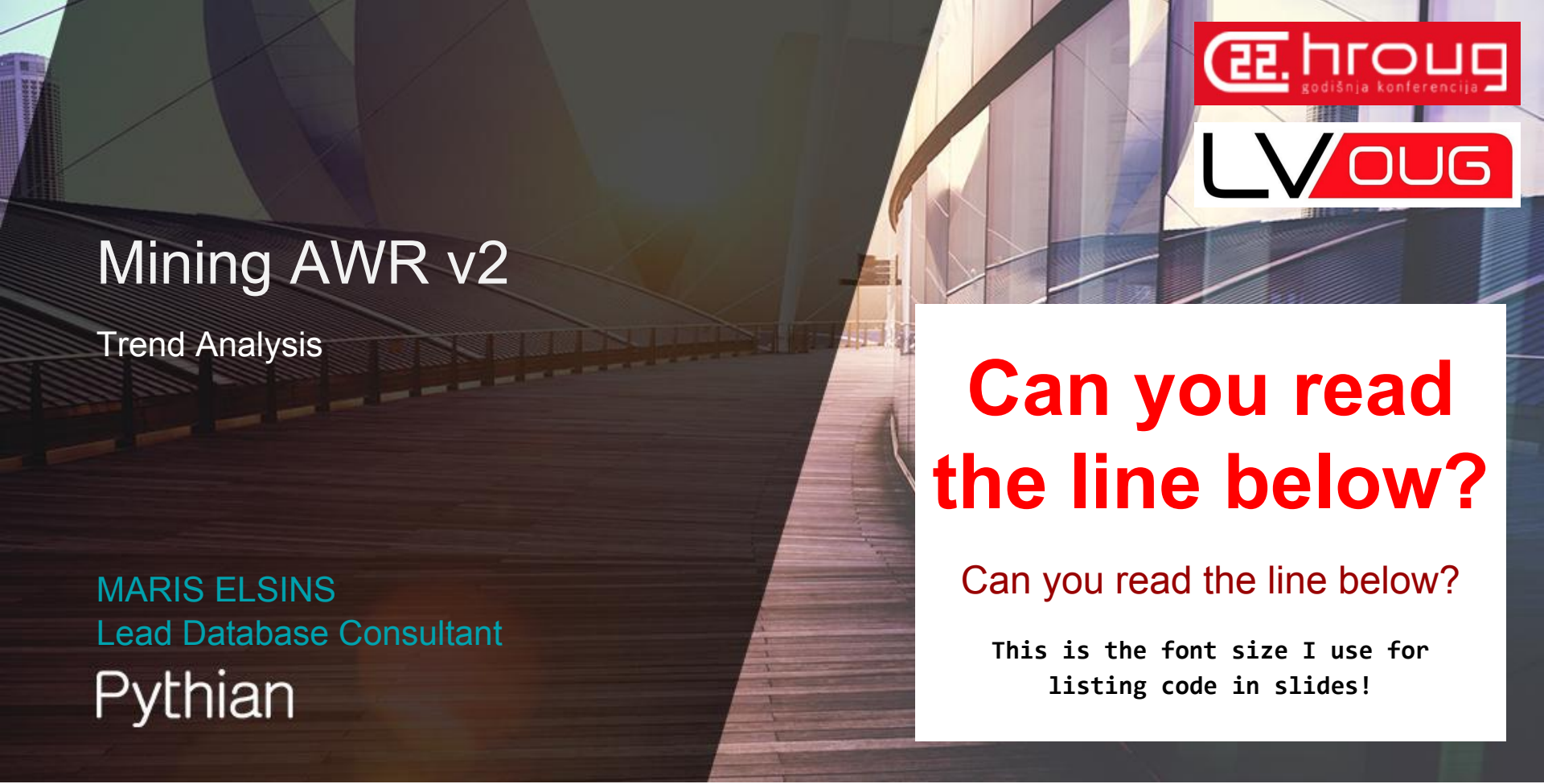


Mining AWR v2

Trend Analysis

MARIS ELSINS
Lead Database Consultant
Pythian

Can you read the line below?

Can you read the line below?

```
This is the font size I use for
listing code in slides!
```

ACE | ORACLE | Apress AUTHOR | ORACLE Certified Master | aws CERTIFIED Solutions Architect Associate

Maris Elsins

Lead Database Consultant

At Pythian since 2011

Located in Riga, Latvia
Oracle [Apps] DBA since 2005
Speaker at conferences since 2007

@MarisDBA elsins@pythian.com

http://bit.ly/getMOSPatchV2

```
PS C:\Users\elsins> java -jar .\getMOSPatch.jar patch=6880880 platform=233P regexp=.*112.* download=all
Enter your MOS username: elsins@pythian.com
Enter your MOS password:
\
Enter Comma separated platforms to list: 233P

We're going to download patches for the following Platforms/Languages:
 233P - Microsoft Windows x64 (64-bit)

Processing patch 6880880 for Microsoft Windows x64 (64-bit) and applying regexp .*112.* to the filenames:
 1 - p6880880_112000_MSWIN-x86-64.zip
 Enter Comma separated files to download: all
 All files will be downloadad because download=all was specified.

Downloading all selected files:
 Downloading p6880880_112000_MSWIN-x86-64.zip: 46MB at average speed of 8203KB/s - DONE!
PS C:\Users\elsins> java -version
java version "1.8.0_66"
Java(TM) SE Runtime Environment (build 1.8.0_66-b18)
Java HotSpot(TM) Client VM (build 25.66-b18, mixed mode, sharing)
PS C:\Users\elsins> _
```

500+ Technical Experts Helping Peers Globally

ORACLE ACE PROGRAM

---

3 Membership Tiers

- Oracle ACE Director
- Oracle ACE
- Oracle ACE Associate

bit.ly/OracleACEProgram

Connect

: oracle-ace_ww@oracle.com

Facebook.com/oracleaces

@oracleace

ORACLE Developer Community

Nominate yourself or someone you know: acenomination.oracle.com

ABOUT PYTHIAN

Pythian's 400+ IT professionals help companies adopt and manage disruptive technologies to better compete

EXPERIENCED

11,800

Systems currently managed by Pythian

GLOBAL

400

Pythian experts in 35 countries

EXPERTS

2

Millennia of experience gathered and shared over 19 years

Agenda

- “1-Dot” Philosophy •
- “2-Dots” Philosophy • •
- Data trends in AWR
- Examples

Let's Oversimplify
What is an AWR Report?

It's a dot !

Let's Oversimplify

Seriously? A Dot???

Let's Oversimplify
AWR Report as A "Dot"! Think About It!

- AWR Report gives you valuable drill-down capabilities (time interval):
  - Top Wait Events
  - Top SQL Statements in different categories
  - Top Segments by Logical Reads
  - ...

Dots and AWR - Let's investigate!

How Did the IO perform on September 9? - AWR Report is the Answer!

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	20534	09-Sep-16 00:00:38	317	15.8
End Snap:	20582	10-Sep-16 00:00:29	312	15.8
Elapsed:				
DB Time:				

Foreground Wait Events

- s - second, ms - millisecond - 1000th of a second
- Only events with Total Wait Time (s) >= .001 are shown
- ordered by wait time desc, waits desc (idle events last)
- %Timeouts: value of 0 indicates value was < .5%. Value of null is truly 0

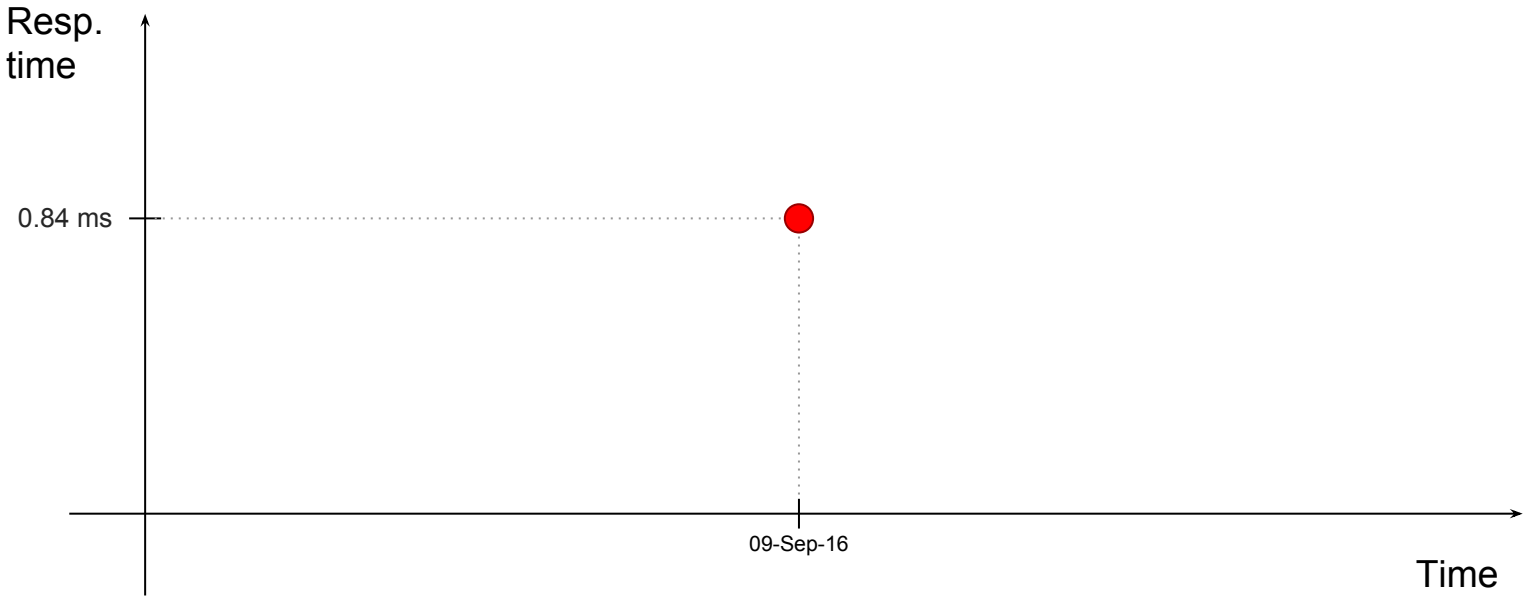
Event	Waits	%Time -outs	Total Wait Time (s)	Avg wait (ms)	Waits /txn	% DB time
db flash cache single block physical read	99,102,156		20,469	0.21	21.84	12.73
db flash cache multiblock physical read	18,608,091		9,755	0.52	4.10	6.07
cursor: pin S wait on X	1,705		5,782	3391.28	0.00	3.60
TCP Socket (KGAS)	66,258	17	4,891	73.82	0.01	3.04
log file sync	2,325,941		4,533	1.95	0.51	2.82
direct path write temp	2,755,278		3,933	1.43	0.61	2.45
direct path read	3,571,633		3,806	1.07	0.79	2.37
direct path read temp	3,502,393		1,696	0.48	0.77	1.05
log buffer space	20,302		1,173	57.21	0.00	0.73
db file sequential read	1,237,682		1,034	0.84	0.27	0.64
enq: TX - row lock contention	598		682	1144.09	0.00	0.42
read by other session	2,818,802		650	0.23	0.62	0.40
control file sequential read	976,939		514	0.53	0.22	0.32

Let's Oversimplify
AWR Report as A "Dot"! Think About It!

- Lot of resource usage / performance information is aggregated down to a single number (a "dot" on a time/value graph)
  - How much time was spent waiting on User I/O? = 42189 s
  - How quickly did the db file sequential reads respond? = 0.84 ms
  - How many transactions per second are completed? = 47.09

Let's Oversimplify

DB File Sequential Reads Performance as A "Dot"



Why This is Important?

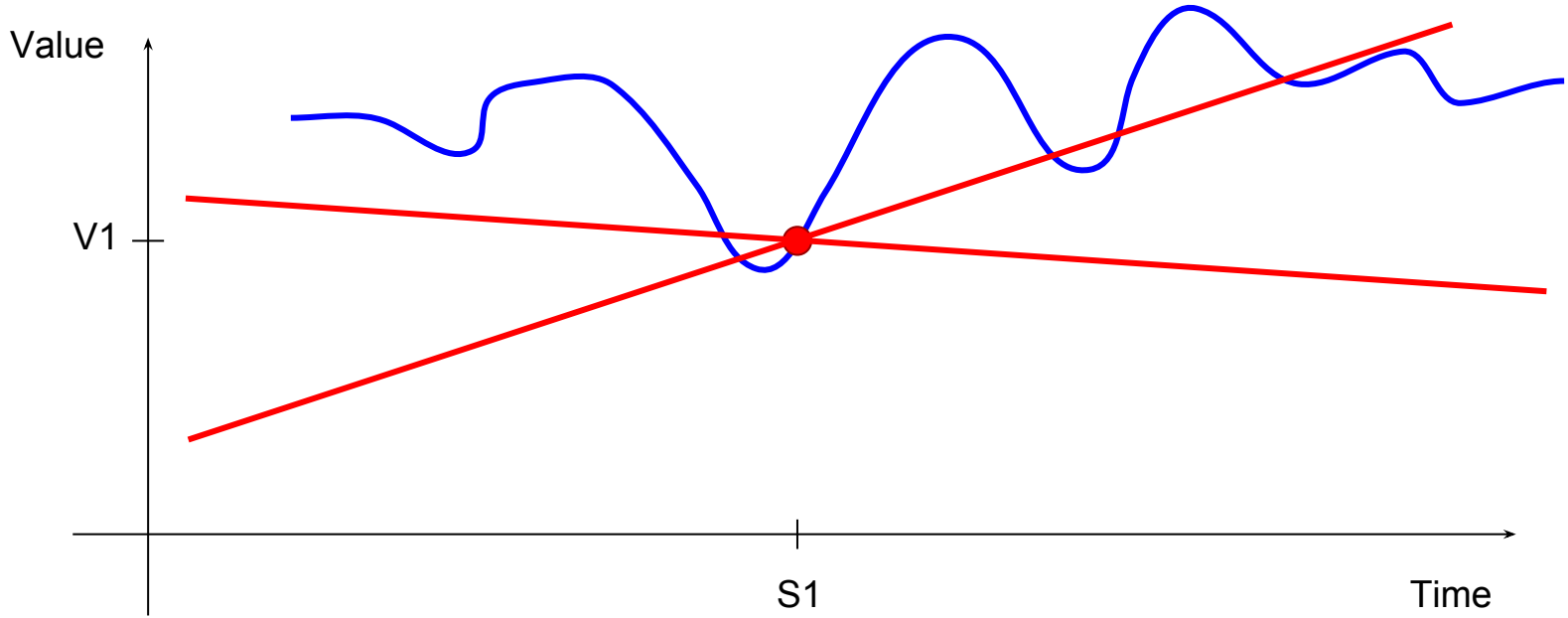
"The FBI Academy teaches new agents that the best predictor of future behavior is past behavior."

Ronald Kessler

Who doesn't want to be an FBI agent
specializing in database performance analysis?

A “dot” doesn't reveal the past behaviour

1-Dot



What does it tell us about how things develop over time?

Dots and AWR

Comparing Two periods of time (AWR Diff report)

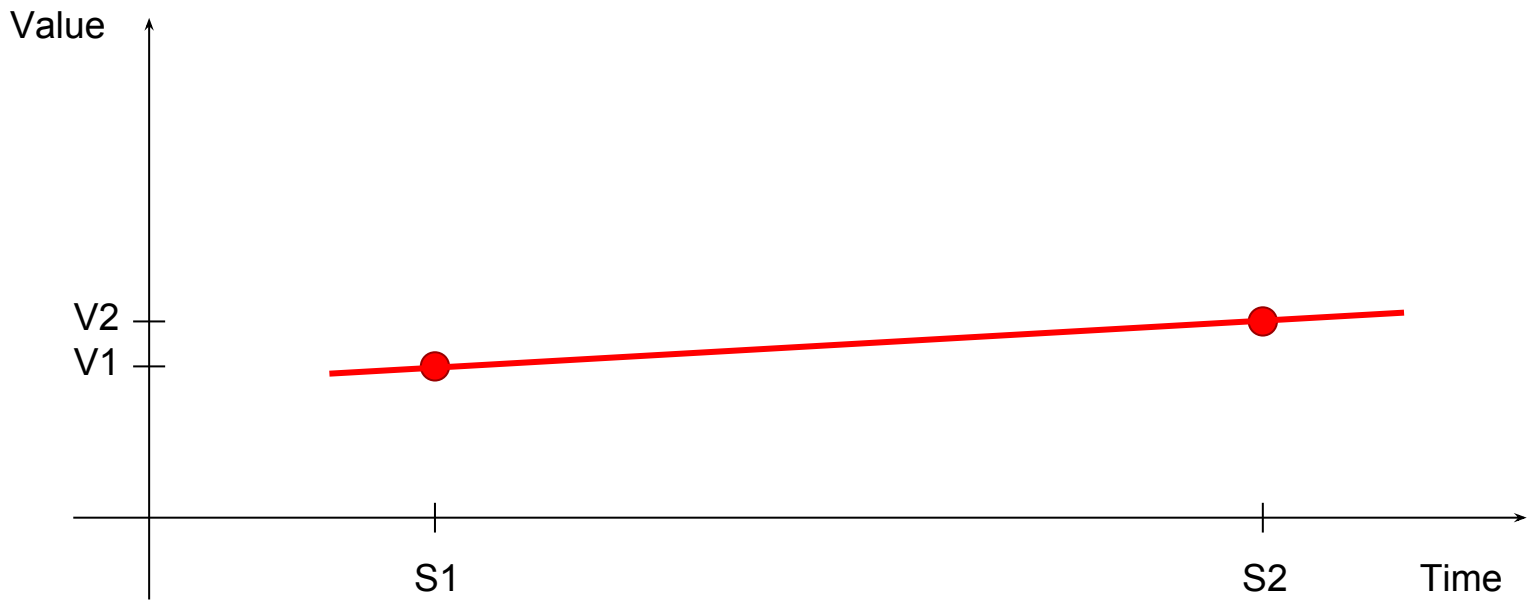
Snapshot Set	Begin Snap Id	Begin Snap Time	End Snap Id	End Snap Time	Avg Active Users	Elapsed Time (min)	DB time (min)
1st	20198	02-Sep-16 00:00:39 (Fri)	20246	03-Sep-16 00:00:49 (Sat)	1.6	1,440.2	2,294.2
2nd	20534	09-Sep-16 00:00:38 (Fri)	20582	10-Sep-16 00:00:29 (Sat)	1.9	1,439.9	2,679.4
% Diff					17.0	0.0	16.8

Wait Events

- Ordered by absolute value of 'Diff' column of '% of DB time' descending (idle events last)

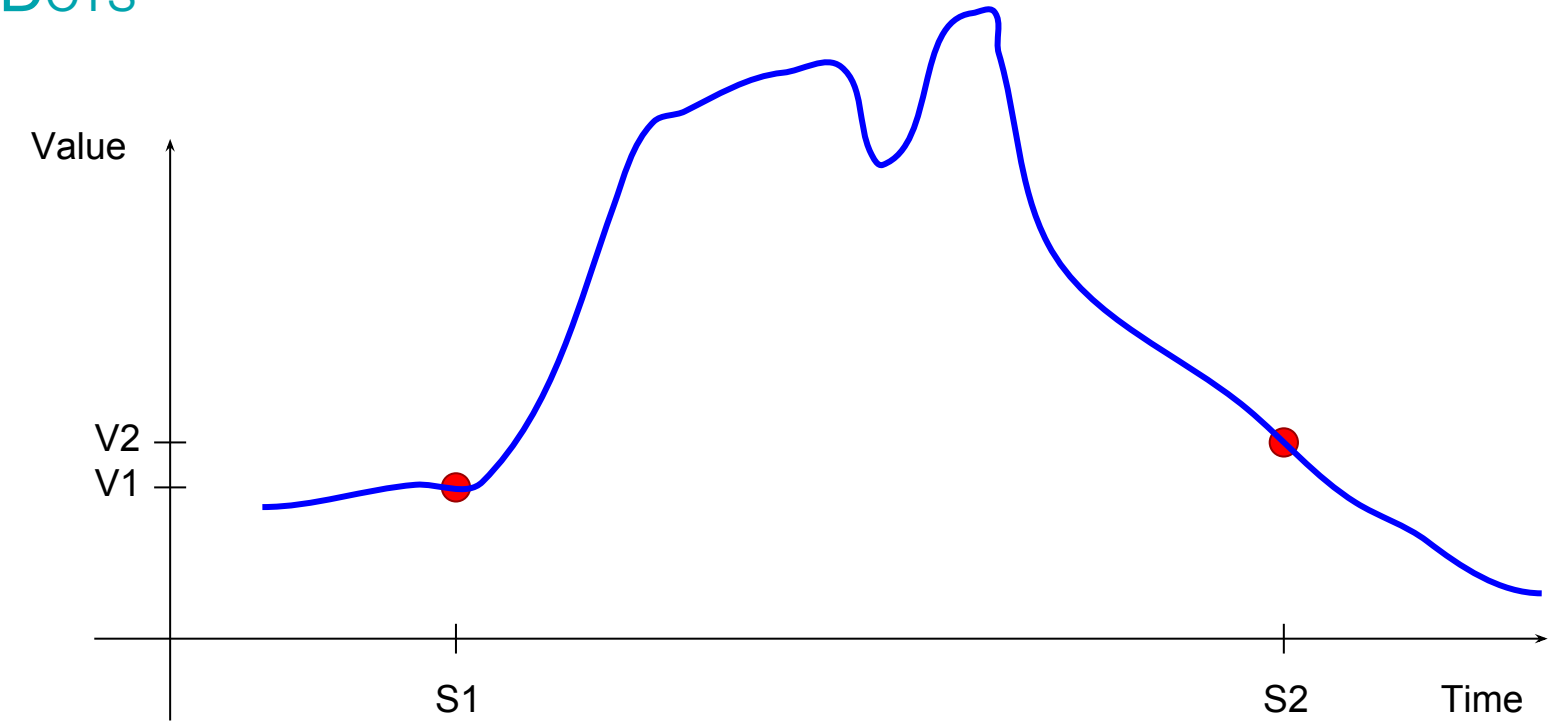
		% of DB time			# Waits/sec (Elapsed Time)			Total Wait Time (sec)			Avg Wait Time (ms)		
Event	Wait Class	1st	2nd	Diff	1st	2nd	%Diff	1st	2nd	%Diff	1st	2nd	%Diff
direct path write temp	User I/O	0.34	2.46	2.12	3.97	32.05	707.30	466.41	3,947.07	746.27	1.36	1.43	5.15
db flash cache single block physical read	User I/O	10.79	12.84	2.04	843.79	1,155.65	36.96	14,856.02	20,636.84	38.91	0.20	0.21	5.00
direct path read	User I/O	4.38	2.37	-2.01	78.59	41.39	-47.33	6,023.72	3,807.73	-36.79	0.89	1.06	19.10
cursor: pin S wait on X	Concurrency	1.90	3.60	1.70	0.02	0.02	0.00	2,612.99	5,782.13	121.28	1,786.05	3,391.28	89.88
log file sync	Commit	1.58	2.82	1.24	26.34	26.94	2.28	2,171.29	4,536.02	108.91	0.95	1.95	105.26
db flash cache multiblock physical read	User I/O	5.15	6.08	0.93	160.33	215.61	34.48	7,084.69	9,771.19	37.92	0.51	0.52	1.96
library cache lock	Concurrency	1.04	0.16	-0.89	0.01	0.01	0.00	1,437.74	251.74	-82.49	1,840.89	342.98	-81.37
direct path read temp	User I/O	0.27	1.05	0.78	7.07	40.54	473.41	377.80	1,695.79	348.86	0.62	0.48	-22.58
log buffer space	Configuration	0.08	0.73	0.65	0.02	0.24	1,100.00	108.97	1,177.77	980.82	58.71	57.22	-2.54
log file sequential read	System I/O	0.22	0.75	0.54	0.98	1.31	33.67	298.89	1,209.77	304.75	3.52	10.70	203.98
db file sequential read	User I/O	1.31	0.83	-0.47	29.25	21.66	-25.95	1,798.68	1,341.07	-25.4	0.71	0.72	1.41
control file sequential read	System I/O	1.95	1.61	-0.34	20.78	20.93	0.72	2,684.25	2,580.70	-3.86	1.49	1.43	-4.03
TCP Socket (KGAS)	Network	3.36	3.04	-0.32	0.73	0.77	5.48	4,630.87	4,891.18	5.62	73.77	73.82	0.07

2-Dots



What does it tell us about how things develop over time?

2-Dots



What does it tell us about how things develop over time?

Dots and AWR

- An AWR report describes the time interval between 2 snapshots i.e.
  - How much time was spent on CPU?
  - How many buffer gets did a particular query do?
  - How many Physical IOs did a top query do?
  - ... etc ...
- A single aggregated value describing the whole reporting interval, is insufficient for trend analysis

Is It Worth Mining AWR?

Trends in AWR

Mining Data Trends in AWR

- Each AWR snapshot represents a moment in time.
- Multitude of statistics are saved for each snapshot
- It's an obvious thing to graph these values!
- And the best about it is you pick the metric to graph!

Dots and AWR

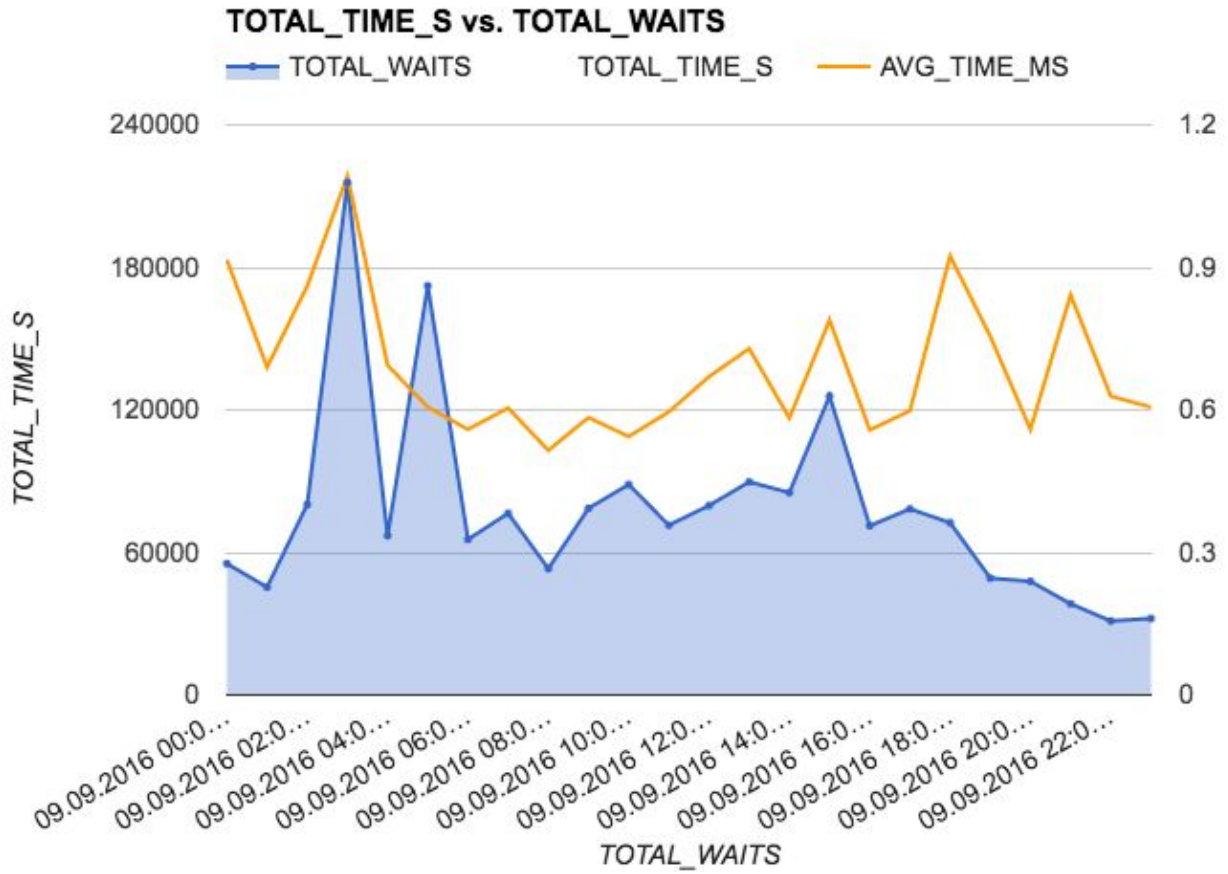
How Did the IO perform on September 9?

```
SQL> @awr_wait_trend.sql "db file sequential read" 10 1

TIME (SNAP)         EVENT_NAME                     TOTAL_WAITS   TOTAL_TIME_S    AVG_TIME_MS
------------------- ------------------------------ ------------- ------------- -------------
                                          ...
09.09.2016 00:00:00 db file sequential read               55302        50.670            .916
09.09.2016 01:00:00 db file sequential read               45372        31.303            .690
09.09.2016 02:00:00 db file sequential read               80171        68.995            .861
09.09.2016 03:00:00 db file sequential read              215704       235.555           1.092
09.09.2016 04:00:00 db file sequential read               67104        46.547            .694
09.09.2016 05:00:00 db file sequential read              172241       104.396            .606
                                          ...
09.09.2016 16:00:00 db file sequential read               71224        39.754            .558
09.09.2016 17:00:00 db file sequential read               78324        46.832            .598
09.09.2016 18:00:00 db file sequential read               72536        67.021            .924
09.09.2016 19:00:00 db file sequential read               49199        37.167            .755
09.09.2016 20:00:00 db file sequential read               47840        26.727            .559
```

Dots and AWR

How Did the IO perform on September 9?



Dots and AWR

Do You Still Remember This Picture?

Snapshot Set	Begin Snap Id	Begin Snap Time	End Snap Id	End Snap Time	Avg Active Users	Elapsed Time (min)	DB time (min)
1st	20198	02-Sep-16 00:00:39 (Fri)	20246	03-Sep-16 00:00:49 (Sat)	1.6	1,440.2	2,294.2
2nd	20534	09-Sep-16 00:00:38 (Fri)	20582	10-Sep-16 00:00:29 (Sat)	1.9	1,439.9	2,679.4
% Diff					17.0	0.0	16.8

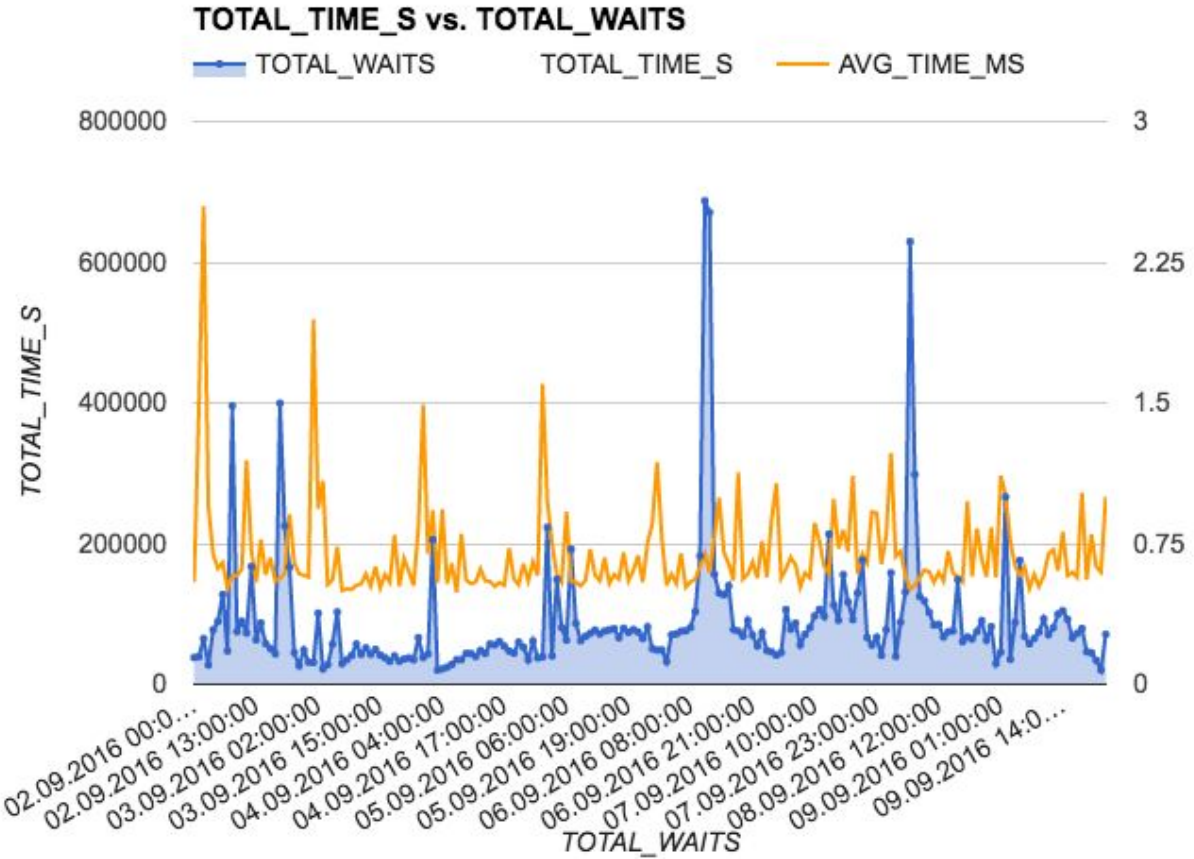
Wait Events

- Ordered by absolute value of 'Diff' column of '% of DB time' descending (idle events last)

		% of DB time			# Waits/sec (Elapsed Time)			Total Wait Time (sec)			Avg Wait Time (ms)		
Event	Wait Class	1st	2nd	Diff	1st	2nd	%Diff	1st	2nd	%Diff	1st	2nd	%Diff
direct path write temp	User I/O	0.34	2.46	2.12	3.97	32.05	707.30	466.41	3,947.07	746.27	1.36	1.43	5.15
db flash cache single block physical read	User I/O	10.79	12.84	2.04	843.79	1,155.65	36.96	14,856.02	20,636.84	38.91	0.20	0.21	5.00
direct path read	User I/O	4.38	2.37	-2.01	78.59	41.39	-47.33	6,023.72	3,807.73	-36.79	0.89	1.06	19.10
cursor: pin S wait on X	Concurrency	1.90	3.60	1.70	0.02	0.02	0.00	2,612.99	5,782.13	121.28	1,786.05	3,391.28	89.88
log file sync	Commit	1.58	2.82	1.24	26.34	26.94	2.28	2,171.29	4,536.02	108.91	0.95	1.95	105.26
db flash cache multiblock physical read	User I/O	5.15	6.08	0.93	160.33	215.61	34.48	7,084.69	9,771.19	37.92	0.51	0.52	1.96
library cache lock	Concurrency	1.04	0.16	-0.89	0.01	0.01	0.00	1,437.74	251.74	-82.49	1,840.89	342.98	-81.37
direct path read temp	User I/O	0.27	1.05	0.78	7.07	40.54	473.41	377.80	1,695.79	348.86	0.62	0.48	-22.58
log buffer space	Configuration	0.08	0.73	0.65	0.02	0.24	1,100.00	108.97	1,177.77	980.82	58.71	57.22	-2.54
log file sequential read	System I/O	0.22	0.75	0.54	0.98	1.31	33.67	298.89	1,209.77	304.75	3.52	10.70	203.98
db file sequential read	User I/O	1.31	0.83	-0.47	29.25	21.66	-25.95	1,798.68	1,341.07	-25.4	0.71	0.72	1.41
control file sequential read	System I/O	1.95	1.61	-0.34	20.78	20.93	0.72	2,684.25	2,580.70	-3.86	1.49	1.43	-4.03
TCP Socket (KGAS)	Network	3.36	3.04	-0.32	0.73	0.77	5.48	4,630.87	4,891.18	5.62	73.77	73.82	0.07

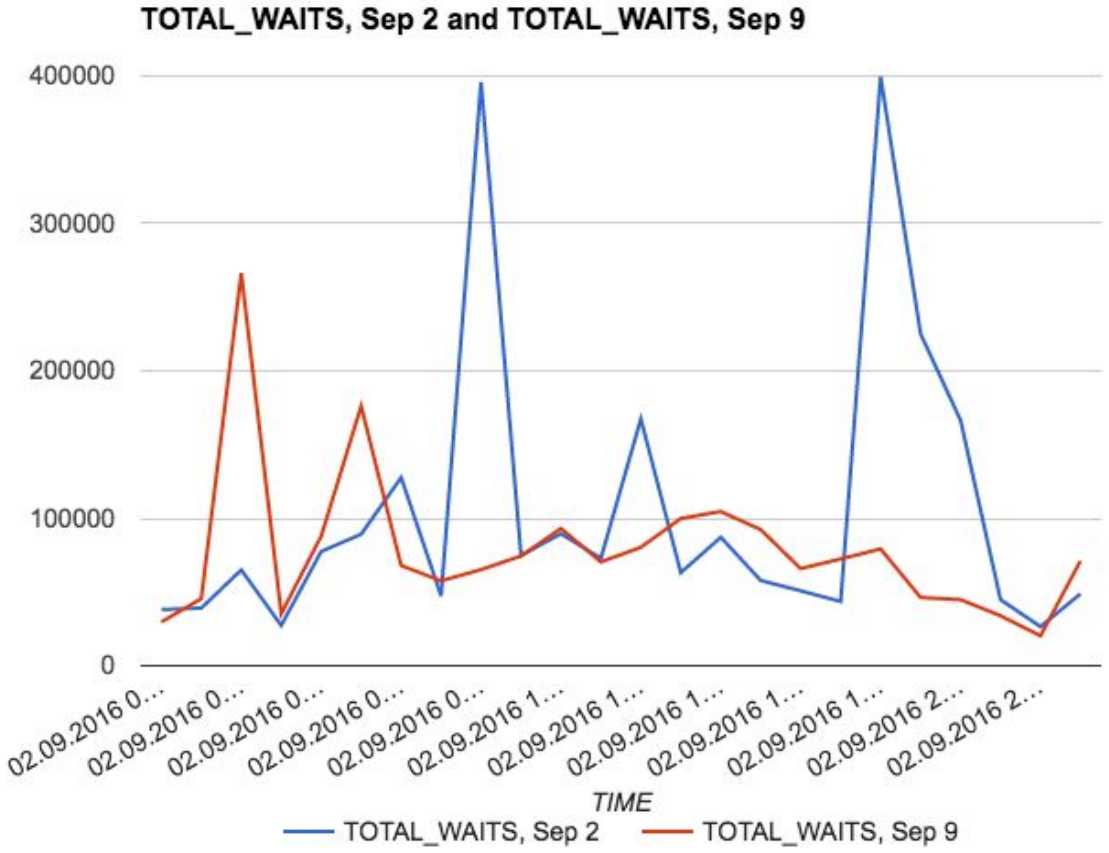
Dots and AWR

How Did the IO perform Between September 2 and 9?



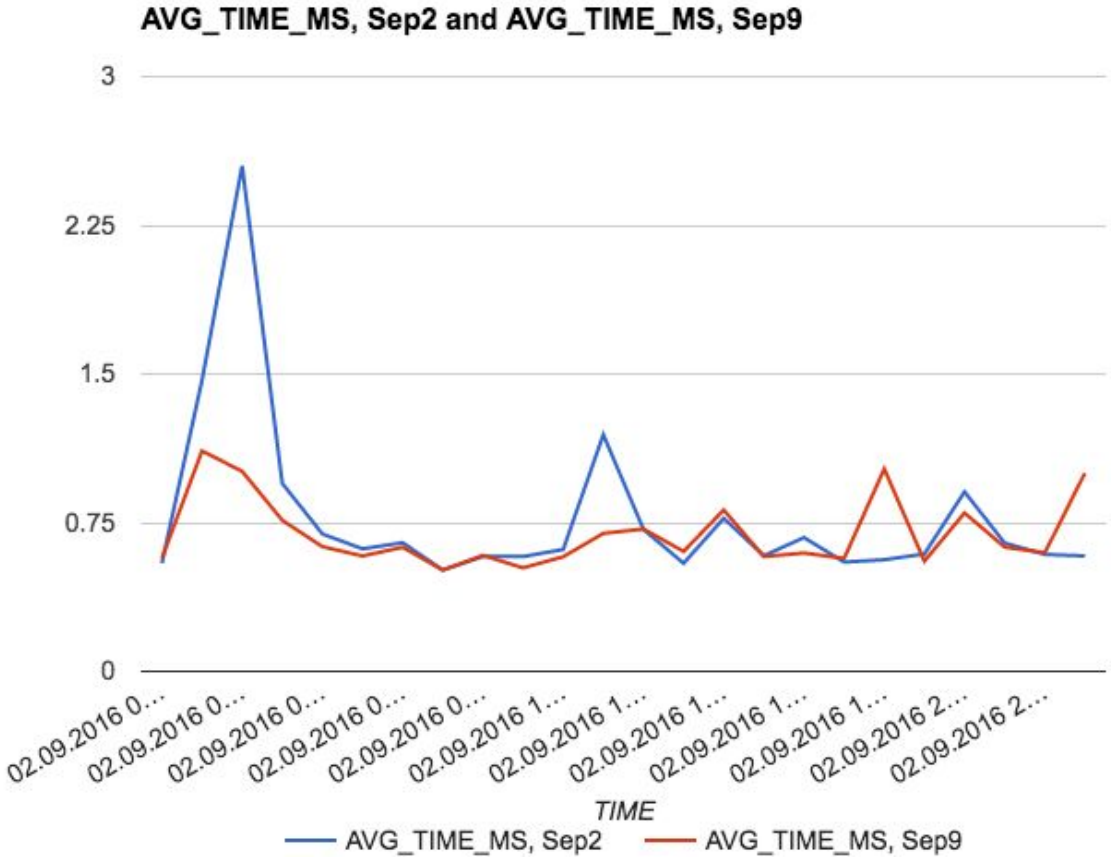
Dots and AWR

Total Waits Comparison Between September 2 and 9?



Dots and AWR

Response Time Comparison Between September 2 and 9?



How Did I Do It?

Extracting the “Past Behaviour” from AWR

STEP 1: PICK THE METRIC AND FIND THE CORRECT DBA_HIST_% VIEW

```
SELECT v.snap_id,
       v.value
FROM   dba_hist_sysstat v
WHERE  v.stat_name = 'user commits'
ORDER  BY v.snap_id;
```

```
   SNAP_ID      VALUE
---------- ----------
     12621     625766
     12622     626608
     12623     627853
....
```

Information on the DBA_HIST_views can be found in Oracle® Database Reference 12c Release 2 (12.2)
http://docs.oracle.com/database/122/REFRN/toc.htm

STEP 2: CALCULATE DELTAS

```
SELECT Lag(snap_id) over (ORDER BY v.snap_id) from_snap,
       v.snap_id to_snap,
       v.value - Lag(v.value) over (ORDER BY v.snap_id) delta_value
FROM   dba_hist_sysstat v
WHERE  v.stat_name = 'user commits'
ORDER  BY v.snap_id;
```

```
 FROM_SNAP    TO_SNAP DELTA_VALUE
---------- ---------- -----------
                12621
     12621      12622         842
     12622      12623        1245
...
```

STEP 3: VIEWER FRIENDLY DATES

```
SELECT Lag(s.end_interval_time)
         over (PARTITION BY s.dbid, s.startup_time ORDER BY v.snap_id) from_time,
       s.end_interval_time to_time,
       v.value - Lag(v.value)
                   over (PARTITION BY s.dbid, s.startup_time ORDER BY v.snap_id)
delta_value
FROM   dba_hist_sysstat v, dba_hist_snapshot s
WHERE  v.stat_name = 'user commits' AND s.snap_id = v.snap_id
ORDER  BY s.end_interval_time;
```

```
FROM_TIME                         TO_TIME                           DELTA_VALUE
--------------------------------- --------------------------------- -----------
                                  09-SEP-16 04.00.44.539000000 PM
09-SEP-16 04.00.44.539000000 PM 09-SEP-16 05.00.58.620000000 PM         842
09-SEP-16 05.00.58.620000000 PM 09-SEP-16 06.00.22.386000000 PM        1245
...
```

STEP 4: TIMELINE FILTERING

```
SELECT * FROM (
  SELECT Lag(s.end_interval_time)
           over (PARTITION BY s.dbid, s.startup_time ORDER BY v.snap_id) from_time,
         s.end_interval_time to_time,
         v.value - Lag(v.value)
                     over (PARTITION BY s.dbid, s.startup_time ORDER BY v.snap_id)
delta_value
        FROM   dba_hist_sysstat v, dba_hist_snapshot s
        WHERE  v.stat_name = 'user commits' AND s.snap_id = v.snap_id)
WHERE  from_time BETWEEN To_date('11092016', 'DDMMYYYY')
                         AND To_date('16092016', 'DDMMYYYY')
ORDER  BY from_time;
```

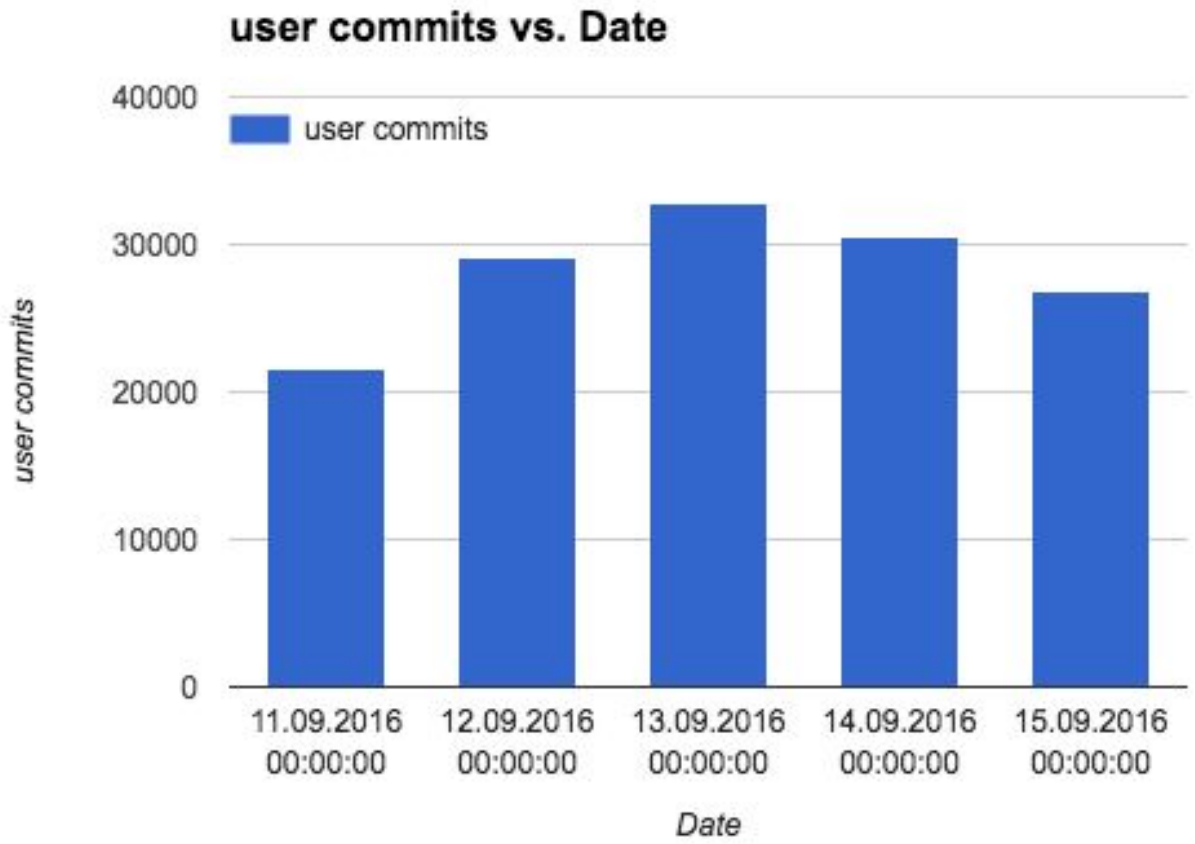
```
FROM_TIME                      TO_TIME                        DELTA_VALUE
------------------------------ ------------------------------ -----------
11-SEP-16 12.00.05.638000000 AM 11-SEP-16 01.00.23.855000000 AM         785
11-SEP-16 01.00.23.855000000 AM 11-SEP-16 02.00.40.073000000 AM        1057
...
15-SEP-16 11.00.35.078000000 PM 16-SEP-16 12.00.51.092000000 AM         831
```

STEP 5: AGGREGATION

```
SELECT Trunc(from_time, 'DD') t_from_time,
       SUM(delta_value)       sum_delta_value
FROM   (
SELECT Lag(s.end_interval_time)
         over (PARTITION BY s.dbid, s.startup_time ORDER BY v.snap_id) from_time,
       S.end_interval_time to_time,
       v.value - Lag(v.value)
         over (PARTITION BY s.dbid, s.startup_time ORDER BY v.snap_id) delta_value
FROM   dba_hist_sysstat v,dba_hist_snapshot s
WHERE  v.stat_name = 'user commits' AND s.snap_id = v.snap_id)
WHERE  from_time BETWEEN To_date('11092016', 'DDMMYYYY') AND To_date('16092016',
'DDMMYYYY')
GROUP  BY Trunc(from_time, 'DD') ORDER  BY t_from_time;

T_FROM_TIME                    SUM_DELTA_VALUE
------------------------------ --------------
11.09.2016 00:00:00                     21602
12.09.2016 00:00:00                     29173
...
```

STEP 6: VISUALIZATION



EXAMPLE 1

SQL Performance

Example 1

The problem

- Amazon RDS for Oracle
  - db.m3.xlarge (4 vCPU, 15GiB)
  - 11.2.0.4
- Alert from the client
  - Could you please take a look at the DB utilization in prod? We're seeing 100% utilization and 795 connections. Could you please let us know if any specific connection is causing high CPU utilization.

EXAMPLE 1
5MIN ASH

```
10:21:04 SQL> @ashtop session_id,sql_id "event is null" sysdate-5/25/60 sysdate

    Total
  Seconds     AAS %This   SESSION_ID SQL_ID        FIRST_SEEN          LAST_SEEN           ...
--------- ------- ------- ---------- ------------- ------------------- ------------------- ...
      285     1.0   12% |        738 4db73upm43ck1 2016-06-06 10:16:40 2016-06-06 10:21:27 ...
      285     1.0   12% |       1336 4db73upm43ck1 2016-06-06 10:16:40 2016-06-06 10:21:27 ...
      285     1.0   12% |       1358 4db73upm43ck1 2016-06-06 10:16:40 2016-06-06 10:21:27 ...
      284     1.0   12% |        745 4db73upm43ck1 2016-06-06 10:16:40 2016-06-06 10:21:27 ...
      284     1.0   12% |        882 4db73upm43ck1 2016-06-06 10:16:40 2016-06-06 10:21:27 ...
      284     1.0   12% |       2046 4db73upm43ck1 2016-06-06 10:16:40 2016-06-06 10:21:27 ...
       99      .3    4% |       1472 grbz54xqkr425 2016-06-06 10:17:04 2016-06-06 10:18:43 ...
       64      .2    3% |       1472 fy4c407vamaks 2016-06-06 10:18:52 2016-06-06 10:21:26 ...
       37      .1    2% |       2176               2016-06-06 10:18:29 2016-06-06 10:19:50 ...
       22      .1    1% |        977 7tdudtm4x03np 2016-06-06 10:16:40 2016-06-06 10:17:01 ...
```

Thank you, Tanel! http://blog.tanelpoder.com/files/scripts/ash/ashtop.sql

EXAMPLE 1
5MIN ASH

```
10:21:04 SQL> @ashtop session_id,sql_id "event is null" sysdate-5/25/60 sysdate

    Total
  Seconds     AAS %This   SESSION_ID SQL_ID        FIRST_SEEN          LAST_SEEN           ...
--------- ------- ------- ---------- ------------- ------------------- ------------------- ...
      285     1.0   12% |        738 4db73upm43ck1 2016-06-06 10:16:40 2016-06-06 10:21:27 ...
      285     1.0   12% |       1336 4db73upm43ck1 2016-06-06 10:16:40 2016-06-06 10:21:27 ...
      285     1.0   12% |       1358 4db73upm43ck1 2016-06-06 10:16:40 2016-06-06 10:21:27 ...
      284     1.0   12% |        745 4db73upm43ck1 2016-06-06 10:16:40 2016-06-06 10:21:27 ...
      284     1.0   12% |        882 4db73upm43ck1 2016-06-06 10:16:40 2016-06-06 10:21:27 ...
      284     1.0   12% |       2046 4db73upm43ck1 2016-06-06 10:16:40 2016-06-06 10:21:27 ...
       99      .3    4% |       1472 grbz54xqkr425 2016-06-06 10:17:04 2016-06-06 10:18:43 ...
       64      .2    3% |       1472 fy4c407vamaks 2016-06-06 10:18:52 2016-06-06 10:21:26 ...
       37      .1    2% |       2176               2016-06-06 10:18:29 2016-06-06 10:19:50 ...
       22      .1    1% |        977 7tdudtm4x03np 2016-06-06 10:16:40 2016-06-06 10:17:01 ...
```

Thank you, Tanel! http://blog.tanelpoder.com/files/scripts/ash/ashtop.sql

Example 1
SQL Performance

Just one query to address!
How hard can it be?

Example 1

The Execution plan

```
10:26:35 SQL> select * from table(dbms_xplan.display_awr('4db73upm43ck1',82218884));

Plan hash value: 82218884

-----------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name                       | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                            |       |       |       | 54163 (100)|          |
|   1 |  TEMP TABLE TRANSFORMATION         |                            |       |       |       |            |          |
|   2 |   LOAD AS SELECT                   |                            |       |       |       |            |          |
|   3 |    VIEW                            |                            |  179K|    13M|       | 10337   (1)| 00:02:05 |
|   4 |     HASH JOIN                      |                            |  179K|  9830K|  5800K| 10337   (1)| 00:02:05 |
|   5 |      TABLE ACCESS FULL             | LEARNING_IMPACT_BY_TOPIC   |  179K|  3686K|       |   794   (1)| 00:00:10 |
|   6 |      TABLE ACCESS FULL             | ACTIVE_USER_VIEWS_BY_TOPIC |  1543K|    51M|       |  5823   (1)| 00:01:10 |
|   7 |     VIEW                           |                            |     1 |    13 |       |     2   (0)| 00:00:01 |
|   8 |      FAST DUAL                     |                            |     1 |       |       |     2   (0)| 00:00:01 |
|   9 |   LOAD AS SELECT                   |                            |       |       |       |            |          |
|  10 |    VIEW                            |                            |  110K|  8415K|       | 10228   (1)| 00:02:03 |
|  11 |     HASH JOIN                      |                            |  110K|  6042K|  3568K| 10228   (1)| 00:02:03 |
|  12 |      TABLE ACCESS FULL             | LEARNING_IMPACT_BY_TOPIC   |  110K|  2265K|       |   794   (1)| 00:00:10 |
|  13 |      TABLE ACCESS FULL             | ACTIVE_USER_VIEWS_BY_TOPIC |  1543K|    51M|       |  5823   (1)| 00:01:10 |
|  14 |     VIEW                           |                            |     1 |    13 |       |     2   (0)| 00:00:01 |
|  15 |      FAST DUAL                     |                            |     1 |       |       |     2   (0)| 00:00:01 |
                                                    ...
```

1

Example 1

The Execution plan

```
                                                   ...
|  16 |   LOAD AS SELECT                        |                            |      |       |       |            |          |
|  17 |    INTERSECTION                         |                            |      |       |       |            |          |
|  18 |     SORT UNIQUE                         |                            | 179K|  7021K|  8472K|  2359   (1)| 00:00:29 |
|  19 |      VIEW                               |                            | 179K|  7021K|       |   518   (1)| 00:00:07 |
|  20 |       TABLE ACCESS FULL                 | SYS_TEMP_0FD9DBDFD_AAF67829 | 179K|  4388K|       |   518   (1)| 00:00:07 |
|  21 |     SORT UNIQUE                         |                            | 110K|  4315K|  5208K|  1452   (1)| 00:00:18 |
|  22 |      VIEW                               |                            | 110K|  4315K|       |   319   (1)| 00:00:04 |
|  23 |       TABLE ACCESS FULL                 | SYS_TEMP_0FD9DBDFE_AAF67829 | 110K|  2697K|       |   319   (1)| 00:00:04 |
|  24 |   LOAD AS SELECT                        |                            |      |       |       |            |          |
|  25 |    WINDOW BUFFER                        |                            |    1 |    27 |       |   860   (1)| 00:00:11 |
|  26 |     HASH GROUP BY                       |                            |    1 |    27 |       |   860   (1)| 00:00:11 |
|  27 |      VIEW                               |                            | 110K|  2913K|       |   856   (1)| 00:00:11 |
|  28 |       TABLE ACCESS FULL                 | SYS_TEMP_0FD9DBDFF_AAF67829 | 110K|  4315K|       |   856   (1)| 00:00:11 |
|  29 |   LOAD AS SELECT                        |                            |      |       |       |            |          |
|  30 |    WINDOW BUFFER                        |                            |   42 |   756 |       |   523   (2)| 00:00:07 |
|  31 |     HASH GROUP BY                       |                            |   42 |   756 |       |   523   (2)| 00:00:07 |
|  32 |      VIEW                               |                            | 179K|  3159K|       |   518   (1)| 00:00:07 |
|  33 |       TABLE ACCESS FULL                 | SYS_TEMP_0FD9DBDFD_AAF67829 | 179K|  4388K|       |   518   (1)| 00:00:07 |
|  34 |   LOAD AS SELECT                        |                            |      |       |       |            |          |
|  35 |    WINDOW BUFFER                        |                            |   42 |   756 |       |   322   (2)| 00:00:04 |
|  36 |     HASH GROUP BY                       |                            |   42 |   756 |       |   322   (2)| 00:00:04 |
|  37 |      VIEW                               |                            | 110K|  1942K|       |   319   (1)| 00:00:04 |
                                                   ...
```

2

EXAMPLE 1

THE EXECUTION PLAN

3

```
                                                    ...
|  38 |       TABLE ACCESS FULL                  | SYS_TEMP_0FD9DBDFE_AAF67829 |   110K|  2697K|       |   319   (1)| 00:00:04 |
|  39 |   FILTER                                 |                             |       |       |       |            |          |
|  40 |    SORT GROUP BY ROLLUP                  |                             |     1 |   579 |       | 28077   (1)| 00:05:37 |
|  41 |     MERGE JOIN OUTER                     |                             |    10 |  5790 |       | 28076   (1)| 00:05:37 |
|  42 |      MERGE JOIN OUTER                    |                             |    10 |  5790 |       | 20379   (1)| 00:04:05 |
|  43 |       HASH JOIN OUTER                    |                             |    10 |  5660 |       | 11818   (1)| 00:02:22 |
|  44 |        HASH JOIN OUTER                   |                             |     1 |   526 |       | 11705   (1)| 00:02:21 |
|  45 |         HASH JOIN OUTER                  |                             |     1 |   495 |       | 11383   (1)| 00:02:17 |
|  46 |          HASH JOIN OUTER                 |                             |     1 |   464 |       | 10860   (1)| 00:02:11 |
|  47 |           HASH JOIN OUTER                |                             |     1 |   407 |       |  6041   (1)| 00:01:13 |
|  48 |            HASH JOIN OUTER               |                             |     1 |   389 |       |  6039   (1)| 00:01:13 |
|  49 |             HASH JOIN OUTER              |                             |     1 |   371 |       |  6037   (1)| 00:01:13 |
|  50 |              HASH JOIN OUTER             |                             |     1 |   327 |       |  5266   (1)| 00:01:04 |
|  51 |               HASH JOIN OUTER            |                             |     1 |   296 |       |  1603   (1)| 00:00:20 |
|  52 |                VIEW                      |                             |     1 |   148 |       |   802   (1)| 00:00:10 |
|  53 |                 HASH GROUP BY            |                             |     1 |    75 |       |   802   (1)| 00:00:10 |
|  54 |                  HASH JOIN OUTER         |                             |     9 |   675 |       |   801   (1)| 00:00:10 |
|  55 |                   TABLE ACCESS BY INDEX ROWID | ACTIVE_USER_VIEWS_BY_TOPIC |     7 |   217 |       |     8   (0)| 00:00:01 |
|  56 |                    INDEX RANGE SCAN      | AUVBT_BY_PPID1_IX1          |     7 |       |       |     5   (0)| 00:00:01 |
|  57 |                    FAST DUAL             |                             |     1 |       |       |     2   (0)| 00:00:01 |
|  58 |                   TABLE ACCESS FULL      | LEARNING_IMPACT_BY_TOPIC    |   179K|  7723K|       |   792   (1)| 00:00:10 |
|  59 |                VIEW                      |                             |     1 |   148 |       |   801   (1)| 00:00:10 |
                                                    ...
```

EXAMPLE 1

THE EXECUTION PLAN

4

```
...
|  60 |                HASH GROUP BY                |                              |     1 |    75 |       |   801   (1)| 00:00:10 |
|  61 |                 HASH JOIN OUTER             |                              |     8 |   600 |       |   800   (1)| 00:00:10 |
|  62 |                  TABLE ACCESS BY INDEX ROWID | ACTIVE_USER_VIEWS_BY_TOPIC   |     7 |   217 |       |     8   (0)| 00:00:01 |
|  63 |                   INDEX RANGE SCAN           | AUVBT_BY_PPID1_IX1           |     7 |       |       |     5   (0)| 00:00:01 |
|  64 |                    FAST DUAL                 |                              |     1 |       |       |     2   (0)| 00:00:01 |
|  65 |                  TABLE ACCESS FULL          | LEARNING_IMPACT_BY_TOPIC     |  110K|  4747K|       |   792   (1)| 00:00:10 |
|  66 |               VIEW                          |                              |     1 |    31 |       |  3663   (1)| 00:00:44 |
|  67 |                HASH JOIN                    |                              |     1 |    62 |       |  3663   (1)| 00:00:44 |
|  68 |                 VIEW                        |                              |     1 |    31 |       |  1658   (1)| 00:00:20 |
|  69 |                  HASH GROUP BY              |                              |     1 |    96 |       |  1658   (1)| 00:00:20 |
|  70 |                   HASH JOIN                 |                              |     1 |    96 |       |  1657   (1)| 00:00:20 |
|  71 |                    NESTED LOOPS             |                              |     1 |    75 |       |   863   (1)| 00:00:11 |
|  72 |                     NESTED LOOPS            |                              |     7 |    75 |       |   863   (1)| 00:00:11 |
|  73 |                      VIEW                   |                              |     1 |    40 |       |   856   (1)| 00:00:11 |
|  74 |                       TABLE ACCESS FULL     | SYS_TEMP_0FD9DBDFF_AAF67829  |     1 |    40 |       |   856   (1)| 00:00:11 |
|  75 |                      INDEX RANGE SCAN       | AUVBT_BY_PPID1_IX1           |     7 |       |       |     4   (0)| 00:00:01 |
|  76 |                       FAST DUAL             |                              |     1 |       |       |     2   (0)| 00:00:01 |
|  77 |                     TABLE ACCESS BY INDEX ROWID| ACTIVE_USER_VIEWS_BY_TOPIC |     1 |    35 |       |     7   (0)| 00:00:01 |
|  78 |                    TABLE ACCESS FULL        | LEARNING_IMPACT_BY_TOPIC     |  179K|  3686K|       |   794   (1)| 00:00:10 |
|  79 |                 VIEW                        |                              |     1 |    31 |       |  2004   (1)| 00:00:25 |
|  80 |                  HASH GROUP BY              |                              |     1 |    96 |       |  2004   (1)| 00:00:25 |
|  81 |                   HASH JOIN                 |                              |     1 |    96 |       |  2003   (1)| 00:00:25 |
...
```

Example 1

The Execution plan

5

```
...
|  82 |                 NESTED LOOPS                |                            |     1 |    75 |       |  1209   (1)| 00:00:15 |
|  83 |                  NESTED LOOPS               |                            |   987 |    75 |       |  1209   (1)| 00:00:15 |
|  84 |                   VIEW                      |                            |     1 |    40 |       |   856   (1)| 00:00:11 |
|  85 |                    TABLE ACCESS FULL        | SYS_TEMP_0FD9DBDFF_AAF67829|     1 |    40 |       |   856   (1)| 00:00:11 |
|  86 |                   INDEX RANGE SCAN          | AUVBT_BY_PPID1_IX1         |   987 |       |       |     4   (0)| 00:00:01 |
|  87 |                    FAST DUAL                |                            |     1 |       |       |     2   (0)| 00:00:01 |
|  88 |                  TABLE ACCESS BY INDEX ROWID| ACTIVE_USER_VIEWS_BY_TOPIC |     1 |    35 |       |   353   (0)| 00:00:05 |
|  89 |                 TABLE ACCESS FULL           | LEARNING_IMPACT_BY_TOPIC   |  110K|  2265K|       |   794   (1)| 00:00:10 |
|  90 |               VIEW                          |                            |    41 |  1804 |       |   771   (1)| 00:00:10 |
|  91 |                HASH GROUP BY                |                            |    41 |  1886 |       |   771   (1)| 00:00:10 |
|  92 |                 HASH JOIN                   |                            |   157 |  7222 |       |   770   (1)| 00:00:10 |
|  93 |                  TABLE ACCESS BY INDEX ROWID| ACTIVE_USER_VIEWS_BY_TOPIC |   157 |  5338 |       |   354   (0)| 00:00:05 |
|  94 |                   INDEX RANGE SCAN          | AUVBT_BY_PPID1_IX1         |   987 |       |       |     5   (0)| 00:00:01 |
|  95 |                    FAST DUAL                |                            |     1 |       |       |     2   (0)| 00:00:01 |
|  96 |                  TABLE ACCESS FULL          | ASSESSMENTS                |  270K|  3166K|       |   415   (1)| 00:00:05 |
|  97 |              VIEW                           |                            |    42 |   756 |       |     2   (0)| 00:00:01 |
|  98 |               TABLE ACCESS FULL             | SYS_TEMP_0FD9DBE01_AAF67829|    42 |   756 |       |     2   (0)| 00:00:01 |
|  99 |             VIEW                            |                            |    42 |   756 |       |     2   (0)| 00:00:01 |
| 100 |              TABLE ACCESS FULL              | SYS_TEMP_0FD9DBE01_AAF67829|    42 |   756 |       |     2   (0)| 00:00:01 |
| 101 |            VIEW                             |                            |    42 |  2394 |       |  4819   (1)| 00:00:58 |
| 102 |             HASH JOIN OUTER                 |                            |    42 |  6678 |       |  4819   (1)| 00:00:58 |
| 103 |              HASH JOIN OUTER                |                            |    42 |  5376 |       |  4816   (1)| 00:00:58 |
...
```

Example 1

The Execution plan

6

```
                                          ...
| 104 |              HASH JOIN OUTER          |                             |   42 | 4074 |      | 4813   (1)| 00:00:58 |
| 105 |               MERGE JOIN OUTER        |                             |   42 | 2394 |      | 4698   (1)| 00:00:57 |
| 106 |                VIEW                   | VW_FOJ_0                    |   42 | 1848 |      |    4   (0)| 00:00:01 |
| 107 |                 HASH JOIN FULL OUTER  |                             |   42 | 2604 |      |    4   (0)| 00:00:01 |
| 108 |                  VIEW                 |                             |   42 | 1302 |      |    2   (0)| 00:00:01 |
| 109 |                   TABLE ACCESS FULL   | SYS_TEMP_0FD9DBE01_AAF67829 |   42 |  756 |      |    2   (0)| 00:00:01 |
| 110 |                  VIEW                 |                             |   42 | 1302 |      |    2   (0)| 00:00:01 |
| 111 |                   TABLE ACCESS FULL   | SYS_TEMP_0FD9DBE02_AAF67829 |   42 |  756 |      |    2   (0)| 00:00:01 |
| 112 |                BUFFER SORT            |                             |    1 |   13 |      | 4698   (1)| 00:00:57 |
| 113 |                 VIEW                  |                             |    1 |   13 |      |  112   (1)| 00:00:02 |
| 114 |                  VIEW                 |                             |    1 |   13 |      |  112   (1)| 00:00:02 |
| 115 |                   SORT AGGREGATE      |                             |    1 |   13 |      |           |          |
| 116 |                    VIEW               |                             | 110K| 1402K|      |  112   (1)| 00:00:02 |
| 117 |                     TABLE ACCESS FULL | SYS_TEMP_0FD9DBE00_AAF67829 | 110K| 2913K|      |  112   (1)| 00:00:02 |
| 118 |               VIEW                    |                             |    1 |   40 |      |  115   (4)| 00:00:02 |
| 119 |                WINDOW BUFFER          |                             |    1 |   40 |      |  115   (4)| 00:00:02 |
| 120 |                 HASH GROUP BY         |                             |    1 |   40 |      |  115   (4)| 00:00:02 |
| 121 |                  VIEW                 |                             | 110K| 4315K|      |  112   (1)| 00:00:02 |
| 122 |                   TABLE ACCESS FULL   | SYS_TEMP_0FD9DBE00_AAF67829 | 110K| 2913K|      |  112   (1)| 00:00:02 |
| 123 |              VIEW                     |                             |   30 |  930 |      |    3  (34)| 00:00:01 |
| 124 |               WINDOW BUFFER           |                             |   30 |  930 |      |    3  (34)| 00:00:01 |
| 125 |                HASH GROUP BY          |                             |   30 |  930 |      |    3  (34)| 00:00:01 |
                                          ...
```

Example 1

The Execution plan

...

```
| 126 |                  VIEW                            |                           |   42 |  1302 |         |     2   (0)| 00:00:01 |
| 127 |                   TABLE ACCESS FULL              | SYS_TEMP_0FD9DBE01_AAF67829 |   42 |   756 |         |     2   (0)| 00:00:01 |
| 128 |              VIEW                                |                           |   30 |   930 |         |     3  (34)| 00:00:01 |
| 129 |               WINDOW BUFFER                      |                           |   30 |   930 |         |     3  (34)| 00:00:01 |
| 130 |                HASH GROUP BY                     |                           |   30 |   930 |         |     3  (34)| 00:00:01 |
| 131 |                 VIEW                             |                           |   42 |  1302 |         |     2   (0)| 00:00:01 |
| 132 |                  TABLE ACCESS FULL               | SYS_TEMP_0FD9DBE02_AAF67829 |   42 |   756 |         |     2   (0)| 00:00:01 |
| 133 |           VIEW                                   |                           |   42 |  1302 |         |   523   (2)| 00:00:07 |
| 134 |            HASH GROUP BY                         |                           |   42 |   756 |         |   523   (2)| 00:00:07 |
| 135 |             VIEW                                 |                           |  179K|  3159K|         |   518   (1)| 00:00:07 |
| 136 |              TABLE ACCESS FULL                   | SYS_TEMP_0FD9DBDFD_AAF67829 |  179K|  4388K|         |   518   (1)| 00:00:07 |
| 137 |          VIEW                                    |                           |   42 |  1302 |         |   322   (2)| 00:00:04 |
| 138 |           HASH GROUP BY                          |                           |   42 |   756 |         |   322   (2)| 00:00:04 |
| 139 |            VIEW                                  |                           |  110K|  1942K|         |   319   (1)| 00:00:04 |
| 140 |             TABLE ACCESS FULL                    | SYS_TEMP_0FD9DBDFE_AAF67829 |  110K|  2697K|         |   319   (1)| 00:00:04 |
| 141 |         VIEW                                     |                           |  110K|  4315K|         |   112   (1)| 00:00:02 |
| 142 |          TABLE ACCESS FULL                       | SYS_TEMP_0FD9DBE00_AAF67829 |  110K|  2913K|         |   112   (1)| 00:00:02 |
| 143 |        BUFFER SORT                               |                           |    1 |    13 |         | 20379   (1)| 00:04:05 |
| 144 |         VIEW                                     |                           |    1 |    13 |         |   856   (1)| 00:00:11 |
| 145 |          VIEW                                    |                           |    1 |    13 |         |   856   (1)| 00:00:11 |
| 146 |           SORT AGGREGATE                         |                           |    1 |       |         |            |          |
| 147 |            VIEW                                  |                           |  110K|       |         |   856   (1)| 00:00:11 |
```

7

...

Example 1

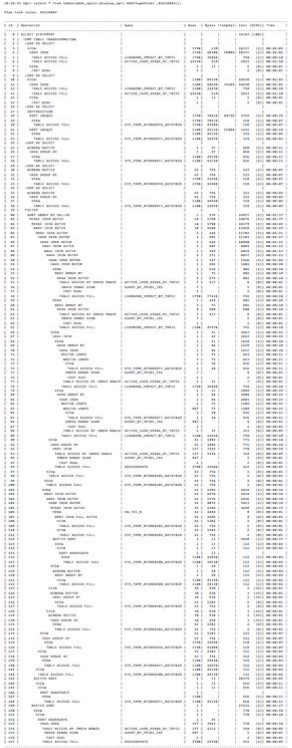
The Execution plan

8

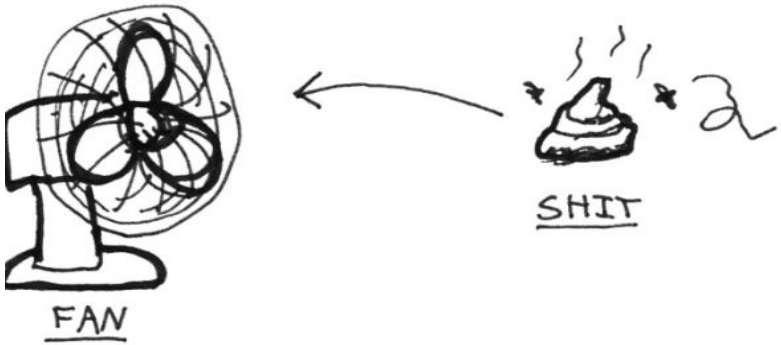
```
                                                                ...
| 148 |           TABLE ACCESS FULL                | SYS_TEMP_0FD9DBDFF_AAF67829 |   110K|  4315K|       |   856   (1)| 00:00:11 |
| 149 |      BUFFER SORT                           |                             |     1 |       |       | 27221   (1)| 00:05:27 |
| 150 |       VIEW                                 |                             |     1 |       |       |   770   (1)| 00:00:10 |
| 151 |        VIEW                                |                             |     1 |       |       |   770   (1)| 00:00:10 |
| 152 |         SORT AGGREGATE                     |                             |     1 |    25 |       |            |          |
| 153 |          HASH JOIN                         |                             |   157 |  3925 |       |   770   (1)| 00:00:10 |
| 154 |           TABLE ACCESS BY INDEX ROWID      | ACTIVE_USER_VIEWS_BY_TOPIC  |   157 |  2512 |       |   354   (0)| 00:00:05 |
| 155 |            INDEX RANGE SCAN                | AUVBT_BY_PPID1_IX1          |   987 |       |       |     5   (0)| 00:00:01 |
| 156 |             FAST DUAL                      |                             |     1 |       |       |     2   (0)| 00:00:01 |
| 157 |           TABLE ACCESS FULL                | ASSESSMENTS                 |   270K|  2374K|       |   415   (1)| 00:00:05 |
-----------------------------------------------------------------------------------------------------------------------------------------
```

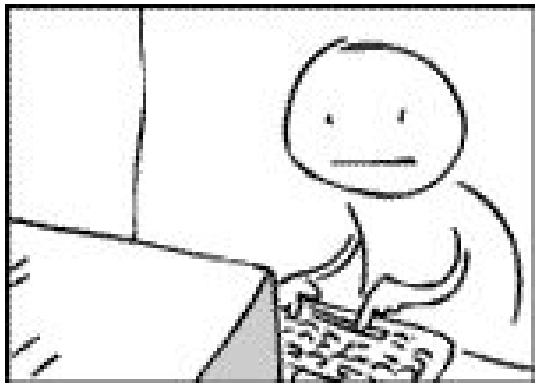
Example 1

The Execution plan -



#SHTF





Example 1
The Options

- What can one do in this case?
  - Tune the query?
  - Check / collect fresh statistics?
  - Flush the shared pool? (hoping it's bind variable peeking)
  - Run SQL Tuning Advisor?
  - or...

- How did the SQL statement execute in the past?

EXAMPLE 1

AWR_SQLID_PERF_TREND_BY_PLAN.SQL

```
select hss.instance_number inst,
       to_char(trunc(sysdate-&days_history+1)+trunc((cast(hs.begin_interval_time as
date)-(trunc(sysdate-&days_history+1)))*24/(&interval_hours))*(&interval_hours)/24,'dd.mm.yyyy
hh24:mi:ss') time,
    plan_hash_value,
    sum(hss.executions_delta) executions,
    round(sum(hss.elapsed_time_delta)/1000000,3) elapsed_time_s,
    round(sum(hss.cpu_time_delta)/1000000,3) cpu_time_s,
...
from dba_hist_sqlstat hss, dba_hist_snapshot hs
where hss.sql_id='&sql_id' and hss.snap_id=hs.snap_id and
hs.begin_interval_time>=trunc(sysdate)-&days_history+1
group by hss.instance_number, trunc(sysdate-&days_history+1)+trunc((cast(hs.begin_interval_time
as date)-(trunc(sysdate-&days_history+1)))*24/(&interval_hours))*(&interval_hours)/24,
plan_hash_value
having sum(hss.executions_delta)>0
```

Example 1
SQL Performance

Looking at 5 days of performance data at once!

```
10:21:28 SQL> @awr_sqlid_perf_trend_by_plan.sql 4db73upm43ck1 5 24
```

TIME	PLAN_HASH_VALUE	EXECUTIONS	ELAPSED_TIME_S	CPU_TIME_S	BUFFER_GETS	DISK_READS
02.06.2016 00:00:00	1597670781	291	278.018	254.272	19787274.000	50740.000
03.06.2016 00:00:00	1597670781	205	204.755	189.479	14986545.000	71451.000
04.06.2016 00:00:00	1597670781	3	8.275	4.483	314594.000	15790.000
04.06.2016 00:00:00	4142332636	2	6.149	3.933	119766.000	47343.000
05.06.2016 00:00:00	82218884	47	11805.808	5766.836	1247372099.000	173254.000
06.06.2016 00:00:00	82218884	127	7216.393	3751.640	522088617.000	24677.000

A few columns have been removed from the outputs to improve the readability!

Example 1
SQL Performance

Looking at 5 days of performance data at once!

```
10:21:28 SQL> @awr_sqlid_perf_trend_by_plan.sql 4db73upm43ck1 5 24
```

TIME	PLAN_HASH_VALUE	EXECUTIONS	ELAPSED_TIME_S	CPU_TIME_S	BUFFER_GETS	DISK_READS
02.06.2016 00:00:00	1597670781	291	278.018	254.272	19787274.000	50740.000
03.06.2016 00:00:00	1597670781	205	204.755	189.479	14986545.000	71451.000
04.06.2016 00:00:00	1597670781	3	8.275	4.483	314594.000	15790.000
04.06.2016 00:00:00	4142332636	2	6.149	3.933	119766.000	47343.000
05.06.2016 00:00:00	82218884	47	11805.808	5766.836	1247372099.000	173254.000
06.06.2016 00:00:00	82218884	127	7216.393	3751.640	522088617.000	24677.000

A few columns have been removed from the outputs to improve the readability!

Example 1
SQL Plan Baseline from AWR

```
exec DBMS_SQLTUNE.CREATE_SQLSET(sqlset_name => 'CR1064802',
                                description  => 'Plan for sql_id 4db73upm43ck1');

DECLARE
  cur sys_refcursor;
BEGIN
  OPEN cur FOR SELECT VALUE(P) FROM
    table(dbms_sqltune.select_workload_repository(11434,12880,'sql_id=''4db73upm43ck1'' and
          plan_hash_value=1597670781' , NULL, NULL, NULL, NULL, NULL, NULL, 'ALL')) P;
  DBMS_SQLTUNE.LOAD_SQLSET(load_option=>'MERGE',sqlset_name => 'CR1064802', populate_cursor => cur);
  CLOSE cur;
END;

VARIABLE cnt NUMBER
EXECUTE :cnt := DBMS_SPM.LOAD_PLANS_FROM_SQLSET(
                sqlset_name  => 'CR1064802',
                basic_filter => 'sql_id=''4db73upm43ck1''');


-- use DBMS_SHARED_POOL.PURGE to flush the cursor
```

EXAMPLE 1
SQL PERFORMANCE

```
11:03:11 SQL>  @awr_sqlid_perf_trend_by_plan.sql 4db73upm43ck1 2 1
```

TIME	PLAN_HASH_VALUE	EXECUTIONS	ELAPSED_TIME_S	CP	BUFFER_GETS	DISK_READS
06.06.2016 00:00:00	82218884	2			425936.000	965.000
06.06.2016 05:00:00					290668.000	4439.000
06.06.2016 06:00:00				.897	45601072.000	79.000
06.06.2016 07:00:00				2.155	165446.000	3369.000
06.06.2016 08:00:00			9.241	9.098	900049.000	61.000
06.06.2016 09:00:00		54	7010.622	3554.801	444951266.000	2593.000
06.06.2016 10:00:00		15	10942.980	4653.979	586951399.000	2078.000
06.06.2016 10:00:00	1597670781	1	7.910	2.047	117577.000	28.000
06.06.2016 11:00:00	1597670781	1	1.028	1.008	117468.000	28.000

Yea, but ...
... what if the good plan is not captured in the AWR?

A few columns have been removed from the outputs to improve the readability!

Example 1

Preserving Good Execution plans

```
-- Create the procedure that will be used to collect the execution plans
create or replace procedure PERF.XXSTABILITY_CAPTURE_PLANS is
  cur sys_refcursor;
  cursor clist is with raw_sqlids as (select sql_id from v$sqlarea
                                      where sql_id in (select sql_id from v$active_session_history
                                                       where sample_time>=sysdate-1/24 and sql_plan_hash_value>0
                                                         and machine like '%-hesvc-app0%')
                                        and plan_hash_value>0 and executions>2
                                      union
                                      select sql_id from v$sqlarea
                                      where sql_id in (select sql_id from v$open_cursor where
                                                       sid in (select sid from v$session where machine like '%-hesvc-app0%'))
                                        and plan_hash_value>0 and executions>2 and parsing_schema_name not in
                                            ('ANONYMOUS','APEX_030200','APEX_040000','APEX_SSO','APPQOSSYS',
'CTXSYS','DBSNMP','DIP','EXFSYS','FLOWS_FILES','MDSYS','OLAPSYS','ORACLE_OCM','ORDDATA','ORDPLUGINS','ORDSYS','OUTLN','OWBSYS'
,'SI_INFORMTN_SCHEMA','SQLTXADMIN','SQLTXPLAIN','SYS','SYSMAN','SYSTEM','TRCANLZR','WMSYS','XDB','XS$NULL')),
                  sqlids as (select distinct ''''||sql_id||'''' sql_id, trunc((rownum-1)/200) rn from raw_sqlids)
                  select listagg(s.sql_id,',') within group (order by s.sql_id) as sql_id_filter from sqlids s group by s.rn;
  c clist%rowtype;
BEGIN
  for c in clist loop
      OPEN cur FOR SELECT VALUE(P) FROM table(dbms_sqltune.select_cursor_cache(basic_filter=>'sql_id
in('||c.sql_id_filter||')', attribute_list=>'ALL')) P;
      DBMS_SQLTUNE.LOAD_SQLSET(load_option=>'MERGE',sqlset_name => 'XXSTABILITY_EXECUTION_PLANS', populate_cursor => cur);
      CLOSE cur;
  end loop;
  Commit;
END;
```

EXAMPLE 1

PRESERVING EXECUTION PLANS

```
-- Create the Scheduler Job which will run on every hour's 47th minute
Begin
 Dbms_scheduler.create_job
 (job_name        => 'COLLECT_XXSTABILITY_PLANS',
  job_type        => 'STORED_PROCEDURE',
  job_action      => 'PERF.XXSTABILITY_CAPTURE_PLANS',
  start_date      => SYSDATE,
  repeat_interval => 'FREQ=HOURLY; INTERVAL=1; BYMINUTE=47;',
  enabled         => TRUE);
End;
/

-- Loading the baseline from SELECT_SQLSET for a specific sql_id and plan
VARIABLE cnt NUMBER
EXECUTE :cnt := DBMS_SPM.LOAD_PLANS_FROM_SQLSET( -
                sqlset_name  => 'XXSTABILITY_EXECUTION_PLANS', -
                basic_filter => 'sql_id=''dtzsc12fbbjk2'' and
plan_hash_value=2896891279');
```

EXAMPLE 2

IO performance

Example 2

The problem

- Oracle e-Business Suite
  - R12.2
  - 12.1.0.2
- Alert from the client
  - Starting from September 5th, everything has become much slower.
  - Entering a sales order can take up to 15 minutes instead of usual 15 seconds.
- Not enough detail to look at any specific process or query

EXAMPLE 2

AWR_SYS_TIME_MODEL_TREND.SQL

```
select to_char(time,'DD.MM.YYYY HH24:MI:SS') time, stat_name,  sum(delta_value) value_delta
from
    (select hss.snap_id,
        trunc(sysdate-&days_history+1)+trunc((cast(hs.end_interval_time as
date)-(trunc(sysdate-&days_history+1)))*24/(&interval_hours))*(&interval_hours)/24 time,
        stat_name, value,
        value-(lag(value,1) over(partition by hs.startup_time, stat_name order by hss.snap_id))
delta_value
    from dba_hist_sys_time_model hss, dba_hist_snapshot hs
    where hss.snap_id=hs.snap_id
        and hss.instance_number=hs.instance_number
        and hs.end_interval_time>=trunc(sysdate)-&days_history+1
        and hss.stat_name like '&stat_name')
group by time, stat_name
order by to_date(time,'DD.MM.YYYY HH24:MI:SS'), 1;
```

EXAMPLE 2
CPU OR WAITING?

```
SQL> @awr_sys_time_model_trend.sql "DB CPU" 14 24

TIME                 STAT_NAME                                VALUE_DELTA
-------------------- ---------------------------------------- -------------
                                        ...
31.08.2016 00:00:00 DB CPU                                    271839933025
01.09.2016 00:00:00 DB CPU                                    398156022996
02.09.2016 00:00:00 DB CPU                                    343376689897
03.09.2016 00:00:00 DB CPU                                    262729582269
04.09.2016 00:00:00 DB CPU                                     79584869551
05.09.2016 00:00:00 DB CPU                                     87671787128
06.09.2016 00:00:00 DB CPU                                    103383834430
07.09.2016 00:00:00 DB CPU                                     16642806685
08.09.2016 00:00:00 DB CPU                                    215983418676
09.09.2016 00:00:00 DB CPU                                    100324311282
10.09.2016 00:00:00 DB CPU                                     11631538710
```

We're spending less time on CPU, so it's likely we're spending more time somewhere else!

So What Are We Waiting On? (Wait Classes)

```
select * from (
select to_char(time,'DD.MM.YYYY HH24:MI:SS') time, wait_class, sum(delta_time_waited)/1000000
wait_time_s from
    (select hse.snap_id, trunc(sysdate-&days_history+1)+trunc((cast(hs.end_interval_time as
date)-(trunc(sysdate-&days_history+1)))*24/(&interval_hours))*(&interval_hours)/24 time,
      EVENT_NAME, WAIT_CLASS,
      TIME_WAITED_MICRO-(lag(TIME_WAITED_MICRO,1) over(partition by hs.STARTUP_TIME, EVENT_NAME
order by hse.snap_id)) delta_time_waited
   from DBA_HIST_SYSTEM_EVENT hse, DBA_HIST_SNAPSHOT hs
   where hse.snap_id=hs.snap_id
      and hs.end_interval_time>=trunc(sysdate)-&days_history+1)
group by time, event_name, wait_class)
pivot
(
 sum(wait_time_s) for wait_class in ('Administrative','Application','Cluster'
                                    ,'Commit','Concurrency','Configuration'
                                    ,'Network','Other','Queueing'
                                    ,'Scheduler','System I/O','User I/O','Idle')
)
order by to_date(time,'DD.MM.YYYY HH24:MI:SS');
```

EXAMPLE 2

SO WHAT ARE WE WAITING ON? (WAIT CLASSES)

```
SQL> @awr_wait_class_hist.sql 14 1

TIME                 'Application' 'Commit' 'Concurrency' 'Other' 'System I/O' 'User I/O'  'Idle'
------------------- ------------ -------- ------------ ------- ------------ ---------- -------
                                                ...
03.09.2016 19:00:00           11       16            1      79          238       2943 1191940
03.09.2016 20:00:00           15       18            2      96          205       3640 1284953
03.09.2016 21:00:00           13       25            1     102          439       3985 1155491
03.09.2016 22:00:00           47     7068          965    1147        10445      11709 1201118
03.09.2016 23:00:00           43    19698          512    2354        56763      16627 1119525
04.09.2016 01:00:00          500    50406         6114   14543       131447      79139 2224995
04.09.2016 04:00:00         1067    65711         4350   16350       165858     199986 3820512
04.09.2016 05:00:00           29    12118          766    2378        13428      55055 1160076
04.09.2016 06:00:00           32     2510           54    1242         7708      47570 1295052
04.09.2016 07:00:00           21      546            1     305         3139      42468 1299331
04.09.2016 08:00:00           18      265           16     200         1788      28516 1242703
04.09.2016 09:00:00           13      377            1     174         1777      13472 1194265
```

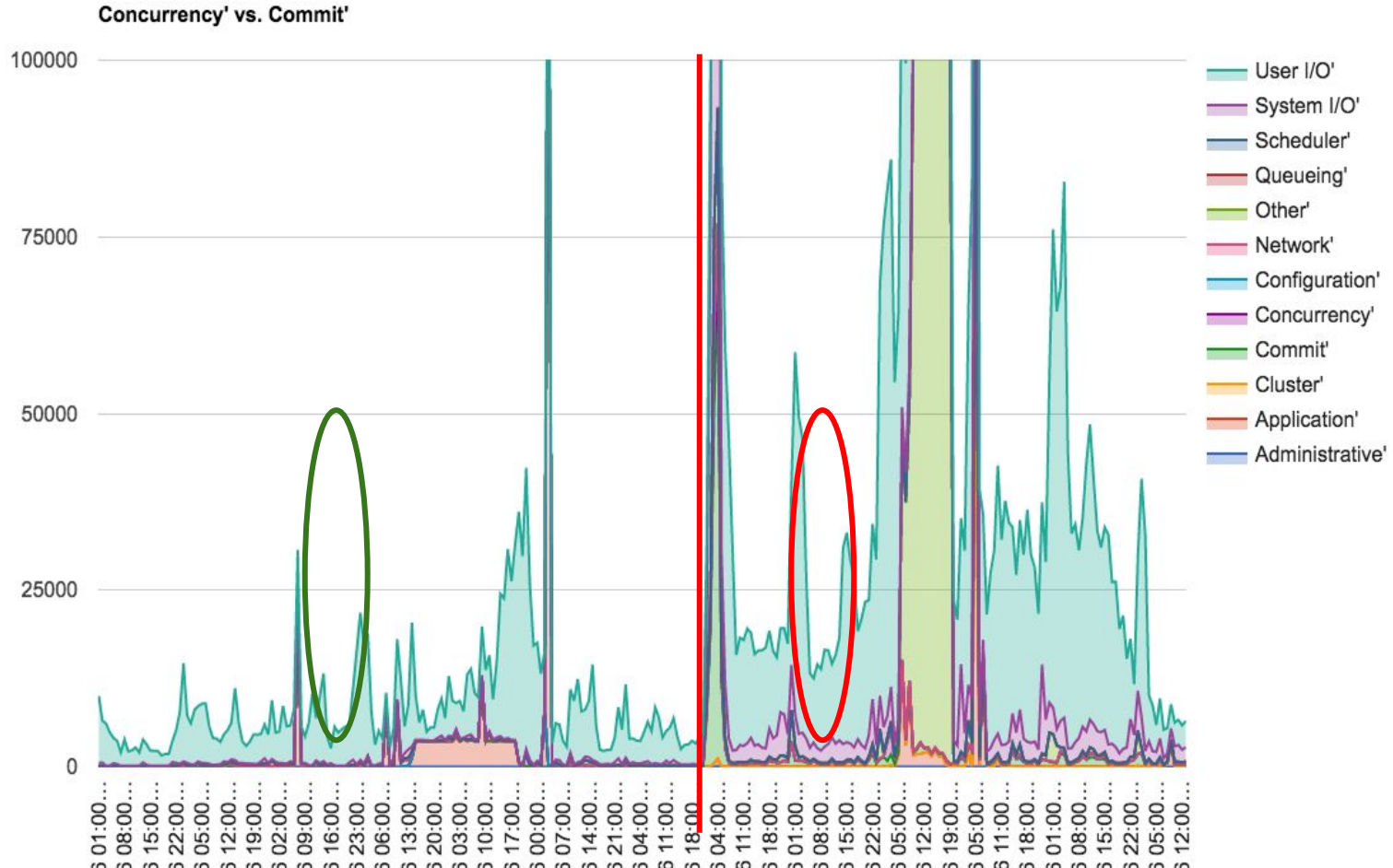
Example 2

So What Are We Waiting On? (Wait Classes)

```
SQL> @awr_wait_class_hist.sql 14 1

TIME                 'Application' 'Commit' 'Concurrency' 'Other' 'System I/O' 'User I/O'  'Idle'
------------------- ------------ -------- ------------ ------- ------------ ---------- -------
                                                ...
03.09.2016 19:00:00           11       16            1      79          238       2943 1191940
03.09.2016 20:00:00           15       18            2      96          205       3640 1284953
03.09.2016 21:00:00           13       25            1     102          439       3985 1155491
03.09.2016 22:00:00           47     7068          965    1147        10445      11709 1201118
03.09.2016 23:00:00           43    19698          512    2354        56763      16627 1119525
04.09.2016 01:00:00          500    50406         6114   14543       131447      79139 2224995
04.09.2016 04:00:00         1067    65711         4350   16350       165858     199986 3820512
04.09.2016 05:00:00           29    12118          766    2378        13428      55055 1160076
04.09.2016 06:00:00           32     2510           54    1242         7708      47570 1295052
04.09.2016 07:00:00           21      546            1     305         3139      42468 1299331
04.09.2016 08:00:00           18      265           16     200         1788      28516 1242703
04.09.2016 09:00:00           13      377            1     174         1777      13472 1194265
```

So What Are We Waiting On? (Wait Classes)



Example 2
"User I/O" is Misbehaving

- What could it be?
  - We're doing way more IOs
  - The IOs are much slower
  - … or both

AWR_WAIT_CLASS_EVENTS_TREND.SQL - WHAT HAPPENED TO THE USER I/O?

```
select to_char(time,'DD.MM.YYYY HH24:MI:SS') time, event_name, sum(delta_total_waits)
total_waits, round(sum(delta_time_waited/1000000),3) total_time_s,
round(sum(delta_time_waited)/decode(sum(delta_total_waits),0,null,sum(delta_total_waits))/1000,3
) avg_time_ms from
    (select hse.snap_id,
      trunc(sysdate-&days_history+1)+trunc((cast(hs.begin_interval_time as
date)-(trunc(sysdate-&days_history+1)))*24/(&interval_hours))*(&interval_hours)/24 time,
      EVENT_NAME,
      WAIT_CLASS,
      TOTAL_WAITS-(lag(TOTAL_WAITS,1) over(partition by hs.STARTUP_TIME, EVENT_NAME order by
hse.snap_id)) delta_total_waits,
      TIME_WAITED_MICRO-(lag(TIME_WAITED_MICRO,1) over(partition by hs.STARTUP_TIME, EVENT_NAME
order by hse.snap_id)) delta_time_waited
   from DBA_HIST_SYSTEM_EVENT hse, DBA_HIST_SNAPSHOT hs
   where hse.snap_id=hs.snap_id
      and hs.begin_interval_time>=trunc(sysdate)-&days_history+1 and hse.WAIT_CLASS like
'&wait_class')
group by time, event_name
order by 2, to_date(time,'DD.MM.YYYY HH24:MI:SS');
```

EXAMPLE 2

AWR_WAIT_CLASS_EVENTS_TREND.SQL - WHAT HAPPENED TO THE USER I/O?

```
SQL> @awr_wait_class_events_trend.sql "User I/O" 14 24

TIME                 EVENT_NAME                TOTAL_WAITS TOTAL_TIME_S AVG_TIME_MS
-------------------- ------------------------- ----------- ------------ -----------
                                        ...
30.08.2016 00:00:00 db file scattered read         710412     8074.224      11.366
31.08.2016 00:00:00 db file scattered read         686109     8779.227      12.796
01.09.2016 00:00:00 db file scattered read        3697362    79473.993      21.495
02.09.2016 00:00:00 db file scattered read         905800    12899.301      14.241
03.09.2016 00:00:00 db file scattered read         181632     5821.909      32.053
04.09.2016 00:00:00 db file scattered read         142537    44193.256     310.048
05.09.2016 00:00:00 db file scattered read         326866    32658.860      99.915
06.09.2016 00:00:00 db file scattered read         258082    29108.440     112.788
07.09.2016 00:00:00 db file scattered read         534720    44248.693      82.751
08.09.2016 00:00:00 db file scattered read         158424    22083.148     139.393
09.09.2016 00:00:00 db file scattered read         313153    47113.132     150.448
10.09.2016 00:00:00 db file scattered read         145106    14431.633      99.456
```

EXAMPLE 2

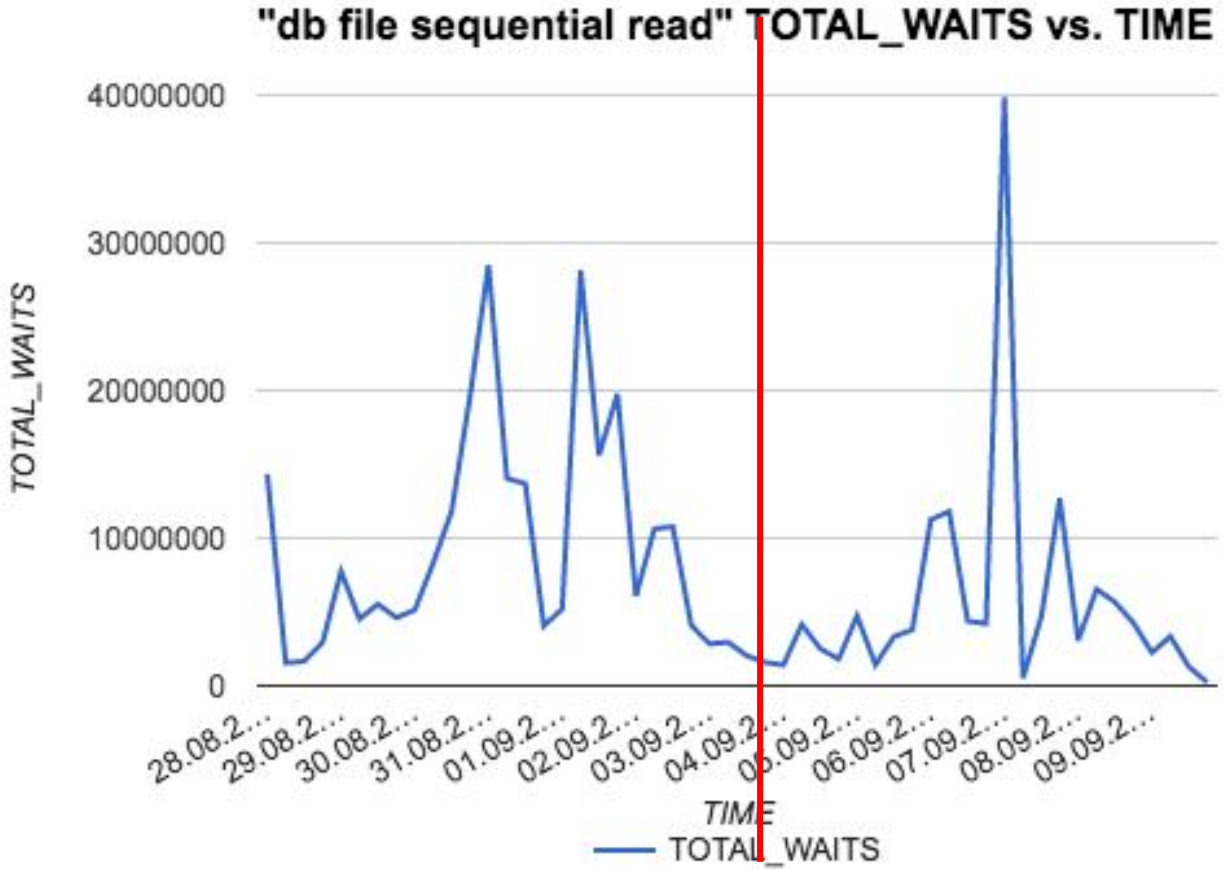
AWR_WAIT_CLASS_EVENTS_TREND.SQL - WHAT HAPPENED TO THE USER I/O?

```
SQL> @awr_wait_class_events_trend.sql "User I/O" 14 24

TIME                 EVENT_NAME                TOTAL_WAITS TOTAL_TIME_S AVG_TIME_MS
-------------------- ------------------------- ----------- ------------ -----------
                                        ...
30.08.2016 00:00:00 db file sequential read     37228593     89856.611       2.414
31.08.2016 00:00:00 db file sequential read     23431904     71298.237       3.043
01.09.2016 00:00:00 db file sequential read     26866886     82063.899       3.054
02.09.2016 00:00:00 db file sequential read     20273240     60593.329       2.989
03.09.2016 00:00:00 db file sequential read      5114798     28039.911       5.482
04.09.2016 00:00:00 db file sequential read      7142647    401434.020      56.202
05.09.2016 00:00:00 db file sequential read      8949913    292546.728      32.687
06.09.2016 00:00:00 db file sequential read     19490699    665357.848      34.137
07.09.2016 00:00:00 db file sequential read     16387403    541792.005      33.061
08.09.2016 00:00:00 db file sequential read      6989352    241881.608      34.607
09.09.2016 00:00:00 db file sequential read     12549290    453219.850      36.115
10.09.2016 00:00:00 db file sequential read      4823556    119185.469      24.709
```

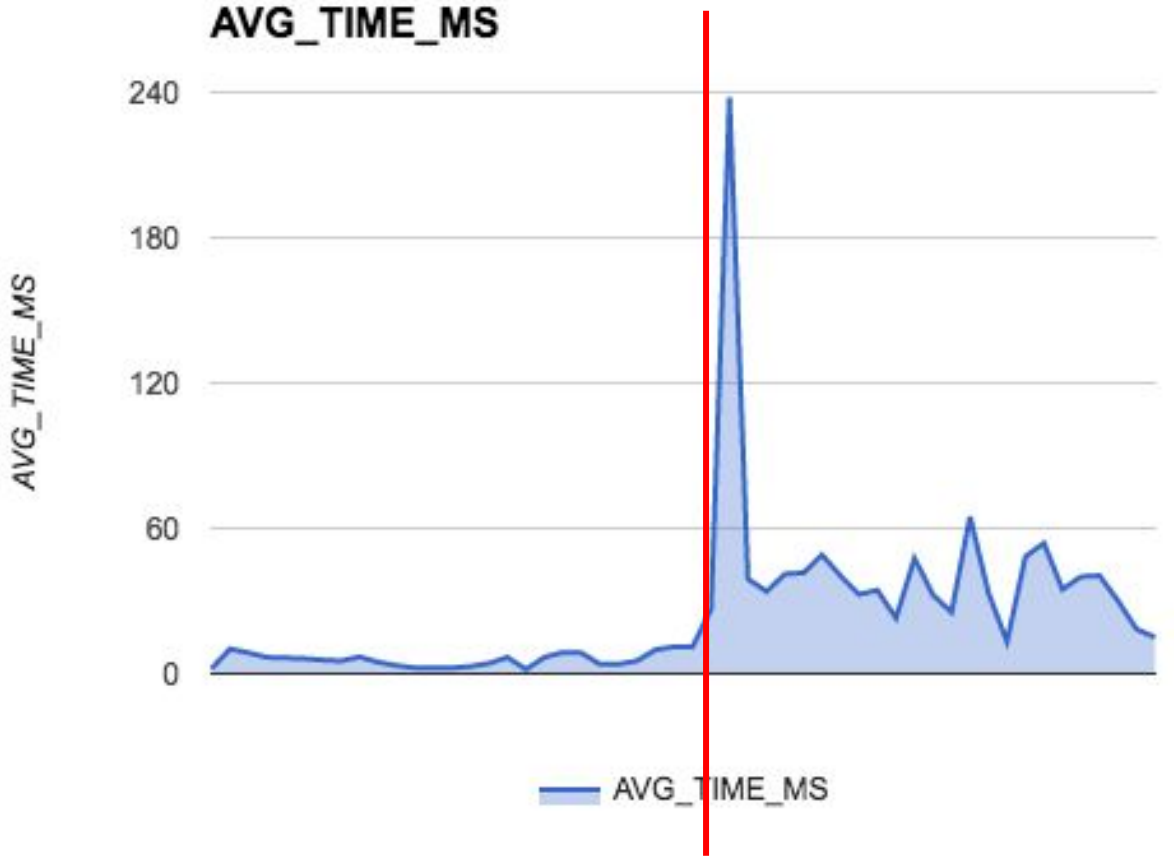
EXAMPLE 2

AWR_WAIT_CLASS_EVENTS_TREND.SQL - WHAT HAPPENED TO THE USER I/O?



Example 2

awr_wait_class_events_trend.sql - What Happened to The User I/O?



Example 2
The Root Cause

The Issue was passed on to the storage team.
They found 2 faulty drives.

EXAMPLE 3

Redo size

Example 3

The problem

- Oracle e-Business Suite
  - R12.1
  - 12.1.0.2
- Alert from the monitoring
  - We're running out of space on the mount point that's used as a long term storage for archived logs (10 days)

EXAMPLE 3

AWR_STAT_TREND.SQL

```
select to_char(time,'DD.MM.YYYY HH24:MI:SS') time, stat_name,  sum(delta_value) value_delta from
    (select hss.snap_id,
        trunc(sysdate-&days_history+1)+trunc((cast(hs.end_interval_time as
date)-(trunc(sysdate-&days_history+1)))*24/(&interval_hours))*(&interval_hours)/24 time,
        stat_name,
        value,
        value-(lag(value,1) over(partition by hs.startup_time, stat_name order by hss.snap_id))
delta_value
    from dba_hist_sysstat hss, dba_hist_snapshot hs
    where hss.snap_id=hs.snap_id
        and hss.instance_number=hs.instance_number
        and hs.begin_interval_time>=trunc(sysdate)-&days_history+1
        and hss.stat_name like ' ')
group by time, stat_name
order by 2, to_date(time,'DD.MM.YYYY HH24:MI:SS');
```

Example 3

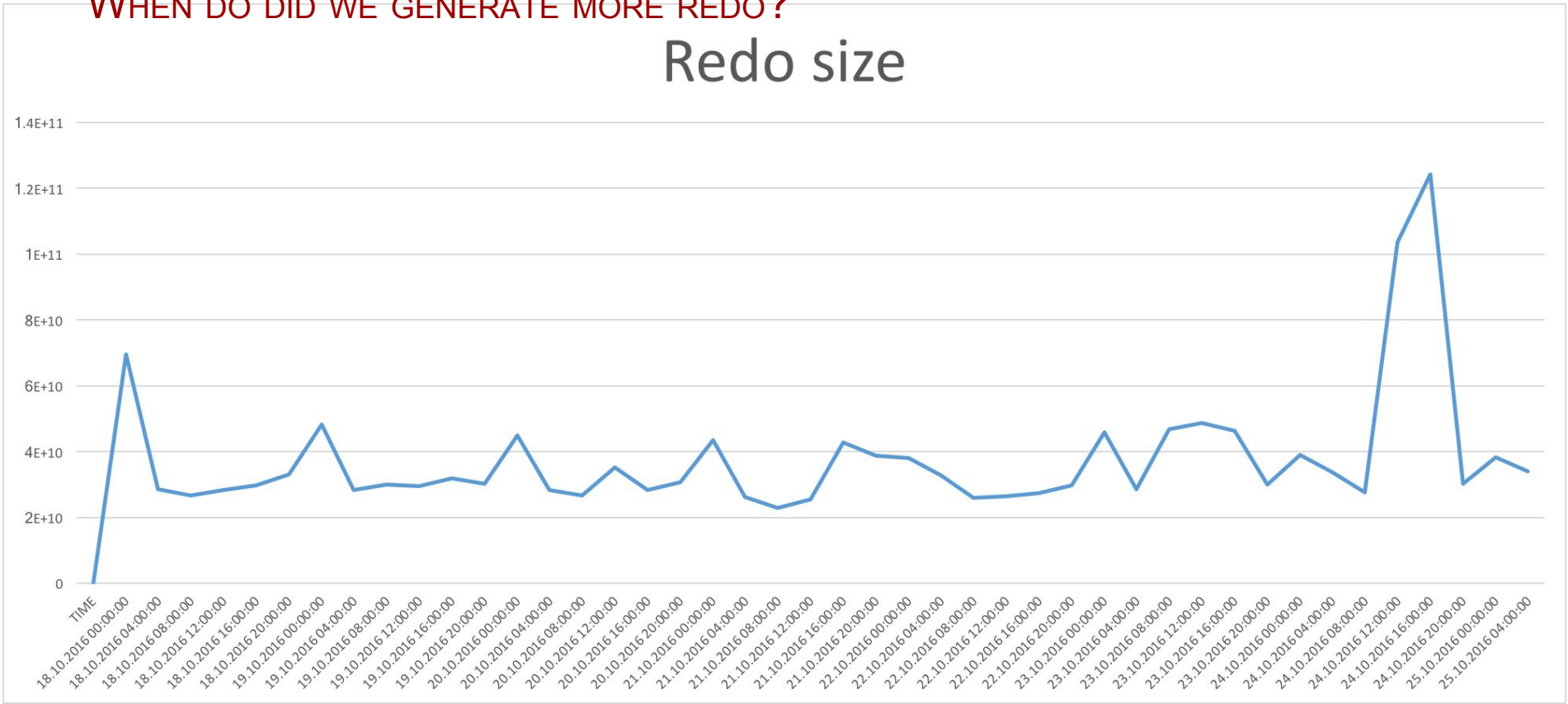
When do did we generate more redo?

```
SQL> @awr_stat_trend "redo size" 30 4

TIME                 STAT_NAME             VALUE_DELTA
-------------------- --------------- -----------------
                                ...
23.10.2016 08:00:00 redo size             46741701432
23.10.2016 12:00:00 redo size             48761724412
23.10.2016 16:00:00 redo size             46302390088
23.10.2016 20:00:00 redo size             29891530168
24.10.2016 00:00:00 redo size             39012726684
24.10.2016 04:00:00 redo size             33850511384
24.10.2016 08:00:00 redo size             27530642800
24.10.2016 12:00:00 redo size            103480430448
24.10.2016 16:00:00 redo size            124196709984
24.10.2016 20:00:00 redo size             30110908608
25.10.2016 00:00:00 redo size             38282203780
25.10.2016 04:00:00 redo size             34092919668
```

EXAMPLE 3

WHEN DO DID WE GENERATE MORE REDO?



Example 3

What caused the excessive redo?

```
EXEC DBMS_LOGMNR.ADD_LOGFILE('/arch_ret/prod/prod1_719619092_385764.arc');
EXEC DBMS_LOGMNR.ADD_LOGFILE('/arch_ret/prod/prod1_719619092_385765.arc');
EXEC DBMS_LOGMNR.ADD_LOGFILE('/arch_ret/prod/prod1_719619092_385766.arc');
EXEC DBMS_LOGMNR.ADD_LOGFILE('/arch_ret/prod/prod1_719619092_385767.arc');
EXEC DBMS_LOGMNR.ADD_LOGFILE('/arch_ret/prod/prod1_719619092_385768.arc');
EXEC DBMS_LOGMNR.ADD_LOGFILE('/arch_ret/prod/prod1_719619092_385769.arc');
EXEC DBMS_LOGMNR.ADD_LOGFILE('/arch_ret/prod/prod1_719619092_385770.arc');
EXEC DBMS_LOGMNR.ADD_LOGFILE('/arch_ret/prod/prod1_719619092_385771.arc');
EXEC DBMS_LOGMNR.ADD_LOGFILE('/arch_ret/prod/prod1_719619092_385772.arc');
EXEC DBMS_LOGMNR.ADD_LOGFILE('/arch_ret/prod/prod1_719619092_385773.arc');
EXEC DBMS_LOGMNR.ADD_LOGFILE('/arch_ret/prod/prod1_719619092_385774.arc');
EXEC DBMS_LOGMNR.ADD_LOGFILE('/arch_ret/prod/prod1_719619092_385775.arc');

EXEC DBMS_LOGMNR.START_LOGMNR(OPTIONS => DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG);
```

EXAMPLE 3

WHAT CAUSED THE EXCESSIVE REDO?

```
set pages 50000 lines 500 tab off time on serverout on
col owner for a30
col object_name for a30
col subobject_name for a30
col object_type for a30

select data_obj#, owner, object_name, subobject_name,
       count(*) REDO_CNT, sum(R_SIZE_B)/1024/1024 REDO_MB
from (select data_obj#, THREAD#, SEQUENCE#, RBABLK, RBABYTE,
             RBABLK*512+RBABYTE b_offset,
            (lead(RBABLK*512+RBABYTE) over (partition by RBASQN order by
RBABLK*512+RBABYTE))-(RBABLK*512+RBABYTE) R_SIZE_B
      from V$LOGMNR_CONTENTS) lc, dba_objects o
      where o.data_object_id(+)=lc.data_obj#
      group by data_obj#, owner, object_name, subobject_name order by 6;
```

What caused the excessive redo?

```
DATA_OBJ# OWNER       OBJECT_NAME                    SUBOBJECT_NAME      REDO_CNT    REDO_MB
--------- ----------- ------------------------------ ---------------- ---------- ----------
...
   4214022 AR          SYS_LOB0004214021C00002$$                          16997  55.844368
   4214007 AR          SYS_LOB0004214006C00002$$                          11991 60.0471535
    160758 APPLSYS     MO_GLOB_ORG_ACCESS_TMP_U1                         431892 80.7151947
    112443                                                               784710 142.132919
         0 CTXSYS      DR$DBO                                            859097 253.653015
    137676                                                              3774665 565.485004
    181691 APPS        XX_OM_PRICING_DETAILS_GTT_N100                   3840811 709.441868
   1026278 APPS        XX_OM_PRICING_DETAILS_GTT_NS                     3870531 772.384396
    181690 APPS        XX_OM_PRICING_DETAILS_GTT_N400                   3886342 817.994175
    181692 APPS        XX_OM_PRICING_DETAILS_GTT_N200                   3943281 882.903255
    880271 APPS        XX_OM_PRICING_DETAIL_GTT_N20                     4118952 1245.52386
    181689 APPS        XX_OM_PRICING_DETAILS_GTT_N300                   4238632 1464.44829

307 rows selected.

EXEC DBMS_LOGMNR.END_LOGMNR;
```

Example 3
The Root Cause

One of the concurrent programs inserting into the GTT
was executed much more often during the time of the issue

A) Added +append and set alter session set temp_undo_enabled=true
B) The frequency of the concurrent program was reduced
C) 4 of 6 indexes on the GTT were dropped

Thanks,
@davidmkurtz

EXAMPLE 3
THE ROOT CAUSE

Redo Size (M)

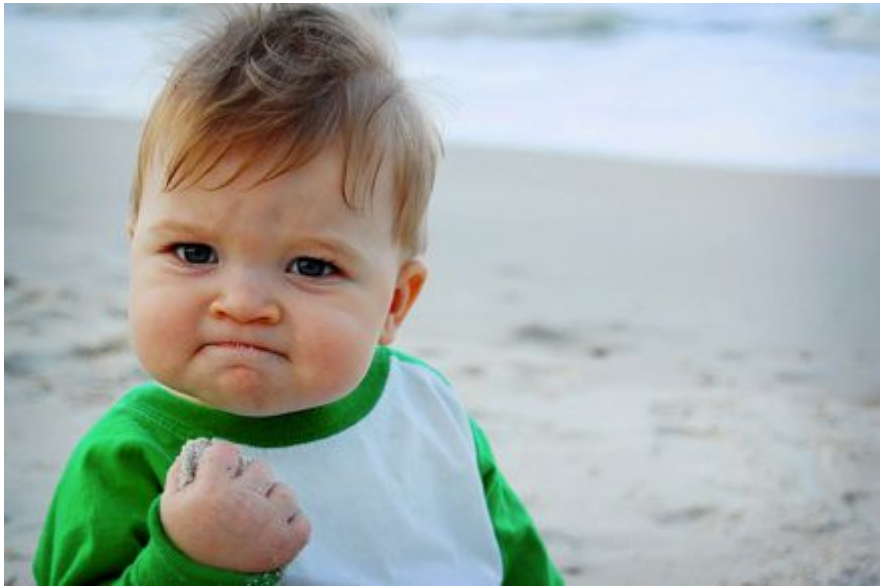
	Conventional load	Direct path load (+append)
Temporary Undo = False	674	200
Temporary Undo = True	13	**8**

SUMMARY

SUMMARY

- AWR reports are not good in displaying how performance changes over time.
- Many performance metrics stored in AWR can be:
  - Put on a timeline to reveal extra detail
  - Useful for Troubleshooting
  - Useful for Prediction
- Remember, you're still working with aggregated data

SUMMARY



AWR is a complex beast, but it’s not necessary to know much to start mining

Be Brave!
But buy the “Diagnostics Pack” licenses first!

The Scripts: http://bit.ly/MiningAWRv2

THANK YOU

@MarisDBA

Elsins@pythian.com