Session 4

Monitoring and Diagnostics

Advanced RAC
Auckland - May 2008

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

Enterprise Manager Database Control

- To check if EM Agent is currently running use:

```
$ emctl status dbconsole
```

- For example:

```
$ emctl status dbconsole
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.
```

- Start EM Agent using:

```
$ emctl start dbconsole
```

- Stop EM Agent using:

```
$ emctl stop dbconsole
```

5 © 2008 Julian Dyke juliandyke.com

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
```

Enterprise Manager Grid Control

- Installed separately from database
- Repository normally stored in a stand-alone database
  - Can now be installed in an existing Oracle 10.2 database
- Uses separate Management Service
  - Does not have to be installed on same node as repository database
- In Oracle 10.2.0.2 and above you can optionally install the Oracle Configuration Manager
  - Automatically sends information about your database to Oracle Support
  - Reduces time required to raise a Service Request
  - Potential security risk

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.

Enterprise Manager
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)
- Can be used with:
  - Enterprise Manager Database Control
  - Enterprise Manager Grid Control
- Cannot be used with:
  - Oracle Database Standard Edition

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.

Enterprise Manager Diagnostic Pack

- Licensed cost option
- Features include
  - Automatic Workload Repository (AWR)
  - Automatic Database Diagnostic Monitor (ADDM)
  - Active Session History (ASH)
  - Enterprise Manager Performance Monitoring pages (database and host)
  - Event notifications: notification methods, rules and schedules
  - Event history and metric history (database and host)
  - Blackouts
  - Dynamic metric baselines
  - Monitoring templates
  - Memory-access based performance monitoring

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

• addmrpt.sql, addmrpti.sql

• awrrpt.sql, awrrpti.sql

• ashrpt.sql, ashrpti.sql

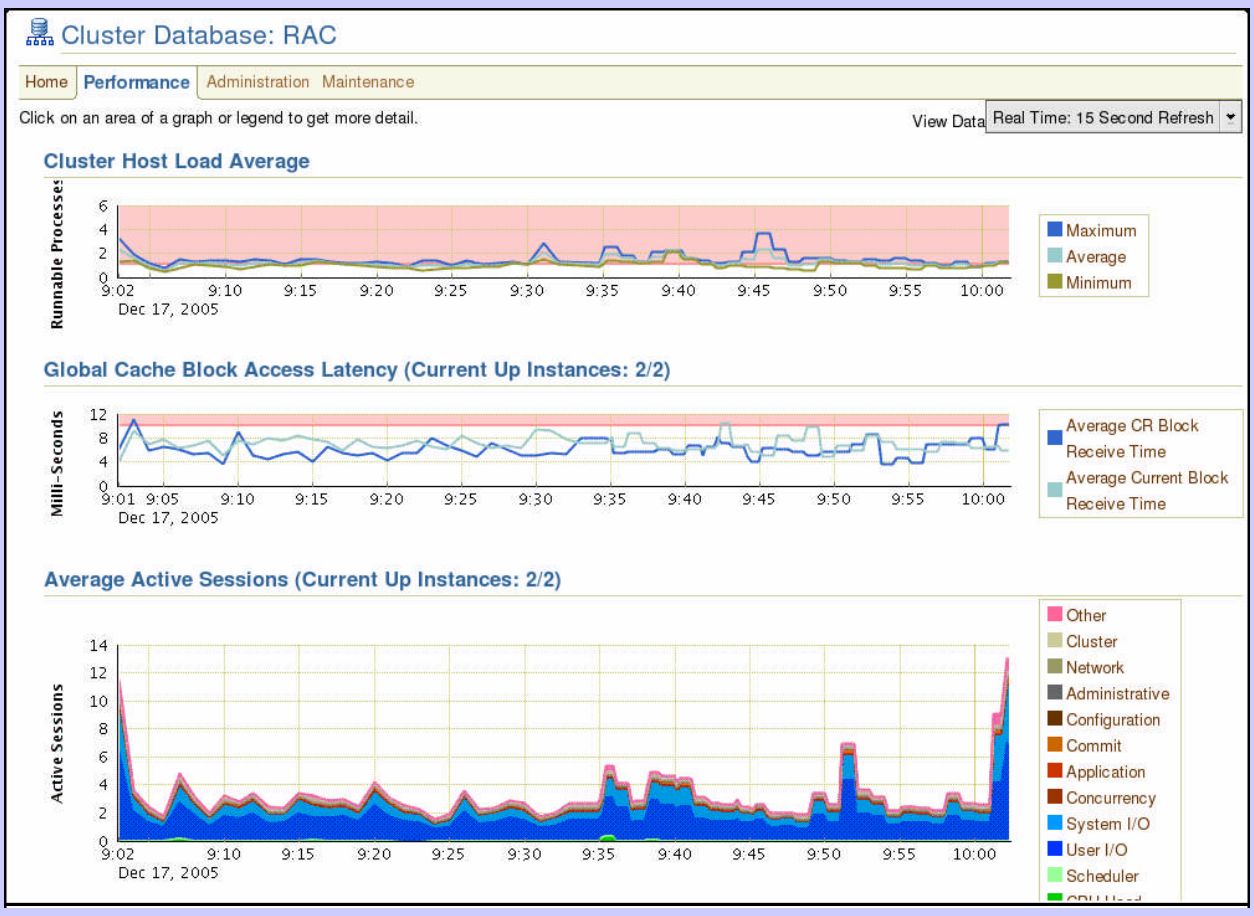
• awrddrpt.sql, awrddrpti.sql

• awrsqrpt.sql, awrsqrpti.sql

Enterprise Manager Tuning Pack

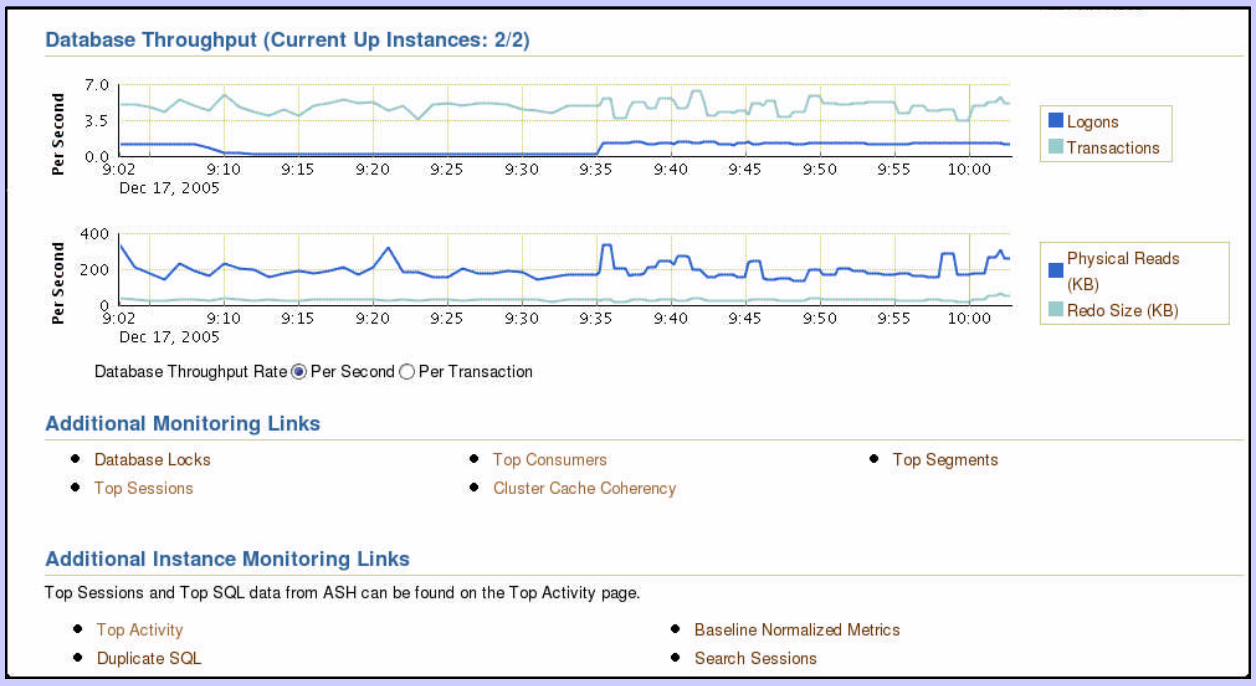
- Licensed cost option
- Features include:
  - SQL Access Advisor
  - SQL Tuning Advisor
  - SQL Tuning Sets
  - Object reorganization

Enterprise Manager Performance Page

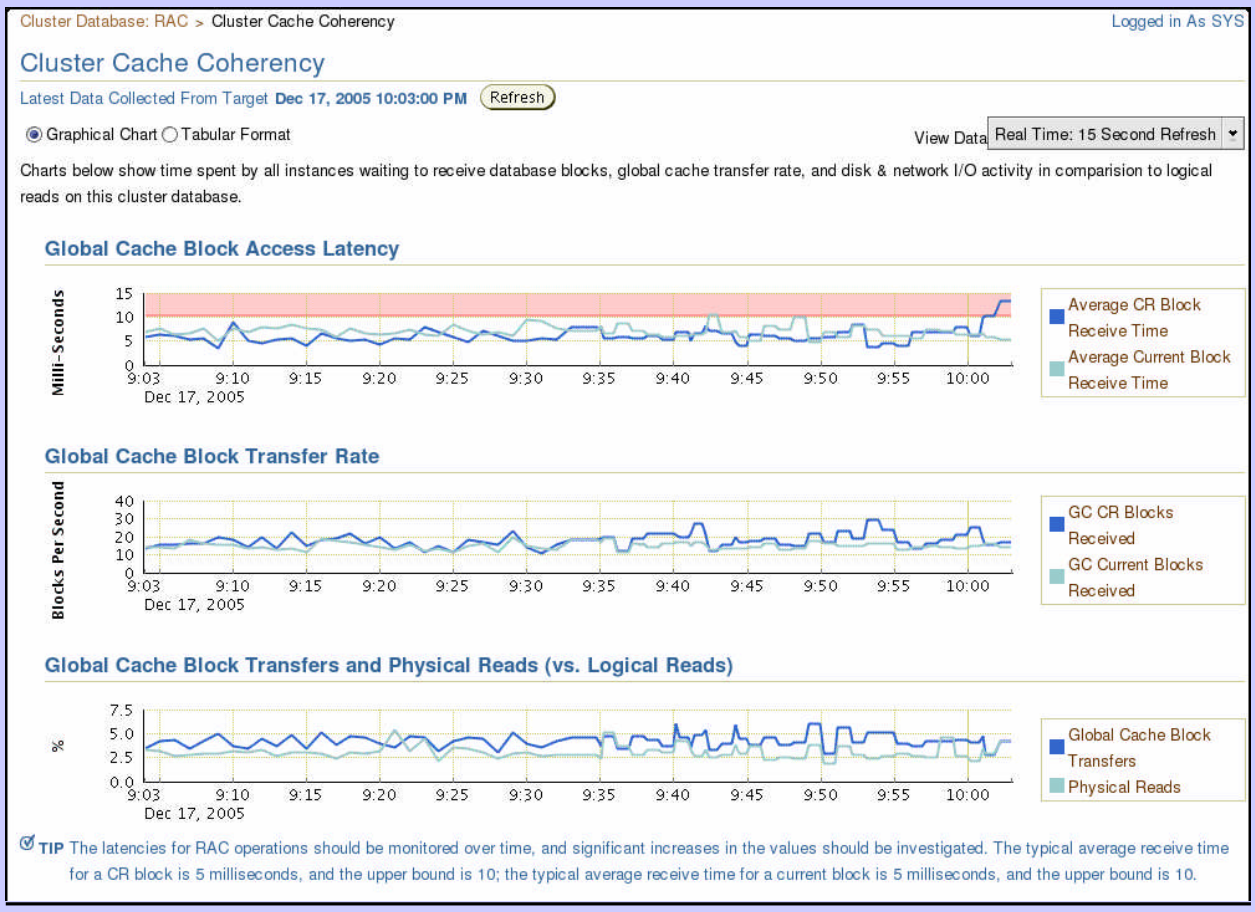


Enterprise Manager Performance Page

- **Continued**



Enterprise Manager
Cluster Cache Coherency Page



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

```
SELECT * FROM dba_hist_wr_control;

DBID       SNAP_INTERVAL     RETENTION         TOPNSQL
---------- ----------------- ----------------- -------
2467148022 +00000 01:00:00.0 +00007 00:00:00.0 DEFAULT
```

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)

```
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);
```

AWR Parameters

- TOPNSQL
  - Determines number of SQL statements to store for each criteria
    - Elapsed time
    - CPU time
    - Gets
    - Reads
    - Executions
    - Parse calls
    - Shareable Memory
    - Version Count
  - To set TOPNSQL value to 50

```
DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS -
(topnsql=>50);
```

17 juliandyke.com

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

AWR Snapshots

- A snapshot represents statistic values at a point in time
- Snapshots are created automatically
  - By default every 60 minutes
- Snapshots can also be created manually
  - Must specify statistics level - can be
    - TYPICAL (default)
    - ALL
- For example:

```
DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT;
```

```
DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT -
(flush_level=>'ALL');
```


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:

```
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

```
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**

DBA_HIST_ACTIVE_SESSION_HISTORY	DBA_HIST_LATCH
DBA_HIST_BASELINE	DBA_HIST_LATCH_CHILDREN
DBA_HIST_BG_EVENT_SUMMARY	DBA_HIST_LATCH_MISSES_SUMMARY
DBA_HIST_BUFFERED_QUEUES	DBA_HIST_LATCH_NAME
DBA_HIST_BUFFERED_SUBSCRIBERS	DBA_HIST_LATCH_PARENT
DBA_HIST_BUFFER_POOL_STAT	DBA_HIST_LIBRARYCACHE
DBA_HIST_COMP_IOSTAT	DBA_HIST_LOG
DBA_HIST_CR_BLOCK_SERVER	DBA_HIST_METRIC_NAME
DBA_HIST_CURRENT_BLOCK_SERVER	DBA_HIST_MTTR_TARGET_ADVICE
DBA_HIST_DATABASE_INSTANCE	DBA_HIST_OPTIMIZER_ENV
DBA_HIST_DATAFILE	DBA_HIST_OSSTAT
DBA_HIST_DB_CACHE_ADVICE	DBA_HIST_OSSTAT_NAME
DBA_HIST_DLM_MISC	DBA_HIST_PARAMETER
DBA_HIST_ENQUEUE_STAT	DBA_HIST_PARAMETER_NAME
DBA_HIST_EVENT_NAME	DBA_HIST_PGASTAT
DBA_HIST_FILEMETRIC_HISTORY	DBA_HIST_PGA_TARGET_ADVICE
DBA_HIST_FILESTATXS	DBA_HIST_PROCESS_MEM_SUMMARY
DBA_HIST_INSTANCE_RECOVERY	DBA_HIST_RESOURCE_LIMIT
DBA_HIST_INST_CACHE_TRANSFER	DBA_HIST_ROWCACHE_SUMMARY
DBA_HIST_JAVA_POOL_ADVICE	DBA_HIST_RULE_SET

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**

DBA_HIST_SEG_STAT	DBA_HIST_SQL_WORKAREA_HSTGRM
DBA_HIST_SEG_STAT_OBJ	DBA_HIST_STAT_NAME
DBA_HIST_SERVICE_NAME	DBA_HIST_STREAMS_APPLY_SUM
DBA_HIST_SERVICE_STAT	DBA_HIST_STREAMS_CAPTURE
DBA_HIST_SERVICE_WAIT_CLASS	DBA_HIST_STREAMS_POOL_ADVICE
DBA_HIST_SESSMETRIC_HISTORY	DBA_HIST_SYSMETRIC_HISTORY
DBA_HIST_SESS_TIME_STATS	DBA_HIST_SYSMETRIC_SUMMARY
DBA_HIST_SGA	DBA_HIST_SYSSTAT
DBA_HIST_SGASTAT	DBA_HIST_SYSTEM_EVENT
DBA_HIST_SGA_TARGET_ADVICE	DBA_HIST_SYS_TIME_MODEL
DBA_HIST_SHARED_POOL_ADVICE	DBA_HIST_TABLESPACE_STAT
DBA_HIST_SNAPSHOT	DBA_HIST_TBSPC_SPACE_USAGE
DBA_HIST_SNAP_ERROR	DBA_HIST_TEMPFILE
DBA_HIST_SQLBIND	DBA_HIST_TEMPSTATXS
DBA_HIST_SQLSTAT	DBA_HIST_THREAD
DBA_HIST_SQLTEXT	DBA_HIST_UNDOSTAT
DBA_HIST_SQL_BIND_METADATA	DBA_HIST_WAITCLASSMET_HISTORY
DBA_HIST_SQL_PLAN	DBA_HIST_WAITSTAT
DBA_HIST_SQL_SUMMARY	DBA_HIST_WR_CONTROL

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

Table Name	View Name
WRM$_BASELINE	DBA_HIST_BASELINE
WRM$_DATABASE_INSTANCE	DBA_HIST_DATABASE_INSTANCE
WRM$_SNAPSHOT	DBA_HIST_SNAPSHOT
WRM$_SNAP_ERROR	DBA_HIST_SNAP_ERROR
WRM$_WR_CONTROL	DBA_HIST_WR_CONTROL

Note that DBA_HIST_BASELINE is a join between WRM$_BASELINE and two instances of WRM$_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

```
$ sqlplus /  as sysdba

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

- 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

Database ID	: 2467148022
Instance Number	: 1
Start snapshot	: 1736
End snapshot	: 1737

```
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
```

AWR Report

- Contains the following sections (Oracle 10.2):
  - Summary
  - Cache Sizes
  - Load Profile
  - Instance Efficiency Percentages
  - Top 5 Timed Events
  - Global Cache Load Profile
  - Global Cache Efficiency Percentages
  - GCS/GES - Workload Characteristics
  - GCS/GES - Messaging Statistics
  - Time Model Statistics
  - Wait Class Statistics
  - Wait Events
  - Background Wait Events
  - Operating System Statistics
  - Service Statistics
  - Service Wait Class Statistics
  - SQL ordered by Elapsed Time
  - SQL ordered by CPU Time
  - SQL ordered by Gets
  - SQL ordered by Reads
  - SQL ordered by Executions
  - SQL ordered by Parse Calls
  - SQL ordered by Sharable Memory
  - SQL ordered by Version Count
  - Parent Latch Statistics
  - Child Latch Statistics
  - Instance Activity Statistics
  - Tablespace I/O Statistics
  - File I/O Statistics
  - Buffer Pool Statistics
  - Instance Recovery Statistics
  - Buffer Pool Advisory
  - PGA Aggregate Summary
  - PGA Aggregate Target Statistics
  - PGA Aggregate Target Histogram
  - PGA Memory Advisory

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.

AWR Report

- 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:

```
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
DB Name         DB Id    Instance     Inst Num Release     RAC Host
------------ ----------- ------------ -------- ----------- --- ------------
PROD         2149809470  PROD1               1 10.2.0.3.0  YES server3

              Snap Id      Snap Time      Sessions Curs/Sess
            --------- ------------------- -------- ---------
Begin Snap:      1768 10-Aug-07 21:00:28        55 #########
  End Snap:      1769 10-Aug-07 22:00:40        57 #########
   Elapsed:               60.19 (mins)
   DB Time:                7.72 (mins)

Cache Sizes
~~~~~~~~~~~                       Begin        End
                             ---------- ----------
               Buffer Cache:     1,232M     1,232M  Std Block Size:         8K
           Shared Pool Size:       224M       224M      Log Buffer:    15,192K
```


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.

AWR
Load Profile

Load Profile

	Per Second	Per Transaction
Redo size:	13,524.27	2,081.01
Logical reads:	548.67	84.43
Block changes:	86.78	13.35
Physical reads:	0.53	0.08
Physical writes:	4.60	0.71
User calls:	107.33	16.51
Parses:	29.34	4.51
Hard parses:	0.00	0.00
Sorts:	2.49	0.38
Logons:	0.30	0.05
Executes:	30.77	4.73
Transactions:	6.50	

% Blocks changed per Read:	15.82	Recursive Call %:	19.29
Rollback per transaction %:	3.23	Rows per Sort:	100.91

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

```
Instance Efficiency Percentages (Target 100%)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

            Buffer Nowait %:   99.98       Redo NoWait %:  100.00
            Buffer  Hit   %:   99.90    In-memory Sort %:  100.00
            Library Hit   %:   99.99        Soft Parse %:  100.00
         Execute to Parse %:    4.66         Latch Hit %:   99.97
Parse CPU to Parse Elapsd %:  104.83     % Non-Parse CPU:   97.01


 Shared Pool Statistics        Begin    End
                              ------  ------
             Memory Usage %:   86.09   86.11
    % SQL with executions>1:   85.59   86.53
  % Memory for SQL w/exec>1:   75.22   76.49
```


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.

AWR
Top 5-timed Events

Top 5 Timed Events

Event	Waits	Time (s)	Avg wait (ms)	%Total Call Time	Wait Class
enq: TX - row lock contention	758	324	428	70.0	Applicatio
CPU time		102		22.0	
reliable message	428	36	84	7.7	Other
gcs log flush sync	7,438	15	2	3.1	Other
log file sync	28,693	14	0	3.0	Commit

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.

AWR
Global Cache Statistics and Ratios

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

                               Begin   End
                               ----- -----
          Number of Instances:     2     2

Global Cache Load Profile
~~~~~~~~~~~~~~~~~~~~~~~~                  Per Second       Per Transaction
                                        ---------------  ---------------
  Global Cache blocks received:                 21.49             3.31
    Global Cache blocks served:                 21.11             3.25
     GCS/GES messages received:                 49.23             7.58
         GCS/GES messages sent:                 49.84             7.67
            DBWR Fusion writes:                  0.88             0.14
 Estd Interconnect traffic (KB)                360.13

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. .

AWR
GCS and GES Workload Characteristics

```
Global Cache and Enqueue Services - Workload Characteristics
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

                    Avg global enqueue get time (ms):      3.8

         Avg global cache cr block receive time (ms):      0.7
    Avg global cache current block receive time (ms):      0.5

           Avg global cache cr block build time (ms):      0.0
            Avg global cache cr block send time (ms):      0.0
     Global cache log flushes for cr blocks served %:     16.8
           Avg global cache cr block flush time (ms):      2.2

        Avg global cache current block pin time (ms):      0.1
       Avg global cache current block send time (ms):      0.0
 Global cache log flushes for current blocks served %:     0.8
      Avg global cache current block flush time (ms):      0.5
```


The Global Cache and Enqneue 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.

AWR
GCS and GES Messaging Statistics

```
Global Cache and Enqueue Services - Messaging Statistics
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                  Avg message sent queue time (ms):      0.1
            Avg message sent queue time on ksxp (ms):      0.2
                Avg message received queue time (ms):      3.3
                  Avg GCS message process time (ms):      0.0
                  Avg GES message process time (ms):      0.0

                         % of direct sent messages:    62.43
                       % of indirect sent messages:    13.36
                    % of flow controlled messages:    24.21
```


The Global Cache and Enqueue Services - Messaging Statistics section describes average time to exchange different categories of inter-instance messages.

AWR
Time Model Statistics

```
Time Model Statistics                  DB/Inst: PROD/PROD1  Snaps: 1768-1769
-> Total time in database user-calls (DB Time): 463.3s
-> Statistics including the word "background" measure background process
   time, and so do not contribute to the DB time statistic
-> Ordered by % or DB time desc, Statistic name

Statistic Name                                       Time (s) % of DB Time
------------------------------------------ ------------------ ------------
sql execute elapsed time                              428.3         92.5
DB CPU                                                101.8         22.0
parse time elapsed                                      3.2           .7
connection management call elapsed time                 1.8           .4
PL/SQL execution elapsed time                           0.4           .1
hard parse elapsed time                                 0.0           .0
repeated bind elapsed time                              0.0           .0
DB time                                               463.3          N/A
background elapsed time                               111.8          N/A
background cpu time                                    30.0          N/A
```


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           DB/Inst: PROD/PROD1  Snaps: 1768-1769
-> Resources reported for PL/SQL code includes the resources used by all SQL
   statements called by the code.
-> % Total DB Time is the Elapsed Time of the SQL statement divided
   into the Total Database Time multiplied by 100

  Elapsed      CPU                  Elap per  % Total
  Time (s)   Time (s)  Executions   Exec (s)  DB Time    SQL Id
---------- ---------- ------------ ---------- ------- -------------
       289          0           16       18.0    62.3 4fpxtnab3p56m
SELECT WL0.NAME, WL0.SCHEDULE_ID FROM ORDER_ITEMS WL0 WHERE (WL0.NAME = :1 )
FOR UPDATE

        27          4        6,064        0.0     5.8 32qvqpcm517s2
SELECT WL0.KEY, WL0.CURRENCY, WL0.ENTRY_DATE, WL0.TOKEN FROM CUSTOMER WL0
WHERE ( WL0.KEY = :1 ) FOR UPDATE

        21          9        3,417        0.0     4.5 4t500f63mp992
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, the 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.

AWR
Dictionary Cache

```
Dictionary Cache Stats                DB/Inst: PROD/PROD1 Snaps: 1768-1769
-> "Pct Misses"  should be very low (< 2% in most cases)
-> "Final Usage" is the number of cache entries being used

                                    Get    Pct    Scan   Pct      Mod      Final
Cache                          Requests   Miss    Reqs  Miss     Reqs      Usage
------------------------- ------------ ------ ------- ----- -------- ----------
dc_awr_control                       66    0.0       0   N/A        2          1
dc_histogram_data                    84    0.0       0   N/A        0      6,243
dc_histogram_defs                    50    0.0       0   N/A        0      5,015
dc_object_ids                       964    0.2       0   N/A        0      1,289
dc_objects                          734    0.3       0   N/A        5      1,586
dc_profiles                         569    0.0       0   N/A        0          1
dc_rollback_segments             30,707    0.0       0   N/A        0        447
dc_segments                         101    0.0       0   N/A        0      1,430
dc_sequences                          1  100.0       0   N/A        1          2
outstanding_alerts                  246   81.3       0   N/A       36          3

          -------------------------------------------------------------
Dictionary Cache Stats (RAC)          DB/Inst: PROD/PROD1 Snaps: 1768-1769

                                    GES          GES          GES
Cache                          Requests    Conflicts     Releases
------------------------- ------------ ------------ ------------
dc_awr_control                       4            0            0
dc_object_ids                        2            0            0
dc_objects                          12            0            0
dc_sequences                         2            1            0
outstanding_alerts                 464          182            0
```


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

```
Library Cache Activity                    DB/Inst: PROD/PROD1  Snaps: 1768-1769
-> "Pct Misses"  should be very low

                        Get    Pct            Pin    Pct             Invali-
Namespace           Requests   Miss       Requests   Miss    Reloads  dations
--------------- ------------ ------ -------------- ------ ---------- --------
BODY                     553    0.0          1,041    0.0          0        0
SQL AREA                   2  100.0        118,512    0.0          2        0
TABLE/PROCEDURE          317    0.0          3,721    0.1          2        0
TRIGGER                   12    0.0            723    0.0          0        0
          -------------------------------------------------------------

Library Cache Activity (RAC)              DB/Inst: PROD/PROD1 Snaps: 1768-1769

                    GES Lock      GES Pin      GES Pin   GES Inval GES Invali-
Namespace           Requests     Requests     Releases    Requests     dations
--------------- ------------ ------------ ------------ ----------- -----------
TABLE/PROCEDURE          995           14            2           9           0
```


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.

AWR
Global Enqueue Statistics

Global Enqueue Statistics DB/Inst: PROD/PROD1 Snaps: 1768-1769

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

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

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

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

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

Statistic         Total   % <1ms  % <10ms % <100ms    % <1s   % <10s
---------- ------------ -------- -------- -------- -------- --------
Pins            35,207    99.83     0.01     0.17     0.00     0.00
Flushes            289    95.85     4.15     0.00     0.00     0.00
Writes           3,184    50.85    49.15     0.00     0.00     0.00
```


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

                                    CR                          Current
                      ----------------------------   ----------------------------
Inst Block              Blocks      %      %      %    Blocks      %      %
  No Class            Received  Immed   Busy Congst  Received  Immed   Busy Congst
---- ----------      -------- ------ ------ ------   -------- ------ ------ ------
   2 data block        39,346   82.9   17.1     .0     33,999   98.9    1.1     .0
   2 undo header        3,167   99.4     .6     .0         65  100.0     .0     .0
   2 undo block           579   95.3    4.7     .0          0    N/A    N/A    N/A
   2 Others                99   99.0    1.0     .0        368  100.0     .0     .0
```


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

41 © 2008 Julian Dyke juliandyke.com

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 Report

- ASH reports can be generated using:
  - the DBMS_WORKLOAD_REPOSITORY package
  - the $ORACLE_HOME/rdbms/admin/ashrpt.sql script
- Oracle recommends using the ashrpt.sql script

```
$ sqlplus /  as sysdba
SQL> @$ORACLE_HOME/rdbms/admin/ashrpt.sql
```

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

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:

```
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

Dynamic Performance Views Statistics

- The following statistics are reported in V$SERVICE_STATS, V$CLIENT_STATS and V$SERV_MOD_ACT_STATS

DB CPU	parse time elapsed
DB time	physical reads
application wait time	physical writes
cluster wait time	redo size
concurrency wait time	session cursor cache hits
db block changes	session logical reads
execute count	sql execute elapsed time
gc cr block receive time	user I/O wait time
gc cr blocks received	user calls
gc current block receive time	user commits
gc current blocks received	user rollbacks
logons cumulative	workarea executions - multipass
opened cursors cumulative	workarea executions - onepass
parse count (total)	workarea executions - optimal

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 valuess

• Querying V$SESSION_WAIT. In earlier versions parameter values are only set while wait in progress

• Enabling event 10046 level 8 trace.

Dynamic Performance Views
Wait Classes

- **The following table shows the number of waits in each class in Oracle 10.2.0.3**

Wait Class	# Events
Administrative	46
Application	12
Cluster	47
Commit	1
Concurrency	25
Configuration	23
Idle	62
Network	27
Other	592
Scheduler	2
System I/O	24
User I/O	17

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**

background elapsed time	inbound PL/SQL rpc elapsed time
background cpu time	Java execution elapsed time
connection management call elapsed time	parse time elapsed
DB time	PL/SQL compilation elapsed time
DB CPU	PL/SQL execution elapsed time
failed parse (out of shared memory) elapsed time	repeated bind elapsed time
failed parse elapsed time	RMAN cpu time (backup/restore)
hard parse (bind mismatch) elapsed time	sequence load elapsed time
hard parse (sharing criteria) elapsed time	sql execute elapsed time
hard parse time elapsed	

The above table was generated using:

```
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

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

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

Dynamic Performance Views
Segment Statistics

- Introduced in Oracle 9.2
- Collected when STATISTICS_LEVEL = TYPICAL or ALL
- Externalized in
  - V$SEGSTAT_NAME
  - V$SEGMENT_STATISTICS
  - V$SEGSTAT
- Recorded in AWR
  - DBA_HIST_SEG_STAT
  - DBA_HIST_SEG_STAT_OBJ
- Reported in
  - AWR (default report)

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

Dynamic Performance Views
Segment Statistics

Statistic Name	Estimated?	Version Introduced
logical reads	Yes	9.2
buffer busy waits	No	9.2
gc buffer busy	No	10.1
db block changes	Yes	9.2
physical reads	No	9.2
physical writes	No	9.2
physical reads direct	No	9.2
physical writes direct	No	9.2
gc cr blocks received	No	9.2
gc current blocks received	No	9.2
ITL waits	No	9.2
row lock waits	No	9.2
space used	No	10.1
space allocated	No	10.1
segment scans	No	10.1

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

Statistic Name (Linux 10.2.0.1)
BUSY_TIME
IDLE_TIME
LOAD
NICE_TIME
NUM_CPUS
PHYSICAL_MEMORY_BYTES
RSRC_MGR_CPU_WAIT_TIME
SYS_TIME
USER_TIME

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

Dynamic Performance Views
Process Memory

- Oracle 10.2 and above includes dynamic performance views reporting use of memory by individual processes
  - V$PROCESS_MEMORY
  - V$PROCESS_MEMORY_DETAIL
  - V$PROCESS_MEMORY_DETAIL_PROG
- Memory usage is reported under the following categories
  - SQL
  - PL/SQL
  - JAVA
  - OLAP
  - Freeable
  - Other

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:

GV$SGA

INST_ID	NUMBER
NAME	VARCHAR2(20)
VALUE	NUMBER

V$SGA

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.

Dynamic Performance Views
Global views

- 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
- Other views with similar properties include
  - V$ACTIVE_INSTANCES
  - V$SPPARAMETER

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$ACTIVE_INSTANCES which returns a list of active instances across the cluster and V$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$CLASS_PING	V$GCSHVMASTER_INFO
V$CLUSTER_INTERCONNECTS	V$GCSPFMASTER_INFO
V$CONFIGURED_INTERCONNECTS	V$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$DLM_LATCH	V$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.

Dynamic Performance Views CATCLUST.SQL

- **Some additional views/synonyms are created for RAC databases using $ORACLE_HOME/rdbms/admin/catclust.sql**

Synonym Name	View Name
V$GES_CONVERT_LOCAL	V$DLM_CONVERT_LOCAL
V$GES_CONVERT_REMOTE	V$DLM_CONVERT_REMOTE
V$GES_LATCH	V$DLM_LATCH
V$GES_RESOURCE	V$DLM_RESS
V$GES_STATISTICS	V$DLM_MISC
V$GES_TRAFFIC_CONTROLLER	V$DLM_TRAFFIC_CONTROLLER
GV$GES_CONVERT_LOCAL	GV$DLM_CONVERT_LOCAL
GV$GES_CONVERT_REMOTE	GV$DLM_CONVERT_REMOTE
GV$GES_LATCH	GV$DLM_LATCH
GV$GES_RESOURCE	GV$DLM_RESS
GV$GES_STATISTICS	GV$DLM_MISC
GV$GES_TRAFFIC_CONTROLLER	GV$DLM_TRAFFIC_CONTROLLER

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**

Group Name	Interval Size	# Metrics
Event Metrics	6000	3
Event Class Metrics	6000	4
File Metrics Long Duration	60000	6
Service Metrics (short)	500	5
Service Metrics	6000	5
Session Metrics Short Duration	1500	10
Session Metrics Long Duration	6000	1
System Metrics Short Duration	1500	41
System Metrics Long Duration	6000	134
Tablespace Metrics Long Duration	6000	2

The above table was generated using the following query:

```
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

Server Alerts Example

- 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

```
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

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

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

```
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:

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

The syntax for DBMS_APPLICATION_INFO.SET_ACTION is:

```
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.

Trace and Diagnostics
DBMS_MONITOR

- Introduced in Oracle 10.1
- Contains the following subroutines
  - SESSION_TRACE_ENABLE
  - SESSION_TRACE_DISABLE
  - DATABASE_TRACE_ENABLE
  - DATABASE_TRACE_DISABLE
  - CLIENT_ID_TRACE_ENABLE
  - CLIENT_ID_TRACE_DISABLE
  - CLIENT_ID_STAT_ENABLE
  - CLIENT_ID_STAT_DISABLE
  - SERV_MOD_ACT_TRACE_ENABLE
  - SERV_MOD_ACT_TRACE_DISABLE
  - SERV_MOD_ACT_STAT_ENABLE
  - SERV_MOD_ACT_STAT_DISABLE

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:

```
EXECUTE DBMS_MONITOR.SESSION_TRACE_ENABLE;
```

- To disable trace in the current session use:

```
EXECUTE DBMS_MONITOR.SESSION_TRACE_DISABLE;
```

- To enable trace in session 42 use:

```
EXECUTE DBMS_MONITOR.SESSION_TRACE_ENABLE
(SESSION_ID => 42);
```

- To disable trace in session 42 use:

```
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:

```
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:

```
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:

```
EXECUTE DBMS_MONITOR.DATABASE_TRACE_ENABLE;
```

- To disable trace for the entire database use:

```
EXECUTE DBMS_MONITOR.DATABASE_TRACE_DISABLE;
```

- To enable trace for instance RAC1 use:

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

- To disable trace for instance RAC1 use:

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


The syntax for DBMS_MONITOR.DATABASE_TRACE_ENABLE is:

```
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:

```
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.

Trace and Diagnostics
DBMS_MONITOR

- Trace can be enabled for using client identifiers
  - Useful when many sessions connect using the same Oracle user
  - Useful with connection caches
- To set a client identifier use DBMS_SESSION.SET_IDENTIFIER
- For example:

```
BEGIN
    DBMS_SESSION.SET_IDENTIFIER ('CLIENT42');
END;
```

- The client identifier for a specific session is reported in V$SESSION.CLIENT_IDENTIFIER

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_TRACE_ENABLE
(
    CLIENT_ID    VARCHAR2         -- Client ID
);
```

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:

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

    );
END;
```

- To enable trace for ACTION1 use:

```
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
);
```

Trace & Diagnostics
DBMS_MONITOR

- In Oracle 10.1 and above, current trace configuration is reported in DBA_ENABLED_TRACES
- TRACE_TYPE column can be
  - CLIENT_ID
  - SERVICE
  - SERVICE_MODULE
  - SERVICE_MODULE_ACTION
  - DATABASE
- Currently enabled trace aggregations are reported in DBA_ENABLED_AGGREGATIONS

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_AGGREGRATIONS 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>
```

Diagnostics Area

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

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

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

```
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.

Trace & Diagnostics
ORADEBUG

- **ORADEBUG includes LKDEBUG**
  - **Must be run by in SQL*Plus by user with SYSDBA privilege**

```
SQL> ORADEBUG LKDEBUG HELP

Usage:lkdebug [options]
 -l [r|p] <enqueue pointer>     Enqueue Object
 -r <resource pointer>          Resource Object
 -b <gcs shadow pointer>        GCS shadow Object
 -p <process id>                client pid
 -P <process pointer>           Process Object
 -O <i1> <i2> <types>           Oracle Format resname
 -a <res/lock/proc>             all <res/lock/proc> pointer
 -A <res/lock/proc>             all <res/lock/proc> contexts
 -a <res> [<type>]              all <res> pointers by an optional type
 -a convlock                    all converting enqueue (pointers)
 -A convlock                    all converting enqueue contexts
 -a convres                     all res ptr with converting enqueues
 -A convres                     all res contexts with converting enqueues
```


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.

Trace & Diagnostics
ORADEBUG

- **LKDEBUG continued...**

-a name	list all resource names
-a hashcount	list all resource hash bucket counts
-t	Traffic controller info
-s	summary of all enqueue types
-k	GES SGA summary info
-m pkey <objectno>	request for remastering this object at current instance
-m dpkey <objectno>	request for dissolving remastering of this object at current instance

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.

Trace & Diagnostics
Numeric Events

- 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

```
ALTER 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:

10425	Global enqueue operations
10426	GES/GCSD reconfiguration
10427	GES traffic controller
10428	GES cached resource
10429	GES IPC calls
10430	GES/GCS dynamic remastering
10432	GCS Fusion calls (part 1)
10435	GES Deadlock detection
10439	GCS Fusion calls (part 2)
10704	Local enqueue manipulation
10706	Global enqueue manipulation
10708	RAC buffer cache

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 | 2 | 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:

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

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**

Level#	Level Name	Description
0	OFF	Disable tracing
4	USER	Include user errors
6	ADMIN	Include administrative level errors
16	SUPPORT	Include packet contents

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

89 juliandyke.com

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