ASH Outliers: Detecting Unusual Events in Active Session History

John Beresniewicz, Oracle USA

Agenda

- ASH Fundamentals
- Wait Events and ASH
- Event Histograms
- Event Significance Levels
- SQL to Find Outliers in ASH
- Handling the GV$ problem

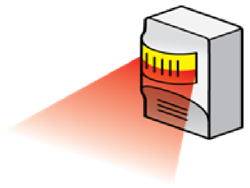
Motivating Use Case

- Unusually long wait events (outliers) suspected to trigger cascade-effect performance incidents
- RAC performance experts claim EM Top Activity not helpful as aggregation masks outlier events
- Can we see if ASH has sampled any such events?
  - If none observed does not mean they have not happened
    - However ASH sampling is biased to longer events

ASH Fundamentals

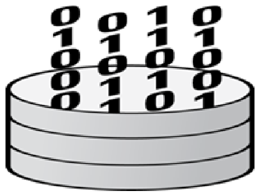
All ‘Active’ sessions captured every second

- Foregrounds and backgrounds are sampled
- Active foregrounds contribute to DB Time



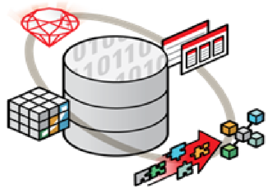
In-memory: *V$ACTIVE_SESSION_HISTORY*

- Sampling interval = 1 second
- Circular SGA buffer with latchless query access



On-disk: *DBA_HIST_ACTIVE_SESS_HISTORY*

- Sub-sampling interval = 10 seconds



ASH is a system-wide record of database activity

- A **FACT** table with multiple dimensions that help diagnose performance issues

ORACLE

ASH and DB Time

Active sessions contribute to DB Time

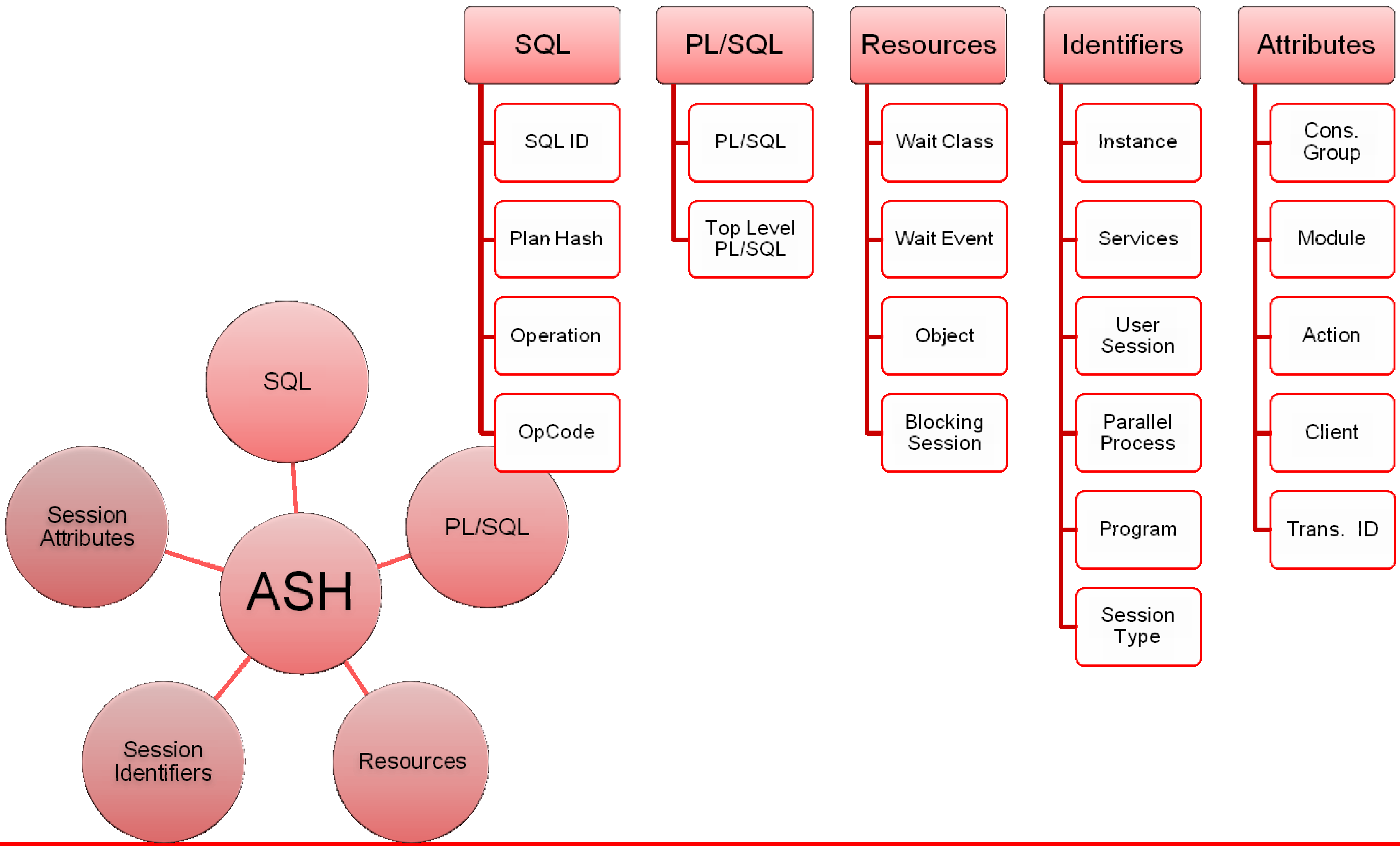
ASH samples active sessions

ASH Math = estimate DB Time by counting ASH samples

COUNT of ASH Samples = Total DB Time in seconds for that time interval

Group by over 70+ performance dimensions

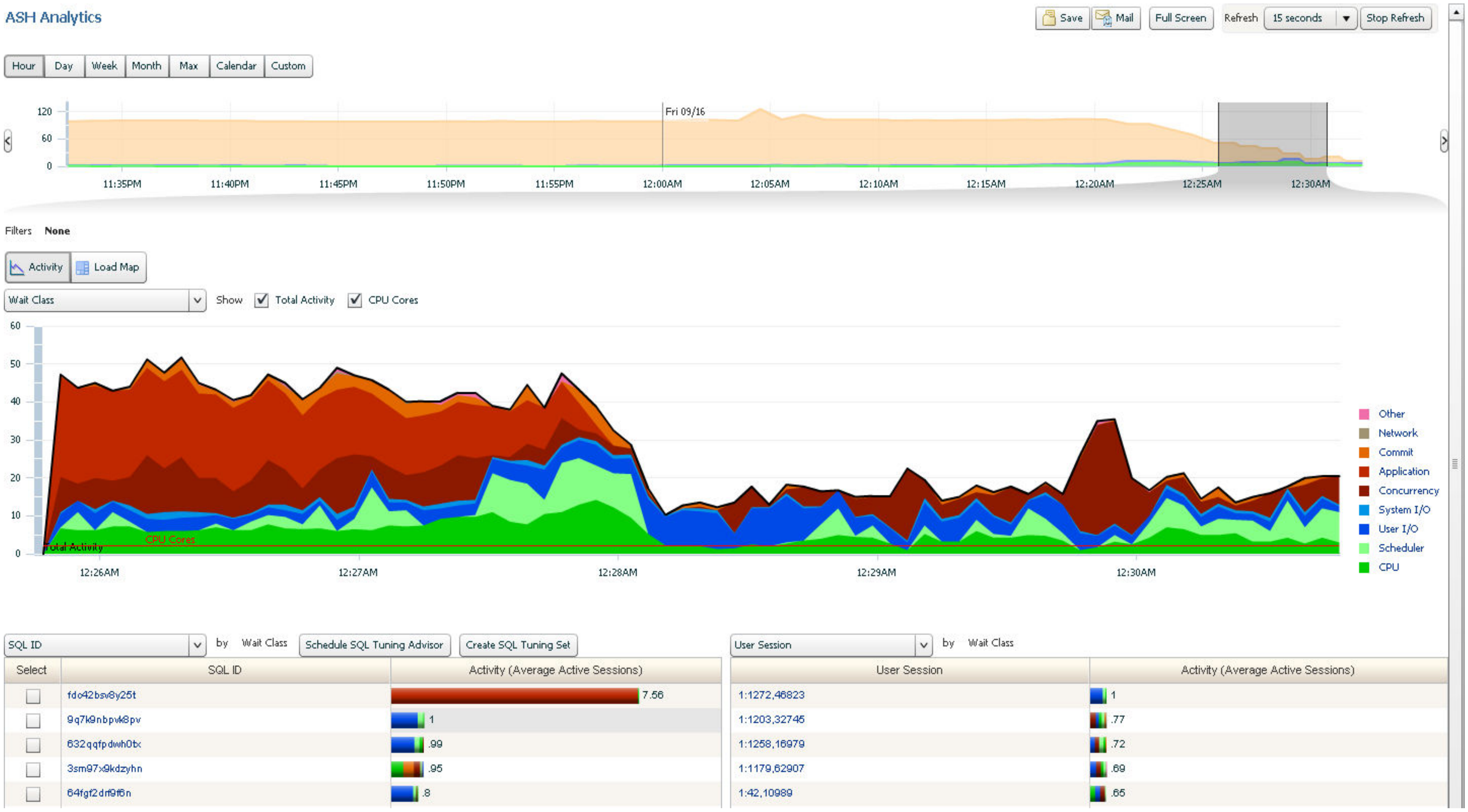
ASH Fact Table Dimensions



ASH Math

- COUNT(*) = DB TIME (secs)
  - Basic in-memory (V$ASH) formula
- The Kurtz Construct is nice
  - SUM(1) = DB Time (secs) for in-memory
  - SUM(10) = DB TIME (secs) for on-disk
- DB Time Method analysis:
  - Dimensional GROUP BY over COUNT(*)

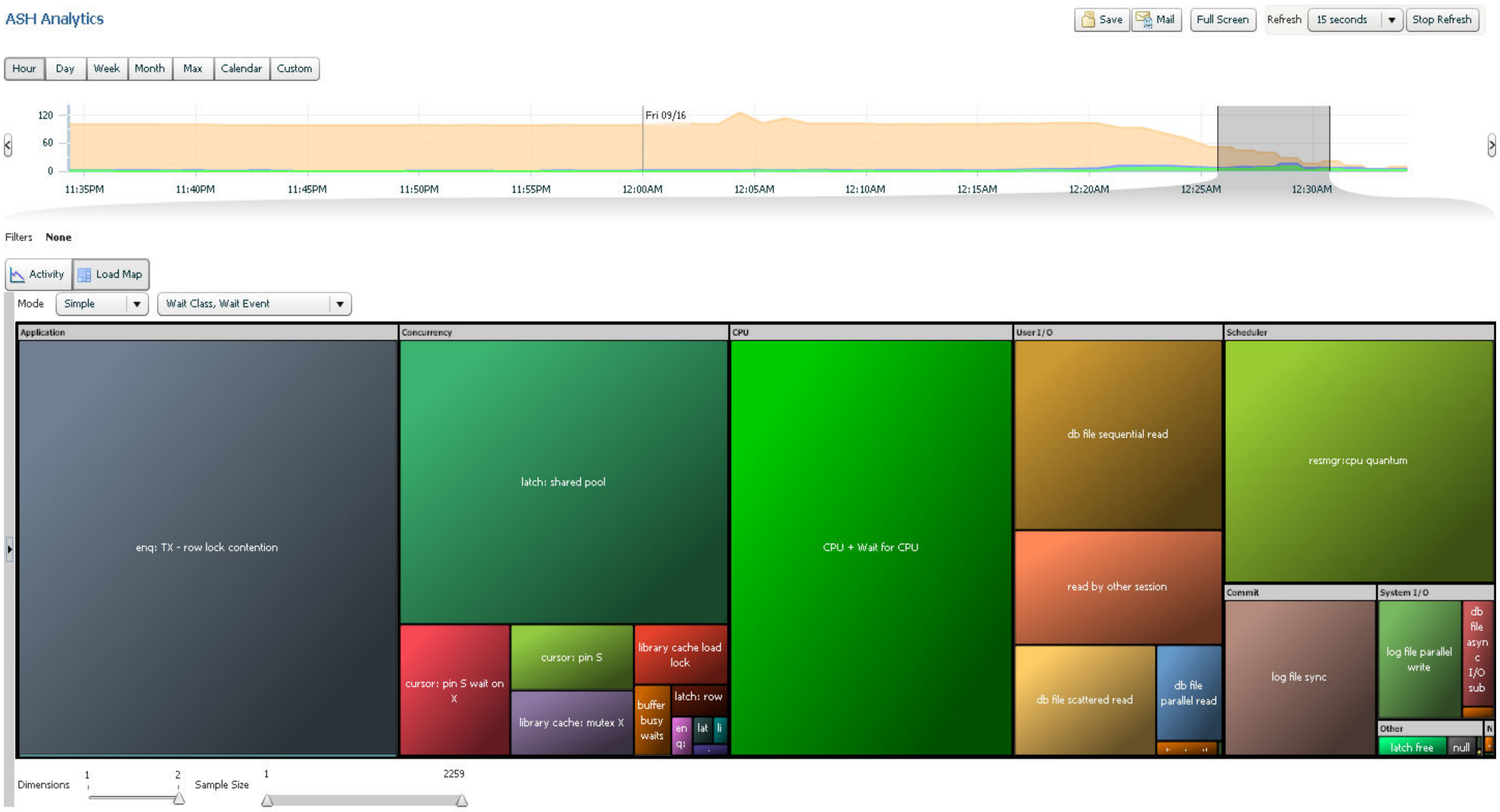
Good ASH Math: ASH Analytics



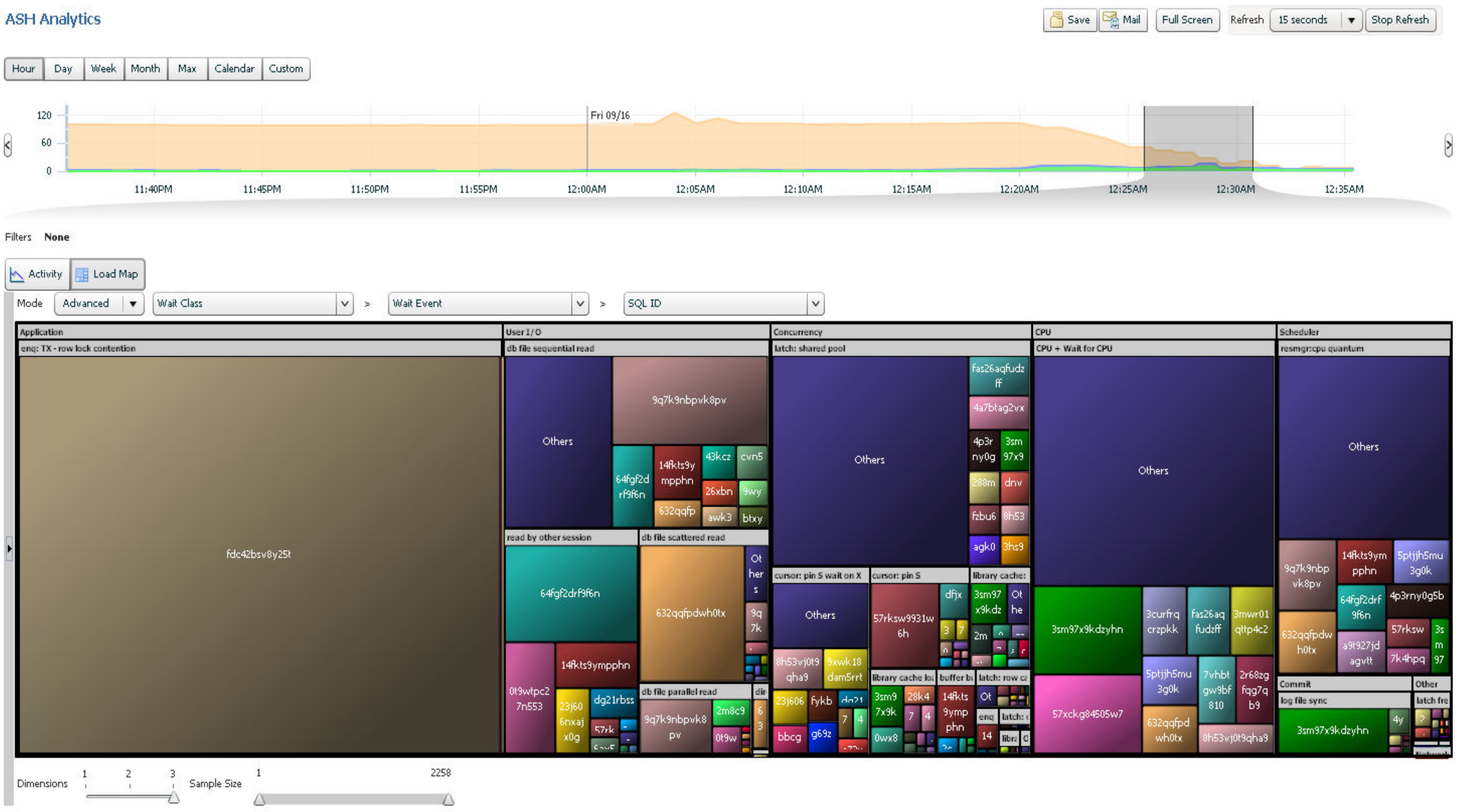
Select	SQL ID	Activity (Average Active Sessions)
☐	fdo42bsv8y25t	7.56
☐	9q7k9nbpvk8pv	1
☐	632qqfpdwh0tx	.99
☐	3sm97x9kdzyhn	.95
☐	64fgf2drf9f6n	.8

User Session	Activity (Average Active Sessions)
1:1272,46823	1
1:1203,32745	.77
1:1258,16979	.72
1:1179,62907	.69
1:42,10989	.65

Group by Wait Event within Wait Class

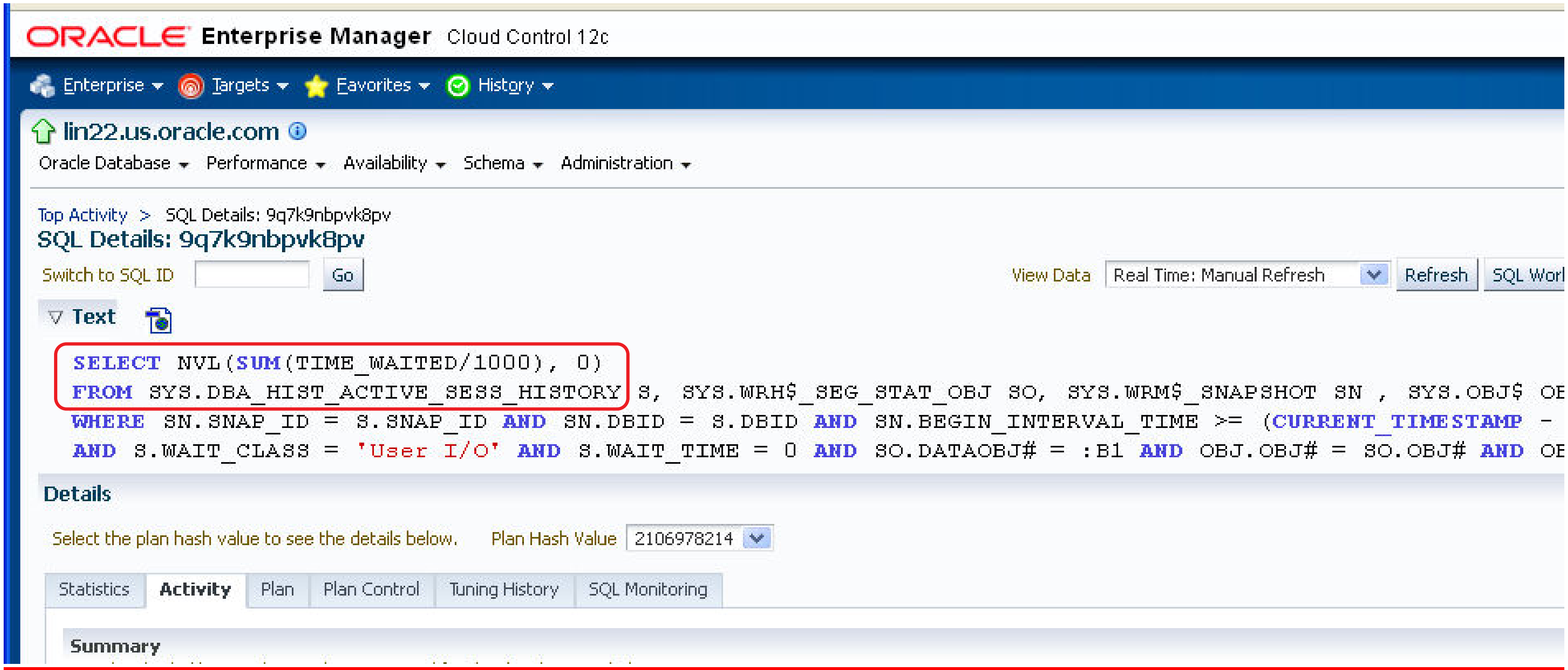


Add SQL_ID Grouping Dimension



Bad ASH Math

- SQL observed using 9 secs of CPU every 10 secs



ORACLE Enterprise Manager Cloud Control 12c

Enterprise ▾ Targets ▾ Favorites ▾ History ▾
lin22.us.oracle.com
Oracle Database ▾ Performance ▾ Availability ▾ Schema ▾ Administration ▾

Top Activity > SQL Details: 9q7k9nbpvk8pv

SQL Details: 9q7k9nbpvk8pv

Switch to SQL ID [ ] Go   View Data: Real Time: Manual Refresh   Refresh   SQL Worl

▽ Text

```
SELECT NVL(SUM(TIME_WAITED/1000), 0)
FROM SYS.DBA_HIST_ACTIVE_SESS_HISTORY S, SYS.WRH$_SEG_STAT_OBJ SO, SYS.WRM$_SNAPSHOT SN , SYS.OBJ$ OE
WHERE SN.SNAP_ID = S.SNAP_ID AND SN.DBID = S.DBID AND SN.BEGIN_INTERVAL_TIME >= (CURRENT_TIMESTAMP -
AND S.WAIT_CLASS = 'User I/O' AND S.WAIT_TIME = 0 AND SO.DATAOBJ# = :B1 AND OBJ.OBJ# = SO.OBJ# AND OE
```

Details

Select the plan hash value to see the details below.   Plan Hash Value: 2106978214

Statistics | Activity | Plan | Plan Control | Tuning History | SQL Monitoring
Summary

Wait Events and ASH

- ASH samples actively waiting sessions
  - Wait time is unknown at sample time
  - The “fix up” writes actual wait time into TIME_WAITED
- ASH is a biased sampler of wait events
  - Longer events have higher probability of being sampled
- Avoid the temptation of TIME_WAITED
  - AVG(time_waited) DOES NOT estimate avg wait times
  - MIN and MAX do not work either
    - Except when MAX exceeds 1-second

The ASH “fix up”

- ASH columns may be unknown at sampling time
    - TIME_WAITED: session is still waiting
    - PLAN_HASH: session is still optimizing SQL
    - GC events: event details unknown at event initiation
    - Certain time model bit vector columns
- ASH “fixes up” missing data during subsequent sample processing
    - TIME_WAITED fixed up in last event sample
- Querying current ASH may return un-fixed rows
    - Should not be a problem generally

ON CPU and ASH

- ASH row status “ON CPU” derived, not observed
  - Session is in a database call
  - Session is NOT in a wait event (idle or non-idle)
- Un-instrumented waits => “ON CPU”
  - These are bugs and should be rare, but have happened
- Session on run queue may be WAITING or ON CPU
  - Depends on state prior to going onto run queue

V$EVENT_HISTOGRAM

- Histogram buckets of event wait times
- Captures statistical distribution of wait times
- All events since instance startup counted in some bucket
- Exponential time bucketing scheme captures long-tail distributions efficiently

V$EVENT_HISTOGRAM

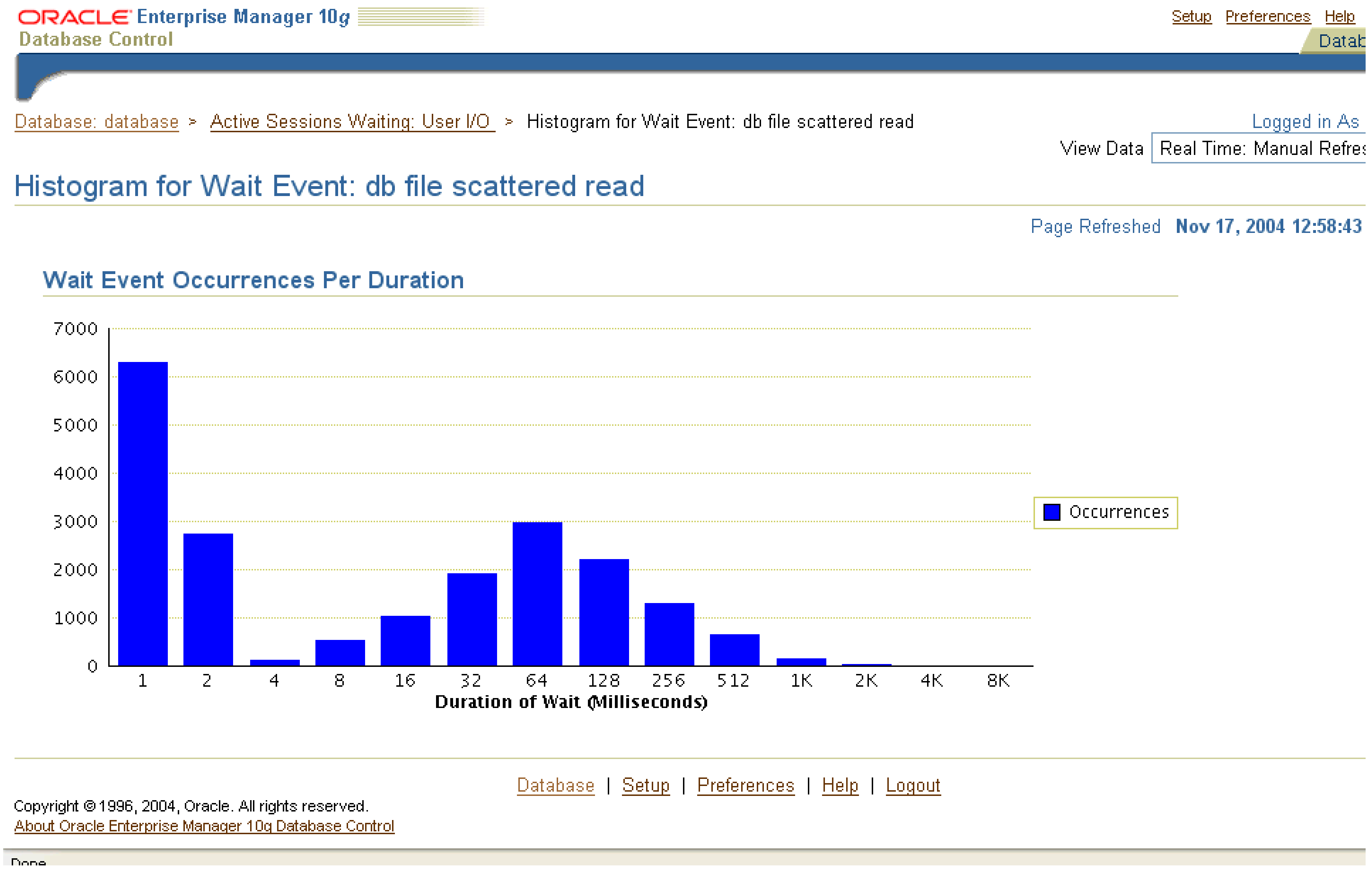
```
SQL> desc v$event_histogram
 Name                            Type
 ------------------------------ ------------------------------
 EVENT#                          NUMBER
 EVENT                           VARCHAR2(64)
 WAIT_TIME_MILLI                 NUMBER
 WAIT_COUNT                      NUMBER
 LAST_UPDATE_TIME                VARCHAR2(64)
```

Event Histogram Time Buckets

```
SQL> select distinct
            wait_time_milli
          ,log(2,wait_time_milli)
     from
            v$event_histogram
     order by 1;
```

```
WAIT_TIME_MILLI LOG(2,WAIT_TIME_MILLI)
--------------- ----------------------
              1                      0
              2                      1
              4                      2
              8                      3
             16                      4
             32                      5
             64                      6
            128                      7
            256                      8
            512                      9
           1024                     10
```

I/O Event Histogram



ORACLE Enterprise Manager 10*g*
Database Control

Setup Preferences Help

Datab

Database: database > Active Sessions Waiting: User I/O > Histogram for Wait Event: db file scattered read

Logged in As

View Data Real Time: Manual Refres

Histogram for Wait Event: db file scattered read

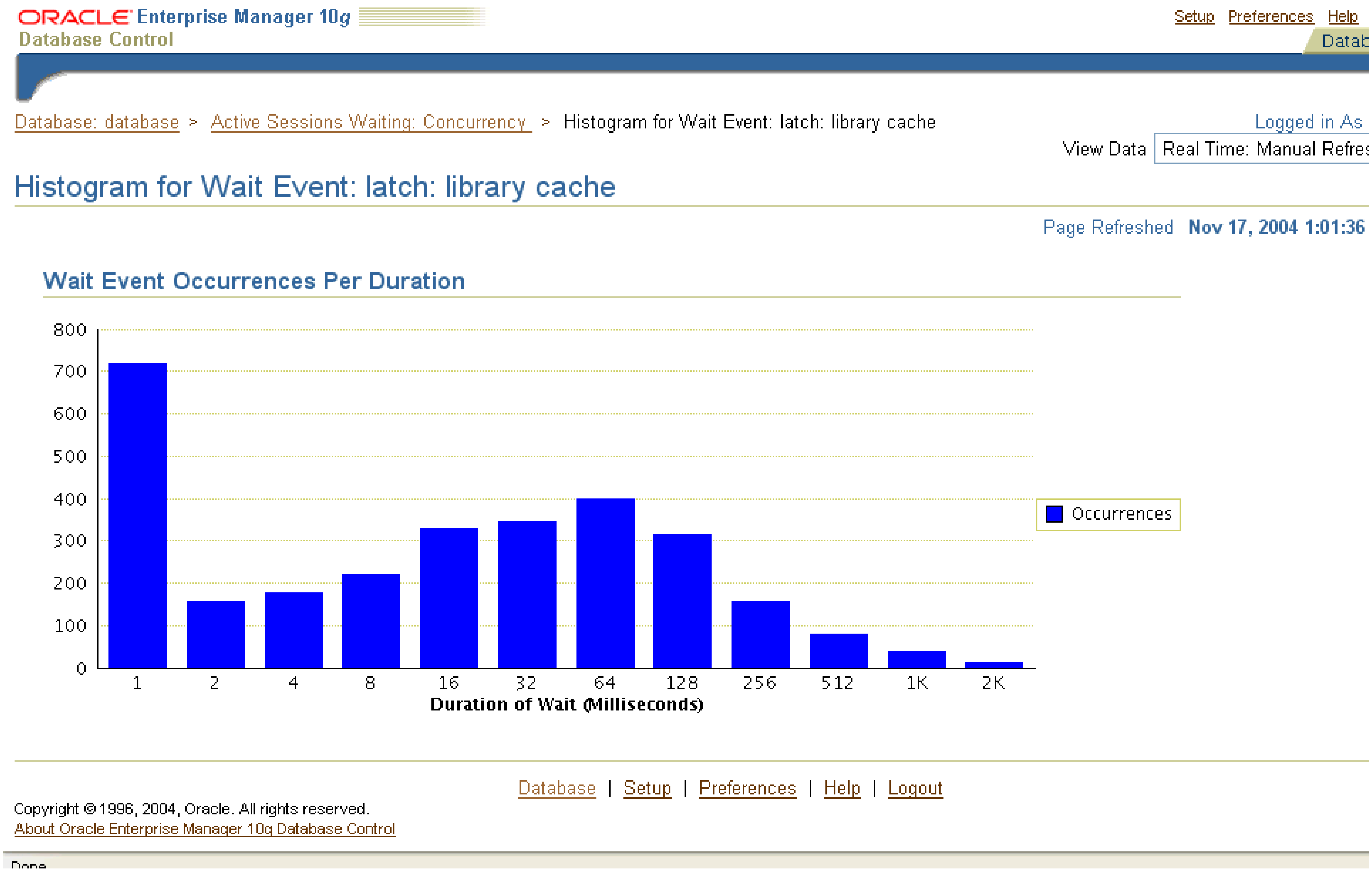
Page Refreshed **Nov 17, 2004 12:58:43**

Wait Event Occurrences Per Duration

Database | Setup | Preferences | Help | Logout

Done

Latch Wait Event Histogram



ORACLE Enterprise Manager 10g
Database Control
Setup Preferences Help
Datab
800
700
600
500
400
300
200
100
0
1
2
4
8
16
32
64
128
256
512
1K
2K
Duration of Wait (Milliseconds)
Occurrences
Database | Setup | Preferences | Help | Logout
Done

Database: database > Active Sessions Waiting: Concurrency > Histogram for Wait Event: latch: library cache

Logged in As

View Data Real Time: Manual Refres

Histogram for Wait Event: latch: library cache

Page Refreshed **Nov 17, 2004 1:01:36**

Wait Event Occurrences Per Duration

Histogram Math

- Histograms capture probability distribution of TIME_WAITED by event over this startup cycle

$$\Pr(time\_waited < bucket_N) = \sum_{bucket<N} WaitCount \Big/ \sum_{allbuckets} WaitCount$$

Significance of Histogram Buckets

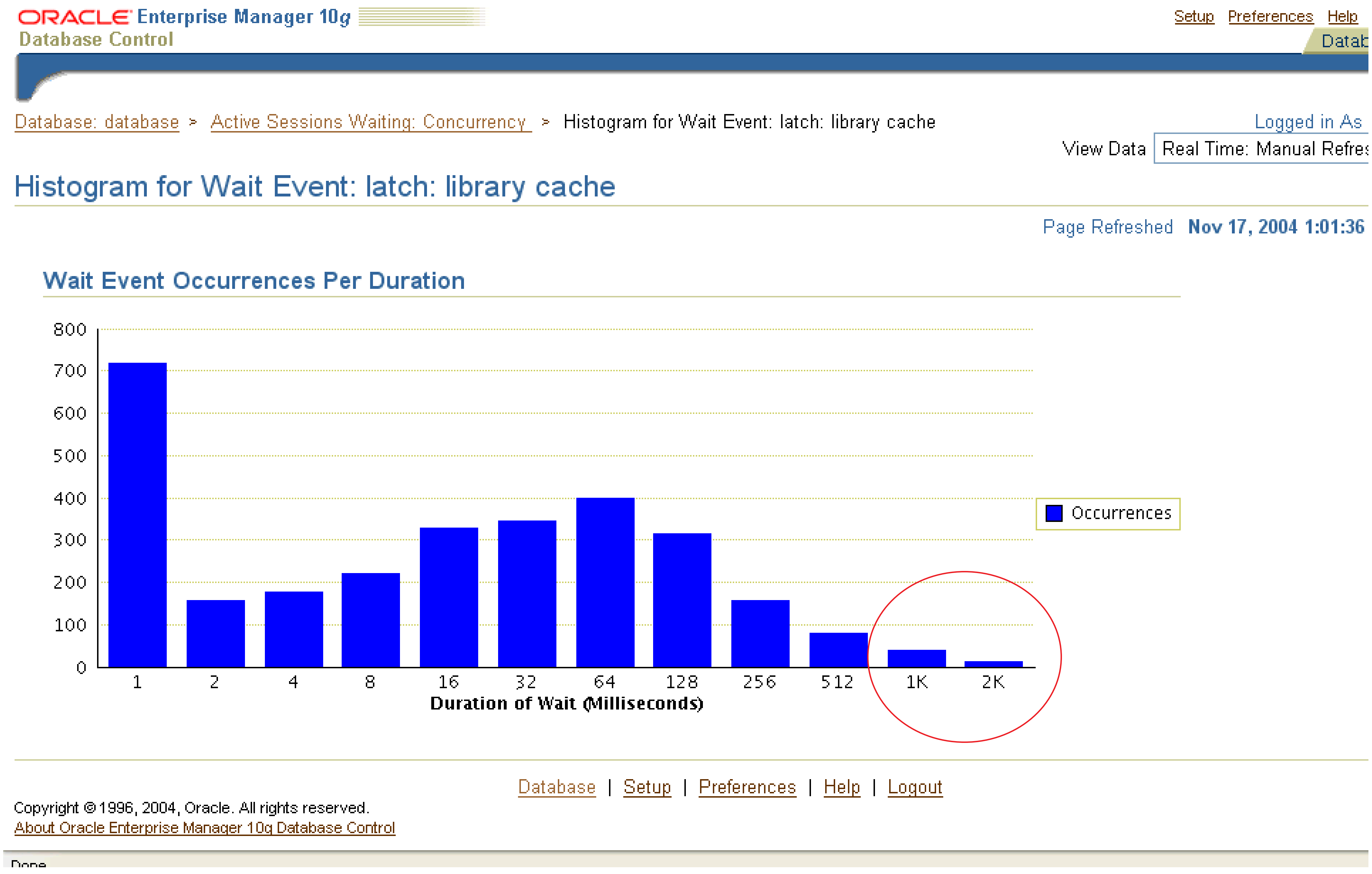
$$Significance_{bucketN} = 1 - \left(\sum^{bucket>=N} WaitCount \Big/ \sum_{allbuckets} WaitCount \right)$$

- Measures the cumulative distribution function of TIME_WAITED probabilities represented by the histograms (per bucket)
- Every event in the bucket has at least this significance

Defining “Outlier Events”

- Events with low probability of occurrence
- Events with high significance value
- Q: Has ASH sampled any such events?

“Outlier” = “Unusual”


ORACLE Enterprise Manager 10g
Database Control
Setup Preferences Help
Datab
800
700
600
500
400
300
200
100
0
1
2
4
8
16
32
64
128
256
512
1K
2K
Duration of Wait (Milliseconds)
Occurrences
Database | Setup | Preferences | Help | Logout
Done

Database: database > Active Sessions Waiting: Concurrency > Histogram for Wait Event: latch: library cache

Logged in As

View Data Real Time: Manual Refres

Histogram for Wait Event: latch: library cache

Page Refreshed Nov 17, 2004 1:01:36

Wait Event Occurrences Per Duration

Finding Outlier Events in ASH

- Which ASH rows (if any) represent wait events with significantly long TIME_WAITED against the event histogram record?

- Two step process:

1. Compute event histogram bucket significance
2. Join ASH to histograms and filter by significance

Step 1: Compute Bucket Significance

```
WITH EH$stats
as
(select
       EH.*
      ,ROUND(1 - (tot_count - bucket_tot_count + wait_count) / tot_count,6)
                                                        as event_bucket_siglevel
from
    (select event#
            ,event
            ,wait_time_milli
            ,wait_count
            ,ROUND(LOG(2,wait_time_milli))              as event_bucket
            ,SUM(wait_count) OVER (PARTITION BY event#) as tot_count
            ,SUM(wait_count) OVER (PARTITION BY event# ORDER BY wait_time_milli
                                                       RANGE UNBOUNDED PRECEDING)
                                                        as bucket_tot_count
       from  v$event_histogram
     ) EH
)
```

Step 2: Join ASH to Buckets and Filter

```
select
      EH.event_bucket
     ,ASH.sample_id
     ,ASH.session_id
     ,EH.event_bucket_siglevel as bucket_siglevel
     ,ASH.event
     ,ASH.time_waited/1000 ASH_time_waited_milli
     ,ASH.sql_id
 from
       EH$stats  EH
      ,v$active_session_history ASH
 where
      EH.event# = ASH.event#
  and EH.event_bucket_siglevel > &siglevel
  and EH.event_bucket = CASE ASH.time_waited
                             WHEN 0 THEN null
                             ELSE TRUNC(LOG(2,ASH.time_waited/1000))+1
                         END
order by
sample_id, event, session_id
;
```

DEMO?

SOFTWARE.
HARDWARE.
COMPLETE.

ORACLE®

The GV$ Problem

- Motivating use case is for RAC
- Execute V$ query on all instances?
  - Too much effort
- Convert V$ query to GV$ query?
  - GV$ remote joins not optimized
    - QC will pull V$ASH and V$EVENT_HISTOGRAM and join locally
- Neither of these “solutions” is acceptable

GV$ Table Function

- Table function distributes V$ cursor across RAC nodes and marshals result sets
- Perfect solution for this use case

```
SELECT … FROM TABLE GV$( (CURSOR( SELECT FROM V$)))
```

DEMO?

Comments and Caveats

- Highly significant events may not be sampled
- Works best for long-tailed distributions
  - Bi-modal, single-bucket, timeout events not well-behaved
- Exponential bucket sizing gets coarse quickly
  - Significance levels increase in big jumps
  - Important distinctions may hide inside large buckets
- An interesting and unusual application of ASH where TIME_WAITED is the key
  - Does it help with the motivating use case?