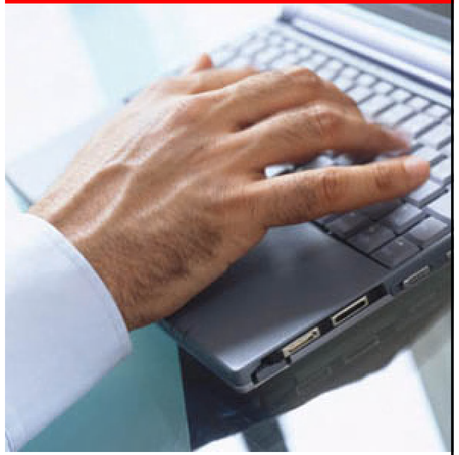
ORACLE®

Advanced Performance Diagnostics: What the GUI (Does and) Doesn't Show You

Nicholas J. Donatone
Principal Grid Sales Consultant

Agenda

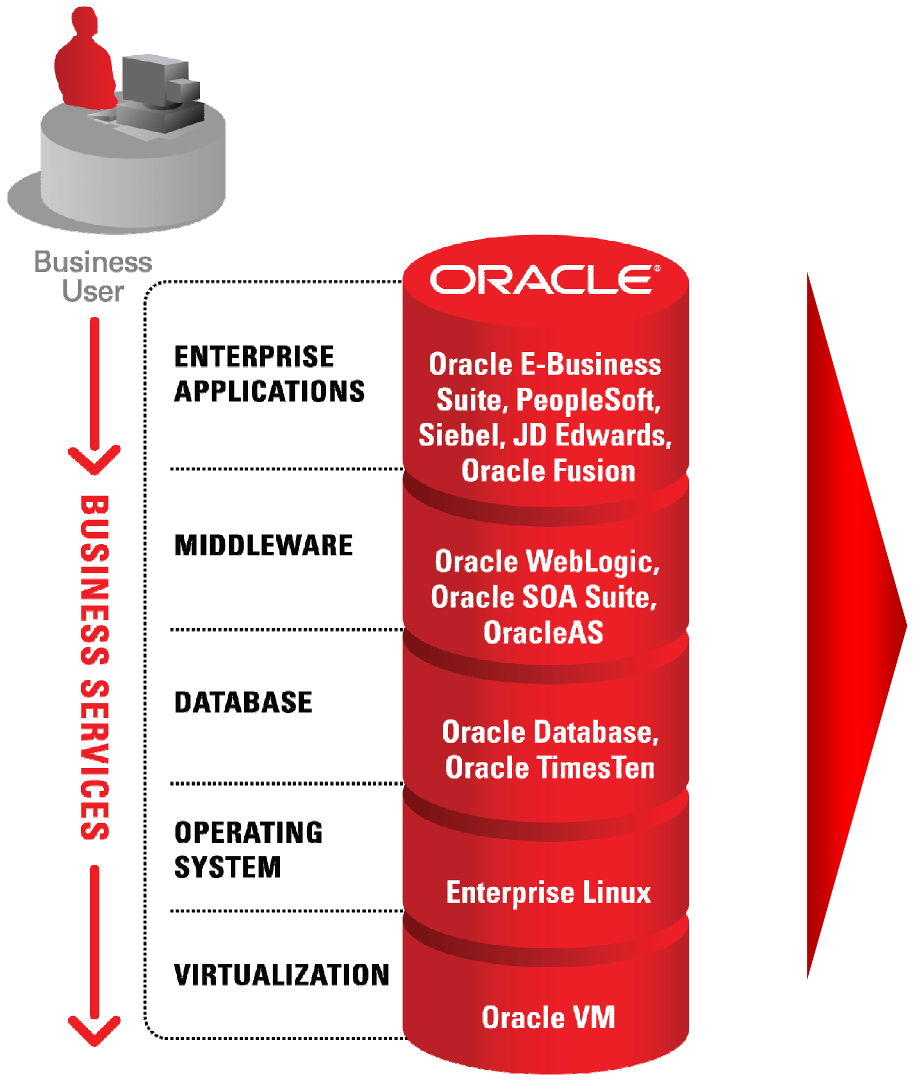


- Review of Performance Methodology
- Review - AWR versus ASH
- Interesting Reports
- Mining your data

Why Oracle Enterprise Manager?

Oracle's Complete Enterprise Software Stack

Built-in & Integrated Manageability



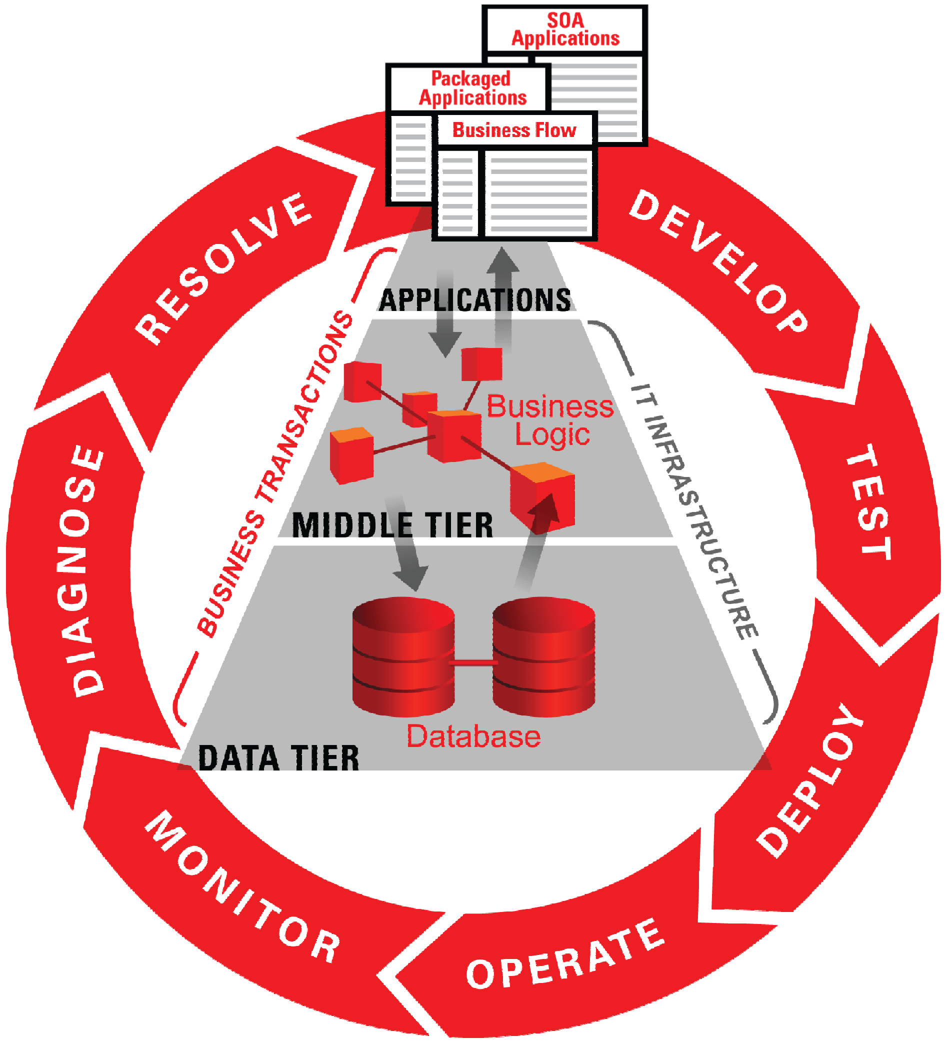
Business User
BUSINESS SERVICES
ENTERPRISE APPLICATIONS
MIDDLEWARE
DATABASE
OPERATING SYSTEM
VIRTUALIZATION
ORACLE
Oracle E-Business Suite, PeopleSoft, Siebel, JD Edwards, Oracle Fusion
Oracle WebLogic, Oracle SOA Suite, OracleAS
Oracle Database, Oracle TimesTen
Enterprise Linux
Oracle VM

- Leader in the complete enterprise application stack
- Built-in manageability in every tier
- Integrated manageability across the entire stack

Oracle Enterprise Manager

Increases Business Efficiency

- **Manage applications top-down, from the business perspective** by understanding user experiences and business impact of IT issues
- **Manage entire application lifecycle to increase business agility** with comprehensive application quality management and compliance solutions
- **Reduce operational costs** through intelligent diagnostics and automated IT processes



ORACLE

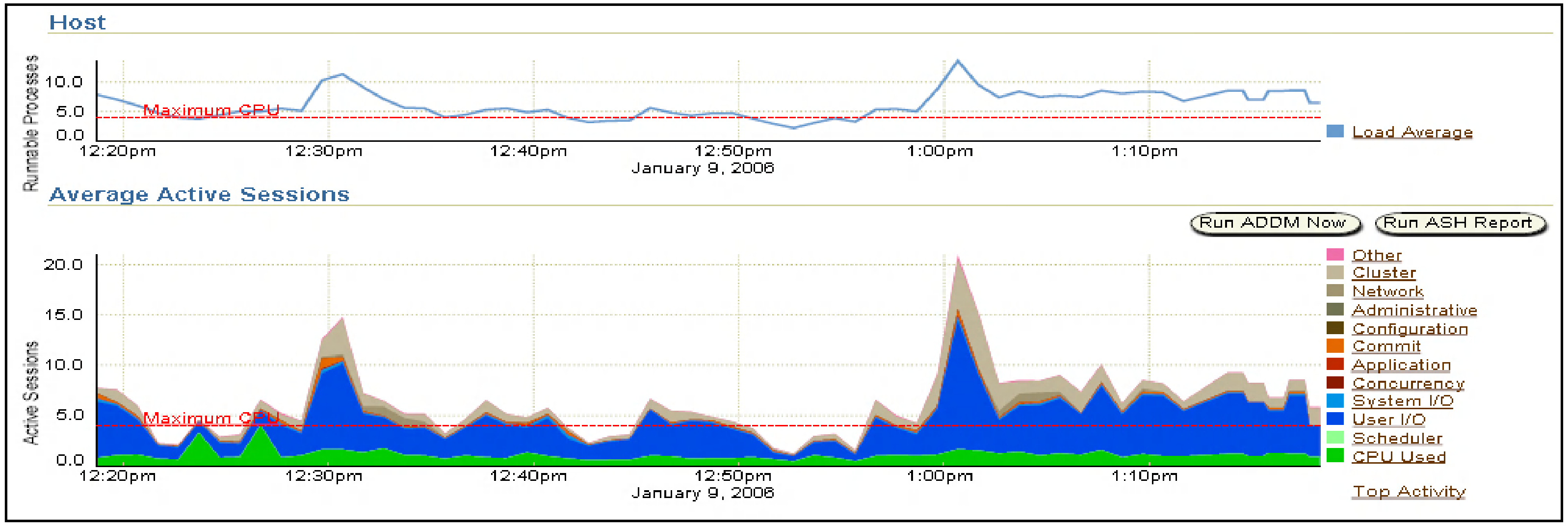
Oracle’s Performance Methodology

- Methodology has evolved with each release
  - Oracle 7
    - Wait events instrumentation
    - BSTAT, ESTAT
  - Oracle 8
    - STATSPACK
  - Oracle 10g and 11g
    - Enhanced Time-Wait Model
    - “Database Time (DB)” Based Methodology

Oracle’s Performance Methodology

- How to tune your system for a given workload?
  - Identify operations consuming most DB Time
  - Identify resource/capacity related bottlenecks
  - Reduce “DB Time” consumed for the workload
- EM embodies Methodology + Best Practice
  - Workflows based on Methodology
  - Problem determination is few mouse clicks away

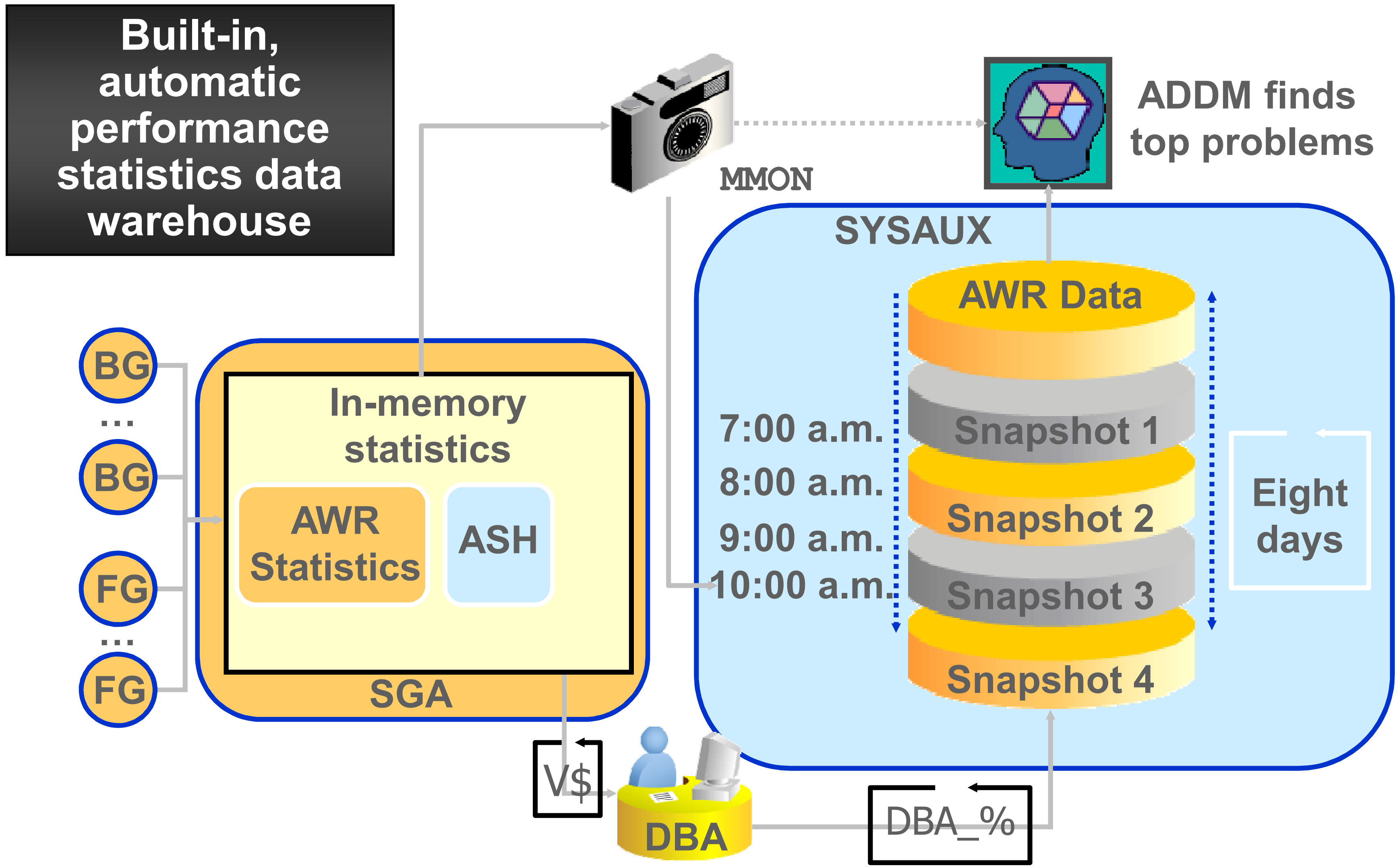
EM Performance Page



- How do you tune an Oracle database using EM's Performance Page?
  - Simplest Answer: "Follow ADDM Recommendations"
  - Simple Answer: "Click on the biggest block of color"

AWR versus ASH

Automatic Workload Repository (AWR)



AWR

- Built-in workload and performance statistics repository in the database
- Automatically Captures Workload Data
- Stores different classes of data:

	Example
Counter Statistics	Number of Executions
Time Statistics	DB Time
Metrics / Rates	Physical Reads / Second
SQL Statistics	Disk Reads (Per SQL statement)
Sampled Data	Session Waits

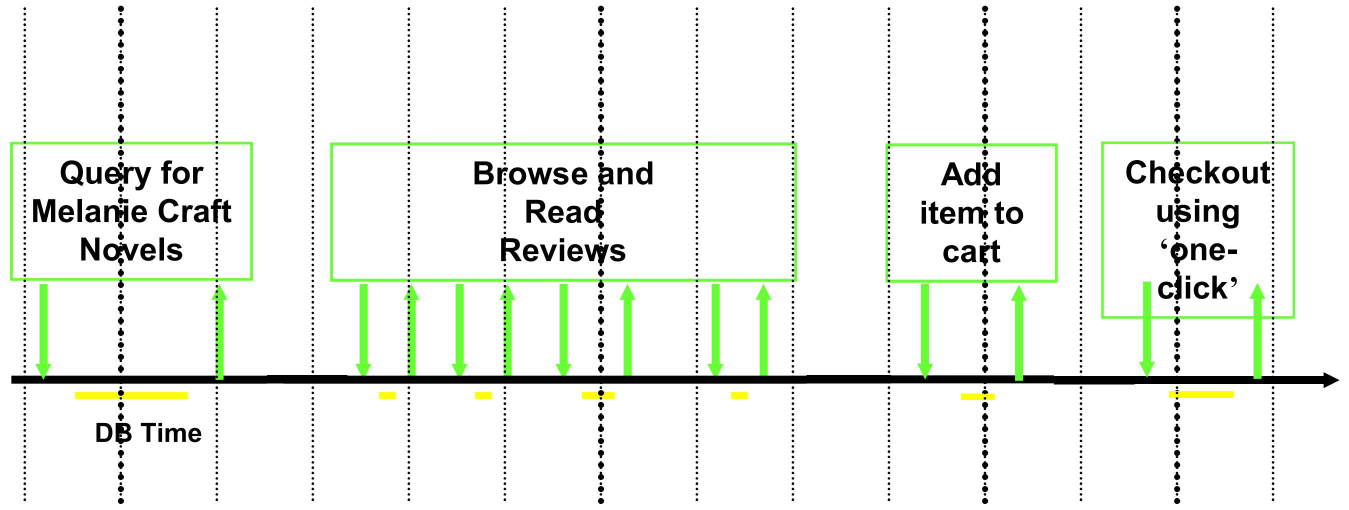
AWR data

- During snapshots, flushed from V$ views to DBA_HIST_* tables
- Interesting Performance tables:
    - DBA_HIST_SNAPSHOT
        - Snapshots in the AWR
        - Join to other tables to constrain the time frame
    - DBA_HIST_SYSTEM_EVENT
        - Information on total waits and times for an event
    - DBA_HIST_SYS_TIME_MODEL
        - System Time Model statistics
    - DBA_HIST_SQLSTAT
        - SQL statistics over time

Active Session History (ASH)

- ASH is session level data
- Active sessions sampled and persisted in-memory
  - Sampling interval = 1 second
  - V$ACTIVE_SESSION_HISTORY
  - Foreground and background sessions are sampled
- On-disk persistence
  - DBA_HIST_ACTIVE_SESS_HISTORY
- ASH is a many-dimensional FACT table
  - Dimensions are V$SESSION columns
  - Fact is that DB time was accumulating over these dimensions
- ASH is a system-wide trace of what happened

Active Session History (ASH)



Time	SID	Module	SQL ID	State	Event
7:38:26	213	Book by author	qa324jffritcf	WAITING	db file sequential read
7:42:35	213	Get review id	aferv5desfzs5	CPU	
7:50:59	213	Add to cart	hk32pekfcbdfr	WAITING	buffer busy wait
7:52:33	213	One click	abngldf95f4de	WAITING	log file sync

ASH

- Can be used for
  - Transient performance problems

  - Targeted performance analysis by various dimensions
    - SQL_ID
    - session
    - module
    - service
    - wait_class

AWR versus ASH Summary

	AWR	ASH
Instance Wide data	Yes	Yes
Time Based data	Yes	Yes
Counts/occurrence data	Yes	No
Analyze any time period	No	Yes
Detailed session level data	No	Yes
Individual wait event data	No	Yes
Sampled data	No	Yes
Time based analysis	Yes	Yes

Resources in $ORACLE_HOME/rdbms/admin

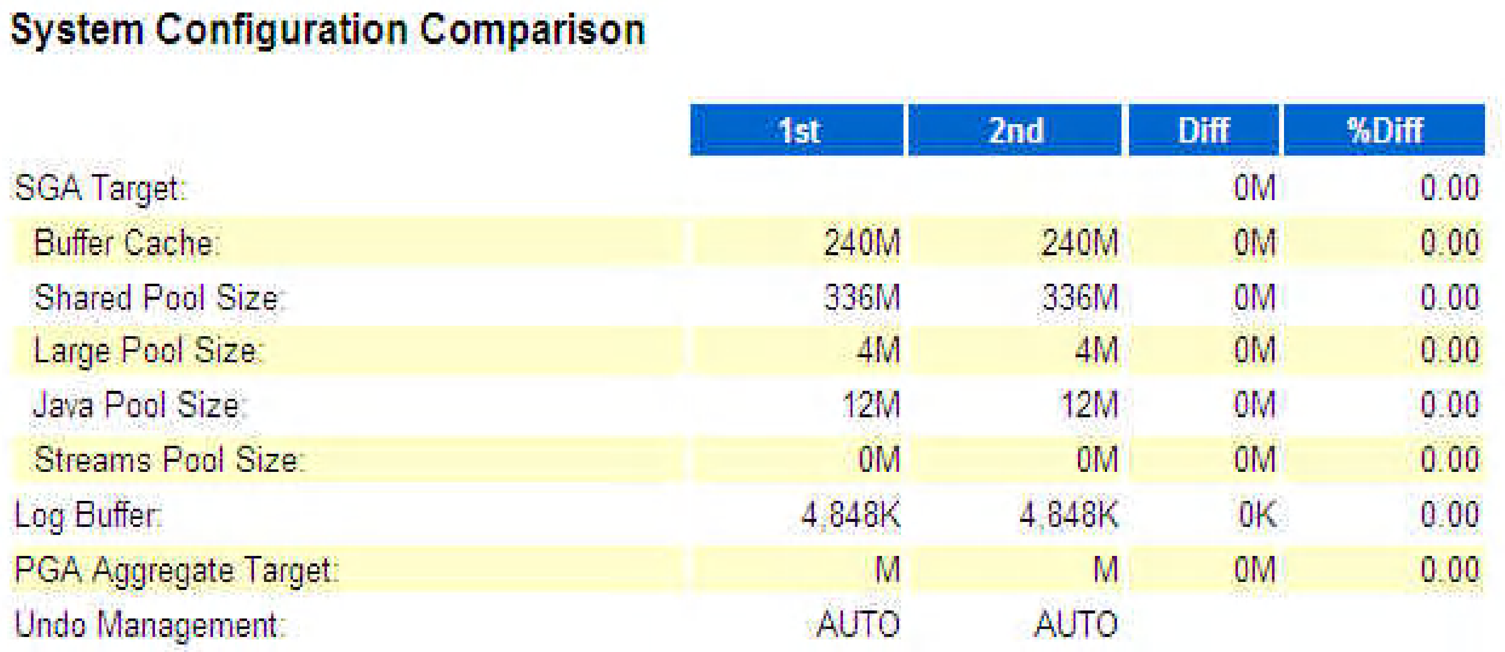
- Available report scripts
  - Common reports
    - awrrpt.sql
    - ashrpt.sql
    - addmrpt.sql
  - Less Well Known reports
    - ashrpti.sql
    - awrddrpt.sql
    - awrsqrpt.sql
    - spawrrac.sql

ashrpti.sql

- ASH report for dimensions in addition to time
  - SQL_ID
  - session
  - service
  - wait_class
  - client_id

awrddrpt.sql

- AWR Compare Periods Report
  - Good for finding out ‘what changed’ in the instance
  - Use Case
    - Overall system performance resulting from SQL tuning
      - Two snapshots - before and after SQL tuning



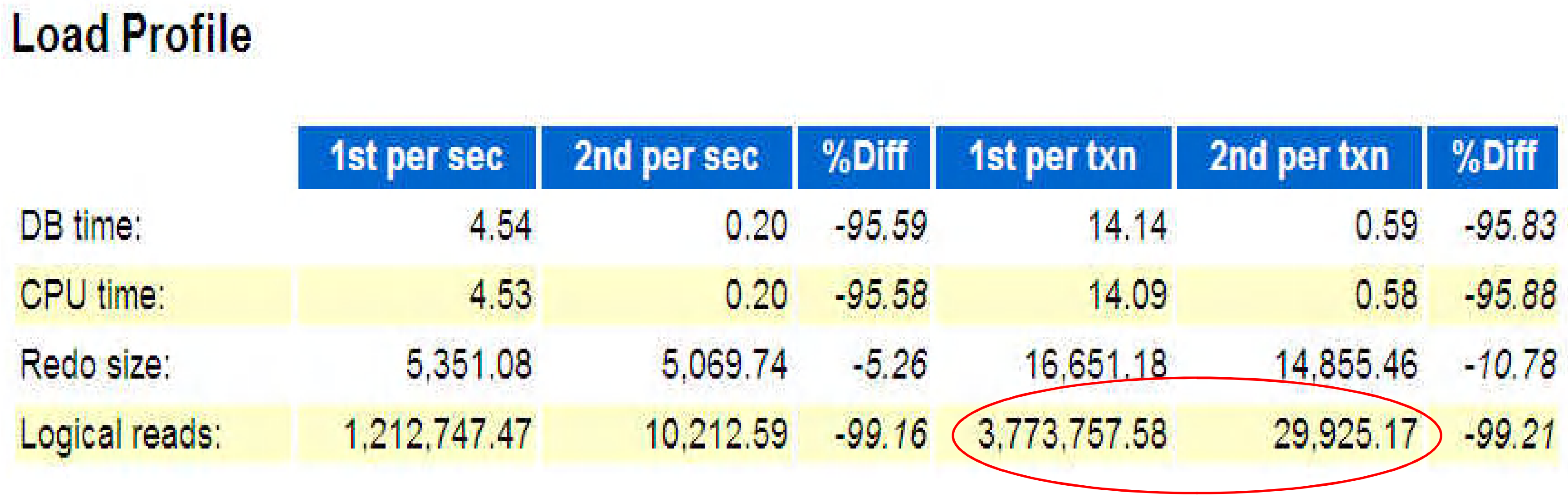
System Configuration Comparison

	1st	2nd	Diff	%Diff
SGA Target:			0M	0.00
Buffer Cache:	240M	240M	0M	0.00
Shared Pool Size:	336M	336M	0M	0.00
Large Pool Size:	4M	4M	0M	0.00
Java Pool Size:	12M	12M	0M	0.00
Streams Pool Size:	0M	0M	0M	0.00
Log Buffer:	4,848K	4,848K	0K	0.00
PGA Aggregate Target:	M	M	0M	0.00
Undo Management:	AUTO	AUTO		

ORACLE

awrddrpt.sql

- System wide ‘Logical Reads per TXN’ significantly reduced



Load Profile

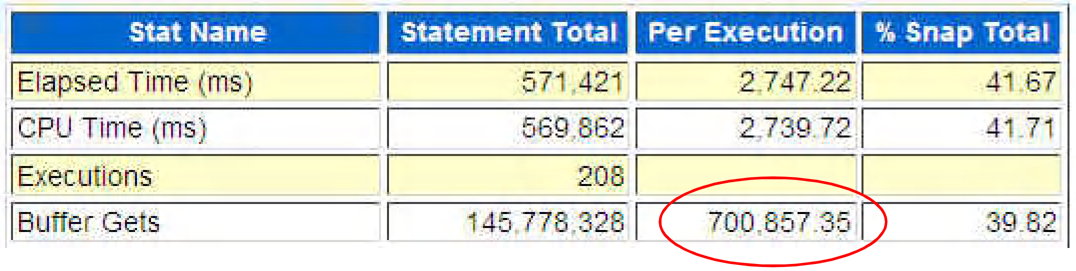
	1st per sec	2nd per sec	%Diff	1st per txn	2nd per txn	%Diff
DB time:	4.54	0.20	-95.59	14.14	0.59	-95.83
CPU time:	4.53	0.20	-95.58	14.09	0.58	-95.88
Redo size:	5,351.08	5,069.74	-5.26	16,651.18	14,855.46	-10.78
Logical reads:	1,212,747.47	10,212.59	-99.16	3,773,757.58	29,925.17	-99.21

awrsqrpt.sql

- AWR Report for a particular SQL Statement
  - Useful for researching individual SQL statement plan changes over time
  - Use Case
    - Single SQL statement, before and after tuning
    - Buffer gets substantially decreased

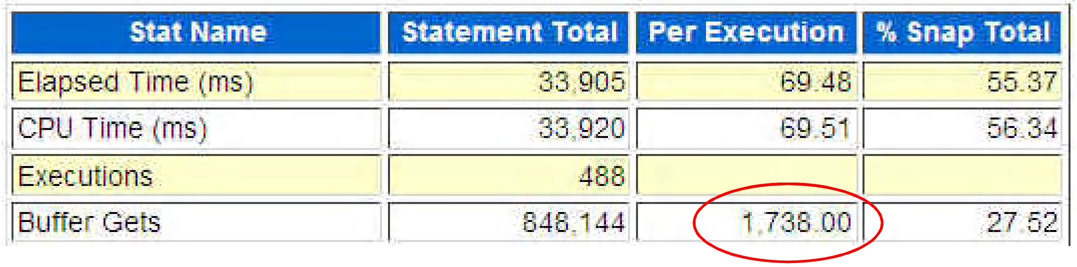
Plan Statistics

Before tuning



Stat Name	Statement Total	Per Execution	% Snap Total
Elapsed Time (ms)	571,421	2,747.22	41.67
CPU Time (ms)	569,862	2,739.72	41.71
Executions	208		
Buffer Gets	145,778,328	700,857.35	39.82

After tuning



Stat Name	Statement Total	Per Execution	% Snap Total
Elapsed Time (ms)	33,905	69.48	55.37
CPU Time (ms)	33,920	69.51	56.34
Executions	488		
Buffer Gets	848,144	1,738.00	27.52

spawrrac.sql

- Generates global AWR report for all nodes on a cluster
- In 11g
- Supplements Global ADDM in 11g
- Has limitations
  - Text only

spawrrac.sql

- Use Cases
  - How localized are my buffer accesses?
  - How evenly is my workload distributed?
  - What is my cluster-wide physical I/O?

```
Global Cache Efficiency Percentages
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     ------ Buffer Access --------
  I#      Local % Remote %   Disk %
---- ----------- -------- --------
   1       92.71     2.86     4.43
   2       95.45     2.14     2.40
   3       97.19     1.60     1.21
   4       96.51     1.41     2.08
```

```
SysStat
~~~~~~~
             Logical     Physical
  I#           Reads        Reads
---- --------------- ------------
   1     134,798,497    5,969,938
   2     140,324,093    3,371,883
   3      39,300,537      477,181
   4      58,850,603    1,227,469
     --------------- ------------
avg       93,318,433    2,761,618
sum      373,273,730   11,046,471
```

ORACLE

spawrrac.sql

- Significant enhancements planned
  - HTML
  - Subset of Instances
  - Global Diff Report

Additional AWR Scripts

- Moving AWR Data
  - Use Cases
    - To offload analysis from production database
    - To preserve data longer than the default on the production system

    - awrextr.sql
      - extract data from awr
    - awrload.sql
      - load data from awrextr dump file

Using AWR Data For Trending

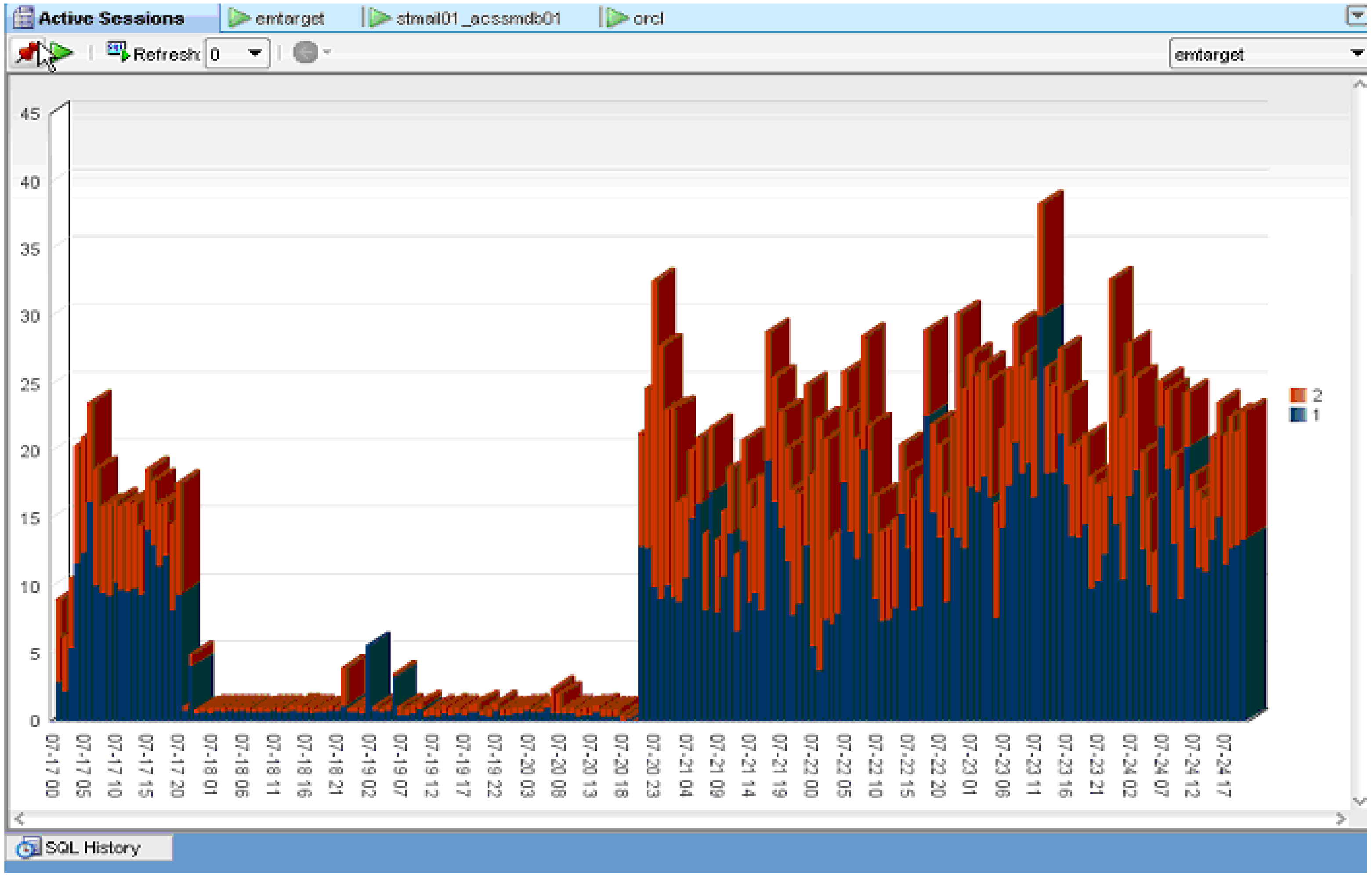
- Common use cases of AWR data are already presented in EM
- Data in DBA_HIST_* tables can be mined to produce data for targeted questions for your company

- Following are some examples to get you started
- These examples were produced using charting capability of SQL Developer
- SQL for these reports are in the appendix

Average Active Sessions

- Average Active Sessions = DBtime / Elapsed Time
  - DBtime
    - Time foreground processes using CPU or non-idle wait events
    - From **dba_hist_sys_time_model**
  - Elapsed Time
    - Calculated from begin / end interval from **dba_hist_snapshot**
- Use Case
  - Longer term trending of RAC cluster
  - Can choose different time ranges
  - Includes data from multiple RAC instances
  - Not broken down by wait events

Average Active Sessions



Active Sessions SQL

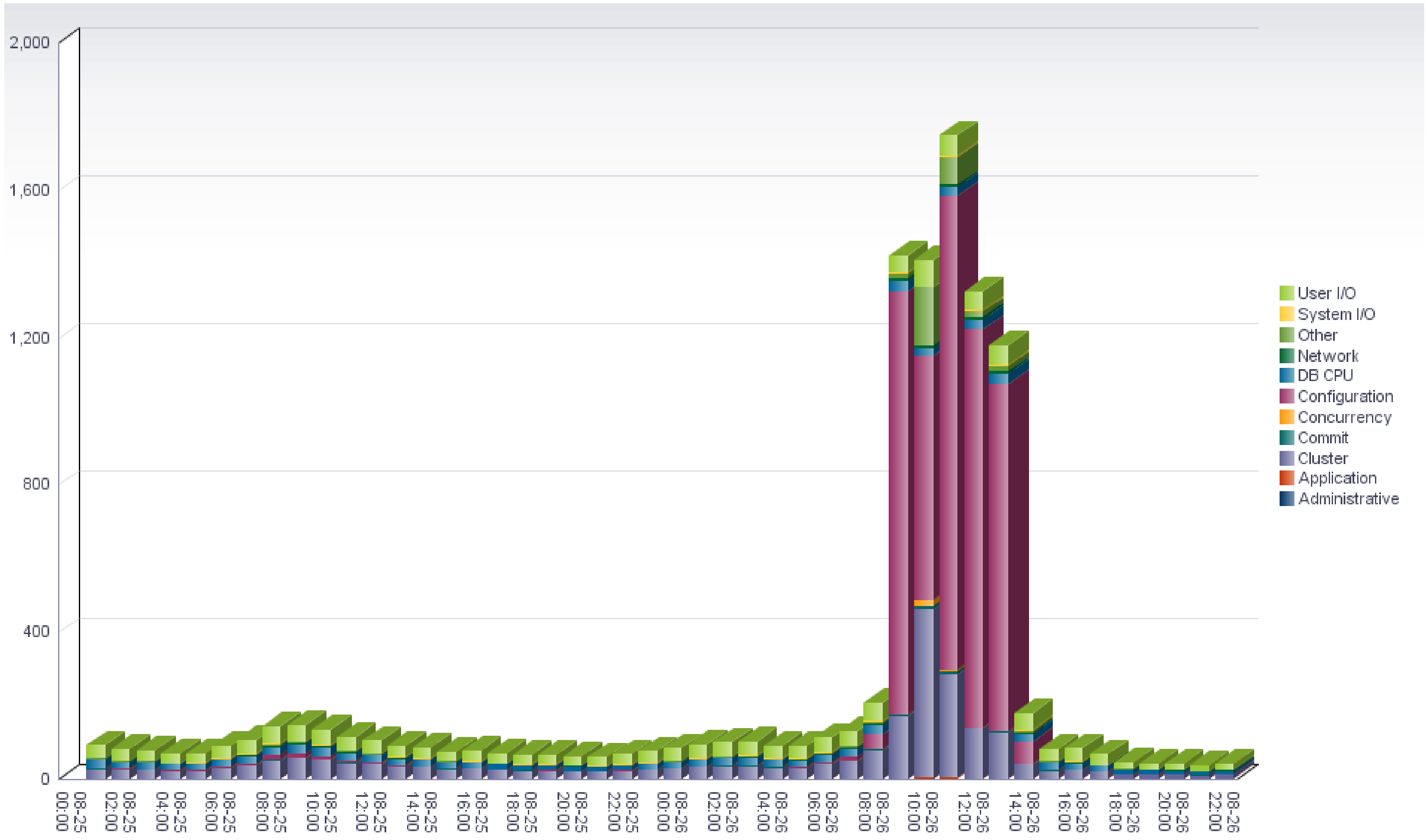
```
define num_days=1
select to_char(end_interval_time,'mm-dd hh24')  snap_time
     , instance_number
     , avg(v_ps)         pSec
  from (
   select end_interval_time
        , instance_number
        , v/ela         v_ps
     from (
      select trunc(s.end_interval_time,'hh24') end_interval_time
           , s.instance_number
           , (case when s.begin_interval_time = s.startup_time
                then value
                else value - lag(value,1) over (partition by sy.stat_id
                                                           , sy.dbid
                                                           , sy.instance_number
                                                           , s.startup_time
                                                   order by sy.snap_id)
            end)/1000000  v
           , (cast(s.end_interval_time as date) - cast(s.begin_interval_time as date))*24*3600   ela
        from dba_hist_snapshot s
           , dba_hist_sys_time_model sy
       where s.dbid = sy.dbid
         and s.instance_number = sy.instance_number
         and s.snap_id = sy.snap_id
         and sy.stat_name = 'DB time'
         and s.end_interval_time > trunc(sysdate) - &num_days))
 group by to_char(end_interval_time,'mm-dd hh24'), instance_number
 order by to_char(end_interval_time,'mm-dd hh24'), instance_number
/
```

Average Active Sessions by Wait Class

- Use Case
  - Longer term trending of RAC cluster
  - Can choose different time ranges
  - Broken down by wait events
  - Includes data from multiple RAC instances
  - Could focus on one class of wait events

- Average Active Sessions = DBtime / Elapsed Time
  - Data comes from
    - **dba_hist_sys_time_model**
    - **dba_hist_snapshot**

Average Active Sessions by Wait Class

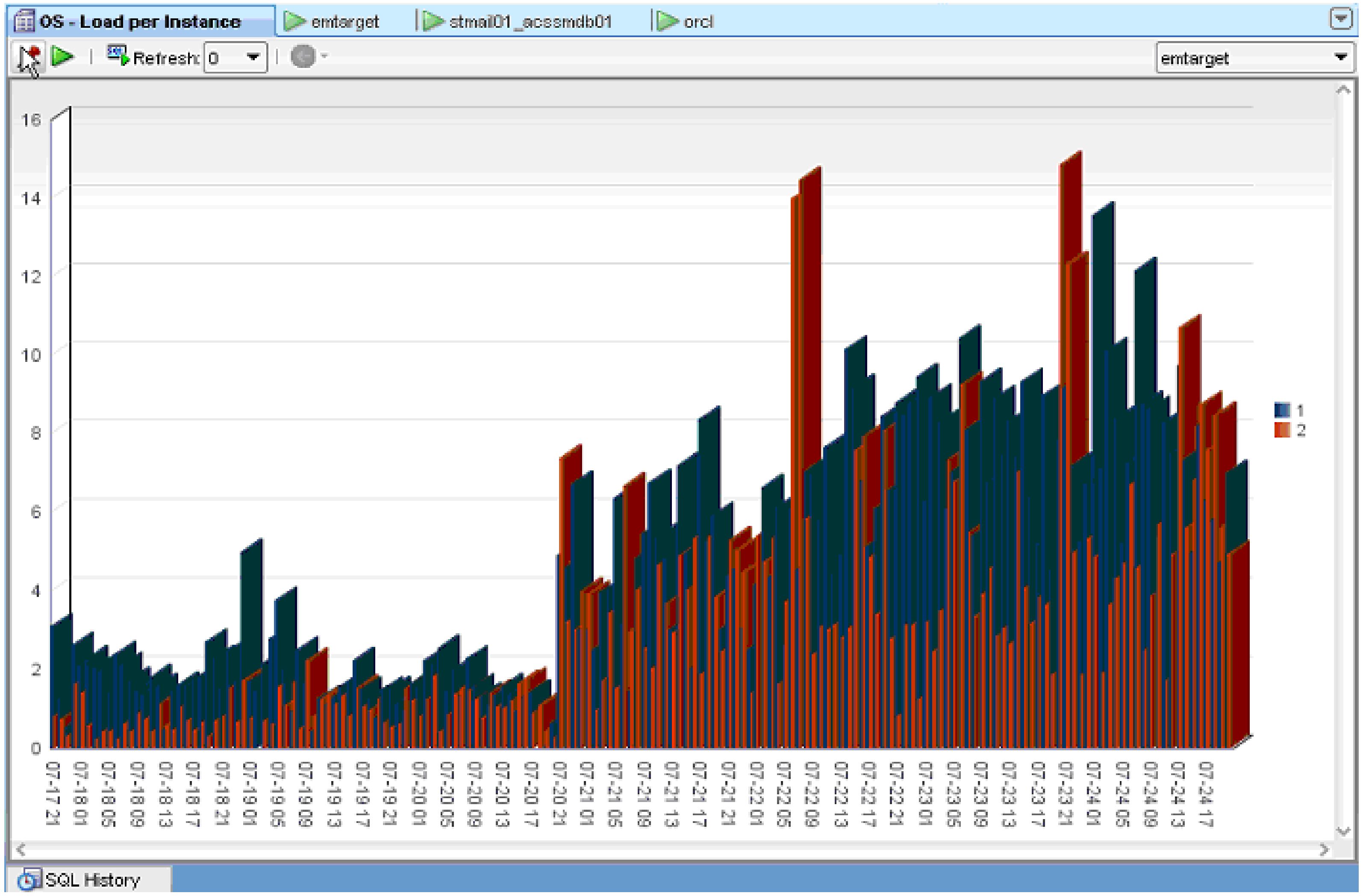


ORACLE

CPU Load

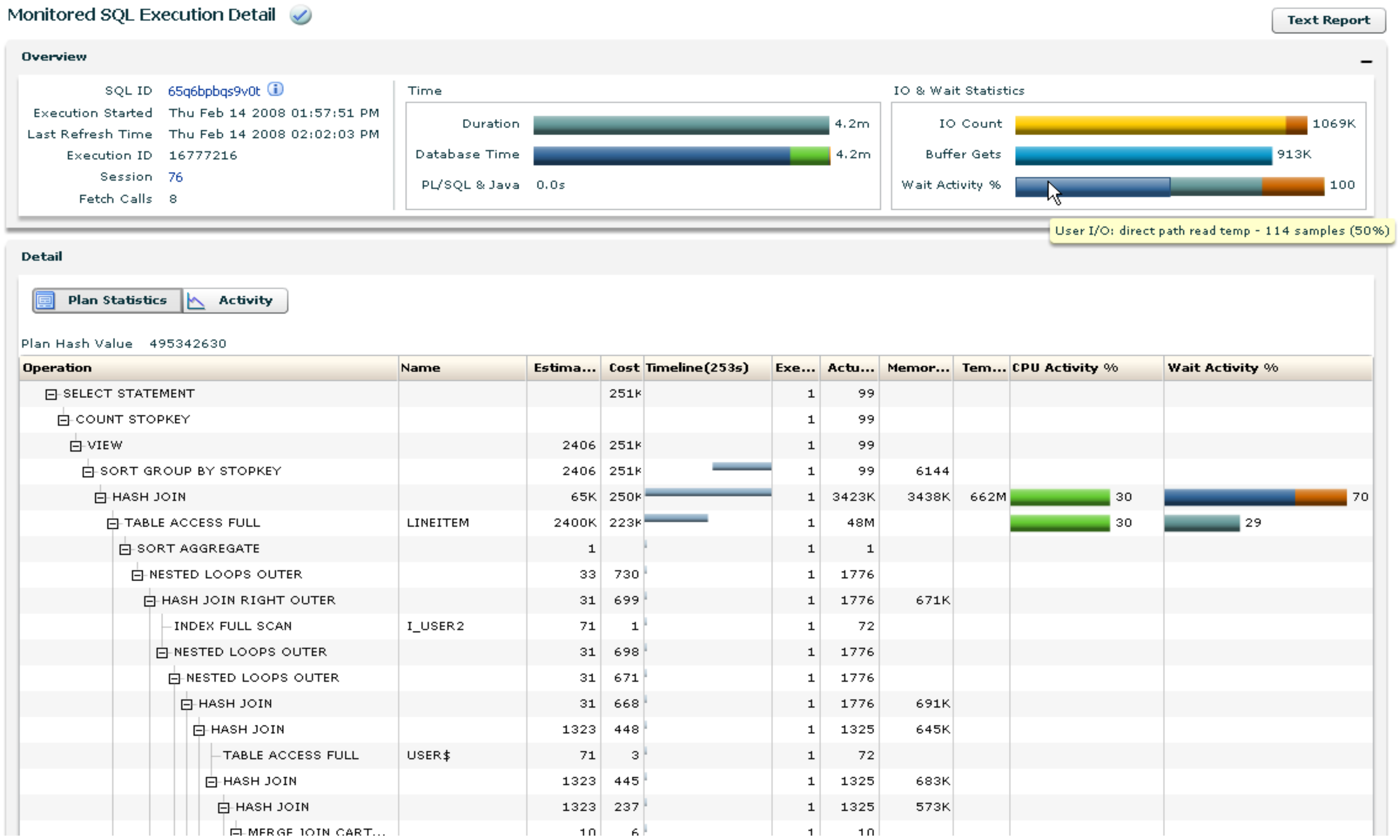
- Data is from **dba_hist_osstat**
- Includes data from two RAC instances
- Data captured during every snapshot, averaged over snapshot time period
- Doesn’t show short term fluctuations

CPU Load


OS - Load per Instance
emtarget
stmail01_acssmdb01
orcl
Refresh: 0
emtarget
16
14
12
10
8
6
4
2
0
07-17 21
07-18 01
07-18 05
07-18 09
07-18 13
07-18 17
07-18 21
07-19 01
07-19 05
07-19 09
07-19 13
07-19 17
07-19 21
07-20 01
07-20 05
07-20 09
07-20 13
07-20 17
07-20 21
07-21 01
07-21 05
07-21 09
07-21 13
07-21 17
07-21 21
07-22 01
07-22 05
07-22 09
07-22 13
07-22 17
07-22 21
07-23 01
07-23 05
07-23 09
07-23 13
07-23 17
07-23 21
07-24 01
07-24 05
07-24 09
07-24 13
07-24 17
1
2
SQL History

Real Time SQL Monitoring

- Explain Plan Shows Progress During SQL Execution
- In 11.1.0.7 DBControl



Monitored SQL Execution Detail

Overview

SQL ID	65q6bpbqs9v0t
Execution Started	Thu Feb 14 2008 01:57:51 PM
Last Refresh Time	Thu Feb 14 2008 02:02:03 PM
Execution ID	16777216
Session	76
Fetch Calls	8

Time

Duration	4.2m
Database Time	4.2m
PL/SQL & Java	0.0s

IO & Wait Statistics

IO Count	1069K
Buffer Gets	913K
Wait Activity %	100

User I/O: direct path read temp - 114 samples (50%)

Detail

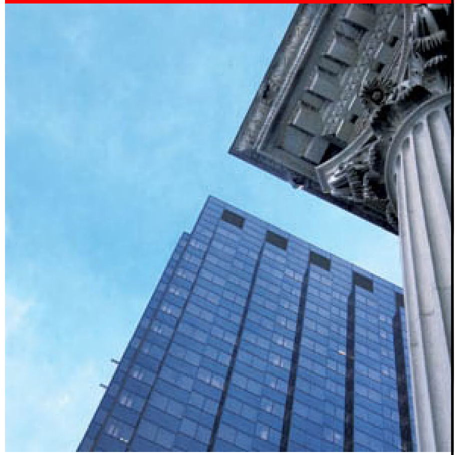
Plan Hash Value 495342630

Operation	Name	Estima...	Cost	Timeline(253s)	Exe...	Actu...	Memor...	Tem...	CPU Activity %	Wait Activity %
SELECT STATEMENT			251K		1	99				
COUNT STOPKEY					1	99				
VIEW		2406	251K		1	99				
SORT GROUP BY STOPKEY		2406	251K		1	99	6144			
HASH JOIN		65K	250K		1	3423K	3438K	662M	30	70
TABLE ACCESS FULL	LINEITEM	2400K	223K		1	48M			30	29
SORT AGGREGATE		1			1	1				
NESTED LOOPS OUTER		33	730		1	1776				
HASH JOIN RIGHT OUTER		31	699		1	1776	671K			
INDEX FULL SCAN	I_USER2	71	1		1	72				
NESTED LOOPS OUTER		31	698		1	1776				
NESTED LOOPS OUTER		31	671		1	1776				
HASH JOIN		31	668		1	1776	691K			
HASH JOIN		1323	448		1	1325	645K			
TABLE ACCESS FULL	USER$	71	3		1	72				
HASH JOIN		1323	445		1	1325	683K			
HASH JOIN		1323	237		1	1325	573K			
MERGE JOIN CART...		10	6		1	10				

Real Time SQL Monitoring

- In 11.1.0.6
  - DBMS_SQLTUNE.REPORT_SQL_MONITOR
- Views
  - v$sql_monitor
  - v$sql_plan_monitor

ORACLE IS THE INFORMATION COMPANY



Appendix

Active Sessions SQL

```
define num_days=1
select to_char(end_interval_time,'mm-dd hh24')  snap_time
     , instance_number
     , avg(v_ps)         pSec
  from (
   select end_interval_time
        , instance_number
        , v/ela         v_ps
     from (
      select trunc(s.end_interval_time,'hh24') end_interval_time
           , s.instance_number
           , (case when s.begin_interval_time = s.startup_time
                  then value
                  else value - lag(value,1) over (partition by sy.stat_id
                                                             , sy.dbid
                                                             , sy.instance_number
                                                             , s.startup_time
                                                      order by sy.snap_id)
             end)/1000000  v
           , (cast(s.end_interval_time as date) - cast(s.begin_interval_time as date))*24*3600   ela
        from dba_hist_snapshot s
           , dba_hist_sys_time_model sy
       where s.dbid = sy.dbid
         and s.instance_number = sy.instance_number
         and s.snap_id = sy.snap_id
         and sy.stat_name = 'DB time'
         and s.end_interval_time > trunc(sysdate) - &num_days))
 group by to_char(end_interval_time,'mm-dd hh24'), instance_number
 order by to_char(end_interval_time,'mm-dd hh24'), instance_number
/
```

Active Sessions Per Wait Class SQL

```
define num_days = 1
select to_char(end_time,'mm-dd hh24') snap_time
     , wait_class
     , sum(pSec)    avg_sess
  from
      (select end_time
        , wait_class
        , p_tmfg/1000000/ela   pSec
     from (
       select trunc(s.end_interval_time,'hh24') end_time
              , (cast(s.end_interval_time as date) - cast(s.begin_interval_time as date))*24*3600 ela
              , s.snap_id
              , wait_class
              , e.event_name
              , case when s.begin_interval_time = s.startup_time
                     then e.time_waited_micro_fg
                     else e.time_waited_micro_fg
                          - lag(time_waited_micro_fg) over (partition by event_id
                                                                        , e.dbid
                                                                        , e.instance_number
                                                                        , s.startup_time
                                                                  order by e.snap_id)
                end    p_tmfg
         from dba_hist_snapshot s
            , dba_hist_system_event e
        where s.dbid = e.dbid
          and s.instance_number = e.instance_number
          and s.snap_id = e.snap_id
          and s.end_interval_time > trunc(sysdate) - &num_days
          and e.wait_class != 'Idle'
        union all
      /* Continued on next slide */
```

Active Sessions Per Wait Class SQL

```
 /* Continued from previous slide */
select trunc(s.end_interval_time,'hh24') end_time
              , (cast(s.end_interval_time as date) - cast(s.begin_interval_time as date))*24*3600 ela
              , s.snap_id
              , t.stat_name   wait_class
              , t.stat_name   event_name
              , case when s.begin_interval_time = s.startup_time
                     then t.value
                     else t.value
                          - lag(value) over (partition by stat_id
                                                        , t.dbid
                                                        , t.instance_number
                                                        , s.startup_time
                                                 order by t.snap_id)
                end    p_tmfg
         from dba_hist_snapshot s
            , dba_hist_sys_time_model t
        where s.dbid = t.dbid
          and s.instance_number = t.instance_number
          and s.snap_id = t.snap_id
          and s.end_interval_time > trunc(sysdate) - &num_days
          and t.stat_name = 'DB CPU'))
  group by to_char(end_time,'mm-dd hh24'), wait_class
  order by to_char(end_time,'mm-dd hh24'), wait_class
/
```

OS CPU Busy SQL

```
define num_days = 1
select to_char(trunc(end_interval_time,'hh24'),'mm-dd hh24') snap_time
     , instance_number
     , busy/decode(busy+idle,0,null,busy+idle)*100 pct_busy
  from (
    select s.snap_id
         , s.instance_number
         , s.dbid
         , s.end_interval_time
         , os.stat_name
         , case when s.begin_interval_time = s.startup_time
                then os.value
                else os.value - lag(os.value,1) over (partition by os.stat_name
                                                                 , os.instance_number
                                                                 , os.dbid
                                                                 , s.startup_time
                                                          order by os.snap_id)
            end delta_v
      from dba_hist_snapshot s
         , dba_hist_osstat os
     where s.snap_id = os.snap_id
       and s.instance_number = os.instance_number
       and s.dbid            = os.dbid
       and s.end_interval_time > trunc(sysdate) - &num_days
       and os.stat_name in ('BUSY_TIME','IDLE_TIME'))
  pivot (sum(delta_v)
         for stat_name in ('BUSY_TIME'   busy
                          ,'IDLE_TIME'   idle))
  order by to_char(trunc(end_interval_time,'hh24'),'mm-dd hh24'), instance_number
/
```