<tr>
<td>Idle</td>
<td>62</td>
</tr>
<tr>
<td>Network</td>
<td>27</td>
</tr>
<tr>
<td>Other</td>
<td>592</td>
</tr>
<tr>
<td>Scheduler</td>
<td>2</td>
</tr>
<tr>
<td>System I/O</td>
<td>24</td>
</tr>
<tr>
<td>User I/O</td>
<td>17</td>
</tr>
</tbody>
</table>

The table was created using the following query:

SELECT wait_class, COUNT(*)
FROM v$event_name
GROUP BY wait_class;

The number of events in each wait class can differ in each patch set. Some RAC waits are allocated to the "Cluster" class. Note, however, that many other cluster waits are allocated to the "Other" class.
Dynamic Performance Views
Time Model Statistics

- Introduced in Oracle 10.1
- Intended to provide more granular information for CPU time

Dynamic Performance Views
- `V$SYS_TIME_MODEL`
- `V$SESS_TIME_MODEL`

System time model statistics collected in AWR
- `DBA_HIST_SYS_TIME_MODEL`

Session time model statistics are not collected in AWR
System time model statistics included in default AWR report

Dynamic Performance Views include
- `V$SYS_TIME_MODEL` - reports time model statistics at system level
- `V$SESS_TIME_MODEL` - reports time model statistics at session level
## Dynamic Performance Views

### Time Model Statistics

The following time model statistics are reported in Oracle 10.2.0.3

<table>
<thead>
<tr>
<th>Time Model Category</th>
<th>Subcategory</th>
</tr>
</thead>
<tbody>
<tr>
<td>background elapsed time</td>
<td>inbound PL/SQL rpc elapsed time</td>
</tr>
<tr>
<td>background cpu time</td>
<td>Java execution elapsed time</td>
</tr>
<tr>
<td>connection management call elapsed time</td>
<td>parse time elapsed</td>
</tr>
<tr>
<td>DB time</td>
<td>PL/SQL compilation elapsed time</td>
</tr>
<tr>
<td>DB CPU</td>
<td>PL/SQL execution elapsed time</td>
</tr>
<tr>
<td>failed parse (out of shared memory) elapsed time</td>
<td>repeated bind elapsed time</td>
</tr>
<tr>
<td>failed parse elapsed time</td>
<td>RMAN cpu time (backup/restore)</td>
</tr>
<tr>
<td>hard parse (bind mismatch) elapsed time</td>
<td>sequence load elapsed time</td>
</tr>
<tr>
<td>hard parse (sharing criteria) elapsed time</td>
<td>sql execute elapsed time</td>
</tr>
<tr>
<td>hard parse time elapsed</td>
<td></td>
</tr>
</tbody>
</table>

The above table was generated using:

```sql
SELECT stat_name
FROM v$sys_time_model
ORDER BY UPPER (stat_name);
```
Dynamic Performance Views
Statistics Level Parameter

- Introduced in Oracle 9.2

- Values can be
  - BASIC
  - TYPICAL
  - ALL

- Default value is TYPICAL

- Levels and defaults reported in V$STATISTICS_LEVEL
Dynamic Performance Views
Statistics Level Parameter

<table>
<thead>
<tr>
<th>Statistic Name</th>
<th>Oracle 9.2</th>
<th>Oracle 10.1</th>
<th>Oracle 10.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Session History</td>
<td></td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
<tr>
<td>Bind Data Capture</td>
<td></td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
<tr>
<td>Buffer Cache Advice</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
<tr>
<td>Cache Stats Monitor</td>
<td>-</td>
<td>TYPICAL</td>
<td>-</td>
</tr>
<tr>
<td>Global Cache Statistics</td>
<td>-</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
<tr>
<td>Longops Statistics</td>
<td>-</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
<tr>
<td>Modification Monitoring</td>
<td>-</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
<tr>
<td>MTTR Advice</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
<tr>
<td>PGA Advice</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
<tr>
<td>Plan Execution Statistics</td>
<td>ALL</td>
<td>ALL</td>
<td>ALL</td>
</tr>
<tr>
<td>Segment Level Statistics</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
<tr>
<td>Shared Pool Advice</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
<tr>
<td>Streams Pool Advice</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
<tr>
<td>Threshold-based Alerts</td>
<td>-</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
<tr>
<td>Timed OS Statistics</td>
<td>ALL</td>
<td>ALL</td>
<td>ALL</td>
</tr>
<tr>
<td>Timed Statistics</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
<tr>
<td>Ultrafast Latch Statistics</td>
<td>-</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
<tr>
<td>Undo Advisor, Alerts and Fast Ramp up</td>
<td>-</td>
<td>TYPICAL</td>
<td>TYPICAL</td>
</tr>
</tbody>
</table>

The above data was obtained using the following query:

```
SELECT statistics_name, activation_level
FROM v$statistics_level
ORDER BY UPPER (statistics_name);
```

The following statistics have been added to Oracle 11.1

- Adaptive Thresholds Enabled
- Automatic Maintenance Tasks
- Plan Execution Sampling
- Session Wait Stack
- SQL Monitoring
- Streams Pool Advice
- Time Model Events
- V$IOSTAT_* statistics

Notes:

- Global Cache Statistics
  - Enables collection of RAC buffer cache statistics
  - Does not directly populate any dynamic performance views
  - Cannot be set at session level
  - Sets _gc_statistics parameter
- Cache Stats Monitor - added in Oracle 10.1; removed in Oracle 10.2
Segment statistics were introduced in Oracle 9.2 and are collected when the STATISTICS_LEVEL parameter is set to TYPICAL or ALL.

Segment statistics are stored in the SGA. They are therefore instance-specific in a RAC environment. Segment statistics are stored for each data object id. The SGA statistics structure can grow dynamically if new objects are created.

Segment statistics are externalized in:
- V$SEGSTAT_NAME - lists names of all statistics
- V$SEGMENT_STATISTICS - reports statistics for each segment - joins back to USER$, OBJ$ etc to provide owner and object names
- V$SEGSTAT - reports statistics for each segment - does not join back to data dictionary tables
Segment statistics are collected in Oracle 9.2 and above.

Two segment statistics are sampled:

- **logical reads**
- **db block changes**

The remaining segment statistics have exact values.

Three additional segment statistics are reported in RAC environments:

- **gc buffer busy**
- **gc cr blocks received**
- **gc current blocks received**

In Oracle 9.2:

- **gc cr blocks received**
- **gc current blocks received**

were known as:

- **global cache cr blocks received**
- **global cache current blocks received**
Dynamic Performance Views
Operating System Statistics

- Introduced in Oracle 10.2
- Reported in V$OSSTAT
- Collected by AWR
  - DBA_HIST_OSSTAT
  - DBA_HIST_OSSTAT_NAME
- Included in default AWR report
- Statistics reported are dependent on operating system and Oracle version

The actual statistics reported are operating system dependent. The table above shows the nine operating system statistics reported in Oracle 10.2.0.1 for Linux 32-bit; the list below shows the 13 operating system statistics reported in Oracle 10.2.0.1 for Windows 32-bit

- AVG_BUSY_TIME
- AVG_IDLE_TIME
- AVG_SYS_TIME
- AVG_USER_TIME
- BUSY_TIME
- IDLE_TIME
- NUM_CPUS
- PHYSICAL_MEMORY_BYTES
- RSRC_MGR_CPU_WAIT_TIME
- SYS_TIME
- USER_TIME
- VM_IN_BYTES
- VM_OUT_BYTES
In Oracle 10.2 and above process memory statistics are reported. This is useful for obtaining a breakdown of memory consumption by individual processes.
Dynamic Performance Views

Global views

- In a RAC environment each V$ view has an equivalent GV$ view
- GV$ view includes INST_ID column containing the instance number. For example:

<table>
<thead>
<tr>
<th>INST_ID</th>
<th>NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>VARCHAR2(20)</td>
</tr>
<tr>
<td>VALUE</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

| NAME   | VARCHAR2(20) |
| VALUE  | NUMBER |

- In Oracle 9.2 and below PARALLEL_MIN_SERVERS must be >= number of instances to use GV$ views
- In Oracle 10.1 and above PZnn background processes are used to return data on remote instances e.g. PZ99

In a RAC environment every instance-specific V$ view has an equivalent global GV$ view. The GV$ view includes an extra column containing the instance number (INST_ID).

The GV$ views are queried using parallel execution to execute a slave on each instance in the cluster. The querying instance then collates the results.

In Oracle 9.2 and below the PARALLEL_MIN_SERVERS parameter must be greater than or equal to number of instances in order to query GV$ views

In Oracle 10.1 and above PZnn background processes are used to return data on remote instances e.g. PZ99, PZ98, PZ97 etc.

In Oracle 10.2 there are a number of V$ views that do not have equivalent GV$ views. These V$ views are mostly related to RMAN.
A number of V$ views externalize the contents of the control files. These include:

- V$DATABASE
- V$DATAFILE
- V$CONTROLFILE
- V$LOG
- V$LOGFILE
- V$THREAD

These views describe the database which will be identical for each instance.

It is not meaningful to select from the GV$ versions of these views as you will receive duplicate rows from each instance.

Other views with similar properties include:

- V$ACTIVE_INSTANCES
- V$SPPARAMETER

These views describe the database which will be identical for each instance.

It is not meaningful to select from the GV$ versions of these views as you will receive duplicate rows from each instance.

Other views with similar properties include V$ACTIVE_INSTANCES which returns a list of active instances across the cluster and V$SPPARAMETER which returns the contents of the server parameter file.
Dynamic Performance Views
RAC-specific views

In Oracle 10.2, the following dynamic performance views are directly related to RAC:

| V$ACTIVE_INSTANCES | V$FILE_CACHE_TRANSFER |
| V$CLASS_CACHE_TRANSFER | V$FILE_PING |
| V$CLASS_PING | V$GC$HVMASTER_INFO |
| V$CLUSTER_INTERCONNECTIONS | V$GCSPPMASTER_INFO |
| V$CONFIGURED_INTERCONNECTIONS | V$GC_ELEMENT |
| V$CR BLOCK_SERVER | V$GC_ELEMENTS_WITH_COLLISIONS |
| V$CURRENT_BLOCK_SERVER | V$GES_BLOCKING_ENQUEUE |
| V$DLM ALL LOCKS | V$GES_ENQUEUE |
| V$DLM_CONVERT_LOCAL | V$GLOBAL_BLOCKED_LOCKS |
| V$DLM_CONVERT_REMOTE | V$GLOBAL_TRANSACTION |
| V$DLMLatch | V$INSTANCE |
| V$DLM_LOCKS | V$INSTANCE_CACHE_TRANSFER |
| V$DLM_MISC | V$INSTANCE_LOG_GROUP |
| V$DLM_RESS | V$INSTANCE_RECOVERY |
| V$DLM_TRAFFIC_CONTROLLER | V$TEMP_PING |

The above list was derived from Oracle 10.2.0.1 All of the above views appear in both single-instance and RAC databases.
Many other V$ views include RAC-specific information.
Some additional views/synonyms are created for RAC databases using `$ORACLE_HOME/rdbms/admin/catclust.sql`

<table>
<thead>
<tr>
<th>Synonym Name</th>
<th>View Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>V$GES_CONVERT_LOCAL</td>
<td>V$DLM_CONVERT_LOCAL</td>
</tr>
<tr>
<td>V$GES_CONVERT_REMOTE</td>
<td>V$DLM_CONVERT_REMOTE</td>
</tr>
<tr>
<td>V$GES_LATCH</td>
<td>V$DLM_LATCH</td>
</tr>
<tr>
<td>V$GES_RESOURCE</td>
<td>V$DLM_RESS</td>
</tr>
<tr>
<td>V$GES_STATISTICS</td>
<td>V$DLM_MISC</td>
</tr>
<tr>
<td>V$GES_TRAFFIC_CONTROLLER</td>
<td>V$DLM_TRAFFIC_CONTROLLER</td>
</tr>
<tr>
<td>GV$GES_CONVERT_LOCAL</td>
<td>GV$DLM_CONVERT_LOCAL</td>
</tr>
<tr>
<td>GV$GES_CONVERT_REMOTE</td>
<td>GV$DLM_CONVERT_REMOTE</td>
</tr>
<tr>
<td>GV$GES_LATCH</td>
<td>GV$DLM_LATCH</td>
</tr>
<tr>
<td>GV$GES_RESOURCE</td>
<td>GV$DLM_RESS</td>
</tr>
<tr>
<td>GV$GES_STATISTICS</td>
<td>GV$DLM_MISC</td>
</tr>
<tr>
<td>GV$GES_TRAFFIC_CONTROLLER</td>
<td>GV$DLM_TRAFFIC_CONTROLLER</td>
</tr>
</tbody>
</table>

The CATCLUST.SQL script should be executed during RAC database creation. It creates a number of synonyms for RAC views in the kernel. The original view names contain the DLM prefix; the equivalent new view names contain the GES prefix, reflecting the change in name of the Dynamic Lock Manager (DLM) to Global Enqueue Services (GES) in Oracle 9.0.1. The original DLM views are still built into the kernel, presumably to provide backwards compatibility.
Server Alerts

Metrics

♦ Introduced in Oracle 10.1

♦ In Oracle 10.2.0.1 there are
  ♦ 10 metric groups
  ♦ 163 distinct metrics

♦ Dependent on actual metric can be calculated:
  ♦ Per Second
  ♦ Per Transaction
  ♦ Per User Call
  ♦ As Ratio
  ♦ As Percentage
  ♦ As Count
  ♦ As Time

The following dynamic performance views contain metric data:

• V$EVENTMETRIC
• V$FILEMETRIC
• V$FILEMETRIC_HISTORY
• V$METRIC
• V$METRICGROUP
• V$METRICNAME
• V$METRIC_HISTORY
• V$SERVICEMETRIC
• V$SERVICEMETRIC_HISTORY
• V$SESSMETRIC
• V$SYSMETRIC
• V$SYSMETRIC_HISTORY
• V$SYSMETRIC_SUMMARY
• V$WAITCLASSMETRIC
• V$WAITCLASSMETRIC_HISTORY
Server Alerts
Metric Groups

- Metrics are grouped together into metric groups
- In Oracle 10.2 there are 10 metric groups

<table>
<thead>
<tr>
<th>Group Name</th>
<th>Interval Size</th>
<th># Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Metrics</td>
<td>6000</td>
<td>3</td>
</tr>
<tr>
<td>Event Class Metrics</td>
<td>6000</td>
<td>4</td>
</tr>
<tr>
<td>File Metrics Long Duration</td>
<td>60000</td>
<td>6</td>
</tr>
<tr>
<td>Service Metrics (short)</td>
<td>500</td>
<td>5</td>
</tr>
<tr>
<td>Service Metrics</td>
<td>6000</td>
<td>5</td>
</tr>
<tr>
<td>Session Metrics Short Duration</td>
<td>1500</td>
<td>10</td>
</tr>
<tr>
<td>Session Metrics Long Duration</td>
<td>6000</td>
<td>1</td>
</tr>
<tr>
<td>System Metrics Short Duration</td>
<td>1500</td>
<td>41</td>
</tr>
<tr>
<td>System Metrics Long Duration</td>
<td>6000</td>
<td>134</td>
</tr>
<tr>
<td>Tablespace Metrics Long Duration</td>
<td>6000</td>
<td>2</td>
</tr>
</tbody>
</table>

The above table was generated using the following query:

```sql
SELECT mg.name, mg.interval_size, COUNT(*)
FROM v$metricgroup mg, v(metricname mn
WHERE mg.group_id = mn.group_id
GROUP BY mg.name, mg.interval_size;
```

RAC-specific metrics include:

- GC CR Block Received Per Second
- GC CR Block Received Per Txn
- GC Current Block Received Per Second
- GC Current Block Received Per Txn
- Global Cache Average CR Get Time
- Global Cache Average Current Get Time
- Global Cache Blocks Corrupted
- Global Cache Blocks Lost
Server Alerts

Thresholds

- In Oracle 10.1 and above thresholds can be specified for individual metrics

- Threshold values can be specified for
  - Warnings
  - Critical alerts

- Currently configured server-alert thresholds are reported in DBA_THRESHOLDS

- Thresholds can be maintained using the following subroutines in the DBMS_SERVER_ALERT package:
  - GET_THRESHOLD
  - SET_THRESHOLD
  - VIEW_THRESHOLDS

Outstanding alerts are reported in the DBA_OUTSTANDING_ALERTS view

A history of alerts is maintained in the DBA_ALERT_HISTORY view

A static list of alert types is available in V$ALERT_TYPES. Each alert is assigned to a specific type of object including:

- ASM INSTANCE
- CLUSTER NODE
- DATABASE
- DATA OBJECT
- EVENT CLASS
- FILE
- GLOBAL SERVICE
- INSTANCE
- QUOTA
- RECOVERY AREA
- ROLLBACK SEGMENT
- SERVICE
- SESSION
- SYSTEM
- TABLESPACE
The following code sets a hard parses per transaction threshold with
- a warning value of 500 per transaction
- a critical value of 1000 per transaction

```sql
DBMS_SERVER_ALERT.SET_THRESHOLD
    (
        metrics_id => DBMS_SERVER_ALERT.HARD_PARSES_TXN,
        warning_operator => DBMS_SERVER_ALERT.OPERATOR_GE,
        warning_value => '500',
        critical_operator => DBMS_SERVER_ALERT.OPERATOR_GE,
        critical_value => '1000',
        observation_period => 1, -- 1 second
        consecutive_occurrences => 1,
        instance_name => '',
        object_type => DBMS_SERVER_ALERT.OBJECT_TYPE_SYSTEM,
        object_name => ''
    );
```

Note that in order to use server-generated alerts you must have an Enterprise Manager Diagnostic Pack license.
Trace and Diagnostics
Modules and Actions

- In Oracle 8.0 and above it is possible to specify a module and action for any session

- Modules and actions allow inefficient SQL statements to be identified and isolated more efficiently

- Modules and actions are reported in
  - STATSPACK / AWR / ASH reports
  - V$SESSION
  - V$SQL
  - V$ACTIVE_SESSION_HISTORY

- Current module and action for a session is reported in
  - V$SESSION.MODULE
  - V$SESSION.ACTION
Trace and Diagnostics
DBMS_MONITOR

❖ To specify a module and action use

```sql
DBMS_APPLICATION_INFO.SET_MODULE
(
    MODULE_NAME => 'MODULE1',
    ACTION_NAME => 'ACTION1'
);
```

❖ To specify a new action within the current module use:

```sql
DBMS_APPLICATION_INFO.SET_ACTION
(
    ACTION_NAME => 'ACTION2'
);
```

❖ Modules and actions can also be specified using OCI calls

The syntax for DBMS_APPLICATION_INFO.SET_MODULE is:

```sql
DBMS_APPLICATION_INFO.SET_MODULE
(
    MODULE_NAME   VARCHAR2,  -- Module
    ACTION_NAME   VARCHAR2  -- Action
);
```

The syntax for DBMS_APPLICATION_INFO.SET_ACTION is:

```sql
DBMS_APPLICATION_INFO.SET_ACTION
(
    ACTION_NAME   VARCHAR2  -- Action
);
```

The current module and action for a session are reported in the MODULE and ACTION columns of V$SESSION.
The DBMS_MONITOR package was introduced in Oracle 10.1 and provides a supported method for enabling and disabling trace:

- for the current session
- for a specified session
- for the entire database
- for a specified instance
- for a specified client identifier
- for a specified service
- for a specified service and module
- for a specified service, module and action

In addition, statistics collection can be enabled for individual clients, for services, modules or actions.
Trace and Diagnostics
DBMS_MONITOR

- Trace is enabled using the following subroutines:
  - SESSION_TRACE_ENABLE
  - DATABASE_TRACE_ENABLE
  - CLIENT_ID_TRACE_ENABLE
  - SERV_MOD_ACT_TRACE_ENABLE

- By default event 10046 level 8 trace will be enabled
  - Includes wait events

- In Oracle 11.1 these subroutines have an additional PLAN_STATS parameter which specifies when row source statistics are dumped. Possible values are
  - NEVER
  - FIRST_EXECUTION (default)
  - ALL_EXECUTIONS

In Oracle 11.1 these subroutines have an additional PLAN_STATS parameter which specifies when row source statistics are dumped. Possible values are

- DBMS_MONITOR.NEVER
- DBMS_MONITOR.FIRST_EXECUTION (default)
- DBMS_MONITOR.ALL_EXECUTIONS
Trace and Diagnostics

DBMS_MONITOR

- Introduced in Oracle 10.1
- To enable trace in the current session use:
  ```sql
  EXECUTE DBMS_MONITOR.SESSION_TRACE_ENABLE;
  ```
- To disable trace in the current session use:
  ```sql
  EXECUTE DBMS_MONITOR.SESSION_TRACE_DISABLE;
  ```
- To enable trace in session 42 use:
  ```sql
  EXECUTE DBMS_MONITOR.SESSION_TRACE_ENABLE (SESSION_ID => 42);
  ```
- To disable trace in session 42 use:
  ```sql
  EXECUTE DBMS_MONITOR.SESSION_TRACE_DISABLE (SESSION_ID => 42);
  ```

By default WAITS (10046 level 8) are traced

The syntax for DBMS_MONITOR.SESSION_TRACE_ENABLE is:

```sql
DBMS_MONITOR.SESSION_TRACE_ENABLE
(
  SESSION_ID  NUMBER,  -- SID
  SERIAL_NUM  NUMBER,  -- Serial Number
  WAITS       BOOLEAN, -- Include Waits
  BINDS       BOOLEAN, -- Include Binds
);
```

With no arguments, DBMS_MONITOR.SESSION_TRACE_ENABLE enables 10046 level 8 trace in the current session.

The syntax for DBMS_MONITOR.SESSION_TRACE_DISABLE is:

```sql
DBMS_MONITOR.SESSION_TRACE_DISABLE
(
  SESSION_ID  NUMBER,  -- SID
  SERIAL_NUM  NUMBER   -- Serial Number
);
```

With no arguments, SESSION_TRACE_DISABLE disables trace in the current session.

It is rarely necessary to specify the serial number for the session.
**Trace and Diagnostics**

**DBMS_MONITOR**

- Introduced in Oracle 10.2
- To enable trace for the entire database use:

  ```sql
  EXECUTE DBMS_MONITOR.DATABASE_TRACE_ENABLE;
  ```

- To disable trace for the entire database use:

  ```sql
  EXECUTE DBMS_MONITOR.DATABASE_TRACE_DISABLE;
  ```

- To enable trace for instance **RAC1** use:

  ```sql
  EXECUTE DBMS_MONITOR.DATABASE_TRACE_ENABLE (INSTANCE_NAME => 'RAC1');
  ```

- To disable trace for instance **RAC1** use:

  ```sql
  EXECUTE DBMS_MONITOR.DATABASE_TRACE_DISABLE (INSTANCE_NAME => 'RAC1');
  ```

The syntax for `DBMS_MONITOR.DATABASE_TRACE_ENABLE` is:

```sql
DBMS_MONITOR.DATABASE_TRACE_ENABLE
  (WAITS BOOLEAN, -- Include Waits
   BINDS BOOLEAN, -- Include Binds
   INSTANCE_NAME VARCHAR2 -- Instance Name
  );
```

With no arguments, `DBMS_MONITOR.DATABASE_TRACE_ENABLE` enables trace for the entire database.

The syntax for `DBMS_MONITOR.DATABASE_TRACE_DISABLE` is:

```sql
DBMS_MONITOR.DATABASE_TRACE_DISABLE
  (INSTANCE_NAME VARCHAR2 -- Instance Name
  );
```

With no arguments, `SESSION_TRACE_DISABLE` disables trace in the entire database.

In early versions of 10.2 (10.2.0.1) I was unable to disable instance-wide trace using this command.
Client identifiers were introduced to allow individual users of connection caches to be identified and traced.

Connection caches usually connect via a single Oracle user. Therefore one Oracle user may have a large number of sessions connected to an instance. Client identifiers can be used to differentiate between these sessions and to identify high resource consumers.

The syntax for DBMS_SESSION.SET_IDENTIFIER is:

```
DBMS_SESSION.SET_IDENTIFIER

(    CLIENT_ID    VARCHAR2    -- Client ID
);
```
Trace and Diagnostics
DBMS_MONITOR

✦ To enable trace for CLIENT42 use:

BEGIN
  DBMS_MONITOR.CLIENT_ID_TRACE_ENABLE
  (CLIENT_ID => 'CLIENT42');
END;

✦ To statistics collection for CLIENT42 use:

BEGIN
  DBMS_MONITOR.CLIENT_ID_STAT_ENABLE
  (CLIENT_ID => 'CLIENT42');
END;

✦ Client statistics are reported in V$CLIENT_STATS

The syntax for DBMS_MONITOR.CLIENT_ID_TRACE_ENABLE is:
DBMS_MONITOR.CLIENT_ID_TRACE_ENABLE
(
  CLIENT_ID VARCHAR2, -- Client ID
  WAITS BOOLEAN, -- Include Waits
  BINDS BOOLEAN -- Include Binds
);

The syntax for DBMS_MONITOR.CLIENT_ID_STAT_ENABLE is:
DBMS_MONITOR.CLIENT_ID_STAT_ENABLE
(
  CLIENT_ID VARCHAR2 -- Client ID
);

juliandyke.com
Trace and Diagnostics
DBMS_MONITOR

- Trace can be enabled for a specific
  - service
  - service and module
  - service, module and action
- To enable trace for SERVICE1 use:

```
BEGIN
  DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE
  (SERVICE_NAME => 'SERVICE1');
END;
```

- To disable trace for SERVICE1 use:

```
BEGIN
  DBMS_MONITOR.SERV_MOD_ACT_TRACE_DISABLE
  (SERVICE_NAME => 'SERVICE1');
END;
```

The syntax for DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE is:

```
DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE
  (SERVICE_NAME VARCHAR2, -- Service Name
   MODULE_NAME VARCHAR2, -- Module
   ACTION_NAME VARCHAR2, -- Action
   WAITS BOOLEAN, -- Waits
   BINDS BOOLEAN, -- Binds
   INSTANCE_NAME VARCHAR2 -- Instance
  );
```

The syntax for DBMS_MONITOR.SERV_MOD_ACT_TRACE_DISABLE is:

```
DBMS_MONITOR.SERV_MOD_ACT_TRACE_DISABLE
  (SERVICE_NAME VARCHAR2, -- Service Name
   MODULE_NAME VARCHAR2, -- Module
   ACTION_NAME VARCHAR2, -- Action
   INSTANCE_NAME VARCHAR2 -- Instance
  );
```
Trace and Diagnostics

DBMS_MONITOR

To enable trace for **MODULE1** use:

```sql
BEGIN
    DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE
    (
        SERVICE_NAME => 'SERVICE1',
        MODULE_NAME => 'MODULE1'
    );
END;
```

To enable trace for **ACTION1** use:

```sql
BEGIN
    DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE
    (
        SERVICE_NAME => 'SERVICE1',
        MODULE_NAME => 'MODULE1',
        ACTION_NAME => 'ACTION1'
    );
END;
```

You must specify a SERVICE_NAME in order to specify a MODULE_NAME; you must specify a SERVICE_NAME and MODULE_NAME in order to specify an ACTION_NAME.

If the ACTION_NAME is not specified then the entire module will be traced.
Trace and Diagnostics
DBMS_MONITOR

- To enable statistics collection for MODULE1 use:

```
BEGIN
  DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE
  (  
    SERVICE_NAME => 'SERVICE1',
    MODULE_NAME => 'MODULE1'
  );
END;
```

- To enable statistics collection for ACTION1 use:

```
BEGIN
  DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE
  (  
    SERVICE_NAME => 'SERVICE1',
    MODULE_NAME => 'MODULE1',
    ACTION_NAME => 'ACTION1'
  );
END;
```

- Statistics are externalized in V$SERV_MOD_ACT_STATS

Statistics are automatically collected at service level and reported in V$SERVICE_STATS. Statistics can also be collected at module and action level.

The syntax for DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE is:

```
DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE
(  
  SERVICE_NAME   VARCHAR2, -- Service Name
  MODULE_NAME    VARCHAR2, -- Module
  ACTION_NAME    VARCHAR2 -- Action
);
```

The syntax for DBMS_MONITOR.SERV_MOD_ACT_STAT_DISABLE is:

```
DBMS_MONITOR.SERV_MOD_ACT_STAT_DISABLE
(  
  SERVICE_NAME   VARCHAR2, -- Service Name
  MODULE_NAME    VARCHAR2, -- Module
  ACTION_NAME    VARCHAR2 -- Action
);
```
The DBA_ENABLED_TRACES view was introduced in Oracle 10.1. It describes the current traces that are enabled in the database. It is based on SYS.WRI$_TRACING_ENABLED. This table is part of the data dictionary and therefore the trace configuration is persistent throughout a reboot.

If the trace type is
• CLIENT_ID - the client identifier will be reported in the PRIMARY_ID column
• SERVICE - the service name will be reported in the PRIMARY_ID column
• SERVICE_MODULE - the service name will be reported in the PRIMARY_ID column and the module name will be reported in the QUALIFIER_ID1 column
• SERVICE_MODULE_ACTION - the service name will be reported in the PRIMARY_ID column, the module name will be reported in the QUALIFIER_ID1 column and the action name will be reported in the QUALIFIER_ID2 column
• DATABASE - introduced in Oracle 10.2. The instance name if specified will be reported in the INSTANCE_NAME column.

The DBA_ENABLED_AGGREGATIONS view is based on SYS.WRI$_AGGREGATION_ENABLED.
Trace and Diagnostics
Automatic Diagnostic Repository

- In Oracle 11.1 and above the diagnostics area has been redesigned

- Diagnostics area is located in $ORACLE_BASE/diag and includes the following top-level directories
  - asm
  - clients
  - crs
  - diagtool
  - lsnrctl
  - netcman
  - ofm
  - rdbms
  - tnslsnr

The RDBMS diagnostics area contains the following subdirectories for each database
- alert
- cdump
- incident
- incpkg
- ir
- lck
- metadata
- stage
- sweep
- trace
Trace and Diagnostics
Automatic Diagnostic Repository

- Trace directory includes
  - server (foreground) process trace files
  - background process trace files
  - alert log (text)

- All trace files and alert log are written to
  - $ORACLE_BASE/diag/rdbms/<database>/<instance>/trace

- For example for database TEST
  - $ORACLE_BASE/diag/rdbms/test/TEST1/trace

- BACKGROUND_DUMP_DEST and USER_DUMP_DEST both specify same trace directory by default
  - Deprecated in Oracle 11.1

In Oracle 11.1 text version of alert log is written to trace directory
XML version of alert log is written to alert directory e.g:

$ORACLE_BASE/diag/rdbms/test/TEST/alert/log.xml

Contents are more verbose due to XML tags. For example:

<msg time='2007-08-13T19:30:00.096+01:00' org_id='oracle'
comp_id='rdbms' msg_id='opistr_real:871:3971575317'
type='NOTIFICATION' group='startup' level='16' pid='16325' version='1'>
  <txt>Starting ORACLE instance (normal)</txt>
</msg>
### Diagnostics Area

- **$V$DIAG_INFO** dynamic performance view
- Introduced in Oracle 11.1
- Returns values for the following diagnostics

<table>
<thead>
<tr>
<th>Name</th>
<th>Example Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR Base</td>
<td>/u01/app/oracle</td>
</tr>
<tr>
<td>ADR Home</td>
<td>/u01/app/oracle/diag/rdbms/test/TEST</td>
</tr>
<tr>
<td>Active Incident Count</td>
<td>2</td>
</tr>
<tr>
<td>Active Problem Count</td>
<td>1</td>
</tr>
<tr>
<td>Default Trace File</td>
<td>/u01/app/oracle/diag/rdbms/test/TEST/trace/TEST_ora_14003.trc</td>
</tr>
<tr>
<td>Diag Alert</td>
<td>/u01/app/oracle/diag/rdbms/test/TEST/alert</td>
</tr>
<tr>
<td>Diag Cdump</td>
<td>/u01/app/oracle/diag/rdbms/test/TEST/cdump</td>
</tr>
<tr>
<td>Diag Enabled</td>
<td>TRUE</td>
</tr>
<tr>
<td>Diag Incident</td>
<td>/u01/app/oracle/diag/rdbms/test/TEST/incident</td>
</tr>
<tr>
<td>Diag Trace</td>
<td>/u01/app/oracle/diag/rdbms/test/TEST/trace</td>
</tr>
<tr>
<td>Health Monitor</td>
<td>/u01/app/oracle/diag/rdbms/test/TEST/hm</td>
</tr>
</tbody>
</table>

In Oracle 11.1 and above trace file for current process can be identified using:

```sql
SELECT value FROM v$diag_info
WHERE name = 'Default Trace File';
```

In Oracle 11.1 and the diagnostics area can be managed using the ADRCI command line utility

- Use HELP to list options
- Use HELP EXTENDED to list additional Oracle internal options

ADRCI is mainly intended to assist generating incident reports for Oracle Support
 Trace & Diagnostics
 ORADEBUG

◆ ORADEBUG
  ◆ undocumented debugging utility
  ◆ available in SQL*Plus (Oracle 8.1.5 and above)
  ◆ requires SYSDBA privilege
  ◆ To list available options

      $ sqlplus / as sysdba
      SQL> ORADEBUG HELP

◆ There are three ways of selecting a process for ORADEBUG
  ◆ To specify current process:
    SQL> ORADEBUG SETMYPID
  ◆ To specify Oracle process:
    SQL> ORADEBUG SETORAPID <oracle pid>
  ◆ To specify operating system process:
    SQL> ORADEBUG SETOSPID <os pid>

ORADEBUG was originally supplied as a standalone utility on Unix (oradbx) and as a standalone utility on VMS (orambx). The executable needed to be linked before execution.

ORADEBUG was subsequently supplied within Server Manager (svrmgr). In Oracle 8.1.5 and above it was also included in SQL*Plus and in Oracle 9.0.1 and above it can only be accessed from SQL*Plus.

When using ORADEBUG you must first specify a process to which ORADEBUG will connect. There are three ways of specifying a process. You can connect to the process for the current session (SETMYPID), using the Oracle PID (SETORAPID) or the operating system process (SETOSPID).

ORADEBUG has some very dangerous functionality. Therefore you should NEVER use it on a production system unless instructed to do so by Oracle Support.
ORADEBUG is an undocumented interface that allows you to obtain extended trace and diagnostics from the database kernel.

ORADEBUG is integrated into SQL*Plus. You must have SYSDBA privilege to execute ORADEBUG commands.

ORADEBUG includes a RAC-specific diagnostic tool called LKDEBUG. You can obtain information about various RAC objects including processes, resources and locks using the LKDEBUG tool.

Output is written to the trace file for the current session as specified by the USER_DUMP_DEST parameter.

This slide shows the first part of the help message for LKDEBUG.
This slide shows the second part of the help message for LKDEBUG.

Note that LKDEBUG contains a couple of commands that enable you to experiment with resource remastering. I recommend that you do not play with these commands in a production system.
To enable a numeric event at instance level
- **PFILE** (init.ora):
  
  ```
  event = '<event> trace name context forever, level <level>';
  ```

- **SPFILE**:
  
  ```
  ALTER SYSTEM SET EVENT = '
  <event> trace name context forever, level <level>' [SCOPE= SPFILE];
  ```

To enable a numeric event at system level

```altr SESSION SET EVENT =
  '<event> trace name context forever, level <level>';
```

- Using **ORADEBUG**

```
oradebug event <event> trace name context forever, level <level>;
```

In Unix events are defined in the following binary file:

```
$ORACLE_HOME/rdbms/mesg/oraus.msg
```

This file can be converted into a text file using:

```
strings $ORACLE_HOME/rdbms/mesg/oraus.msg > oraus.txt
```
Trace & Diagnostics
Numeric Events

The following numeric trace events can produce some useful output in RAC environments:

<table>
<thead>
<tr>
<th>Event Code</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10425</td>
<td>Global enqueue operations</td>
</tr>
<tr>
<td>10426</td>
<td>GES/GCSD reconfiguration</td>
</tr>
<tr>
<td>10427</td>
<td>GES traffic controller</td>
</tr>
<tr>
<td>10428</td>
<td>GES cached resource</td>
</tr>
<tr>
<td>10429</td>
<td>GES IPC calls</td>
</tr>
<tr>
<td>10430</td>
<td>GES/GCS dynamic remastering</td>
</tr>
<tr>
<td>10432</td>
<td>GCS Fusion calls (part 1)</td>
</tr>
<tr>
<td>10435</td>
<td>GES Deadlock detection</td>
</tr>
<tr>
<td>10439</td>
<td>GCS Fusion calls (part 2)</td>
</tr>
<tr>
<td>10704</td>
<td>Local enqueue manipulation</td>
</tr>
<tr>
<td>10705</td>
<td>Global enqueue manipulation</td>
</tr>
<tr>
<td>10708</td>
<td>RAC buffer cache</td>
</tr>
</tbody>
</table>

You should not attempt to enable numeric events on a production database. Take extreme care when using numeric events on any database.

Most of the above events operate at multiple levels. These levels are not documented and their usefulness can only really be assessed by experimentation.

Levels are either numbered sequentially e.g. 1,2,3,... The maximum level for older events is often 10.

Levels can also be specified in terms of bit values e.g. 1, 2, 4, 8, 16. These values can be combined using a bitwise OR operation e.g.

\[ 1 \lor 2 \lor 4 = 7 \]

Some events must be set when processes are started. Therefore to set an event in a background process such as LMD0 or LMS0 it may be necessary to restart the instance.
Trace & Diagnostics
Symbolic Events

- Useful symbolic event dumps include
  - GC_ELEMENTS
  - HEAPDUMP
  - HEAPDUMP_ADDR
  - LIBRARY_CACHE
  - ENQUEUES
  - LATCHES
  - HANGANALYZE
  - PROCESSSTATE
  - SYSTEMSTATE
  - GES_STATE

- A full list of symbolic dumps can be obtained using
  `SQL> ORADEBUG DUMPLIST`

You should not attempt to take symbolic dumps on a production database.
Take extreme care when using symbolic dumps on any database.

As with the numeric events, symbolic event dumps have levels. Older events tend to have levels from 1 to 10; newer events use bit values.

For example, to list all global cache elements currently mastered by an instance use

`ALTER SESSION SET EVENTS 'immediate trace name gc_elements level 1';`
Trace & Diagnostics
Network

♦ Trace can be enabled for the Oracle Net client or server
  ♦ Configured in $TNS_ADMIN/sqlnet.ora
  ♦ The following parameters can be specified:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRACE_DIRECTORY_CLIENT</td>
<td>Specifies the directory for the client trace file</td>
</tr>
<tr>
<td>TRACE_FILE_CLIENT</td>
<td>Specifies the name of the client trace file</td>
</tr>
<tr>
<td>TRACE_FILELEN_CLIENT</td>
<td>Specifies the size of each client trace file in KB</td>
</tr>
<tr>
<td>TRACE_FILENO_CLIENT</td>
<td>Specifies the number of client trace files</td>
</tr>
<tr>
<td>TRACE_LEVEL_CLIENT</td>
<td>Specifies the level of detail of client trace</td>
</tr>
<tr>
<td>TRACE_TIMESTAMP_CLIENT</td>
<td>Includes a timestamp (microseconds) for each event in client trace</td>
</tr>
<tr>
<td>TRACE_UNIQUE_CLIENT</td>
<td>Creates an individual client trace file for each process</td>
</tr>
<tr>
<td>TRACE_DIRECTORY_SERVER</td>
<td>Specifies the directory for the server trace file</td>
</tr>
<tr>
<td>TRACE_FILE_SERVER</td>
<td>Specifies the name of the server trace file</td>
</tr>
<tr>
<td>TRACE_FILELEN_SERVER</td>
<td>Specifies the size of each server trace file in KB</td>
</tr>
<tr>
<td>TRACE_FILENO_SERVER</td>
<td>Specifies the number of server trace files</td>
</tr>
<tr>
<td>TRACE_LEVEL_SERVER</td>
<td>Specifies the level of detail of server trace</td>
</tr>
<tr>
<td>TRACE_TIMESTAMP_SERVER</td>
<td>Includes a timestamp (microseconds) for each event in server trace</td>
</tr>
</tbody>
</table>

For both the client and server trace files, the default directory is $ORACLE_HOME/network/trace.
For the client, the default trace file name is sqlnet.trc; for the server the default trace file name is svr_pid.trc

When both TRACE_FILELEN_CLIENT and TRACE_FILENO_CLIENT are set to non-zero values, the trace files are used cyclically. When one file is full, output continues in the next file; when all files are full output continues in the first file. A sequence number is included in the file name. For example if TRACE_FILE_CLIENT is client and TRACE_FILENO_CLIENT is 5 then the files will be:

• client1_pid.trc
• client2_pid.trc
• client3_pid.trc
• client4_pid.trc
• client5_pid.trc

TRACE_FILELEN_SERVER and TRACE_FILENO_SERVER work in a similar way to TRACE_FILELEN_CLIENT and TRACE_FILENO_CLIENT.

If TRACE_UNIQUE_CLIENT is set to ON then a separate trace file will be created for each client. The pid is appended to the file name e.g. client_123.trc. Note that this appears to be the default behaviour in recent versions.
Trace & Diagnostics

Network

- Trace can be enabled for the Oracle Net client or server
  - Configured in $TNS_ADMIN/sqlnet.ora

- Trace levels are

<table>
<thead>
<tr>
<th>Level#</th>
<th>Level Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OFF</td>
<td>Disable tracing</td>
</tr>
<tr>
<td>4</td>
<td>USER</td>
<td>Include user errors</td>
</tr>
<tr>
<td>6</td>
<td>ADMIN</td>
<td>Include administrative level errors</td>
</tr>
<tr>
<td>16</td>
<td>SUPPORT</td>
<td>Include packet contents</td>
</tr>
</tbody>
</table>

For both TRACE_LEVEL_CLIENT and TRACE_LEVEL_SERVER, the parameter can take a numeric value between 0 and 16 where 0 is disabled and 16 is the most detailed. Alternatively these parameters can also take a scalar value as shown above.

Level 16 (SUPPORT) is the most detailed trace level. Take care when enabling this level of detail as it will consume disk space very rapidly.
Trace & Diagnostics
Listener

- Listener trace levels are same as network trace levels
  - OFF, USER, ADMIN, SUPPORT
- Listener trace can be
  - configured in LISTENER.ORA
    
    TRACE_LEVEL_LISTENER_SERVER6 = admin
    TRACE_DIRECTORY_LISTENER_SERVER6 = /tmp
    TRACE_FILE_LISTENER_SERVER6 = listener.logv

- enabled dynamically in LSNRCTL
  
  LSNRCTL> SET TRC_LEVEL ADMIN
  LSNRCTL> SET TRC_DIRECTORY /tmp
  LSNRCTL> SET TRC_FILE listener.log

- Dynamic changes can optionally be written to LISTENER.ORA using:
  
  LSNRCTL> SAVE_CONFIG

You can also use TRACE <level> in LSNRCTL e.g.
  
  LSNRCTL> TRACE ADMIN

This is equivalent to SET TRC_LEVEL ADMIN

Alternatively you can specify the trace level at the command line:
  
  lsnrctl TRACE ADMIN

However, changes using TRACE instead of SET_TRC_LEVEL cannot be saved to LISTENER.ORA
Trace & Diagnostics
CLUVFY

- To enable trace in CLUVFY use:
  
  export SRVM_TRACE = true

- Trace files are written to the \$CV_HOME/cv/log directory

- By default this directory is removed immediately after CLUVFY is execution

- On Linux/Unix comment out the following line in runcluvfy.sh

  ```
  # $RM -rf $CV_HOME
  ```

- Pathname of CV_HOME directory is based on operating system process e.g:

  ```
  /tmp/18124
  ```

- It can be useful to echo value of CV_HOME in runcluvfy.sh:

  ```
  echo CV_HOME=$CV_HOME
  ```

In Windows use:

SET SRVM_TRACE = TRUE

By default trace files will be written to C:\WINDOWS\TEMP e.g.
reqproc_891812.log
Trace & Diagnostics

SRVCTL

- In Oracle 10.1 and above, to enable trace in SRVCTL use
  
  ```
  export SRVM_TRACE = true
  ```

- By default trace is written to standard output

- In Oracle 10.1 and above, the same environment variable can be used to trace:
  - NETCA
  - VIPCA
  - SRVCONFIG
  - GSDCTL
  - CLUVFY
  - CLUUTIL

By default the VIPCA trace file is written to

```
$ORA_CRS_HOME/cfgtoollogs/vipca/vipca.log
```

By default the NETCA trace file is written to

```
$ORACLE_HOME/cfgtoollogs/netca/trace.log
```

In Oracle 9.2 and below, to enable trace in SRVCTL

1 - Edit $ORACLE_HOME/bin/srvctl (srvctl.bat in Windows)
2 - Find the following line

```
$JRE -classpath $CLASSPATH oracle.ops.opsctl.OPSCTLDriver "$@
```

3 - Add the following immediately after $JRE and before -classpath

```
-DTRACING.ENABLED=true -DTRACING.LEVEL=2
```

In Oracle 9.2 and below similar arguments can be used to trace:

- GSD
- GSDCTL
- SRVCONFIG
Trace & Diagnostics
DBCA

♦ To enable trace for the DBCA in Oracle 9.0.1 and above
♦ Edit $ORACLE_HOME/bin/dbca and change

# Run DBCA
$JRE_DIR/bin/jre -DORACLE_HOME=$OH -DJDBC_PROTOCOL=thin
-mx64m -classpath $CLASSPATH oracle.sysman.assistants.dbca.Dbca
$ARGUMENTS

♦ to

# Run DBCA
$JRE_DIR/bin/jre -DORACLE_HOME=$OH -DJDBC_PROTOCOL=thin
-mx64m -DTRACING.ENABLED=true -DTRACING.LEVEL=2
-classpath $CLASSPATH oracle.sysman.assistants.dbca.Dbca
$ARGUMENTS

♦ Redirect standard output to a file e.g.

$ dbca > dbca.out &

Before making any changes backup the original dbca file e.g

cp $ORACLE_HOME/bin/dbca $ORACLE_HOME/bin/dbca.orig

The text starting with $JRE_DIR/bin/jre and ending with $ARGUMENTS should be on a single line.

In Oracle 10.2 trace information is written automatically to
$ORACLE_HOME/cfgtoollogs/dbca/trace.log

In Oracle 10.2 you can also add the -DDEBUG flag so that output is written interactively

See Metalink note: 188134.1 Tracing the Database Configuration Assistant (DBCA)
Trace & Diagnostics
Oracle Universal Installer (OUI)

- On Unix/Linux to launch the OUI with tracing enabled use:
  
  ```
  ./runInstaller -J-DTRACING.ENABLED=true -J-DTRACING.LEVEL=2
  ```

- Log files will be written to `$ORACLE_BASE/oraInventory/logs`

- To trace `root.sh` execute it using:
  
  ```
  sh -x root.sh
  ```

- Note that it may be necessary to cleanup the CRS installation before executing `root.sh` again

On Windows to launch the OUI with tracing enabled use:

  ```
  setup.exe -J-DTRACING.ENABLED=true -J-DTRACING.LEVEL=2
  ```

Logs will be written to:

  `C:\Program Files\oracle\Inventory\logs`

See Metalink note 269837.1: Tracing the OUI from 9.2.0.5 to 10g

See Metalink note 240001.1: 10g RAC: Troubleshooting CRS Root.sh Problems