Taming the AWR Tsunami

Roger Cornejo

roger.d.cornejo@gsk.com

eco East Coast Oracle Users Conference

Raleigh/Durham, NC

November 6, 2013

Roger Cornejo Speaker Background

- Computer Science degree (Rutgers Univ.)
- Working with Oracle over 28 years (since V4)
  - Building/supporting large database applications
  - 1000's of tuning opportunities
- Professional presentations:
  - Benchmarking
  - Data Integration
  - Performance Tuning Bag-of-Tricks

Outline

- AWR Introduction
- How to Use AWR Data
- Tuning Methodology In Practice (examples)
- Exploring AWR data (deeper dive)
- Advanced topics
- Questions

Zip file to be posted

eastcoastoracle.org

AWR Introduction: Can I use AWR?

- Licensing requirements
  - Diagnostics Pack
  - 11g parameter to determine if licensed

    ```
    show parameter control_management_pack_access
    ```
  - 10g – tracked externally

**Note: Could be running, but not licensed
=>
don’t use it if not licensed**

- Is AWR running on this DB?
  - 10g: Scheduled job:

    ```
    select * from dba_scheduler_jobs
    where job_name = 'GATHER_STATS_JOB';
    ```
  - 11g: 3 standard processes in the auto task window

    -- Gather stats, segment space advisor and sql tuning advisor.

    ```
    select * from dba_autotask_operation;
    ```

;

AWR Introduction: Why use AWR?

- Identify Root Cause of performance problems

- Quantify the root cause ***quickly***

- AWR Overcomes issues with timing of monitoring
  - Investigate past performance issues instead of limiting ourselves to present moment monitoring (V$ views)

- AWR is Superior to StatsPack
  - AWR has stats on session
  - Has other stats not in StatPack

AWR Introduction: What’s in AWR?

- Automated Workload Repository
- Gathers and persists performance metrics
  - DBA_HIST Views:
    - 78 in 10g … 111 in 11g
  - Sessions: SQL, waits, blockers, …
  - Workload metrics (e.g. IO; Memory; CPU; …)
  - Object Statistics (e.g. Library Cache; File; Temp; Undo; Latches; Segments; Tablespace; …)
- Standard Tools: AWR, ASH, ADDM, … Reports
  - AWR report consists of dozens of sections; can contain a 100 screens of data or more

How do we make sense of all this information?

How to Use AWR Data:
Methodical Tuning Approach

How to Use AWR Data:
Methodical Tuning Approach

- **Identify**
  - What application issue? [Talk to the user]
    Error? Time? Duration? Instance? User? Module? …
  - *Don’t solve the wrong problem!*
- **Quantify** [using AWR tools and data]
  - Gather information from AWR that applies to ID’d problem
- **Analyze** [your experience along with ADDM Report]
  - Analyzing SQL by examining:
    execution plan; table cardinality (size / # rows); waits; …
  - root cause – known / hypothesized
- **Tune** [Implement Tuning solutions]
- **Test/Evaluate**
- **Monitor** over time using AWR Data

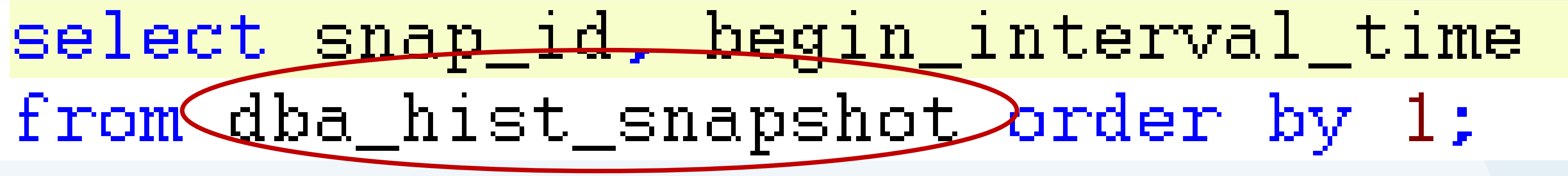
Taming the AWR Tsunami: Case Studies

Case Study 1: Long Running Process Alert e-mail

- Identify
  - Problem: Long Running Process Alert e-mail
  - Instance: UKPRD662
  - Connect Username: GWDMPR61
  - Other: connect time, SQL Id’s, session id’s … reported

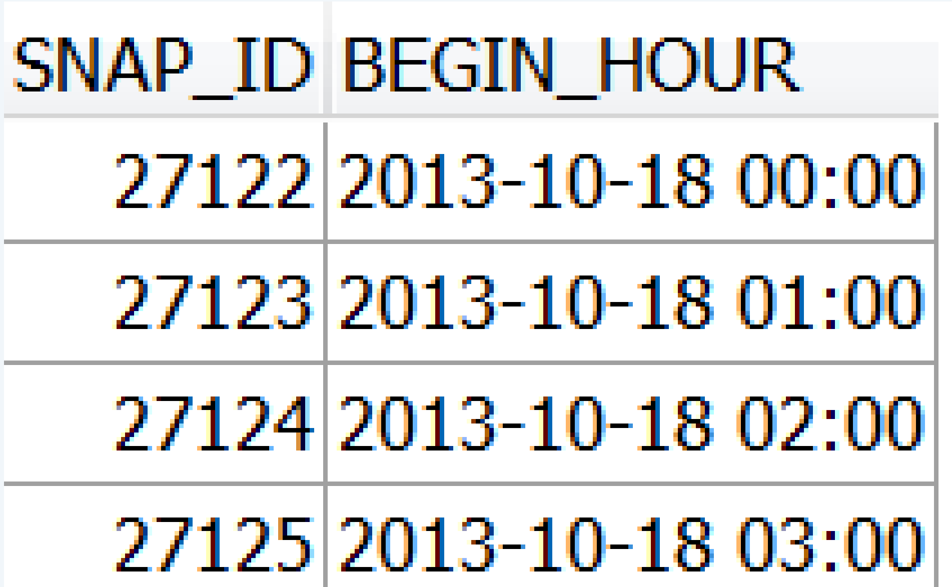
Case Study 1: Long Running Process Alert e-mail

- Quantify – get the snap_id's from AWR dba_hist_snapshot



```
select snap_id, begin_interval_time
from dba_hist_snapshot order by 1;
```

  - snap_id from **start** of the interval
  - snap_id from the **end** of the interval
  - Note and use these snap_id's:
    to subset the AWR data in subsequent analysis

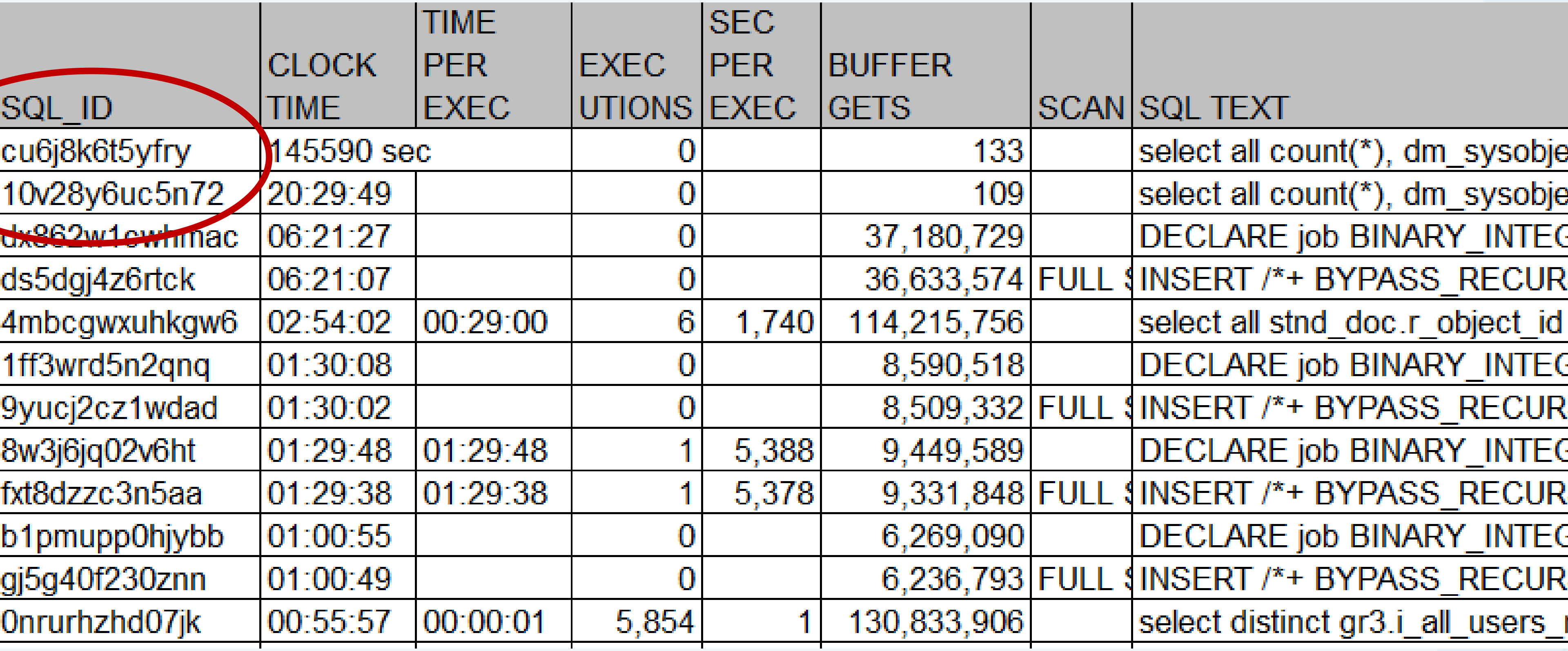


SNAP_ID	BEGIN_HOUR
27122	2013-10-18 00:00
27123	2013-10-18 01:00
27124	2013-10-18 02:00
27125	2013-10-18 03:00

Case Study 1: Long Running Process Alert e-mail

- Quantify - SQL

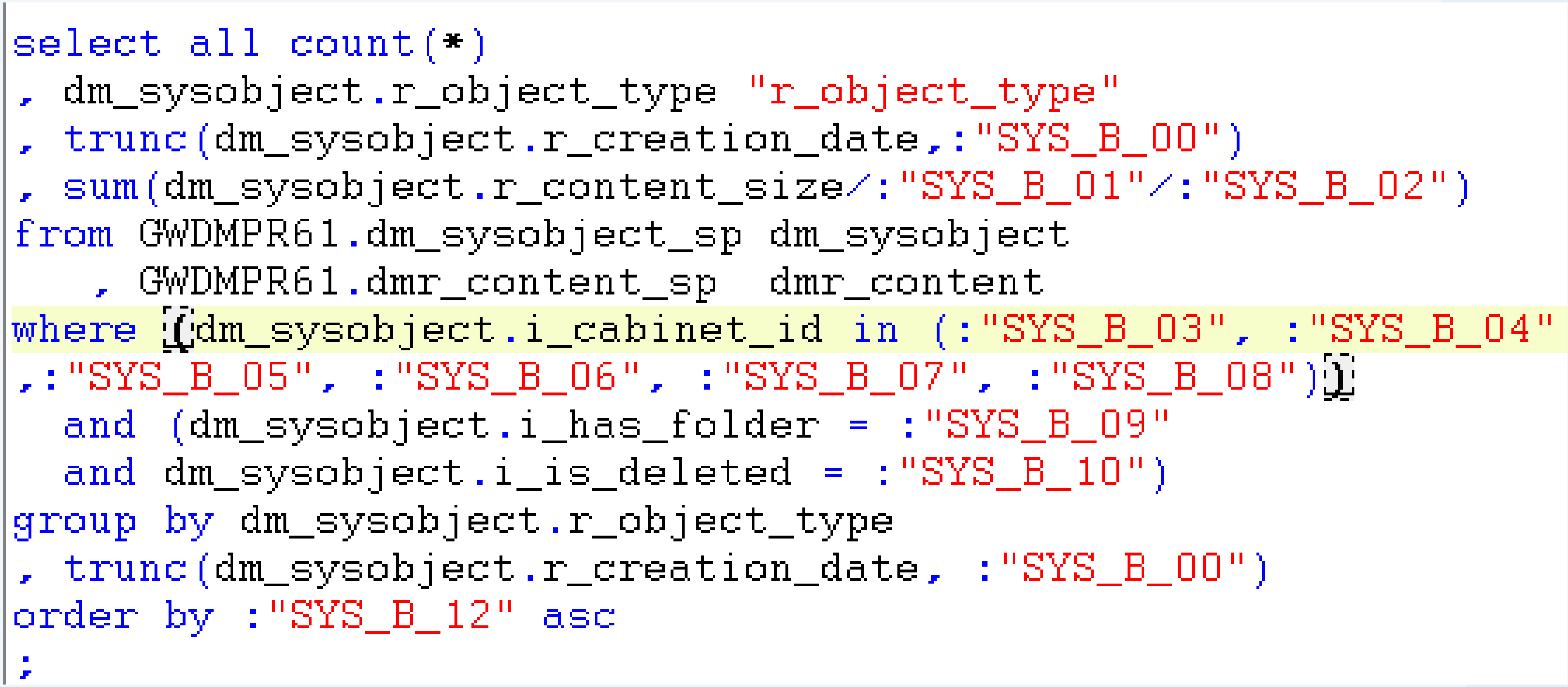
AWR - Expensive SQL for a snap_id.sql



		TIME		SEC			
	CLOCK	PER	EXEC	PER	BUFFER		
SQL_ID	TIME	EXEC	UTIONS	EXEC	GETS	SCAN	SQL TEXT
cu6j8k6t5yfry	145590 sec		0		133		select all count(*), dm_sysobje
10v28y6uc5n72	20:29:49		0		109		select all count(*), dm_sysobje
dx862w1cwhmac	06:21:27		0		37,180,729		DECLARE job BINARY_INTEG
ds5dgj4z6rtck	06:21:07		0		36,633,574	FULL $	INSERT /*+ BYPASS_RECUR
4mbcgwxuhkgw6	02:54:02	00:29:00	6	1,740	114,215,756		select all stnd_doc.r_object_id
1ff3wrd5n2qnq	01:30:08		0		8,590,518		DECLARE job BINARY_INTEG
9yucj2cz1wdad	01:30:02		0		8,509,332	FULL $	INSERT /*+ BYPASS_RECUR
8w3j6jq02v6ht	01:29:48	01:29:48	1	5,388	9,449,589		DECLARE job BINARY_INTEG
fxt8dzzc3n5aa	01:29:38	01:29:38	1	5,378	9,331,848	FULL $	INSERT /*+ BYPASS_RECUR
b1pmupp0hjybb	01:00:55		0		6,269,090		DECLARE job BINARY_INTEG
gj5g40f230znn	01:00:49		0		6,236,793	FULL $	INSERT /*+ BYPASS_RECUR
0nrurhzhd07jk	00:55:57	00:00:01	5,854	1	130,833,906		select distinct gr3.i_all_users_

Case Study 1: Long Running Process Alert e-mail

- Analyze: Query from **DBA_HIST_SQLTEXT**

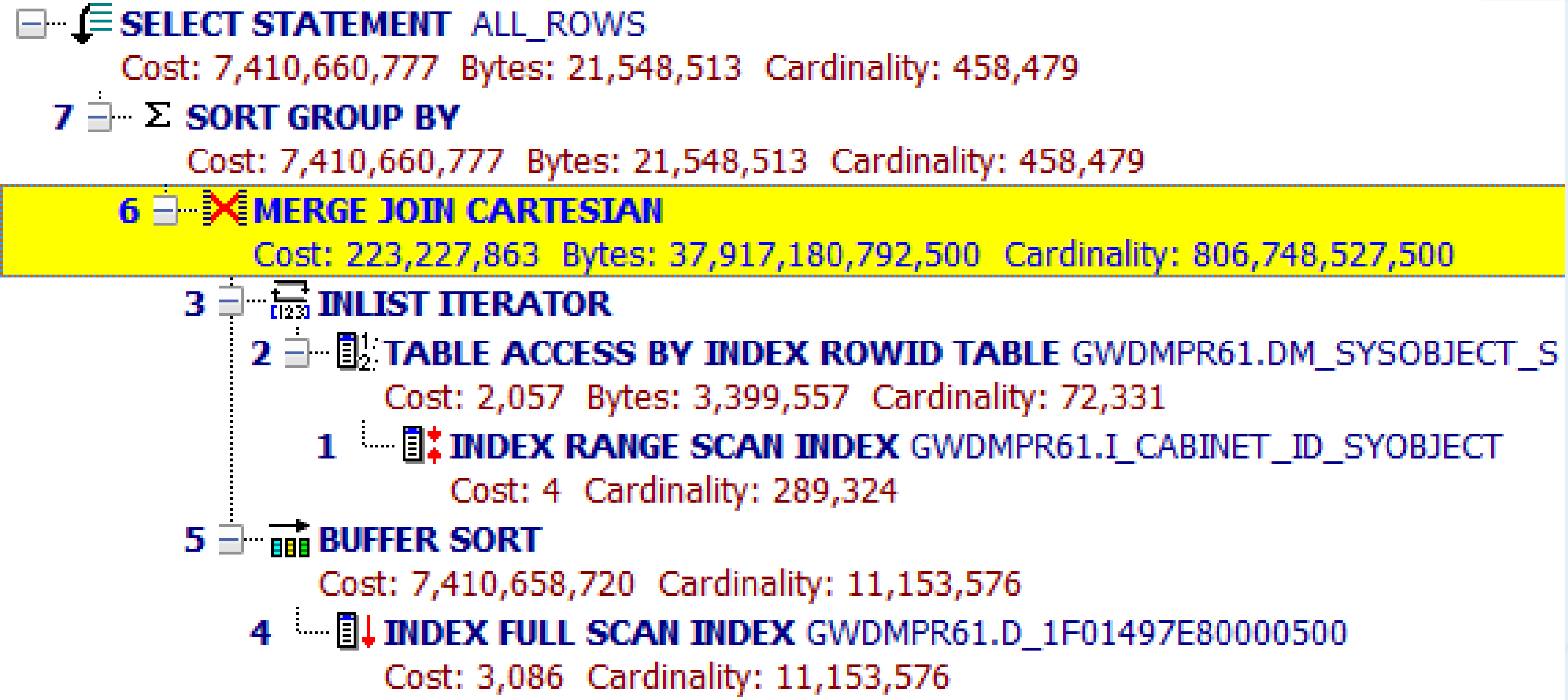


```
select all count(*)
, dm_sysobject.r_object_type "r_object_type"
, trunc(dm_sysobject.r_creation_date,:"SYS_B_00")
, sum(dm_sysobject.r_content_size/:"SYS_B_01"/:"SYS_B_02")
from GWDMPR61.dm_sysobject_sp dm_sysobject
    , GWDMPR61.dmr_content_sp  dmr_content
where (dm_sysobject.i_cabinet_id in (:"SYS_B_03", :"SYS_B_04"
,:"SYS_B_05", :"SYS_B_06", :"SYS_B_07", :"SYS_B_08"))
  and (dm_sysobject.i_has_folder = :"SYS_B_09"
  and dm_sysobject.i_is_deleted = :"SYS_B_10")
group by dm_sysobject.r_object_type
, trunc(dm_sysobject.r_creation_date, :"SYS_B_00")
order by :"SYS_B_12" asc
;
```

Case Study 1: Long Running Process Alert e-mail

- Analyze: Execution Plan from **`DBA_HIST_SQL_PLAN`**

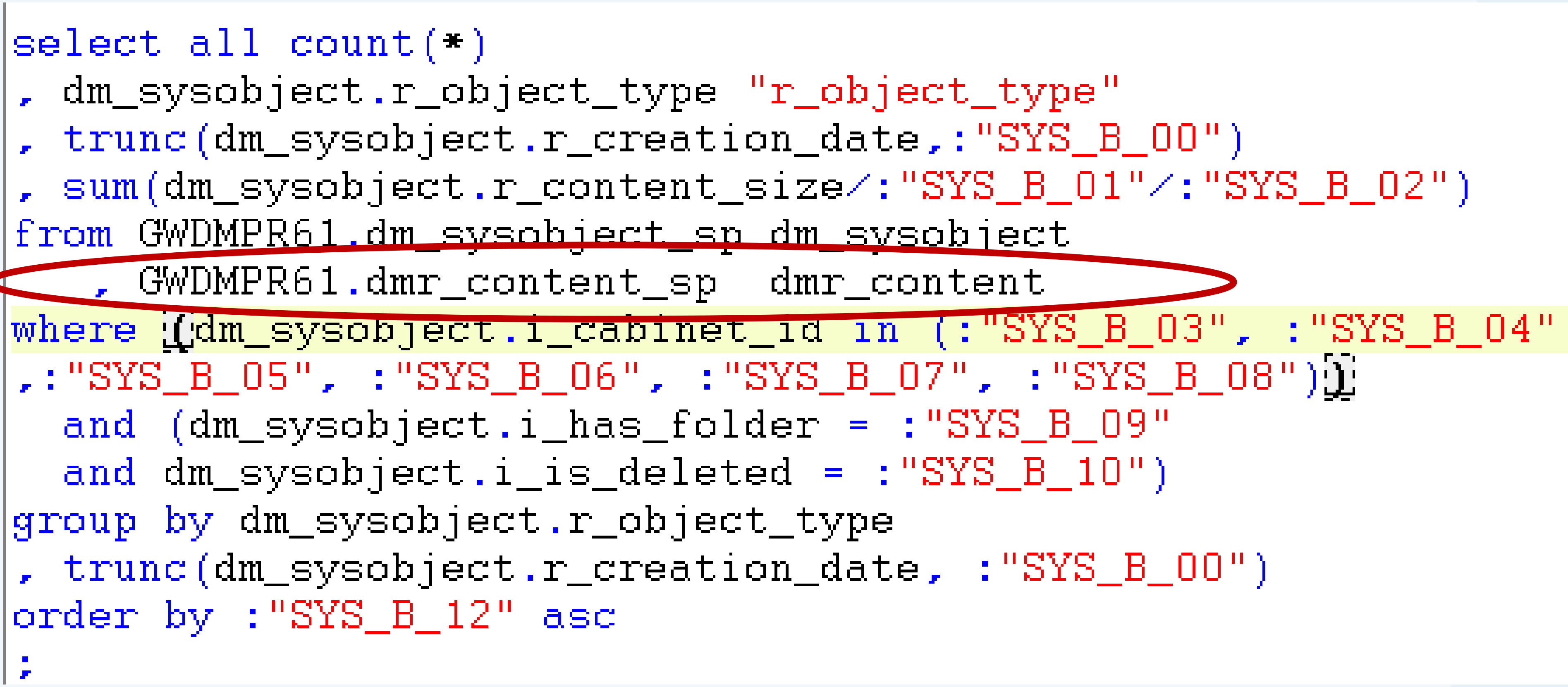
  `dbms_xplan.display_awr('&SQLID')`



```
SELECT STATEMENT  ALL_ROWS
 Cost: 7,410,660,777  Bytes: 21,548,513  Cardinality: 458,479
 7 SORT GROUP BY
   Cost: 7,410,660,777  Bytes: 21,548,513  Cardinality: 458,479
   6 MERGE JOIN CARTESIAN
     Cost: 223,227,863  Bytes: 37,917,180,792,500  Cardinality: 806,748,527,500
     3 INLIST ITERATOR
       2 TABLE ACCESS BY INDEX ROWID TABLE GWDMPR61.DM_SYSOBJECT_S
         Cost: 2,057  Bytes: 3,399,557  Cardinality: 72,331
         1 INDEX RANGE SCAN INDEX GWDMPR61.I_CABINET_ID_SYOBJECT
           Cost: 4  Cardinality: 289,324
     5 BUFFER SORT
       Cost: 7,410,658,720  Cardinality: 11,153,576
       4 INDEX FULL SCAN INDEX GWDMPR61.D_1F01497E80000500
         Cost: 3,086  Cardinality: 11,153,576
```

Case Study 1: Long Running Process Alert e-mail

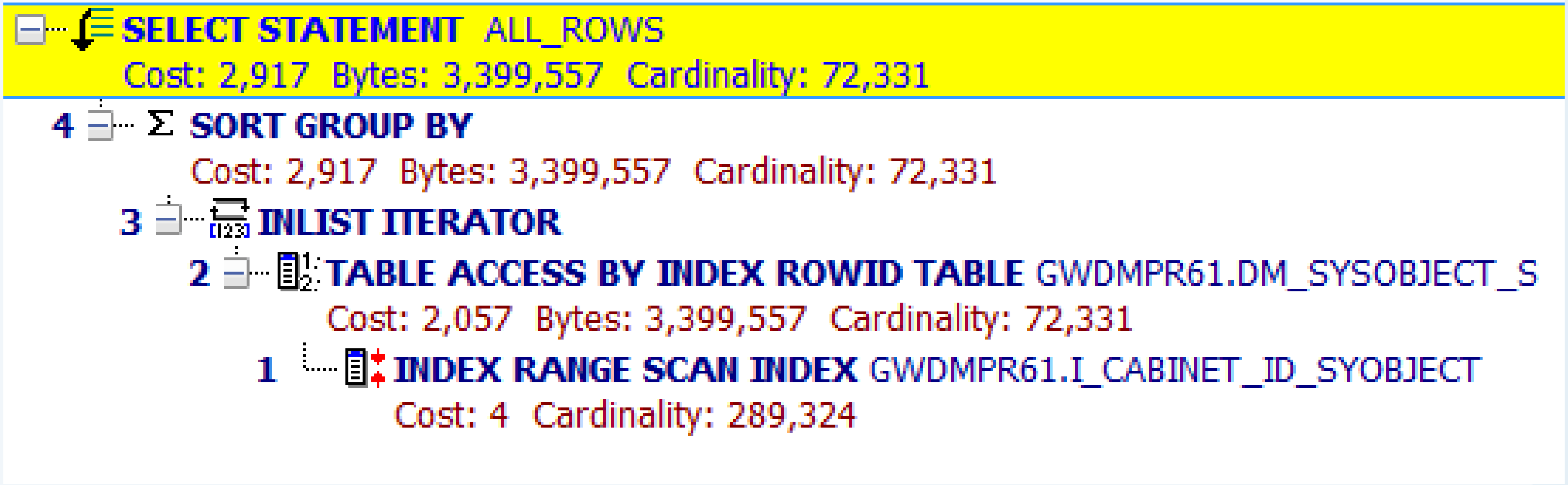
- Analyze / Tune



```
select all count(*)
, dm_sysobject.r_object_type "r_object_type"
, trunc(dm_sysobject.r_creation_date,:"SYS_B_00")
, sum(dm_sysobject.r_content_size/:"SYS_B_01"/:"SYS_B_02")
from GWDMPR61.dm_sysobject_sp dm_sysobject
   , GWDMPR61.dmr_content_sp  dmr_content
where (dm_sysobject.i_cabinet_id in (:"SYS_B_03", :"SYS_B_04"
,:"SYS_B_05", :"SYS_B_06", :"SYS_B_07", :"SYS_B_08"))
  and (dm_sysobject.i_has_folder = :"SYS_B_09"
  and dm_sysobject.i_is_deleted = :"SYS_B_10")
group by dm_sysobject.r_object_type
, trunc(dm_sysobject.r_creation_date, :"SYS_B_00")
order by :"SYS_B_12" asc
;
```

Case Study 1: Long Running Process Alert e-mail

- Test/Evaluate

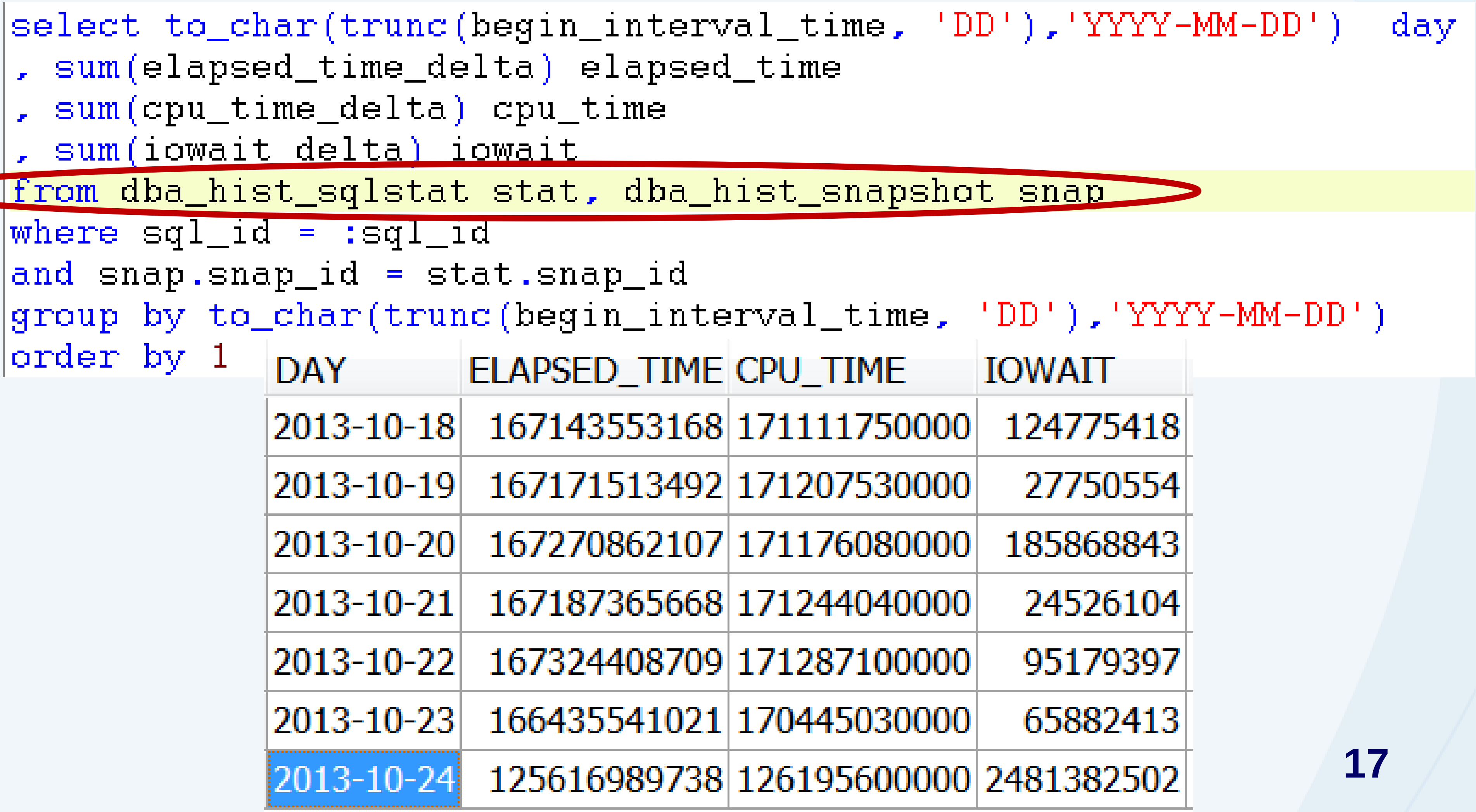


```
SELECT STATEMENT  ALL_ROWS
Cost: 2,917  Bytes: 3,399,557  Cardinality: 72,331
4   SORT GROUP BY
    Cost: 2,917  Bytes: 3,399,557  Cardinality: 72,331
  3   INLIST ITERATOR
    2   TABLE ACCESS BY INDEX ROWID TABLE GWDMPR61.DM_SYSOBJECT_S
        Cost: 2,057  Bytes: 3,399,557  Cardinality: 72,331
      1   INDEX RANGE SCAN INDEX GWDMPR61.I_CABINET_ID_SYOBJECT
          Cost: 4  Cardinality: 289,324
```

- Tuned version ran in 3 ½ minutes

Case Study 1: Long Running Process Alert e-mail

- Monitor over time



```
select to_char(trunc(begin_interval_time, 'DD'),'YYYY-MM-DD')  day
, sum(elapsed_time_delta) elapsed_time
, sum(cpu_time_delta) cpu_time
, sum(iowait_delta) iowait
from dba_hist_sqlstat stat, dba_hist_snapshot snap
where sql_id = :sql_id
and snap.snap_id = stat.snap_id
group by to_char(trunc(begin_interval_time, 'DD'),'YYYY-MM-DD')
order by 1
```

DAY	ELAPSED_TIME	CPU_TIME	IOWAIT
2013-10-18	167143553168	171111750000	124775418
2013-10-19	167171513492	171207530000	27750554
2013-10-20	167270862107	171176080000	185868843
2013-10-21	167187365668	171244040000	24526104
2013-10-22	167324408709	171287100000	95179397
2013-10-23	166435541021	170445030000	65882413
2013-10-24	125616989738	126195600000	2481382502

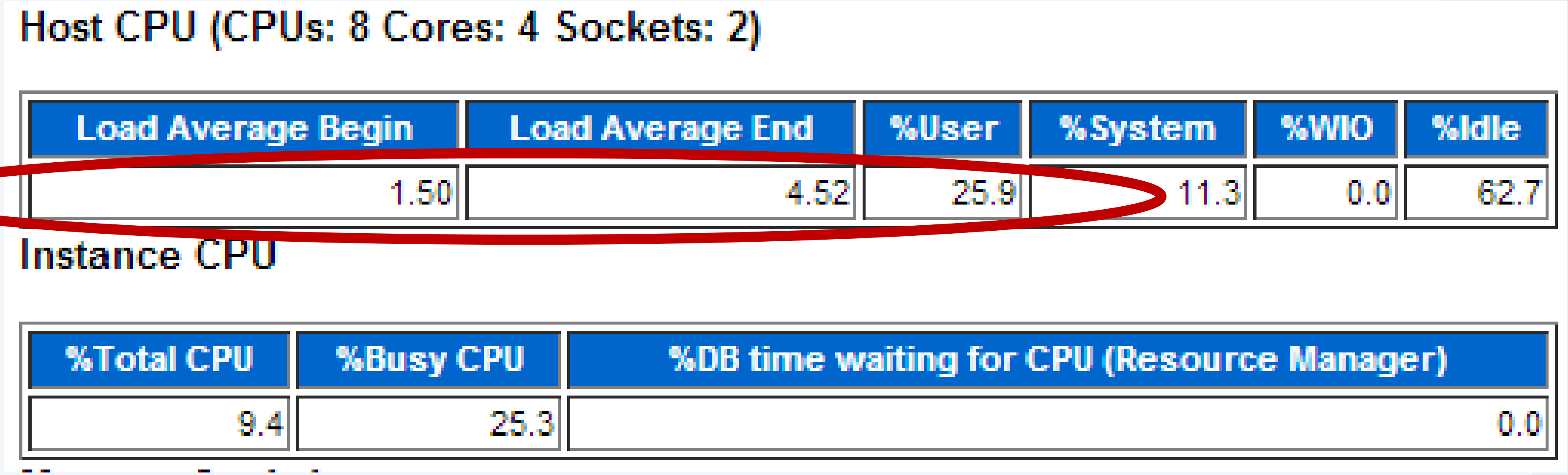
Taming the AWR Tsunami: Case Studies

Case Study 2: Client Application slow - times out

- Identify
  - Problem:
    - Client Application slow and times out,
    - usually with large dataset
  - Client error:
    - "Requested operation failed…”
    - “connection timed out".
  - Instance: USPRD775
  - Connect Username: DESIGNER
  - Time and Interval: “random” / not specific

Case Study 2: Client Application slow - times out

- Quantify – Load (from standard AWR Report)

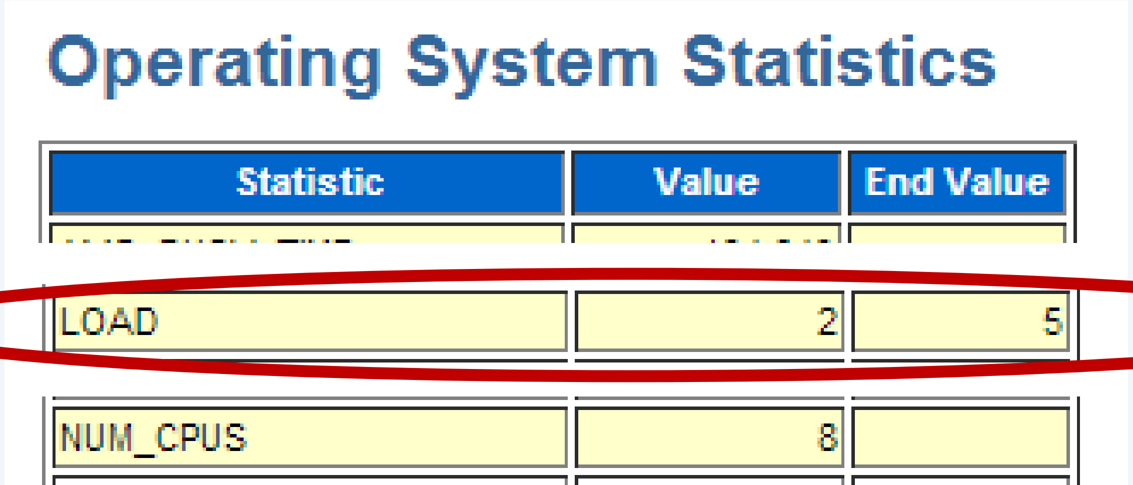


Host CPU (CPUs: 8 Cores: 4 Sockets: 2)

Load Average Begin	Load Average End	%User	%System	%WIO	%Idle
1.50	4.52	25.9	11.3	0.0	62.7

Instance CPU

%Total CPU	%Busy CPU	%DB time waiting for CPU (Resource Manager)
9.4	25.3	0.0



Operating System Statistics

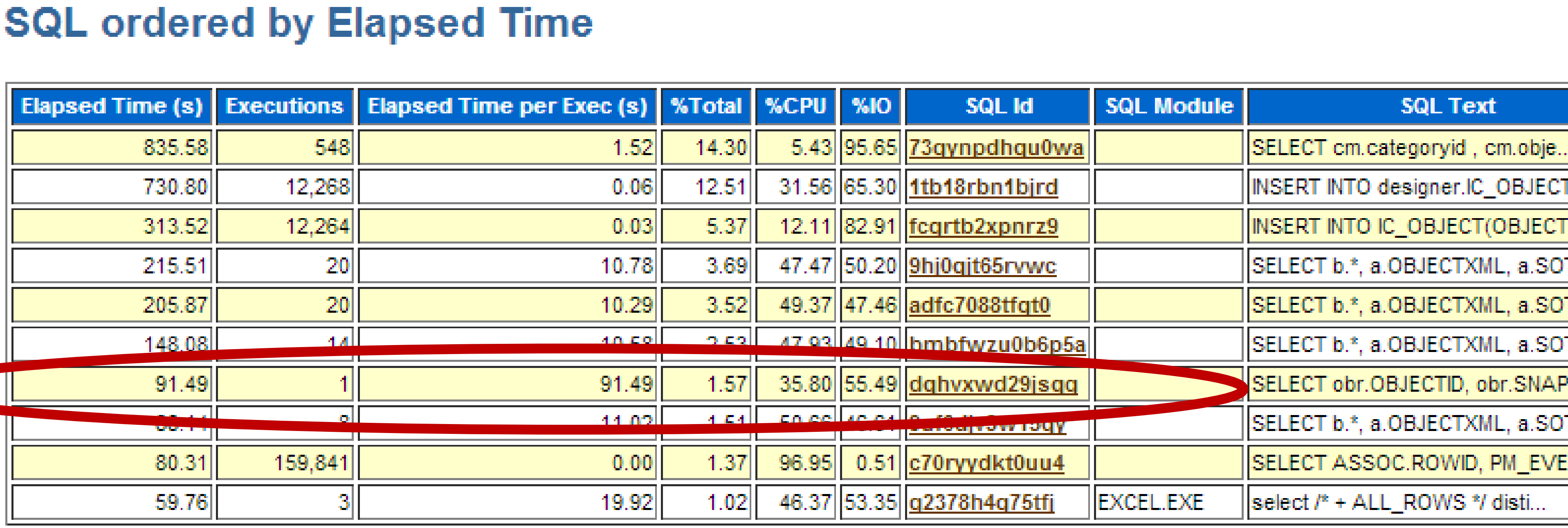
Statistic	Value	End Value
[illegible]	[illegible]	
LOAD	2	5
NUM_CPUS	8	

Load appears to be low

So let's look at what SQL is taking the longest time:

Case Study 2: Client Application slow - times out

- Quantify – Slow SQL (from standard AWR Report)



SQL ordered by Elapsed Time

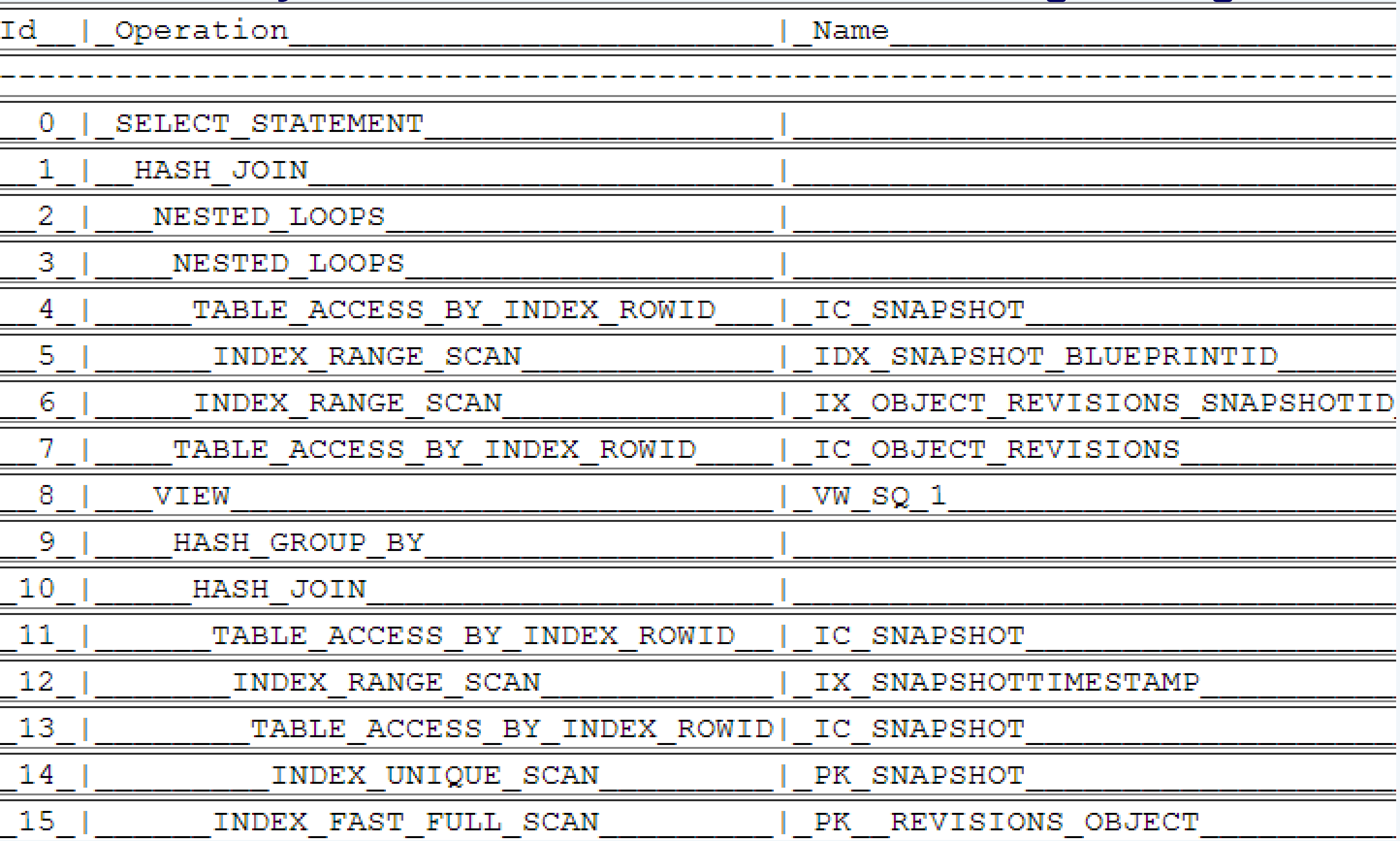
Elapsed Time (s)	Executions	Elapsed Time per Exec (s)	%Total	%CPU	%IO	SQL Id	SQL Module	SQL Text
835.58	548	1.52	14.30	5.43	95.65	73qynpdhqu0wa		SELECT cm.categoryid , cm.obje..
730.80	12,268	0.06	12.51	31.56	65.30	1tb18rbn1bjrd		INSERT INTO designer.IC_OBJECT
313.52	12,264	0.03	5.37	12.11	82.91	fcqrtb2xpnrz9		INSERT INTO IC_OBJECT(OBJECT
215.51	20	10.78	3.69	47.47	50.20	9hj0qjt65rvwc		SELECT b.*, a.OBJECTXML, a.SO
205.87	20	10.29	3.52	49.37	47.46	adfc7088tfqt0		SELECT b.*, a.OBJECTXML, a.SO
148.08	14	10.58	2.53	47.93	49.10	bmbfwzu0b6p5a		SELECT b.*, a.OBJECTXML, a.SO
91.49	1	91.49	1.57	35.80	55.49	dqhvxwd29jsqg		SELECT obr.OBJECTID, obr.SNAP
[illegible]	8	11.02	1.51	50.66	46.81	[illegible]		SELECT b.*, a.OBJECTXML, a.SO
80.31	159,841	0.00	1.37	96.95	0.51	c70ryydkt0uu4		SELECT ASSOC.ROWID, PM_EVE
59.76	3	19.92	1.02	46.37	53.35	g2378h4q75tfj	EXCEL.EXE	select /* + ALL_ROWS */ disti...

- Sql_id: dqhvxwd29jsqg
- Time per execution: 92 sec

Case Study 2: Client Application slow - times out

- Analyze

AWR - sqlid profile.sql



```
Id  | Operation                            | Name
-------------------------------------------------------------------------
  0 | SELECT STATEMENT                     |
  1 |  HASH JOIN                           |
  2 |   NESTED LOOPS                       |
  3 |    NESTED LOOPS                      |
  4 |     TABLE ACCESS BY INDEX ROWID      | IC_SNAPSHOT
  5 |      INDEX RANGE SCAN                | IDX_SNAPSHOT_BLUEPRINTID
  6 |     INDEX RANGE SCAN                 | IX_OBJECT_REVISIONS_SNAPSHOTID
  7 |    TABLE ACCESS BY INDEX ROWID       | IC_OBJECT_REVISIONS
  8 |   VIEW                               | VW_SQ_1
  9 |    HASH GROUP BY                     |
 10 |     HASH JOIN                        |
 11 |      TABLE ACCESS BY INDEX ROWID     | IC_SNAPSHOT
 12 |       INDEX RANGE SCAN               | IX_SNAPSHOTTIMESTAMP
 13 |        TABLE ACCESS BY INDEX ROWID   | IC_SNAPSHOT
 14 |         INDEX UNIQUE SCAN            | PK_SNAPSHOT
 15 |      INDEX FAST FULL SCAN            | PK__REVISIONS_OBJECT
```

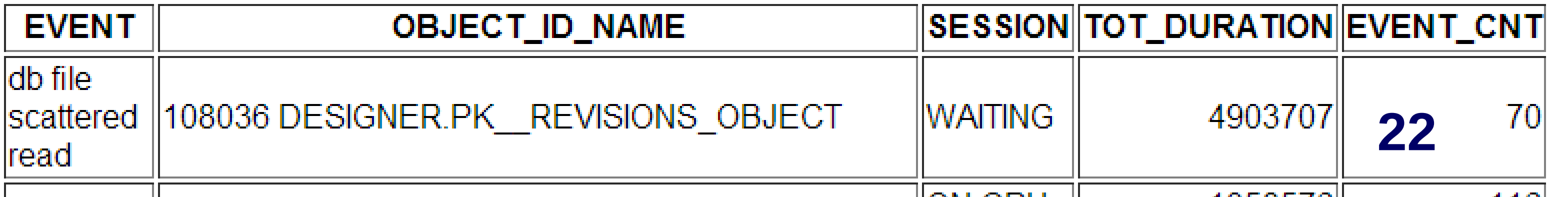
Execution Plan:

Full scan of index on 10 Gb table

Aggregate Events:

Most waits on "db file scattered read"

Same index



EVENT	OBJECT_ID_NAME	SESSION	TOT_DURATION	EVENT_CNT
db file scattered read	108036 DESIGNER.PK__REVISIONS_OBJECT	WAITING	4903707	70

Case Study 2: Client Application slow - times out

- Analyze
  - Load is low; plenty of capacity on machine/DB
  - Vendor correlated Slow Sql with the application function
- Tune
  - on vendor to-do list for next version
- Test/Evaluate
  - N/A
- Monitor over time
  - Monitor via **`DBA_HIST_SQLSTAT`**

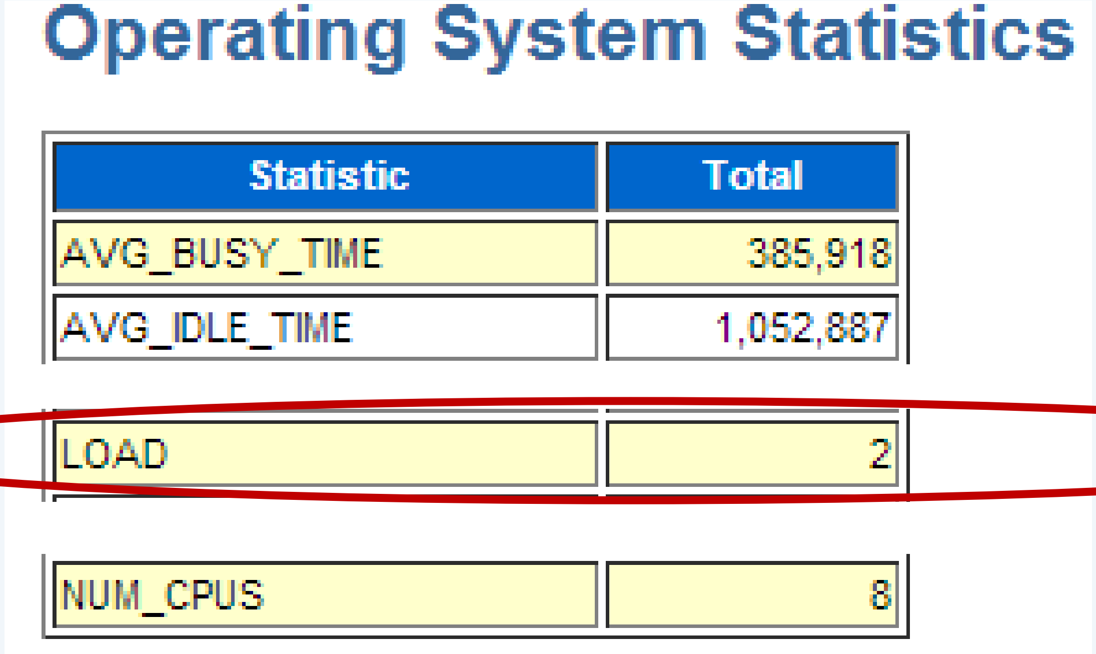
Taming the AWR Tsunami: Case Studies

Case Study 3: Batch Jobs for Sales Force App.

- Identify
  - Problem: Daily job takes several hours to execute; delays other job processing
  - Interval: runs between 1:30 AM and 5:30 AM (approximately; kicked off by prior job)
  - Instance: USPRD661
  - Connect Username: CRM_CH
  - Code/Module: PrmMy0014Calc.exe;
    runs from 4 separate sessions simultaneously
  - Table: Bpldata and others (user does not have SQL)
  - Basic flow: fetches the data using select queries;
    performs a calculation;
    updates the tables with the calculated values.

Case Study 3: Batch Jobs for Sales Force App.

- Quantify – Load (from standard AWR Report)



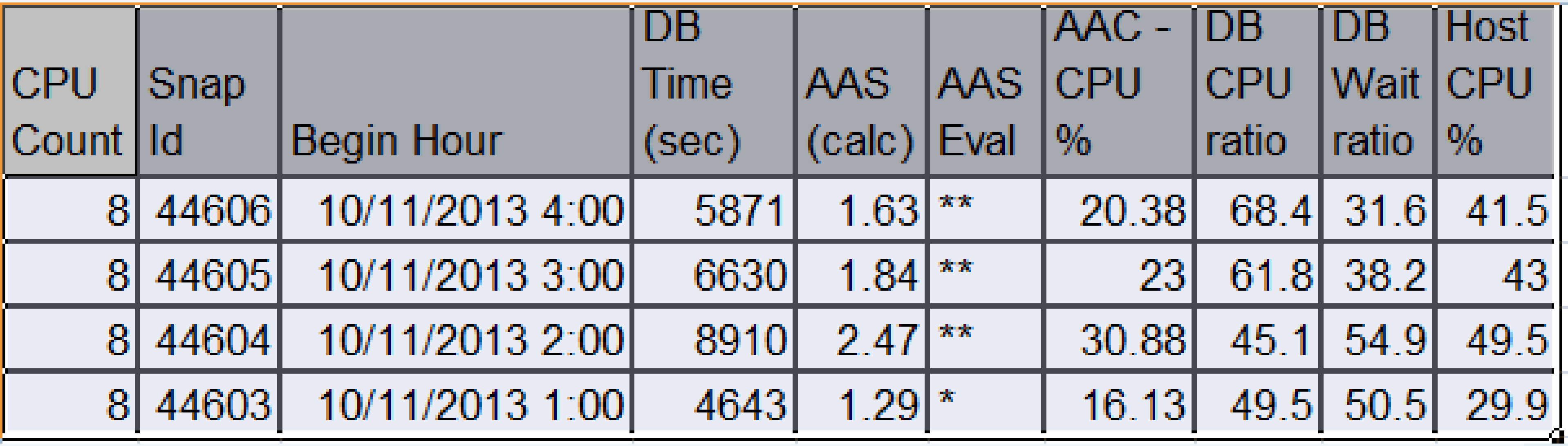
Operating System Statistics

Statistic	Total
AVG_BUSY_TIME	385,918
AVG_IDLE_TIME	1,052,887
LOAD	2
NUM_CPUS	8

Load seems to be ok / low

Case Study 3: Batch Jobs for Sales Force App.

- Quantify – Load using `AAS – Average Active Sessions.sql`

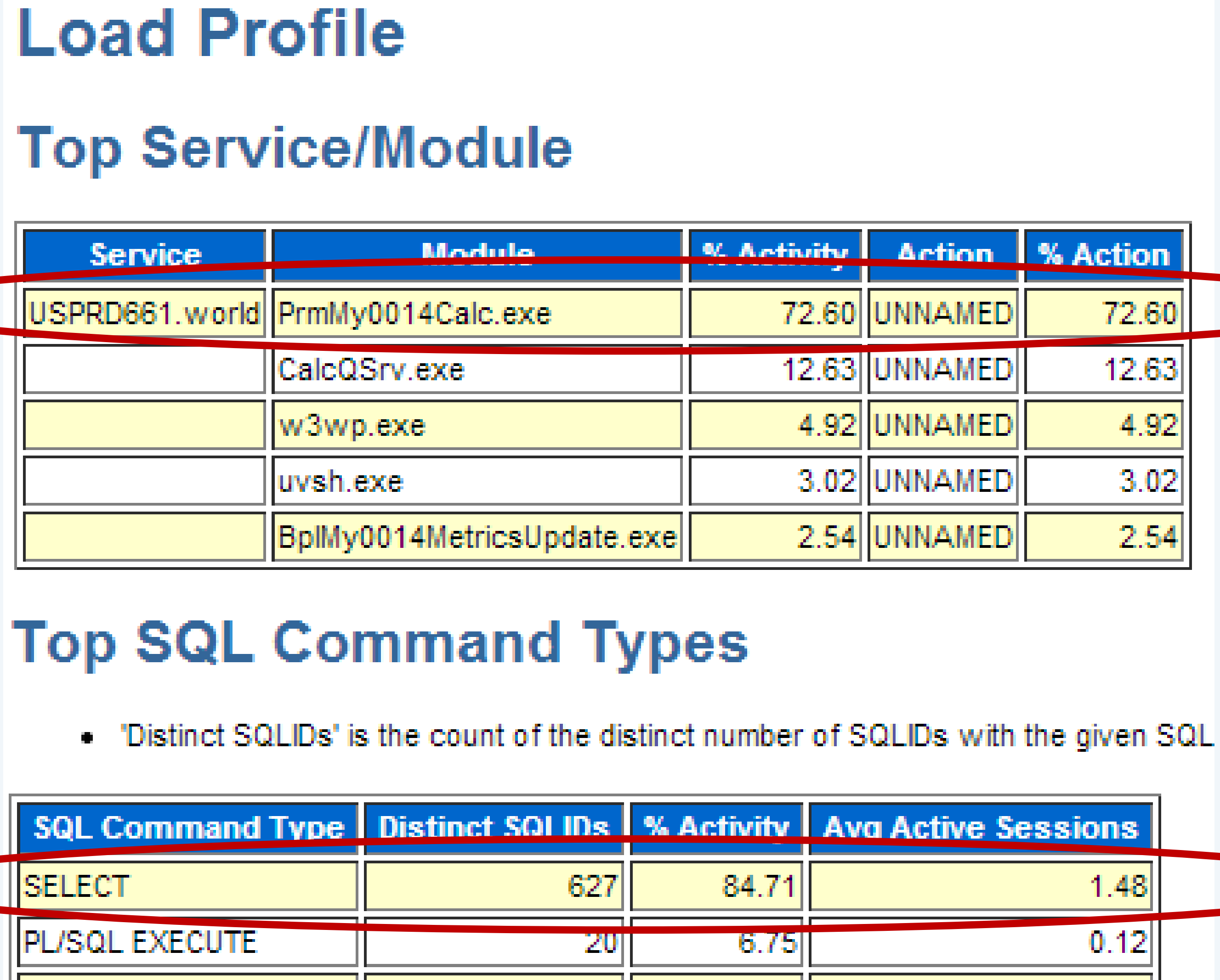


CPU Count	Snap Id	Begin Hour	DB Time (sec)	AAS (calc)	AAS Eval	AAC - CPU %	DB CPU ratio	DB Wait ratio	Host CPU %
8	44606	10/11/2013 4:00	5871	1.63	**	20.38	68.4	31.6	41.5
8	44605	10/11/2013 3:00	6630	1.84	**	23	61.8	38.2	43
8	44604	10/11/2013 2:00	8910	2.47	**	30.88	45.1	54.9	49.5
8	44603	10/11/2013 1:00	4643	1.29	*	16.13	49.5	50.5	29.9

This confirms that the load is not a problem.
So now I want to know some more details about the load to see if that can help me focus in on the root cause.

Case Study 3: Batch Jobs for Sales Force App.

- Quantify – Load details (from standard ASH Report)



Load Profile

Top Service/Module

Service	Module	% Activity	Action	% Action
USPRD661.world	PrmMy0014Calc.exe	72.60	UNNAMED	72.60
	CalcQSrv.exe	12.63	UNNAMED	12.63
	w3wp.exe	4.92	UNNAMED	4.92
	uvsh.exe	3.02	UNNAMED	3.02
	BplMy0014MetricsUpdate.exe	2.54	UNNAMED	2.54

73% db activity this module

Top SQL Command Types

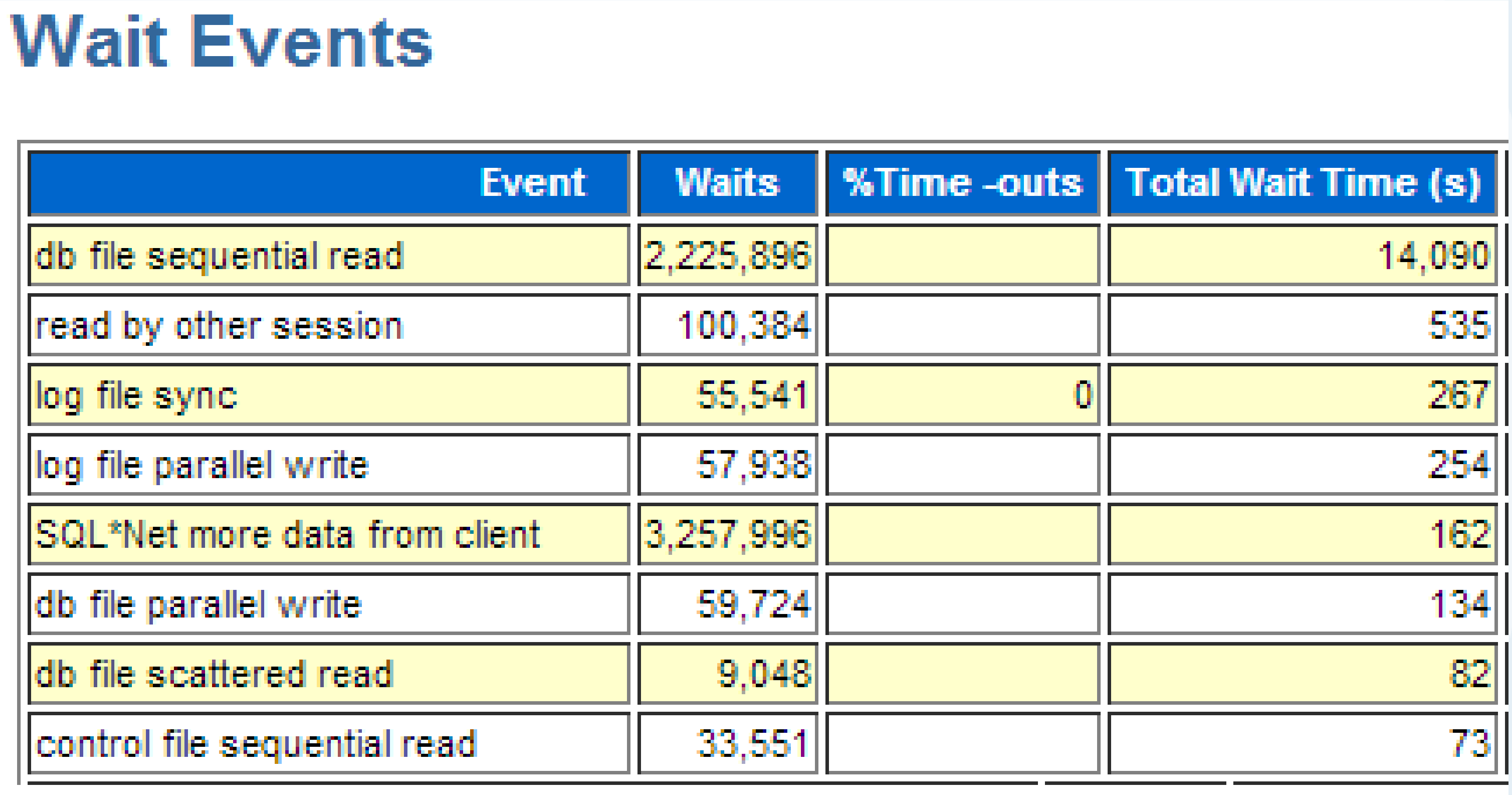
- 'Distinct SQLIDs' is the count of the distinct number of SQLIDs with the given SQL

SQL Command Type	Distinct SQLIDs	% Activity	Avg Active Sessions
SELECT	627	84.71	1.48
PL/SQL EXECUTE	20	6.75	0.12

85% of db activity is select

Case Study 3: Batch Jobs for Sales Force App.

- Quantify – Waits (from standard AWR Report)

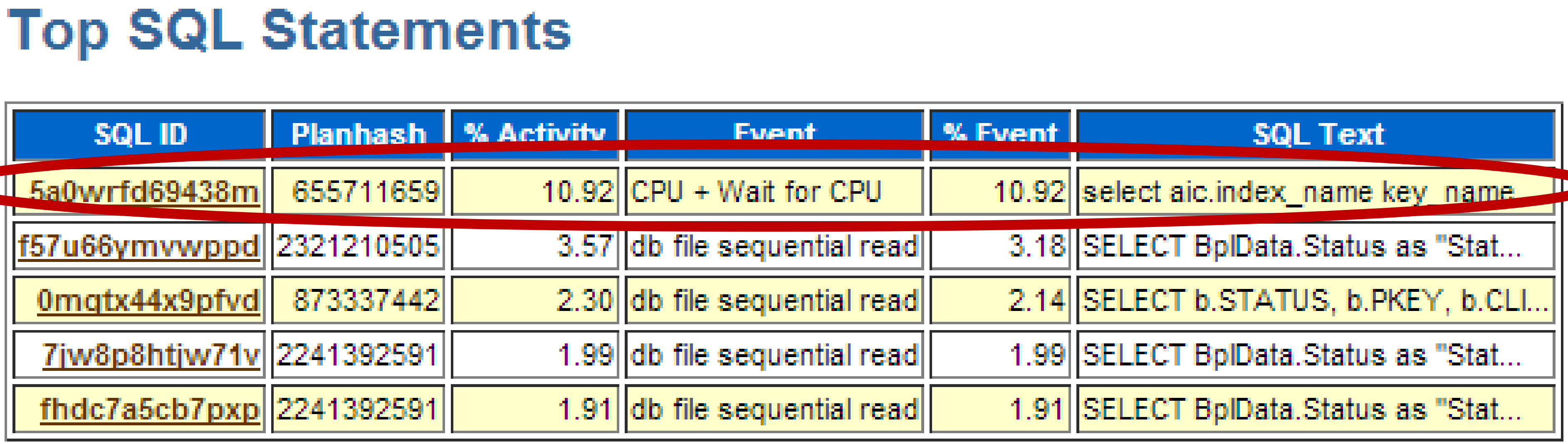


Wait Events

Event	Waits	%Time -outs	Total Wait Time (s)
db file sequential read	2,225,896		14,090
read by other session	100,384		535
log file sync	55,541	0	267
log file parallel write	57,938		254
SQL*Net more data from client	3,257,996		162
db file parallel write	59,724		134
db file scattered read	9,048		82
control file sequential read	33,551		73

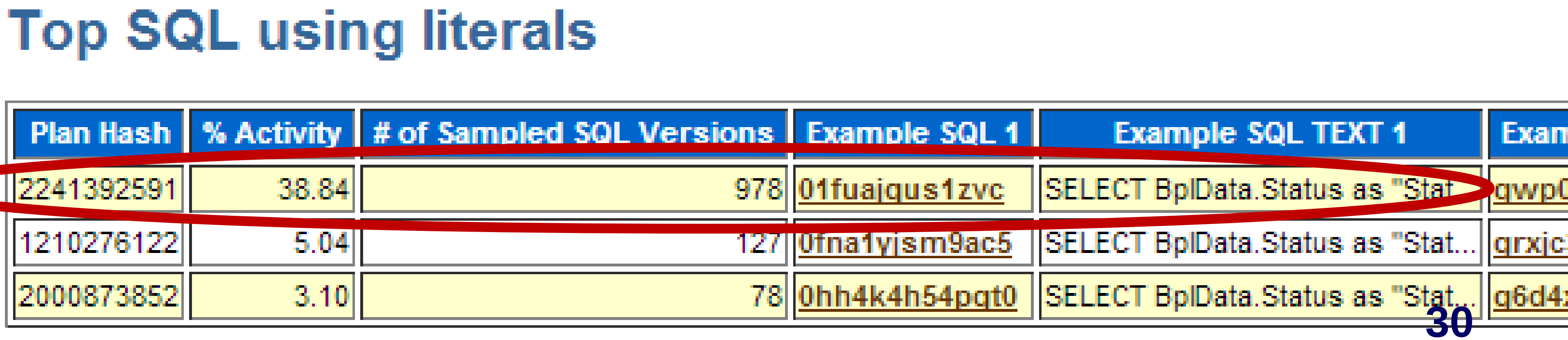
Case Study 3: Batch Jobs for Sales Force App.

- Quantify – SQL (from standard ASH report)



Top SQL Statements

SQL ID	Planhash	% Activity	Event	% Event	SQL Text
5a0wrfd69438m	655711659	10.92	CPU + Wait for CPU	10.92	select aic.index_name key_name
f57u66ymvwppd	2321210505	3.57	db file sequential read	3.18	SELECT BplData.Status as "Stat...
0mqtx44x9pfvd	873337442	2.30	db file sequential read	2.14	SELECT b.STATUS, b.PKEY, b.CLI...
7jw8p8htjw71v	2241392591	1.99	db file sequential read	1.99	SELECT BplData.Status as "Stat...
fhdc7a5cb7pxp	2241392591	1.91	db file sequential read	1.91	SELECT BplData.Status as "Stat...

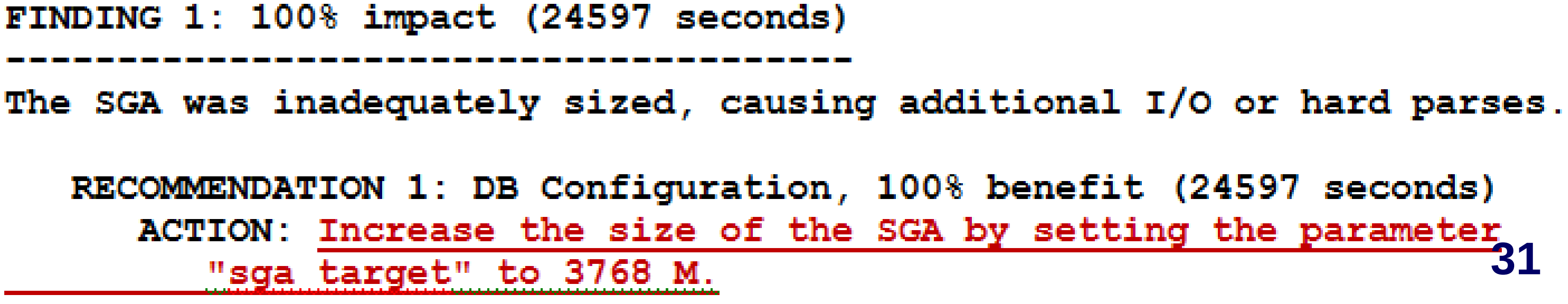


Top SQL using literals

Plan Hash	% Activity	# of Sampled SQL Versions	Example SQL 1	Example SQL TEXT 1	Exam
2241392591	38.84	978	01fuajqus1zvc	SELECT BplData.Status as "Stat	qwp0
1210276122	5.04	127	0fna1yjsm9ac5	SELECT BplData.Status as "Stat...	qrxjc
2000873852	3.10	78	0hh4k4h54pqt0	SELECT BplData.Status as "Stat...	q6d4

Case Study 3: Batch Jobs for Sales Force App.

- Analyze
  - Load is low (never more that 50%)
  - Minor log file related waits
  - App does not make use of bind variables
  - Big issue is I/O on 14 gig table BPLDATA
  - Vendor indicated fragmentation and row chaining
    - High row chaining reads confirmed by looking in **`dba_hist_sysstat.stat_name = 'table fetch continued row’`**
  - Standard ADDM Report: DBA_HIST_SGA_TARGET_ADVICE

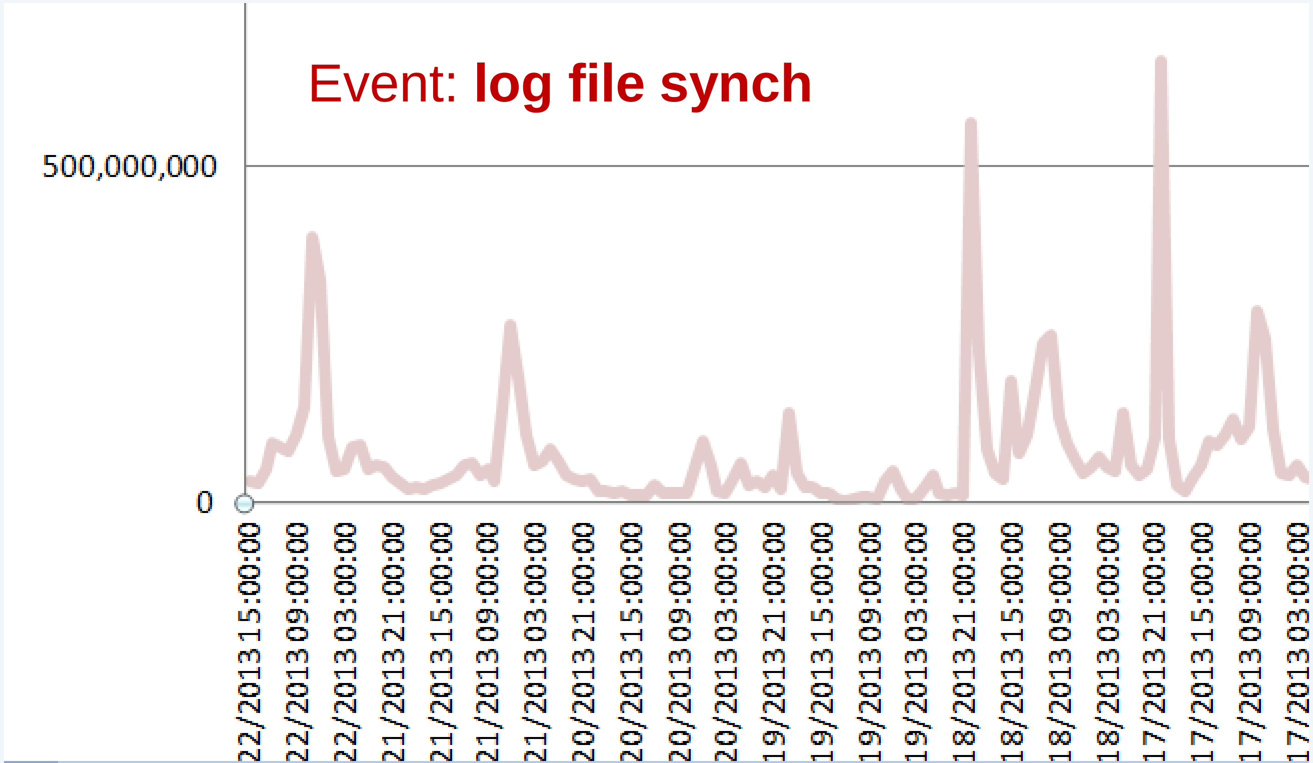


```
FINDING 1: 100% impact (24597 seconds)
--------------------------------------
The SGA was inadequately sized, causing additional I/O or hard parses.

   RECOMMENDATION 1: DB Configuration, 100% benefit (24597 seconds)
      ACTION: Increase the size of the SGA by setting the parameter
         "sga_target" to 3768 M.
```

Case Study 3: Batch Jobs for Sales Force App.

- Tune
  - Increase log file size
  - Re-organize or cluster BPLDATA
- Test/Evaluate (in progress w/ application team)
- Monitor over time – `DBA_HIST_SYSTEM_EVENT`



Method Recap

1] Identify Problem

2] Quantify using AWR

3] Analysis

4] Tune

5] Test/Evaluate

6] Monitor

Tools Recap

- Quantify Load:
  - Operating System Stats of Standard AWR
  - Average Active Sessions script
  - ASH report
- Quantify Wait events - standard AWR Report
- Quantify SQL
  - ASH and AWR Report
  - custom script querying DBA_HIST_SQLSTAT
- Custom Quantifying of row chaining reads from DBA_HIST_SYSSTAT ‘table fetch continued row’ statistic
- Analysis - ADDM report
- Monitor over time with some custom scripts

Exploring AWR Data

Taming the AWR Tsunami

Exploring AWR Data: Metrics vs Statistics

Metric

- Value in that period
- e.g.: `DBA_HIST_` **`SYSMETRIC_SUMMARY`**

Statistic

- Cumulative value
- e.g.: `DBA_HIST_` **`SYSTEM_EVENT`** `&` **`SYSSTAT`**
- Use analytic functions:
  - LAG
    - compute deltas values
  - ROW_NUMBER
    - e.g. Top-n Events script
  - RATIO_TO_REPORT
    - % of total

Exploring AWR Data: Comparison of Views

V$ Views	**StatsPack Views**	**AWR Views**
V$_SYNONYM_NAME	**STATS$_SYNONYM_NAME**	**DBA_HIST_SYNONYM_NAME**
V$SESSION		DBA_HIST_ACTIVE_SESS_HISTORY
V$SESSION_EVENT		DBA_HIST_ACTIVE_SESS_HISTORY
V$DATAFILE		DBA_HIST_DATAFILE
V$DB_CACHE_ADVICE	STATS$DB_CACHE_ADVICE	DBA_HIST_DB_CACHE_ADVICE
V$FILESTAT	STATS$FILESTATXS	DBA_HIST_FILESTATXS
V$LATCH_PARENT	STATS$LATCH_PARENT	DBA_HIST_LATCH_PARENT
V$LIBRARYCACHE	STATS$LIBRARYCACHE	DBA_HIST_LIBRARYCACHE
V$LOG		DBA_HIST_LOG
V$OSSTAT	STATS$OSSTAT	DBA_HIST_OSSTAT
V$PARAMETER	STATS$PARAMETER	DBA_HIST_PARAMETER
V$SGA	STATS$SGA	DBA_HIST_SGA
V$SQLSTATS	**STATS$SQL_SUMMARY**	**DBA_HIST_SQLSTAT**
V$SQLTEXT	STATS$SQLTEXT	DBA_HIST_SQLTEXT
V$SQL_PLAN	STATS$SQL_PLAN	DBA_HIST_SQL_PLAN
V$SYSMETRIC_SUMMARY		DBA_HIST_SYSMETRIC_SUMMARY
V$SYSSTAT	STATS$SYSSTAT	DBA_HIST_SYSSTAT
V$SYS_TIME_MODEL	STATS$SYS_TIME_MODEL	DBA_HIST_SYS_TIME_MODEL

- Jumpstart AWR data exploration:
  - Leverage wealth of knowledge of v$ views

Exploring AWR Data: Key DBA_HIST Views

- DBA_HIST_SNAPSHOT
  - Maps a snap_id to the Date/Time

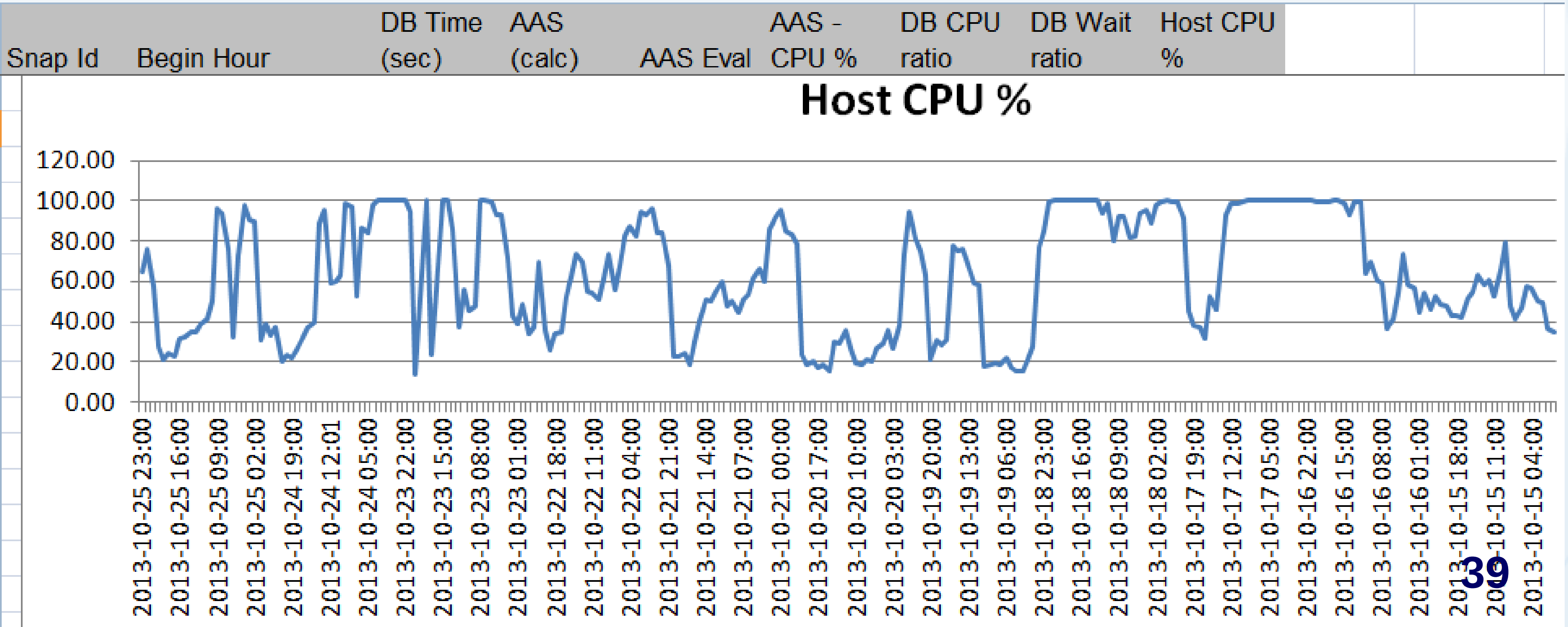
- DBA_HIST_SQLSTAT
  - Statistics on all SQL statements picked up by the DB
  - Use this to find the Expensive SQL

- DBA_HIST_ACTIVE_SESS_HISTORY
  - All the sessions and what they were running
  - Samples rolled up every 10 seconds

- DBA_HIST_SYSMETRIC_SUMMARY (135 10g -158 11g)
  - Various metrics (response time, I/O, …)

Tracking Measurements Over Time

- **AAS – Average Active Sessions.sql**
  - Uses: dba_hist_sysmetric_summary.metric_name:
    - 'Average Active Sessions'
    - 'Host CPU Utilization (%) '
    - 'Database CPU Time Ratio'
    - 'Database Wait Time Ratio'



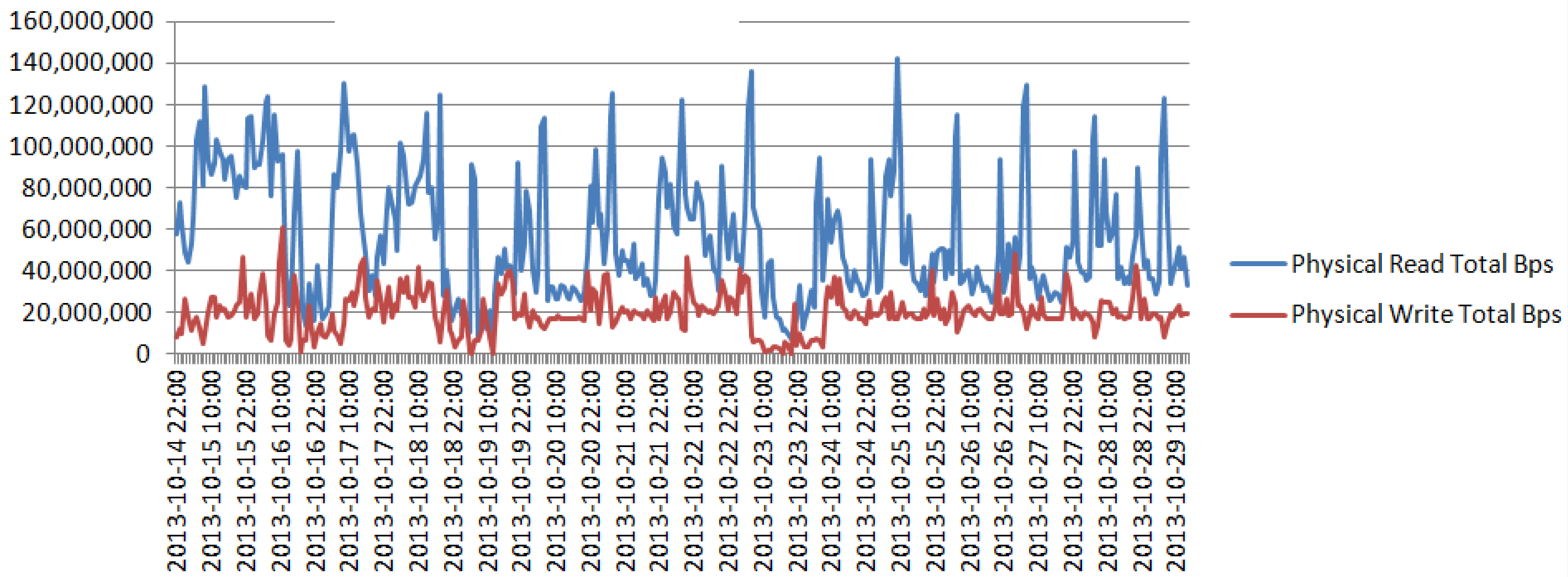
Snap Id	Begin Hour	DB Time (sec)	AAS (calc)	AAS Eval	AAS - CPU %	DB CPU ratio	DB Wait ratio	Host CPU %

Tracking Measurements Over Time

- Load metrics - I/O Workload <query in sql script>

`AWR - dba_hist_sysmetric_summary - various metrics.sql`

  - Physical Read Total Bytes Per Sec
  - Physical Write Total Bytes Per Sec

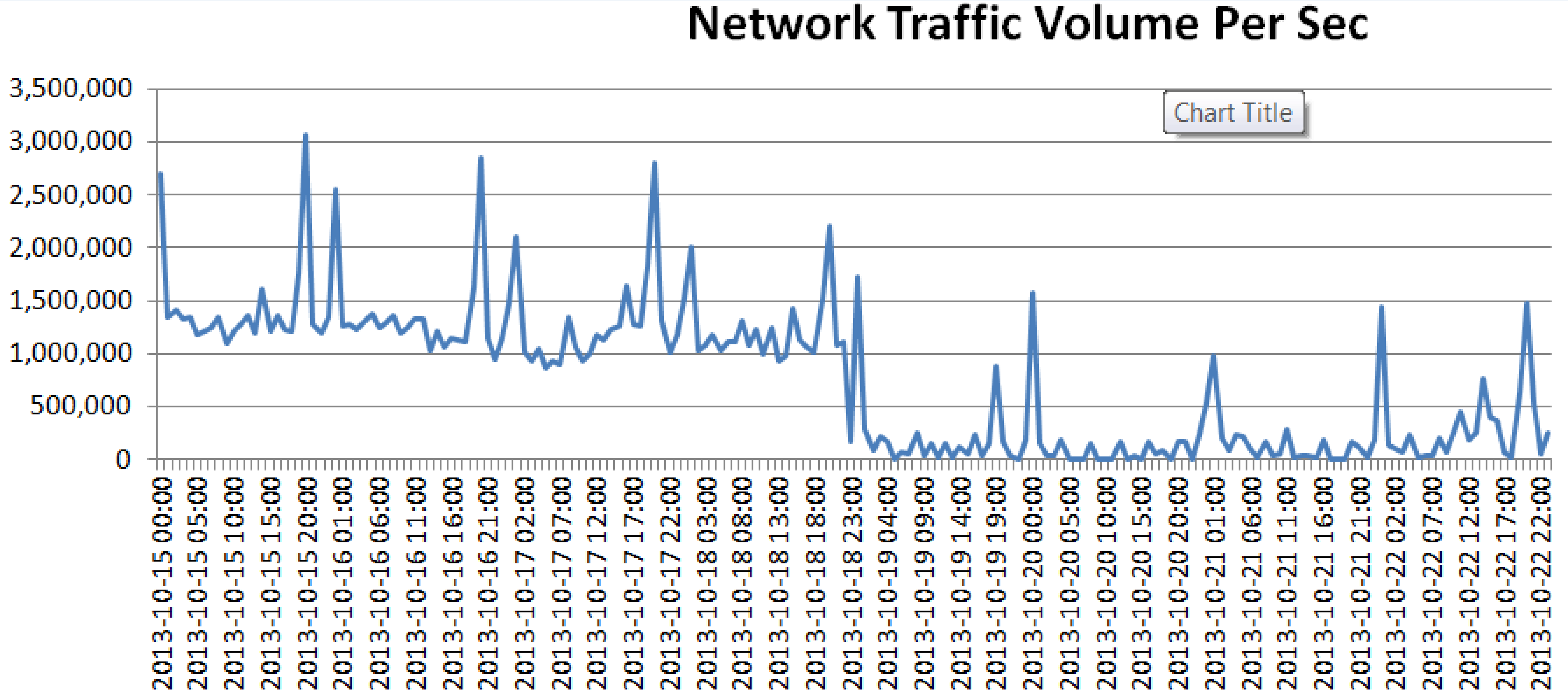


Tracking Measurements Over Time

- Load metrics - Workload volume/throughput <query in script>

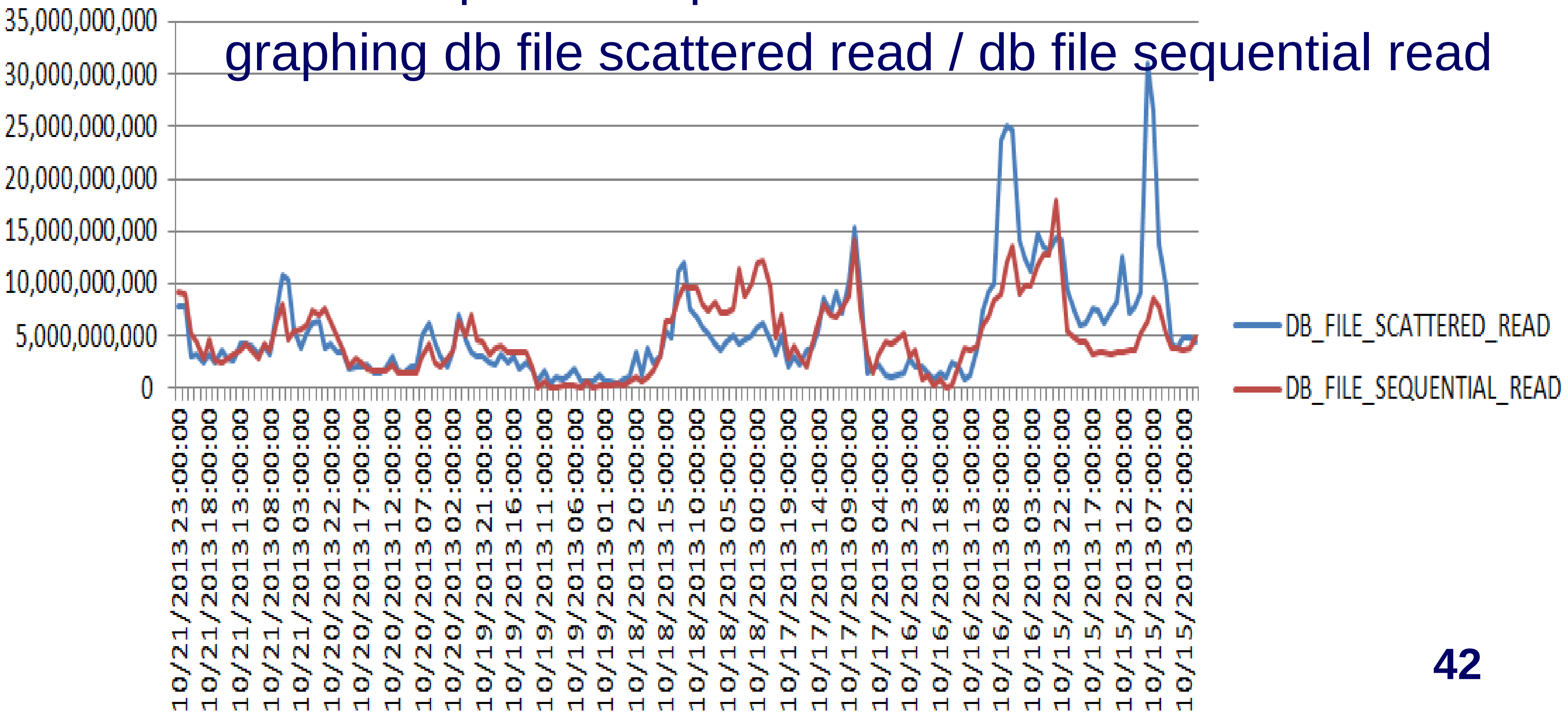
`AWR - dba_hist_sysmetric_summary - various metrics.sql`

  – Network Traffic Volume Per Sec



Tracking Measurements Over Time

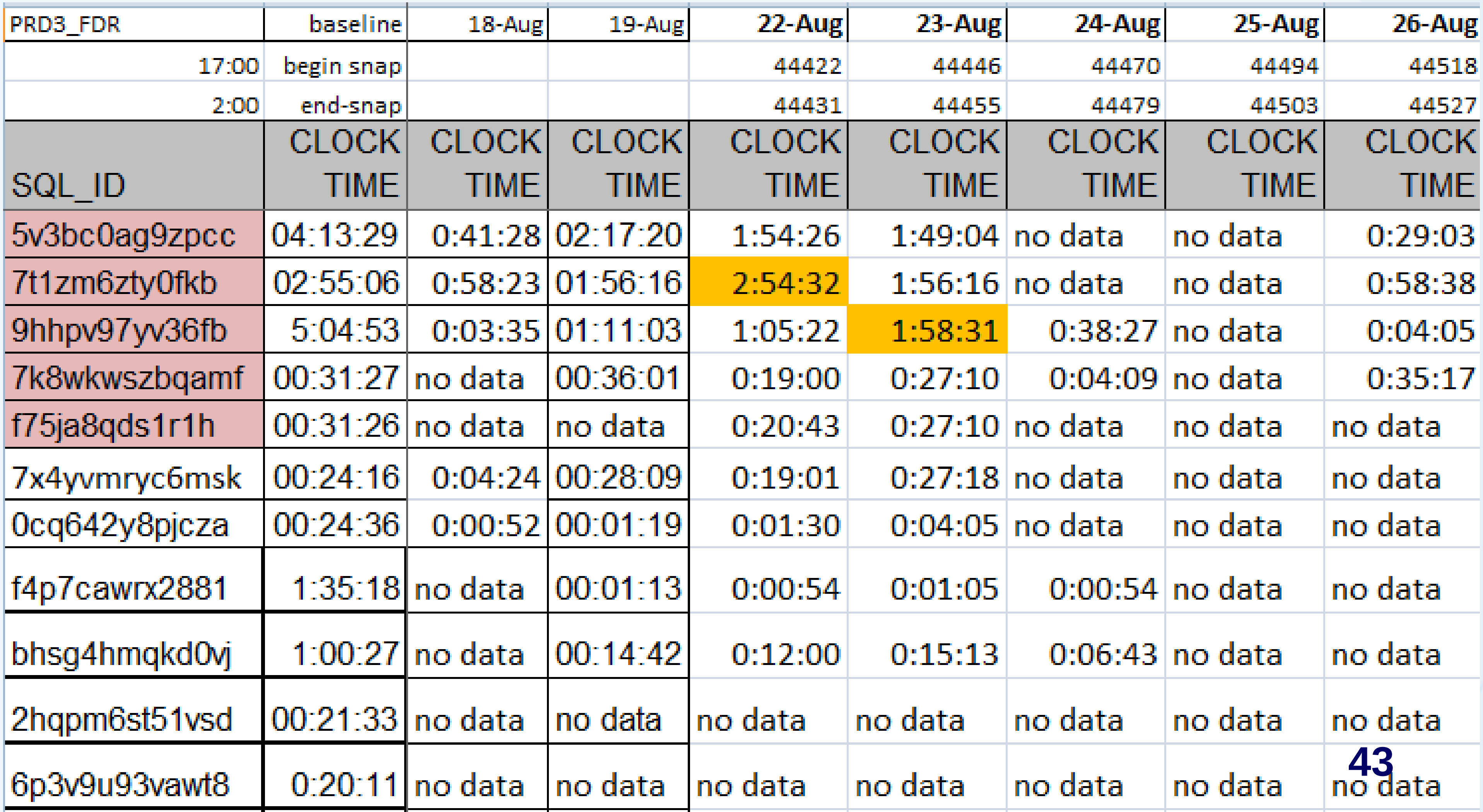
- **`AWR - Top-n waits by snap_id.SQL`**
- Top-n events – pivoted on Event Name <instructions in code>
  1. Generate code fragments
  2. Edit into “pivot” template



graphing db file scattered read / db file sequential read

Tracking Measurements Over Time

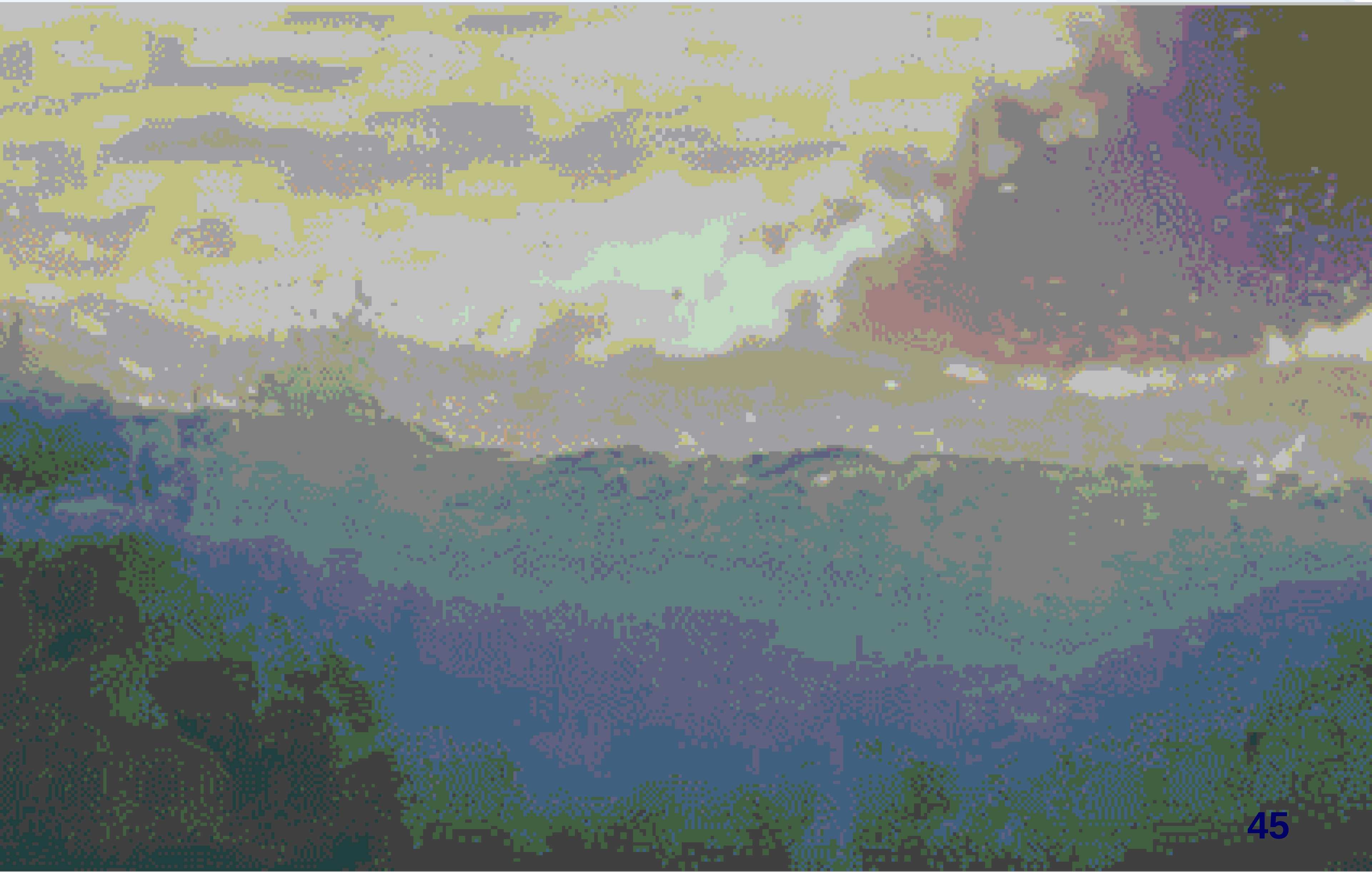
- SQL statistics - **AWR - Expensive SQL for a snap_id.sql**



PRD3_FDR	baseline	18-Aug	19-Aug	22-Aug	23-Aug	24-Aug	25-Aug	26-Aug
17:00	begin snap			44422	44446	44470	44494	44518
2:00	end-snap			44431	44455	44479	44503	44527
SQL_ID	CLOCK TIME	CLOCK TIME	CLOCK TIME	CLOCK TIME	CLOCK TIME	CLOCK TIME	CLOCK TIME	CLOCK TIME
5v3bc0ag9zpcc	04:13:29	0:41:28	02:17:20	1:54:26	1:49:04	no data	no data	0:29:03
7t1zm6zty0fkb	02:55:06	0:58:23	01:56:16	2:54:32	1:56:16	no data	no data	0:58:38
9hhpv97yv36fb	5:04:53	0:03:35	01:11:03	1:05:22	1:58:31	0:38:27	no data	0:04:05
7k8wkwszbqamf	00:31:27	no data	00:36:01	0:19:00	0:27:10	0:04:09	no data	0:35:17
f75ja8qds1r1h	00:31:26	no data	no data	0:20:43	0:27:10	no data	no data	no data
7x4yvmryc6msk	00:24:16	0:04:24	00:28:09	0:19:01	0:27:18	no data	no data	no data
0cq642y8pjcza	00:24:36	0:00:52	00:01:19	0:01:30	0:04:05	no data	no data	no data
f4p7cawrx2881	1:35:18	no data	00:01:13	0:00:54	0:01:05	0:00:54	no data	no data
bhsg4hmqkd0vj	1:00:27	no data	00:14:42	0:12:00	0:15:13	0:06:43	no data	no data
2hqpm6st51vsd	00:21:33	no data	no data	no data	no data	no data	no data	no data
6p3v9u93vawt8	0:20:11	no data	no data	no data	no data	no data	no data	no data

Tracking Measurements Over Time

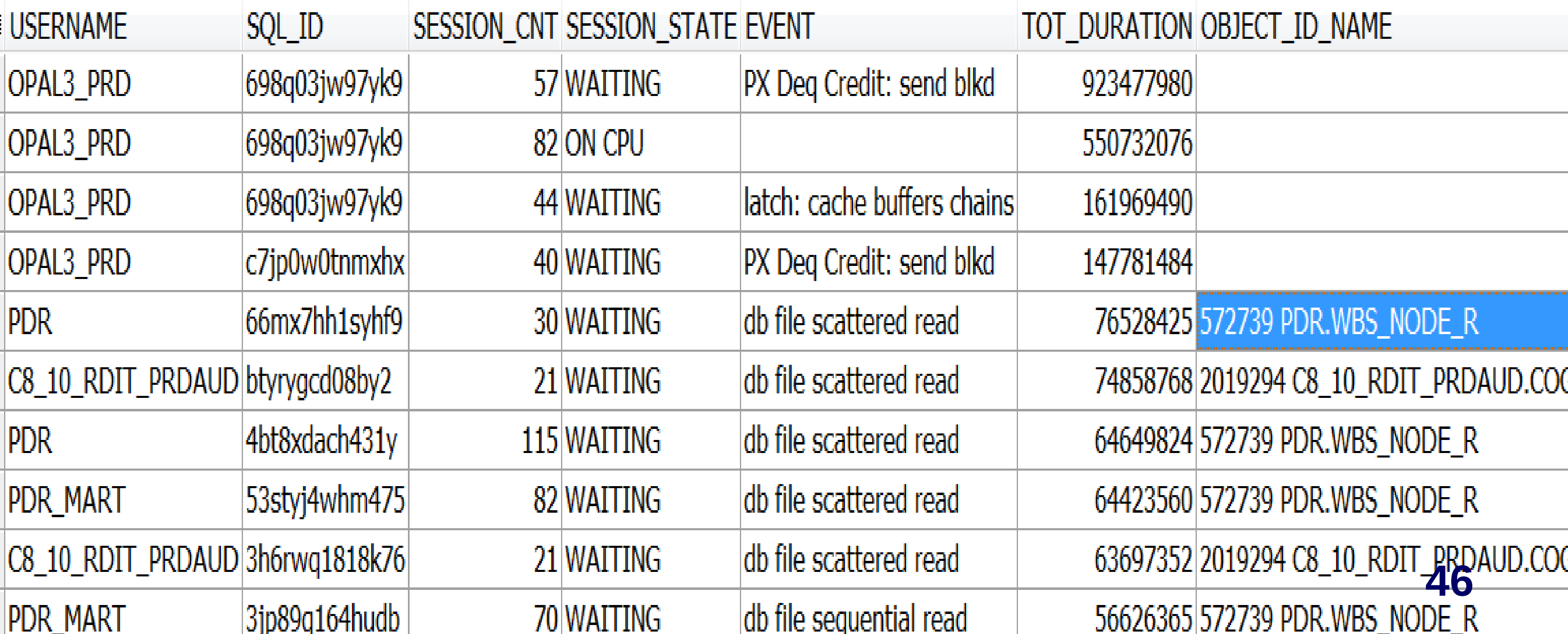
Fun stuff



All the SQL associated with an event

- **`ASH – SQL for SnapId – event.sql`**
- Usage Scenario:
  Have the event name from Top-n waits, but would like to know what SQL is causing that and to what extent

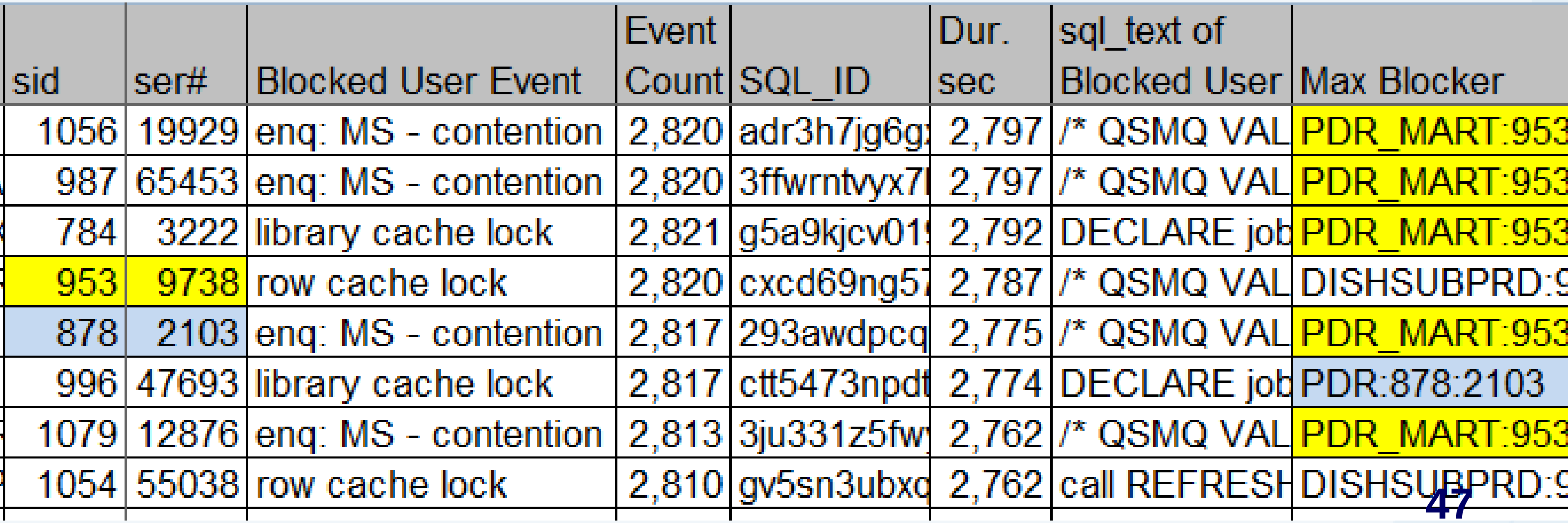
Example execution with: **'direct path read temp'**



USERNAME	SQL_ID	SESSION_CNT	SESSION_STATE	EVENT	TOT_DURATION	OBJECT_ID_NAME
OPAL3_PRD	698q03jw97yk9	57	WAITING	PX Deq Credit: send blkd	923477980	
OPAL3_PRD	698q03jw97yk9	82	ON CPU		550732076	
OPAL3_PRD	698q03jw97yk9	44	WAITING	latch: cache buffers chains	161969490	
OPAL3_PRD	c7jp0w0tnmxhx	40	WAITING	PX Deq Credit: send blkd	147781484	
PDR	66mx7hh1syhf9	30	WAITING	db file scattered read	76528425	572739 PDR.WBS_NODE_R
C8_10_RDIT_PRDAUD	btyrygcd08by2	21	WAITING	db file scattered read	74858768	2019294 C8_10_RDIT_PRDAUD.COC
PDR	4bt8xdach431y	115	WAITING	db file scattered read	64649824	572739 PDR.WBS_NODE_R
PDR_MART	53styj4whm475	82	WAITING	db file scattered read	64423560	572739 PDR.WBS_NODE_R
C8_10_RDIT_PRDAUD	3h6rwq1818k76	21	WAITING	db file scattered read	63697352	2019294 C8_10_RDIT_PRDAUD.COC
PDR_MART	3jp89q164hudb	70	WAITING	db file sequential read	56626365	572739 PDR.WBS_NODE_R

Sessions that were Blocked by other Sessions

- **ASH - Blocked Sessions.sql**
- Usage Scenario:
  App slow, but SQL seemed tuned.
  Found blocker and related info:
  Duration; sql_text of Blocked User;
  Blocker sid/event/sql_id; sql_text …



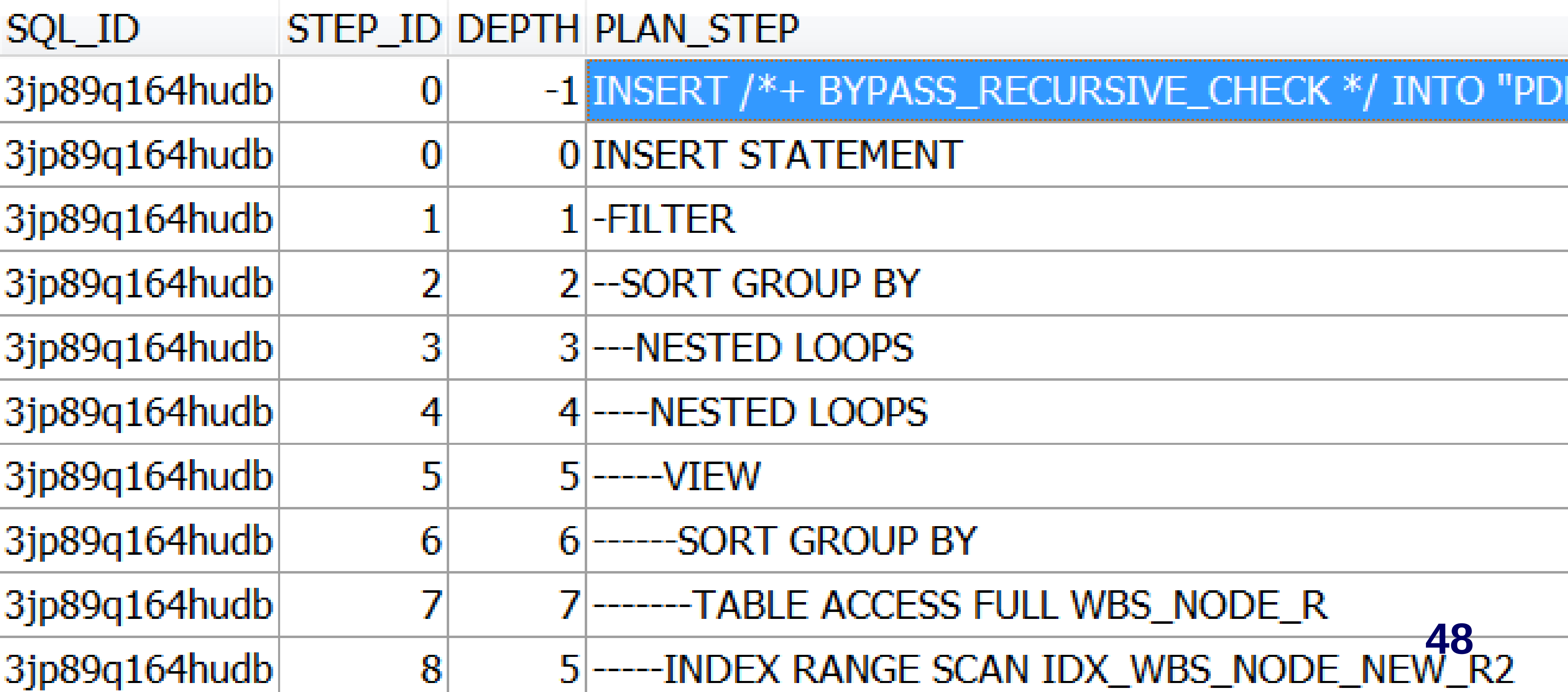
sid	ser#	Blocked User Event	Event Count	SQL_ID	Dur. sec	sql_text of Blocked User	Max Blocker
1056	19929	enq: MS - contention	2,820	adr3h7jg6g	2,797	/* QSMQ VAL	PDR_MART:953
987	65453	enq: MS - contention	2,820	3ffwrntvyx7	2,797	/* QSMQ VAL	PDR_MART:953
784	3222	library cache lock	2,821	g5a9kjcv01	2,792	DECLARE job	PDR_MART:953
953	9738	row cache lock	2,820	cxcd69ng5	2,787	/* QSMQ VAL	DISHSUBPRD:
878	2103	enq: MS - contention	2,817	293awdpcq	2,775	/* QSMQ VAL	PDR_MART:953
996	47693	library cache lock	2,817	ctt5473npdt	2,774	DECLARE job	PDR:878:2103
1079	12876	enq: MS - contention	2,813	3ju331z5fw	2,762	/* QSMQ VAL	PDR_MART:953
1054	55038	row cache lock	2,810	gv5sn3ubxq	2,762	call REFRESH	DISHSUBPRD:

Get all the SQL that involve a particular Object

- **AWR - sql and plan for an object.sql**
- Usage Scenario:

  High I/O on a particular table or index …

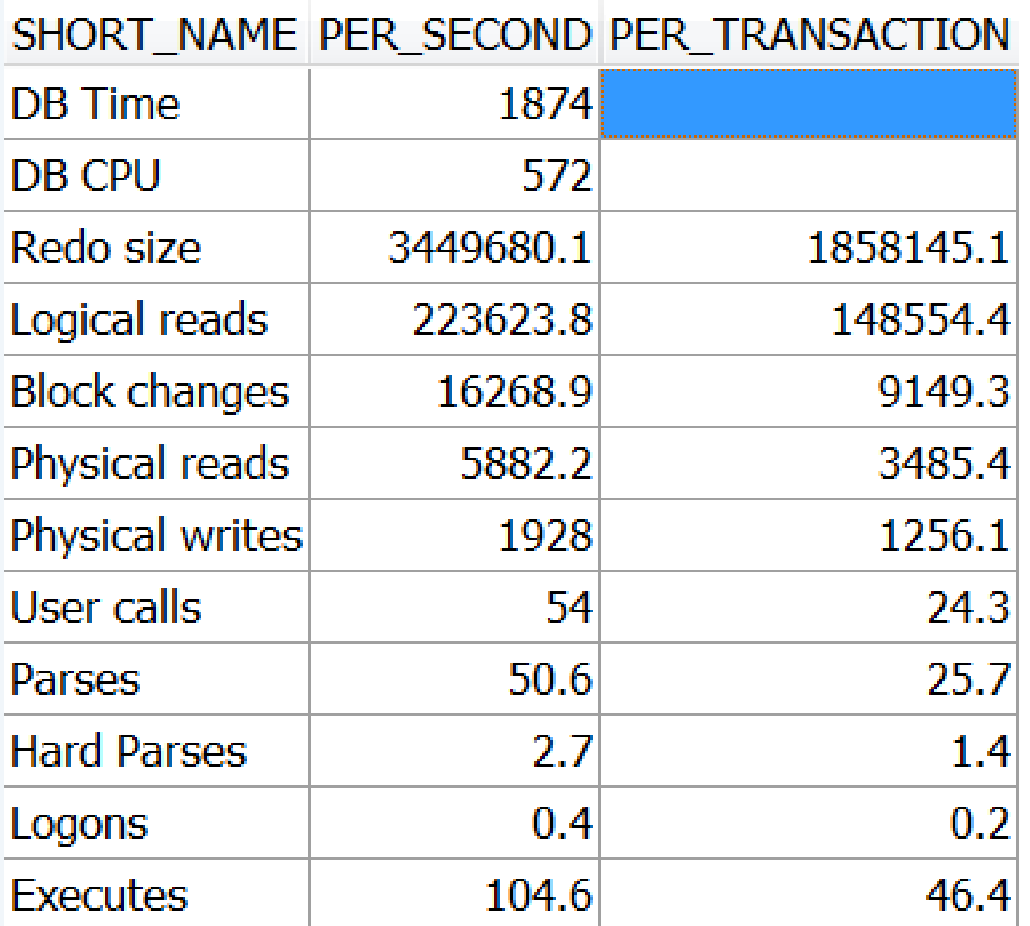
  What other SQL hits that same object?



SQL_ID	STEP_ID	DEPTH	PLAN_STEP
3jp89q164hudb	0	-1	INSERT /*+ BYPASS_RECURSIVE_CHECK */ INTO "PD
3jp89q164hudb	0	0	INSERT STATEMENT
3jp89q164hudb	1	1	-FILTER
3jp89q164hudb	2	2	--SORT GROUP BY
3jp89q164hudb	3	3	---NESTED LOOPS
3jp89q164hudb	4	4	----NESTED LOOPS
3jp89q164hudb	5	5	-----VIEW
3jp89q164hudb	6	6	------SORT GROUP BY
3jp89q164hudb	7	7	-------TABLE ACCESS FULL WBS_NODE_R
3jp89q164hudb	8	5	-----INDEX RANGE SCAN IDX_WBS_NODE_NEW_R2

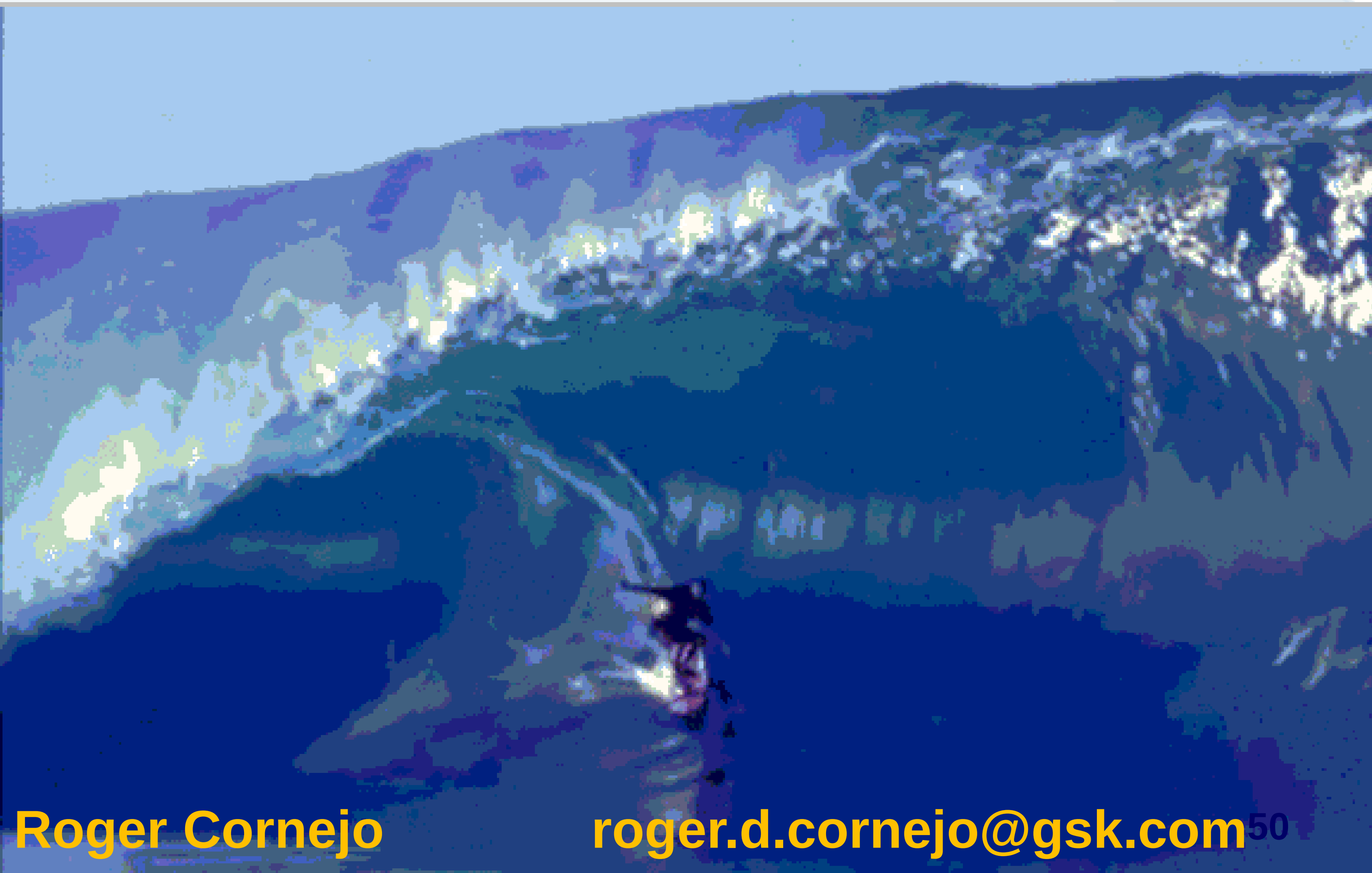
Reverse Engineering an AWR Report

- **AWR** - Report Queries.sql
- **Usage Scenario:**
  Data for Load Profile Section of AWR Report



SHORT_NAME	PER_SECOND	PER_TRANSACTION
DB Time	1874	
DB CPU	572	
Redo size	3449680.1	1858145.1
Logical reads	223623.8	148554.4
Block changes	16268.9	9149.3
Physical reads	5882.2	3485.4
Physical writes	1928	1256.1
User calls	54	24.3
Parses	50.6	25.7
Hard Parses	2.7	1.4
Logons	0.4	0.2
Executes	104.6	46.4

Taming the AWR Tsunami



Roger Cornejo roger.d.cornejo@gsk.com