DOAG November 2014

Performance monitoring in SQL*Plus using AWR and analytic functions

Marcin Przepiórowski

- Principal Oracle DBA
- Geek
- Blogger
- RAC Attack Ninja

OakTable®

ORACLE ACE

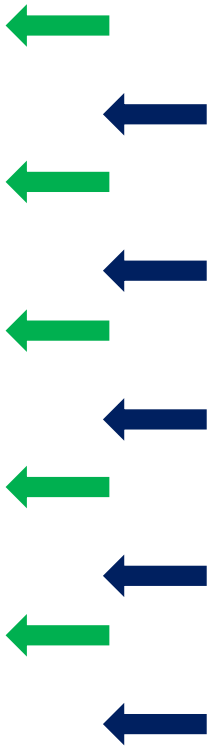
ORACLE Certified Professional

- Analytic functions
- Automatic Workload Repository – AWR
- AWR and SQL
- DB Time / Active Session History
- Q & A

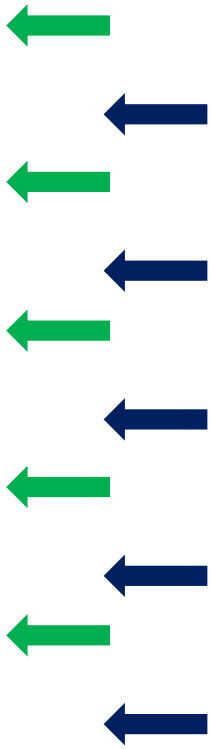
Analytic functions

- function over ()
- function over (partition by col1)
- function over (partition by col1 order by col2)

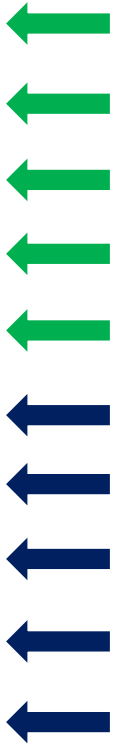
```
SQL> select r, e from (
       select rownum r, mod(rownum,2) e
       from dba_source where rownum < 11 );

         R          E
---------- ----------
         1          1
         2          0
         3          1
         4          0
         5          1
         6          0
         7          1
         8          0
         9          1
        10          0
```



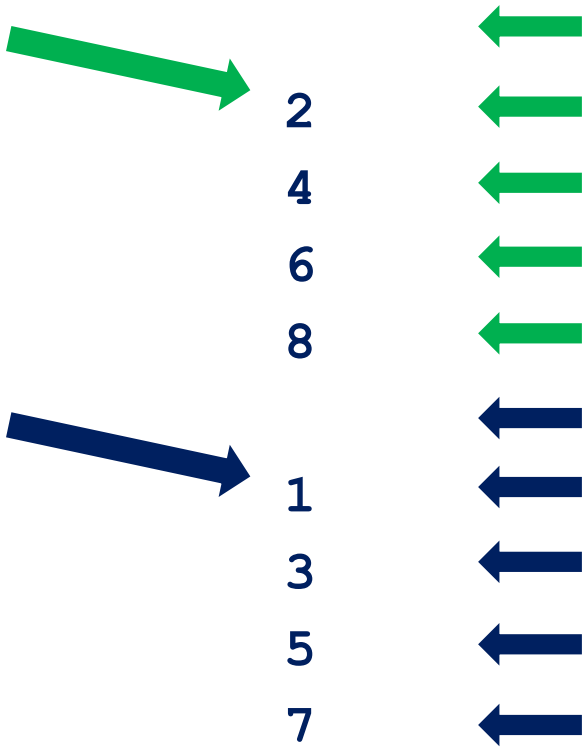
```
SQL> select r, e, sum(e) over () from (
       select rownum r, mod(rownum,2) e
       from dba_source where rownum < 11);

         R          E SUM(R)OVER()
---------- ---------- ------------
         1          1           55
         2          0           55
         3          1           55
         4          0           55
         5          1           55
         6          0           55
         7          1           55
         8          0           55
         9          1           55
        10          0           55
```



```
SQL> select r, e, sum(r) over (partition by e)
       from (select rownum r, mod(rownum,2) e
       from dba_source where rownum < 11);

         R          E SUM(R)OVER(PARTITIONBYE)
---------- ---------- --------------------------
         4          0                         30
         8          0                         30
         2          0                         30
         6          0                         30
        10          0                         30
         9          1                         25
         7          1                         25
         3          1                         25
         1          1                         25
         5          1                         25
```



```
SQL> select r, lag(r) over
       (partition by e order by r) prev_r
       from (…);

         R     PREV_R
---------- ----------
         2
         4          2
         6          4
         8          6
        10          8
         1
         3          1
         5          3
         7          5
         9          7
```



```
SQL> select r, rank() over
       (partition by e order by r) r_order
       from (…);

         R    R_ORDER
---------- ----------
         2          1
         4          2
         6          3
         8          4
        10          5
         1          1
         3          2
         5          3
         7          4
         9          5
```

RANK LAG

SUM MAX MIN DENSE_RANK

LEAD

Automatic Workload Repository - AWR

- Configured out of the box
- Capture current system status and load into repository
- Can be used for real time / historic and trend analysis

DBA_HIST_WR_CONTROL

```
SQL> col RETENTION format a25

SQL> col SNAP_INTERVAL format a25

SQL> select SNAP_INTERVAL, RETENTION, TOPNSQL from
DBA_HIST_WR_CONTROL;


SNAP_INTERVAL               RETENTION                 TOPNSQL
--------------------------- ------------------------- ----------
+00000 01:00:00.0           +00008 00:00:00.0         DEFAULT
```

DBMS_WORKLOAD_REPOSITORY

MODIFY_SNAPSHOT_SETTINGS

```
SQL> exec DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS
(RETENTION=>30*24*60, INTERVAL=>15);

SQL> select SNAP_INTERVAL, RETENTION, TOPNSQL from
DBA_HIST_WR_CONTROL;
SNAP_INTERVAL              RETENTION                      TOPNSQL
-------------------------- ------------------------------ ----------
+00000 00:15:00.0          +00030 00:00:00.0              DEFAULT
```

DBMS_WORKLOAD_REPOSITORY

ADD_COLORED_SQL

```
SQL> exec
DBMS_WORKLOAD_REPOSITORY.ADD_COLORED_SQL(SQL_ID=>'xxxxxxxxxxxx
x');


SQL> exec
DBMS_WORKLOAD_REPOSITORY.REMOVE_COLORED_SQL(SQL_ID=>'xxxxxxxxx
xxxx');
```

AWR and SQL

- **DBA_HIST_xxxx – views**
- **V$ are saved as DBA_HIST_**

  V$SYSSTAT - DBA_HIST_SYSSTAT
- **DBA_HIST_SNAPSHOT**

```
select STAT_NAME, value, …
from dba_hist_sysstat
where stat_name in ….
```

STAT_NAME	VALUE	SNAP_ID	
physical reads	439412650	45377	
consistent gets	5436891942	45377	
db block gets	42173568	45377	
physical reads	443619497	45378	
consistent gets	5694658872	45378	
db block gets	42298564	45378	

```
select snap_id, stat_name,
value -
lag(value) over (partition by stat_name order by snap_id)
delta
from ();
```

```
 SNAP_ID STAT_NAME                DELTA
---------- ---------------- ----------
     45378 consistent gets    257766930
     45378 db block gets         124996
     45378 physical reads       4206847
```



```
select s.snap_id, cast(begin_interval_time as date),
decode(stat_name, 'physical reads', delta, 0) pr,
decode(stat_name, 'physical reads', 0, delta ) lr
from () s, dba_hist_snapshot ss where …..
```

SNAP_ID	BEGIN_INTERVAL_TIME	PR	LR
45378	17-SEP-13 12.00.44	0	257766930
45378	17-SEP-13 12.00.44	0	124996
45378	17-SEP-13 12.00.44	4206847	0

```
select begin_interval_time,
(
min(begin_interval_time)-
(lag(min(begin_interval_time))
over (order by snap_id))
)*24*60*60 sec,
sum(pr) pr,
sum(lr) lr
from ()
group by begin_interval_time
```

```
BEGIN_INTERVAL_TIME    SEC         PR           LR
------------------- ------ ---------- ----------
17-SEP-13 12.00.44    3596    4206847  257891926
```

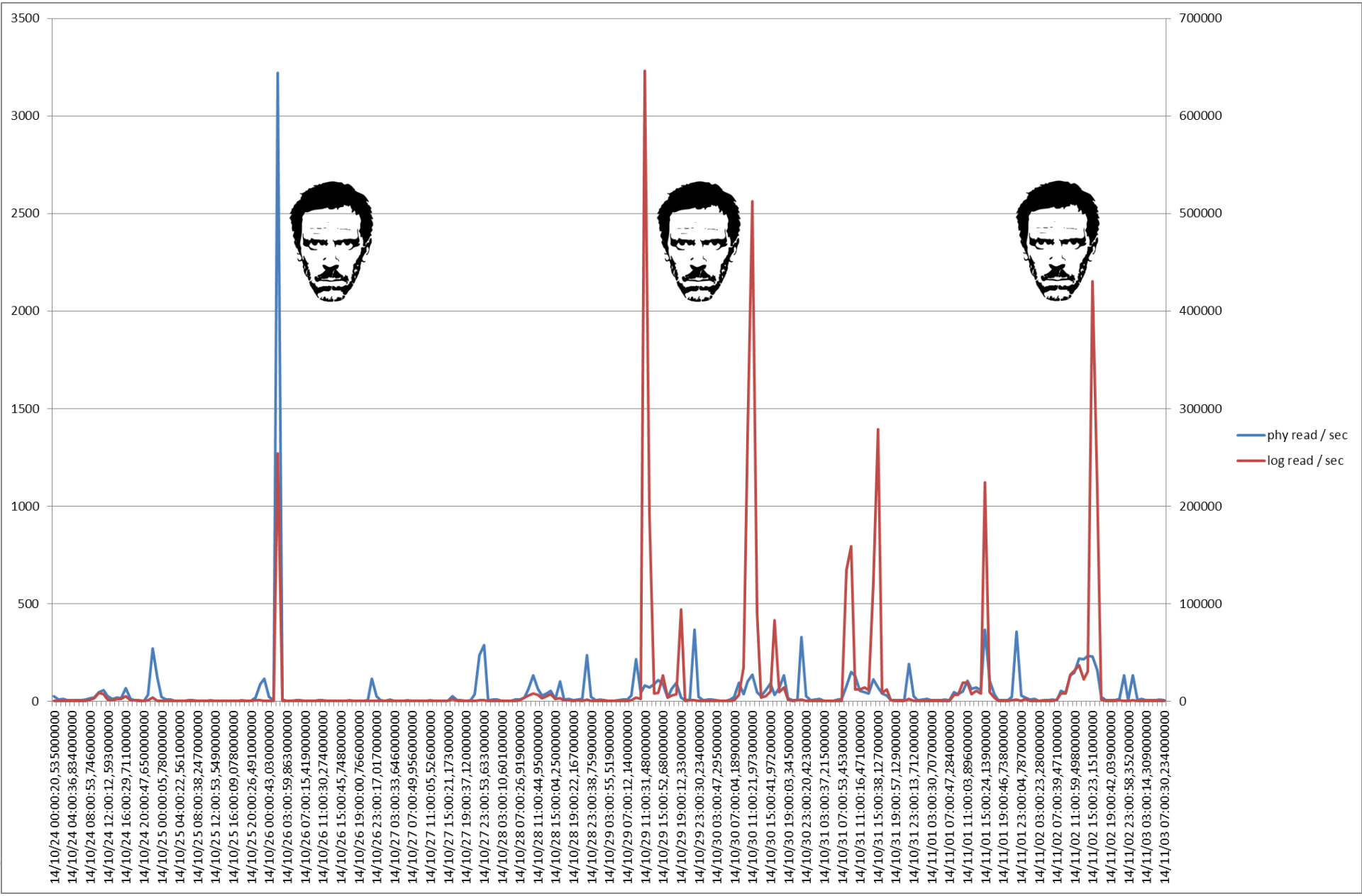
```
select begin_interval_time,
pr/sec "phy read / sec",
lr/sec "log read / sec"
from ()


BEGIN_INTERVAL_TIME  phy read / sec log read / sec
-------------------  -------------- --------------
17-SEP-13 12.00.44           1169.87       71716.33
```

	Per Second
DB Time(s):	1.1
DB CPU(s):	0.8
Redo size:	4,948.6
Logical reads:	71,771.2
Block changes:	26.3
Physical reads:	1,170.8

https://github.com/pioro/ashmasters

load_awr.sql

"Anomalies bug me"


3500
3000
2500
2000
1500
1000
500
0
700000
600000
500000
400000
300000
200000
100000
0
phy read / sec
log read / sec
14/10/24 00:00:20,535000000
14/10/24 04:00:36,834000000
14/10/24 08:00:53,746000000
14/10/24 12:00:12,593000000
14/10/24 16:00:29,711000000
14/10/24 20:00:47,650000000
14/10/25 00:00:05,780000000
14/10/25 04:00:22,561000000
14/10/25 08:00:38,247000000
14/10/25 12:00:53,549000000
14/10/25 16:00:09,078000000
14/10/25 20:00:26,491000000
14/10/26 00:00:43,030000000
14/10/26 03:00:59,863000000
14/10/26 07:00:15,419000000
14/10/26 11:00:30,274000000
14/10/26 15:00:45,748000000
14/10/26 19:00:00,766000000
14/10/26 23:00:17,017000000
14/10/27 03:00:33,646000000
14/10/27 07:00:49,956000000
14/10/27 11:00:05,526000000
14/10/27 15:00:21,173000000
14/10/27 19:00:37,120000000
14/10/27 23:00:53,633000000
14/10/28 03:00:10,601000000
14/10/28 07:00:26,919000000
14/10/28 11:00:44,950000000
14/10/28 15:00:04,250000000
14/10/28 19:00:22,167000000
14/10/28 23:00:38,759000000
14/10/29 03:00:55,519000000
14/10/29 07:00:12,140000000
14/10/29 11:00:31,480000000
14/10/29 15:00:52,680000000
14/10/29 19:00:12,330000000
14/10/29 23:00:30,234000000
14/10/30 03:00:47,295000000
14/10/30 07:00:04,189000000
14/10/30 11:00:21,973000000
14/10/30 15:00:41,972000000
14/10/30 19:00:03,345000000
14/10/30 23:00:20,423000000
14/10/31 03:00:37,215000000
14/10/31 07:00:53,453000000
14/10/31 11:00:16,471000000
14/10/31 15:00:38,127000000
14/10/31 19:00:57,129000000
14/10/31 23:00:13,712000000
14/11/01 03:00:30,707000000
14/11/01 07:00:47,284000000
14/11/01 11:00:03,896000000
14/11/01 15:00:24,139000000
14/11/01 19:00:46,738000000
14/11/01 23:00:04,787000000
14/11/02 03:00:23,280000000
14/11/02 07:00:39,471000000
14/11/02 11:00:59,498000000
14/11/02 15:00:23,151000000
14/11/02 19:00:42,039000000
14/11/02 23:00:58,352000000
14/11/03 03:00:14,309000000
14/11/03 07:00:30,234000000

Event histogram

history with deltas

```
select s.begin_interval_time,
decode (a.wait_time_milli,1,wait_count,0) ms1,
..
from DBA_HIST_EVENT_HISTOGRAM a, dba_hist_snapshot s
Where

BEGIN_INTERVAL_TIME                MS1    MS2    MS4
--------------------------- ------- ------ ------
17-SEP-13 12.00.44.511 PM    454819      0      0
17-SEP-13 12.00.44.511 PM         0 454819      0
17-SEP-13 12.00.44.511 PM         0      0 454819
```

```
select begin_interval_time,
max(ms1) ms1,
max(ms2) ms2,
…
from ( )
group by begin_interval_time
order by begin_interval_time

BEGIN_INTERVAL_TIME                MS1     MS2     MS4
--------------------------- ------- ------ ------
17-SEP-13 12.00.44.511 PM  454819  31622 454819
```

```
select begin_interval_time,
ms1 - lag(ms1) over (order by begin_interval_time)
ms2 - lag(ms2) over (order by begin_interval_time)
from ( )
```

BEGIN_INTERVAL_TIME	MS1	MS2	MS4
17-SEP-13 12.00.44.511 PM	3413	222	496
17-SEP-13 01.00.37.764 PM	4409	377	842
17-SEP-13 02.00.26.110 PM	648	44	284

https://github.com/pioro/ashmasters

event_histograms_delta_from_AWR.sql

DB Time / Active Session History

- Oracle doc

"*Database time represents the total time spent in database calls, and is an indicator of the total instance workload*"

- ASH samples counts

= DB Time in seconds

(proof in DB Time Oracle Performance Tuning: Theory and Practice by Graham Wood, John Beresniewicz)

V$ACTIVE_SESSION_HISTORY vs DBA_HIST_ACTIVE_SESS_HISTORY

Flushed history of ASH – persistent table

- Sampling rate 10 s (while ASH has 1 s)
- Partitioned for easier purging

Using ASH

SESSION STATE – ON CPU / WAITING

EVENT

CURRENT_OBJ#, etc

P1, P2, P3

BLOCKING SESSION

- Use count(*) for calculate wait or ON CPU time
- ASH is a sample DO NOT use time_waited column in calculations like:
  - sum(time_waited)
  - avg(time_waited)
  - etc.

TOP n SQL

with wait class

```
select sql_id,
decode(session_state,'WAITING',wait_class,'ON CPU')
sum(count(*)) over (partition by sql_id) /
sum(count(*)) over () pct,
count(*) cnt,
sum(count(*)) over () totalsum
from v$active_session_history where

SQL_ID        WAIT_CLASS          PCT CNT TOTAL
------------- ---------- ---------- --- -----
00natu70t45aa ON CPU      .001515152   1   660
0ff1rbxqa9su9 ON CPU      .013636364   3   660
0ff1rbxqa9su9 User I/O    .013636364   6   660
0n2ms1wttb1dh User I/O    .016666667  11   660
0rrrjbqmjhc87 ON CPU      .172727273 114   660
```

```
select sql_id,
round(pct*100,2),
nvl("'ON CPU'",0) "CPU"
from ()
pivot ( sum(round(cnt/totalsum*100,2))
for (wait_class) in ('..') order by 2 desc
```

SQL_ID	PCT	CPU	SCHEDULER	User I/O
1vur9dhsf7whh	30.79	30.34	0	.45
0rrrjbqmjhc87	10.79	10.79	0	0
82y81yfy7pu8z	9.66	9.66	0	0
9dqca4uwg467x	6.07	5.84	0	.22
0n2ms1wttb1dh	2.7	.45	0	2.02

```
SQL_ID              PCT     CPU   SCHEDULER   User I/O
------------- ------ ------ ---------- ----------
1vur9dhsf7whh   30.79   30.34          0         .45
```

SQL ID	Planhash	Sampled # of Executions	% Activity	Event
1vur9dhsf7whh	1286904735	137	30.79	CPU + Wait for CPU
0rrrjbqmjhc87	1408037256	48	10.79	CPU + Wait for CPU
82y81yfy7pu8z	2218240246	43	9.66	CPU + Wait for CPU
9dqca4uwq467x	3401411368	27	6.07	CPU + Wait for CPU
0n2ms1wttb1dh	2718274967	12	2.70	direct path read

https://github.com/pioro/ashmasters

topsql.sql

Top N SQL

with events over AWR samples

```
select begin_interval_time, sql_id,
decode(session_state,'WAITING',event,'ON CPU')
count(*) cnt,
sum(count(*)) over
(partition by begin_interval_time,sql_id) total_sql
from dba_hist_active_sess_history ash
```

SQL_ID	EVENT	CNT	TOTAL_SQL	
1vur9dhsf7whh	ON CPU	136	138	
0rrrjbqmjhc87	ON CPU	47	47	
d15cdr0zt3vtp	direct path read	4	8	
d15cdr0zt3vtp	ON CPU	4	8	
8h3tn77rzmccd	direct path read	3	8	
5zb6bsuyn13g5	ON CPU	3	3	
1vur9dhsf7whh	db file sequential	2	138	

```
select begin_interval_time, sql_id, event, cnt,
rank() over
(partition by begin_interval_time
 order by total_sql desc) r
from ()
```

SQL_ID	EVENT	CNT	R	TOTAL	
1vur9dhsf7whh	db file sequential	2	1	138	←
1vur9dhsf7whh	ON CPU	136	1	138	←
0rrrjbqmjhc87	ON CPU	47	3	47	←
82y81yfy7pu8z	ON CPU	43	4	43	
9dqca4uwg467x	ON CPU	25	5	25	
0n2ms1wttb1dh	direct path read	8	6	10	
0n2ms1wttb1dh	ON CPU	2	6	10	

```
select begin_interval_time, sql_id, event, cnt
from () where r < 10
order by begin_interval_time, cnt desc;

SQL_ID        EVENT                    CNT
------------- ------------------------ ----
1vur9dhsf7whh ON CPU                   136   
1vur9dhsf7whh db file sequential         2   
0rrrjbqmjhc87 ON CPU                    47   
82y81yfy7pu8z ON CPU                    43
9dqca4uwg467x ON CPU                    25
0n2ms1wttb1dh direct path read           8
```

https://github.com/pioro/ashmasters

top10_sql_with_events_from_AWR.sql

AWR / ASH availability

- Oracle 10g onwards
- Oracle Enterprise Edition only ☹
- Diagnostic and Tuning Pack required

- 3rd party products
- Free solution – Simulating-ASH

  https://github.com/pioro/orasash
- Scripts like Snapper, MOATS, TopAAS

Q & A

oracleprof.blogspot.com