Taking ASH Analysis To The Next Level

Plan Changes – Parsing Issues

Craig Shallahamer
craig@orapub.com

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ASH provides DBAs with an amazing storehouse of detailed session activity.

ORAPUB

ASH provides DBAs with an amazing storehouse of detailed session activity.

I have focused on random intense performance incidents, which takes advantage of pure "sample nature" of ASH data.

This presentation was given by Craig Shallahamer (craig@orapub.com) at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

ASH Scratch Pad

BLOOD HOUND Toolkit

> orapub.com
>resources
> tools

ORAPUB

Question:

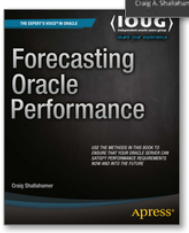
Are there some non-standard, unusual, unexpected yet powerful ways we can leverage ASH data?

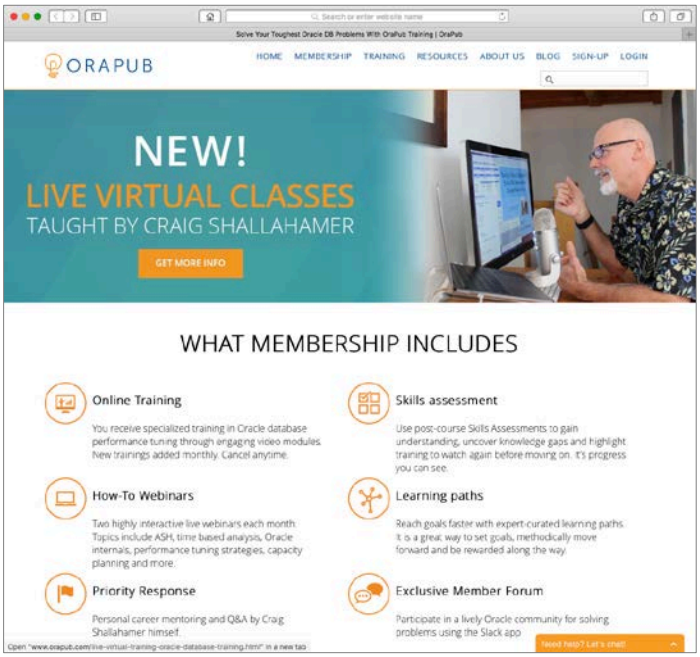
This presentation was given by Craig Shallahamer (craig@orapub.com) at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

About Me...

- Long time Oracle DBA
- Specialize in Oracle Database performance and performance engineering
- Performance researcher
- Blogger: A Wider View About Oracle Performance Tuning
- Author: Oracle Performance Firefighting and Forecasting Oracle Performance.
- Conference speaker
- Teacher and mentor
- Oracle ACE Director
- IOUG DBA Track Manager





ORAPUB

OraPub works closely with DBAs enabling them to solve every Oracle performance problem they encounter.

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

Your core Oracle performance analysis options

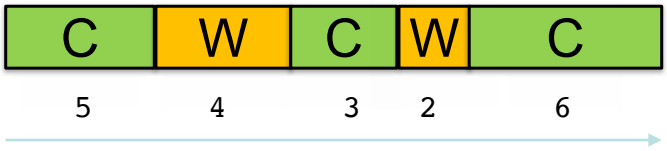
Our three core performance data sources.

- Timing Oracle activity (70%)
  - Oracle time model (AWR, tracing, etc.)
  - Application instrumentation
- Sampling Oracle sessions (25%)
  - Active session history (ASH)
- Oracle Tracing (5%)
  - Makes use of Oracle's time model
  - DBMS_MONITOR
  - Set event…

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

Oracle Time. What is it?

Database Time = CPU Time + Non-Idle Wait Time




```
        T I M E                W O R K
CPU                  = 14
Non-Idle Wait        =  6     300K LIOs
----------------------
Elapsed = DB time    = 20
Idle Wait            =  0
Experience           = 20
```

Single Serial Session

ASH Analysis

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

The types of questions can we ask

- **Summarize/Profile** anything. instance, sql_id, module, session, program, state,…
- **Top** anything (on cpu or waiting). instance, sql_id, module, session,…
- **Interval timeline** instance, module, etc.
- **Timeline** a session, sample by sample
- **Visualize** a complex situation at a given time

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

Can we do more?

ORAPUB

How to use ASH to pinpoint **plan changes**

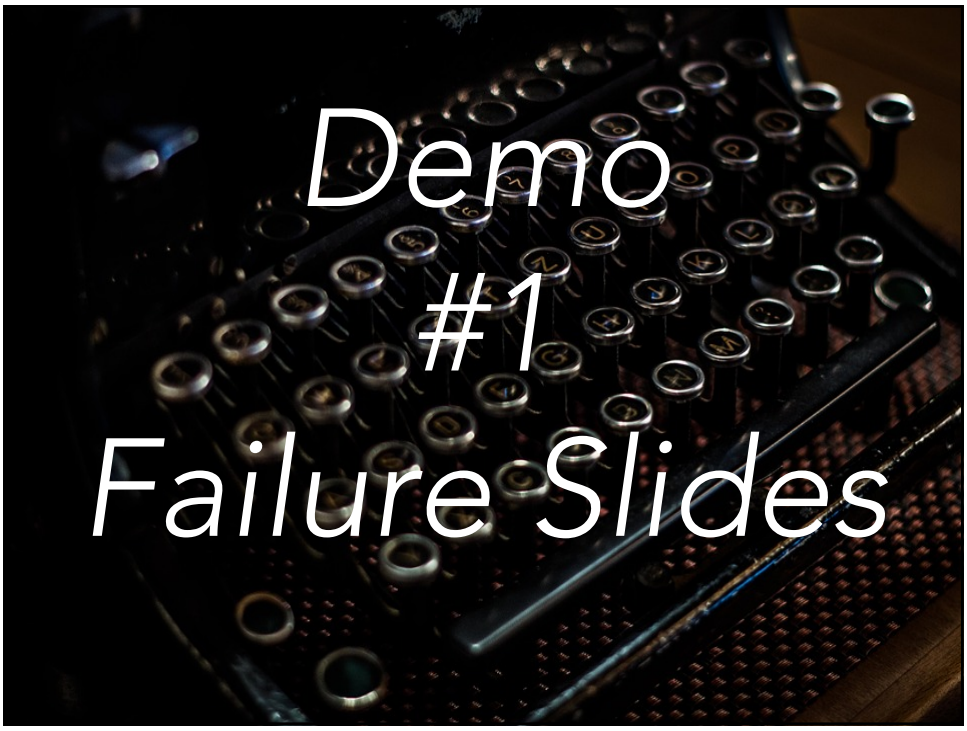
This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

Situation and objectives

- Situation. Operations called us to say:
  - User called
  - surprising slow performance,
  - occurred July 17 @ 19:30,
  - OEM shows the SQL_ID
- Objectives
  - Investigate looking for reasons for poor performance
  - Contact appropriate development group with your findings.

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.



The key columns.

```
SQL> desc dba_hist_active_sess_history
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
SNAP_ID                                   NOT NULL NUMBER
DBID                                      NOT NULL NUMBER
INSTANCE_NUMBER                           NOT NULL NUMBER
SAMPLE_ID                                 NOT NULL NUMBER
SAMPLE_TIME                               NOT NULL TIMESTAMP(3)
SESSION_ID                                NOT NULL NUMBER
SESSION_SERIAL#                                    NUMBER
SESSION_TYPE                                       VARCHAR2(10)
FLAGS                                              NUMBER
USER_ID                                            NUMBER
SQL_ID                                             VARCHAR2(13)
IS_SQLID_CURRENT                                   VARCHAR2(1)
SQL_CHILD_NUMBER                                   NUMBER
SQL_OPCODE                                         NUMBER
…
TOP_LEVEL_SQL_OPCODE                               NUMBER
SQL_PLAN_HASH_VALUE                                NUMBER
SQL_FULL_PLAN_HASH_VALUE                           NUMBER
SQL_ADAPTIVE_PLAN_RESOLVED                         NUMBER
SQL_PLAN_LINE_ID                                   NUMBER
…
SQL_EXEC_ID                                        NUMBER
SQL_EXEC_START                                     DATE
```


This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

Reducing the frequent clutter.

```
def filter2 = "curr_sql_plan_hash_value != prev_sql_plan_hash_value"
def ashCols = "sql_id, sql_child_number, to_char(SQL_EXEC_START,'MI:SS') START_T,
               sql_exec_id, session_id, session_serial#"

SQL_ID        CHILD START SQL_EXEC_ID SESSION_ID    SERIAL# SAMPLE_TIME            CUR PLAN   PRE PLAN CHA
------------- ----- ----- ----------- ---------- ---------- --------------------- ---------- ---------- ---
234j2z9h7m01f     1 13:13    18428491       1276       8223 2014-Jul-17 21:13:13 3515140844 2974043169 Yes
234j2z9h7m01f     1 05:04    18441771       2188      64871 2014-Jul-18 00:05:04 1718532790 3515140844 Yes
234j2z9h7m01f     0 29:17    18457848        531      42329 2014-Jul-18 00:29:17 2616175562 1718532790 Yes
234j2z9h7m01f     1 47:40    18463296       2188      64871 2014-Jul-18 00:47:40 3238610579 2616175562 Yes
234j2z9h7m01f     1 22:46    18499815       1276       8225 2014-Jul-18 15:22:47 1718532790 3238610579 Yes
234j2z9h7m01f     1 14:58    18510612        149      17159 2014-Jul-18 18:14:59 3238610579 1718532790 Yes
```

Someone called at 3:30pm on July 18th about a problem!
We know the SQL_ID from OEM… something changed!
We know the session_id and serial number.
We know when the SQL started.
Now we can "timeline" the session
to see more session level details.

ORAPUB

Let's timeline the key session!

```
def ashCols="event,sql_id,SQL_PLAN_HASH_VALUE,to_char(SQL_EXEC_START,'MI:SS') DT,sql_exec_id"
def ashWhere='session_id=1589 and session_serial#=52847'

 SAMPLE_ID DT            SID SESSION_STATE EVENT                    SQL_ID          SQL_PLAN_HASH_VALUE DT    SQL
---------- ----------- ----- ------------- ------------------------ --------------- ------------------- ----- ---
  53937912 17 19:33:10  1589 WAITING       db file sequential read  bwua6ab574qz5           1652481407 30:53
  53937922 17 19:33:20  1589 WAITING       db file sequential read  4z3zwn3mc0ku7           3317200625 33:17
  53937932 17 19:33:30  1589 WAITING       db file sequential read  80v83uf003p3z           4132976507 33:26
  53937942 17 19:33:40  1589 WAITING       db file sequential read  80v83uf003p3z           4132976507 33:26
  53937952 17 19:33:50  1589 WAITING       db file sequential read  234j2z9h7m01f           3515140844 33:49
  53937962 17 19:34:00  1589 WAITING       db file sequential read  234j2z9h7m01f           3515140844 33:59
  53937972 17 19:34:10  1589 WAITING       db file sequential read  234j2z9h7m01f           3515140844 34:09
  53937982 17 19:34:20  1589 ON CPU                                 2qa602puysys3           3515140844
  53937992 17 19:34:30  1589 ON CPU                                                                   0
  53938012 17 19:34:50  1589 ON CPU                                 9guhygusvgrfy            421693023 34:49
  53938022 17 19:35:00  1589 WAITING       gc current grant 2-way   5qx6a6d70m0xb            421693023 34:59
```

We can see the key SQL is executed 3 times! How can we find the Developers?

The output above is based on OraPub's ASH Scratch Pad "timeline" script.

ASH Scratch Pad

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

Let's timeline the key session!

```
def ashCols="event,sql_id,module,to_char(SQL_EXEC_START,'MI:SS') DT"

 SAMPLE_ID DT            SID SESSION_STATE EVENT                    SQL_ID          MODULE          DT
---------- ----------- ----- ------------- ------------------------ --------------- --------------- -----
  53937932 17 19:33:30  1589 WAITING       db file sequential read  80v83uf003p3z   PSAE.AP_PSTVCHR 33:26
  53937942 17 19:33:40  1589 WAITING       db file sequential read  80v83uf003p3z   PSAE.AP_PSTVCHR 33:26
  53937952 17 19:33:50  1589 WAITING       db file sequential read  234j2z9h7m01f   PSAE.AP_PSTVCHR 33:49
  53937962 17 19:34:00  1589 WAITING       db file sequential read  234j2z9h7m01f   PSAE.AP_PSTVCHR 33:59
  53937972 17 19:34:10  1589 WAITING       db file sequential read  234j2z9h7m01f   PSAE.AP_PSTVCHR 34:09
  53937982 17 19:34:20  1589 ON CPU                                 2qa602puysys3   PSAE.AP_PSTVCHR
  53937992 17 19:34:30  1589 ON CPU                                                 PSAE.AP_PSTVCHR
  53938012 17 19:34:50  1589 ON CPU                                 9guhygusvgrfy   PSAE.AP_PSTVCHR 34:49
```

We know the module. Now it's time to talk with the Developers and show them what we discovered.

It would be helpful to know if these SQL run times where unusually long.

ASH Scratch Pad

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

Using ASH to detect plan changes.

We used the key ASH plan related columns to detect a plan change around the reported performance situation for the given SQL_ID.

We drilled down to the session level, confirmed what we saw and found the key SQL was executed multiple times in succession by the same session.

We also determined the relevant module and are contacting the appropriate development team with our findings.

ORAPUB

That is

how to use ASH

to pinpoint

plan changes

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

How to use ASH

to detect

parsing issues

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

Oracle time problem

ORAPUB

48.4K

Time Model Statistics

- DB Time represents total time in user calls
- DB CPU represents CPU time of foreground processes
- Total CPU Time represents foreground and background processes
- Statistics including the word "background" measure background
- Ordered by % of DB time in descending order, followed by

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	82,924.74	96
DB CPU	48,443.49	56
parse time elapsed	15,672.18	18
hard parse elapsed time	15,534.96	18
hard parse (sharing criteria) elapsed time	255.99	0
PL/SQL execution elapsed time	45.66	0
connection management call elapsed time	35.38	0
sequence load elapsed time	24.94	0.03
PL/SQL compilation elapsed time	3.33	0.00
failed parse elapsed time	1.48	0.00
repeated bind elapsed time	0.75	0.00
hard parse (bind mismatch) elapsed time	0.31	0.00
DB time	86,270.46	
background elapsed time	431.18	
background cpu time	121.45	
total CPU time	48,564.94	

86.3K

Foreground Wait Class

14.2K

- s - second, ms - millisecond - 1000th of a second
- ordered by wait time desc, waits desc
- %Timeouts: value of 0 indicates value was < .5%. Value of null is truly 0
- Captured Time accounts for 72.5% of Total DB time 86,270.46 (s)
- Total FG Wait Time: 14,131.32 (s) DB CPU time 48,443.49 (s)

Wait Class	Waits	%Time -outs	Total Wait Time (s)	Avg wait (ms)	%DB time
DB CPU			48,443		56.15
User I/O	5,906,084	0	9,260	1.57	10.73
Application	14,203	0	3,739	263.26	4.33
Other	47,314	0	791	16.71	0.92
Concurrency	51,365	0	183	3.57	0.21
Commit	80,982	0	94	1.16	0.11
System I/O	72,921	0	57	0.79	0.07
Network	7,225,651	0	6	0.00	0.01
Configuration	31	6	1	18.62	0.00

```
DB CPU              =  48.4K
Non-Idle Wait       =  14.2K
-------------------------
Derived DB time     =  62.6K
Reported DB time    =  86.3K
```

?

Oracle time problem

ORAPUB

5.3K

Time Model Statistics

- DB Time represents total time in user calls
- DB CPU represents CPU time of foreground processes
- Total CPU Time represents foreground and background process
- Statistics including the word "background" measure background statistic
- Ordered by % of DB time in descending order, followed by Statis

Statistic Name	Time (s)	% of DB Time
sql execute elapsed time	11,621.32	64.62
DB CPU	5,258.74	29.24
parse time elapsed	187.56	1.04
hard parse elapsed time	159.21	0.89
PL/SQL execution elapsed time	89.30	0.50
PL/SQL compilation elapsed time	13.95	0.08
connection management call elapsed time	1.20	0.01
repeated bind elapsed time	0.83	0.00
sequence load elapsed time	0.04	0.00
hard parse (sharing criteria) elapsed time	0.02	
failed parse elapsed time	0.00	
DB time	17,985.44	
background elapsed time	842.50	
background cpu time	333.28	
total CPU time	5,592.02	

18.0K

Wait Classes by Total Wait Time

Wait Class	Waits	Total Wait Time (sec)	A
DB CPU		5,259	
Network	69,188,140	4,351	
User I/O	1,910,950	2,518	
System I/O	304,232	574	
Concurrency	384,021	203	
Other	1,023,467	73	
Cluster	422,409	36	
Application	4,346	3	
Commit	3,539	2	
Configuration	486	2	
Scheduler	26	0	

7.8K

```
DB CPU              =  5.3K
Non-Idle Wait       =  7.8K
-------------------------
Derived DB time     = 13.1K
Reported DB time    = 18.0K
```

?

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

Parsing issue AWR report.

ORAPUB

Instance Efficiency Percentages (Target 100%)

Buffer Nowait %:	99.97	Redo NoWait %:	100.00
Buffer Hit %:	94.96	In-memory Sort %:	100.00
Library Hit %:	91.59	Soft Parse %:	86.73
Execute to Parse %:	76.68	Latch Hit %:	99.95
Parse CPU to Parse Elapsd %:	22.13	% Non-Parse CPU:	96.92
Flash Cache Hit %:	0.00		

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Avg(ms)	% DB time	Wait Class
DB CPU		2194.7		84.8	
cursor: pin S wait on X	402	233.8	581.52	9.0	Concurrency
direct path read	1,294,974	41.9	0.03	1.6	User I/O
log file sync	5,039	12.5	2.48	.5	Commit
db file sequential read	467,628	12.3	0.03	.5	User I/O
SQL*Net break/reset to client	6,934	7.6	1.09	.3	Application
db file parallel read	12,337	6.9	0.56	.3	User I/O
db file scattered read	70,528	3.3	0.05	.1	User I/O
direct path sync	722	2	2.82	.1	User I/O
SQL*Net more data to client	7,084	1.5	0.21	.1	Network

ORAPUB

Parsing analysis problems using a time time analysis.

- DB CPU. What does "one second" of CPU consumption actually mean in the world of "virtual CPUs"? Does one second of CPU equates to one second of Oracle wait time?
- Parsing CPU ratio is helpful when CPU time is correct, to alert us of a [potential] problem, but what SQL is involved? …thousands of similar SQL.
- Latch/Mutex acquisition spinning consumes CPU, with no wait time. Oracle does not instrument CPU consumption. So, no early detection.

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

What we need is a non time based parsing detecting solution.

ASH solve the problem because ASH analysis is based on analyzing active session samples.

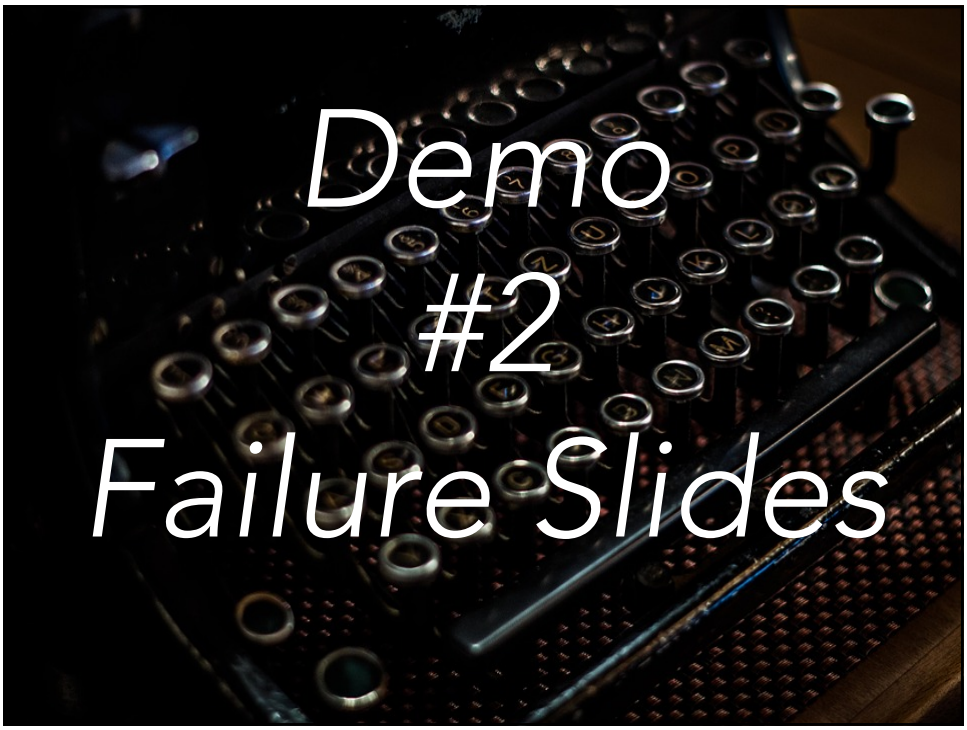
A fundamentally different approach.

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

Situation and objectives

- Situation. The vendor application
  - Is not usefully instrumented
  - Does not use bind variables
  - Cursor_sharing force is NOT supported by vendor application
  - DBAs believe there are major parsing issues.
- Objectives
  - Investigate and confirm the reported parsing situation.



This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

Is there a parsing issue?

```
def datasource=v$active_session_history
def dbahistdetails=' and 1=1'
def timingdetails="sample_time >= current_timestamp - interval '1' minute"

--def ashstate='ON CPU'
--def ashstate='WAITING'
def ashstate='%'


Select
  count(session_id) active_sessions,
  sum(decode(in_parse,'N',1,0)) not_in_parse,
  sum(decode(in_parse,'Y',1,0)) in_parse,
  sum(decode(in_hard_parse,'Y',1,0)) in_hard_parse,
  sum(decode(in_parse,'Y',1,0))/count(session_id) in_parse_ratio,
  sum(decode(in_hard_parse,'Y',1,0))/sum(decode(in_parse,'Y',1,0)) in_parse_hard_par
  sum(decode(in_hard_parse,'Y',1,0))/count(session_id) hard_parse_ratio
From
      &datasource
Where
      &timingdetails
      &dbahistdetails
  and session_state like '&ashstate'
```

ORAPUB

Is there a parsing issue?

```
set tab off verify off feedback off
col active_sessions heading "Active|Sessions|(A,B+C)"
col not_in_parse heading "AS Not|Parsing|(B,A-C)"
col in_parse heading "AS|Parsing|(C,A-B)"
col in_hard_parse heading "AS|Parsing|Hard|(D,C-E)"
col in_parse_ratio heading "AS|Parsing|Ratio|(F,C/A)" format 990.000
col in_parse_hard_parse_ratio heading "AS|Parsing|Hard Ratio|(G,D/C)" format
col hard_parse_ratio heading "AS|Hard Parse|Ratio|(H,D/A)" format 990.000
                                          AS       AS         AS         AS
    Active     AS Not         AS    Parsing  Parsing    Parsing Hard Parse
  Sessions    Parsing    Parsing       Hard    Ratio Hard Ratio      Ratio
   (A,B+C)    (B,A-C)    (C,A-B)    (D,C-E)  (F,C/A)    (G,D/C)    (H,D/A)
---------- ---------- ---------- ---------- -------- ---------- ----------
(1)    461         77        384        270    0.833      0.703      0.586
(2)    346        249         97          1    0.280      0.010      0.003

(1)With Cursor Sharing = EXACT ← Default and vendor supported option.
(2)With Cursor Sharing = FORCE
```

Lots of parsing and lots of hard parsing… super bad!

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

One minute instance profile.

```
def datasource=v$active_session_history
def dbahistdetails=' and 1=1'
def timingdetails="sample_time >= current_timestamp - interval '1' minute
def ashcolkey=sample_id
def ashcolval='%'

SQL> /
     Total    CPU         CPU    CPU HARD WAITING WAITING   WAITING
        AS    PCT PARSING PCT PARSING PCT     PCT  IO PCT OTHER PCT
---------- ------ ----------- ----------- ------- ------- ---------
       382   89.8        89.8        65.9    10.2     0.0     100.0

  COUNT(*) EVENT
---------- -------------------------
        18 library cache: mutex X
        17 latch: shared pool
         3 latch: row cache objects
         1 resmgr:cpu quantum
```

Source:
OraPub ASH Scratch Pad
Profile script

Only 10% of sessions waiting. Parsing is hidden within CPU, then spills over into wait time.

ORAPUB

Find the top parsing SQL_ID

```
col event format a25

--def ashstate='ON CPU'
--def ashstate='WAITING'
def ashstate='%'

def ashWhere="1=1"
def ashCols='in_parse,in_hard_parse,sql_id'

select count(*), &ashCols
from  &datasource
where &timingdetails
      &dbahistdetails
  and session_state like '&ashstate'
  and &ashWhere
group by &ashCols
order by 1 desc
```


This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

Lots of single count ASH entries.

```
  COUNT(*) P H SQL_ID
---------- - - ---------------
        35 N N az5pch9r972w3
        23 Y N az5pch9r972w3
        13 N N
         1 Y Y 363c6z8xscg50
         1 Y N 2158kabbdxbwk
         1 Y Y 15d6wh794c66k
         1 Y N 088n1d29k1t5f
         1 Y Y 213w4cuv18fws
         1 Y Y 9q9a89pjpm70r
         1 Y Y a1m2na65s9ds5
         1 Y Y 5f74vyh494kng
...
328 rows selected
```

ORAPUB

Let's get the SQL text.

```
SQL> @sqlgettext sql_id 8fnbn6a5pjxwy
old   4: where lower(&id_key) like '&id_value'
new   4: where lower(sql_id) like '8fnbn6a5pjxwy'

no rows selected

SQL> @sqlgettext sql_id 9dxwh2jkh69mz
old   4: where lower(&id_key) like '&id_value'
new   4: where lower(sql_id) like '9dxwh2jkh69mz'

no rows selected

SQL> @sqlgettext sql_id 3dwmzm1bsd87d
old   4: where lower(&id_key) like '&id_value'
new   4: where lower(sql_id) like '3dwmzm1bsd87d'

no rows selected
```

GONE!

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

Let's try some dynamic SQL magic.

```
select '@sqlgettext sql_id '||sql_id
from   &datasource
Where  &timingdetails
       &dbahistdetails
   and session_state like '&ashstate'
   and in_hard_parse = 'Y'
/
'@SQLGETTEXTSQL_ID'||SQL_ID
--------------------------------
@sqlgettext sql_id 2yv7axp3ac1w3
@sqlgettext sql_id d77w5uc0760uy
@sqlgettext sql_id bhs3xf89vh73d
@sqlgettext sql_id cq7aa3gdggag2
…
237 rows selected.
```

ORAPUB

Let's get the SQL text.

```
SQL> @sqlgettext sql_id 2yv7axp3ac1w3
@sqlgettext sql_id d77w5uc0760uy
@sqlgettext sql_id bhs3xf89vh73d
@sqlgettext sql_id cq7aa3gdggag2
@sqlgettext sql_id gqvfkm1z5tm[illegible]old [illegible] : w[illegible]r
lower(&id_key) like '&id_value'
new   4: where lower(sql_[illegible] like '2yv7a[illegible]p3ac1w3'

no rows selected
no rows selected
no rows selecte[illegible]
no rows selecte[illegible]d
```

GONE!

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

Let's try force matching signature

```
col FORCE_MATCHING_SIGNATURE format 99999999999999999999

--def ashstate='ON CPU'
--def ashstate='WAITING'
def ashstate='%'

def ashCols='in_parse,in_hard_parse,force_matching_signature,event'

select count(*), &ashCols
from  &datasource
where &timingdetails
      &dbahistdetails
  and session_state like '&ashstate'
group by &ashCols
order by 1 desc
```

Nailed it using FMS!

```
  COUNT(*) P H FORCE_MATCHING_SIGNATURE MODULE          EVENT
---------- - - ------------------------ --------------- ----------------------
       212 Y Y      7740087316336291928 SQL*Plus
        59 Y N      7740087316336291928 SQL*Plus
        38 N N                        0 SQL*Plus
        27 Y N                        0 SQL*Plus
        10 Y Y      7740087316336291928 SQL*Plus        library cache: mutex X
        10 N N      7740087316336291928 SQL*Plus
         9 Y N      7740087316336291928 SQL*Plus        library cache: mutex X
         7 Y Y      7740087316336291928 SQL*Plus        latch: shared pool
         6 Y N      7740087316336291928 SQL*Plus        latch: shared pool
         5 N N                        0 SQL*Plus        library cache: mutex X
         4 Y N                        0 SQL*Plus        latch: shared pool
         2 Y Y      7740087316336291928 SQL*Plus        latch: row cache objects
         1 Y Y                        0 SQL*Plus

13 rows selected.
```

We have a "picture" of 212 actives sessions over the last minute, using similar yet unique statements that were caught in the middle of a a hard parse! And, they are all in the same module, SQL*Plus.

This presentation was given by Craig Shallahamer (craig@orapub.com) at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

The SQL Please…

```
def myfms = 7740087316336291928

set linesize 300

col fms format 99999999999999999999
col st format a90

select sql_id,
       force_matching_signature fms,
       hash_value,
       plan_hash_value,
       child_number,
       sql_text st
from   v$sql
where  lower(sql_text) like '%toomuch%'
  and  sql_text not like '%v$sql%'
  and  force_matching_signature = '&myfms'
```

ORAPUB

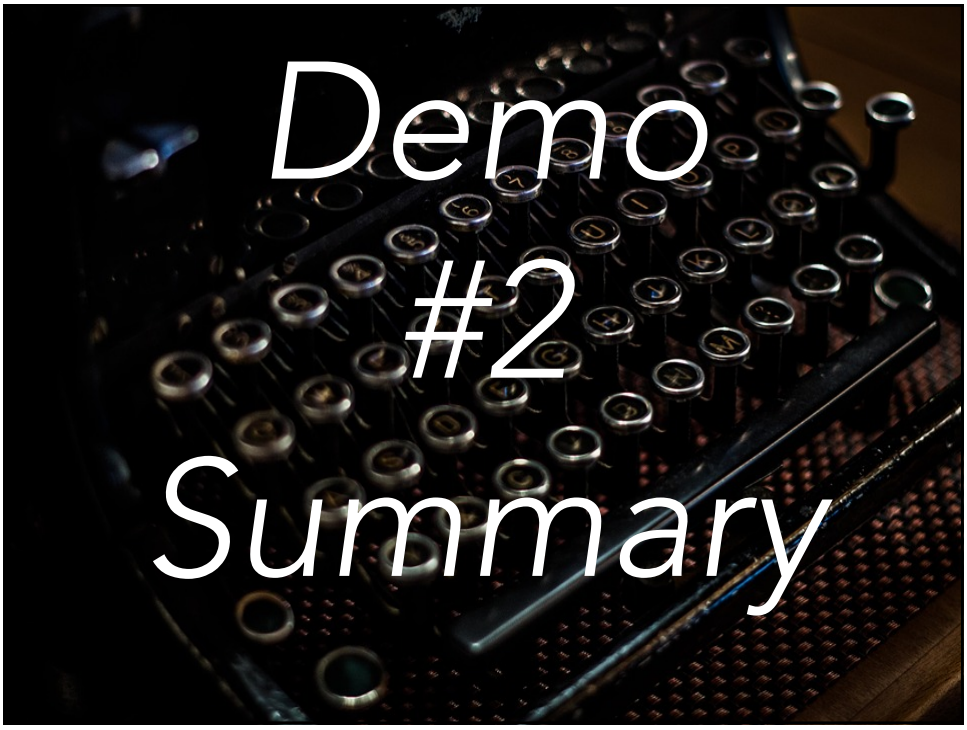
The SQL Please…

```
SQL_ID                          FMS HASH_VALUE PLAN_HASH_VALUE CHILD_NUMBER ST
------------- -------------------- ---------- --------------- ------------ --------------------------------------------------------------------------------
7nyqpw9ngvzj8   7740087316336291928 1761476136        31690868            0 select c1 from toomuch where c1 =6024073.184701063008832281283357812206 99
3jycy34m3vzq8   7740087316336291928  641597128        31690868            0 select c1 from toomuch where c1 =6984935.158659193377191281601238745533 32
…
fbnvrkz8hgzs2   7740087316336291928 3506962178        31690868            0 select c1 from toomuch where c1 =8755885.135828045848950786799462973008 94
3qkb27ux77zs6   7740087316336291928 3128164102        31690868            0 select c1 from toomuch where c1 =5320357.543926843306126440614074502661 94
dy81f0skyrzsg   7740087316336291928  636223247        31690868            0 select c1 from toomuch where c1 =8020906.777638477764456478815948540121 52
557ku2hq97zsh   7740087316336291928  747896592        31690868            0 select c1 from toomuch where c1 =1241321.162586807251839397703572082745 59
f6c033uh77zsh   7740087316336291928 2691956496        31690868            0 select c1 from toomuch where c1 =3663859.729802878218422093780319197894 11
fpf5w2mq6gzsg   7740087316336291928 3966238485        31690868            0 select c1 from toomuch where c1 =2147620.069876397148067182768170380739 12
5nc0qxnbsrztf   7740087316336291928  395050798        31690868            0 select c1 from toomuch where c1 =7909247.363097946334513054564041134284 4
dp25vd3x2bztm   7740087316336291928 4196794163        31690868            0 select c1 from toomuch where c1 =5302228.793875144133741279166299174126 85
0n2026819zztr   7740087316336291928   44039991        31690868            0 select c1 from toomuch where c1 =2805308.028071555038522707690692340174
00t68y4rubzts   7740087316336291928  799407928        31690868            0 select c1 from toomuch where c1 =4955626.619836546522340259610788338175 81
6qv192gcsbztw   7740087316336291928 3649437500        31690868            0 select c1 from toomuch where c1 =7565992.357012983822269834513580780941 09
6cd0trds6rztw   7740087316336291928 1886125884        31690868            0 select c1 from toomuch where c1 =9985636.641947928540901468476877758630 55
gl3ap9r9v3zul   7740087316336291928 3551657793        31690868            0 select c1 from toomuch where c1 =4129699.936596048033504685606982240091 33
9xznqhbv2zzu3   7740087316336291928 4130340675        31690868            0 select c1 from toomuch where c1 =3013378.288899317695384701419300593146 11
7ufxf6wqsvzu6   7740087316336291928  764280646        31690868            0 select c1 from toomuch where c1 =4220144.071002192559481650008753171374 62
b5jknust5rzuc   7740087316336291928  844889932        31690868            0 select c1 from toomuch where c1 =9152760.256207402386122128850416174869 26
8sn4upwh7zzuz   7740087316336291928  545259359        31690868            0 select c1 from toomuch where c1 =9505518.526526826964600827121234564312 8
agvdhw09tvzv2   7740087316336291928  329121634        31690868            0 select c1 from toomuch where c1 =7629304.014491093887011481271167330898 93
2rf07mg46vzv5   7740087316336291928 3362652005        31690868            0 select c1 from toomuch where c1 =2577201.588574941816655393119950761953 75
3n5bdy9dnrzv6   7740087316336291928 1531107238        31690868            0 select c1 from toomuch where c1 =293363.106404277855372206843996250314 1771
6jgayzg99gzvg   7740087316336291928 3533176687        31690868            0 select c1 from toomuch where c1 =2409071.855656596850438059993569635434 93
ghbakvqqyrzvz   7740087316336291928 2917924735        31690868            0 select c1 from toomuch where c1 =1392367.049764692807358226630276940570 53
4mhnrztyvbzwm   7740087316336291928 2109079443        31690868            0 select c1 from toomuch where c1 =1096222.802057215799562026108026367365 33
7m0h2nsntrzwt   7740087316336291928  698089369        31690868            0 select c1 from toomuch where c1 =1094531.948409581822183189092971684279 83
cf5nr4jn63zxp   7740087316336291928 1751252917        31690868            0 select c1 from toomuch where c1 =8240847.404091048874295711406216636217 03
cl4gltzqk3zxy   7740087316336291928 3978428350        31690868            0 select c1 from toomuch where c1 =8568476.325320595317302004425443260115 6
5xxd5c8htrzy3   7740087316336291928  563871683        31690868            0 select c1 from toomuch where c1 =6943311.294500149708944890163782985781 61
dqsfjhvbgmzy6   7740087316336291928 3606708166        31690868            0 select c1 from toomuch where c1 =1790397.543603685182304717928477762673 3
frhy5ut50mzy7   7740087316336291928 1242169287        31690868            0 select c1 from toomuch where c1 =2376214.161307567860120544370359918221 41
7yzwakk0ymzy8   7740087316336291928 2179596232        31690868            0 select c1 from toomuch where c1 =5130313.527513035673018447701323947376 4
8z5f16tn3rzyn   7740087316336291928 1748762580        31690868            0 select c1 from toomuch where c1 =6758010.518594595697091629345234473547 52
gg1704rqbzzk    7740087316336291928  795213810        31690868            0 select c1 from toomuch where c1 =9522566.314615603455865981064469149553 83
36ww2gx8fmzzp   7740087316336291928 1357512693        31690868            0 select c1 from toomuch where c1 =1469772.634221767965364163787604641809 11
fu4163bp3bzzq   7740087316336291928 3929407478        31690868            0 select c1 from toomuch where c1 =9547779.020540496224913837630707827144 28
f3f03jy0dgzzy   7740087316336291928 2161639422        31690868            0 select c1 from toomuch where c1 =8983020.781097266300777477725681777228 3

18084 rows selected.
```

Clearly part of the issue is
similar yet unique SQL statements.

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.



ORAPUB

We used ASH to:

Detect a real parsing issue, both from a high level "instance profile" and more detailed perspective.

We clearly saw the parsing impact on CPU usage and also the spill over into Oracle waits.

All this without reliance on Oracle's time model, that is, CPU consumption time or wait event time.

Also, we found the top parsing SQL_IDs... too many. Found the top parsing FMS, pulled the SQL text and discovered thousands of similar yet unique text and SQL_IDs.

This presentation was given by Craig Shallahamer (craig@orapub.com) at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

Time to talk with the application vendor.

Either get more and faster CPU cores, reduce the workload, implement bind variables or support cursor sharing FORCE.

ORAPUB

My research has shown:

ASH will show if there is a parsing issue brewing; CPU parsing activity and wait events.

ASH will show the parsing intense SQL.

ASH may not show much parsing, when parsing is light... sampling problem

ASH counts correlates with v$sysstat hard parse counts and CPU parse time ...but I do see a discrepancy when a non-production (light) load… sampling problem

Will work even when CPU "problems" exist, like with AIX… very cool

This presentation was given by Craig Shallahamer (craig@orapub.com) at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

That is how to use ASH to detect **parsing issues**

ORAPUB

Next Steps...

ASH Scratch Pad

BLOOD HOUND Toolkit

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

ORAPUB

Thank You!

Taking ASH Analysis To The Next Level

Plan Changes – Parsing Issues

Craig Shallahamer
craig@orapub.com

This presentation was given by Craig Shallahamer (craig@orapub.com)
at the September 2018 East Coast Oracle User Group (ECO) conference in Raleigh, NC USA.

You can upload the most recent version this presentation at OraPub.com

Craig Shallahamer
craig@orapub.com