Getting the Most Out of Oracle’s Active Session History, Time Model and Wait Events

Tanel Poder
Intro: About me

• Tanel Põder
  • Oracle Database Performance geek (18+ years)
  • Exadata Performance geek
  • Linux Performance geek
  • Hadoop Performance geek
  • CEO & co-founder:

Instant promotion

Expert Oracle Exadata book
(2nd edition is out now!)
“The database is slow!”

1. Everyone complains
2. System monitoring tools show a clear difference
“The database is slow”?

1. Only one user or a few users complain
   - Why hasn’t my batch job completed?

2. System monitoring does not show any visible difference
   - You might be tempted to compare today 8am to “last Friday” 8am
Getting the Most Out of ASH

© Tanel Poder

Measuring SQL execution activity over time – V$SQLSTATS?

V$SQL, V$SQLSTATS, Statspack/AWR Top SQL reports measure SQL performance, not sessions’ performance

Instance-level detail (not session-level)
SQL statement level breakdown
Measuring session activity details over time – SQL trace

Session-level detail
Breakdown by SQL Statements +
Breakdown by Wait Events
Basic performance counters (logical IOs, parses etc)
Full chronological detail of session activity
No instance-wide coverage in realistic cases
What is Oracle Database activity about?

1. A bunch of *sessions*
   - You need a session for doing anything in Oracle

1. Applications use these sessions to execute (PL)SQL *cursors*
   - Most application->database calls require the use of a cursor
   - Including PL/SQL stored procedure calls (unless using direct RPC)

1. The cursor execution will do some *work*
   - During the execution, the processes either:
     - Work on CPU
       - Or at least want to work on CPU, but OS scheduler doesn’t allow them
     - Do not Work on CPU
       - They have been put to sleep by OS because of a blocking system call
       - They have gone to sleep voluntarily, waiting for some event completion
Getting the Most Out of ASH
© Tanel Poder

V$SESSION

SGA

V$SQLSTATS

Total CPU, Elapsed
Time, IO consumed
by this SQL ID / Child

AWR, Statspack
“TOP SQL” reports

Currently this
session is running SQL ID
c13sma6rkr27

Regular SQL cursor
execution metrics
(CPU, Elapsed, Exec#, LIO/PIO)

Oracle Process
(thread)

Oracle Process
(thread)

Oracle Process
(thread)

V$SQLSTATS

8dq0v1mjngj7
7hk2m2702ua
c13sma6rkr27
...
0ruh367af7gb

Currently this
session is running SQL ID
c13sma6rkr27

Oracle Process
(thread)

Oracle Process
(thread)

Oracle Process
(thread)

V$SESSION

SID 1  SID 2  SID 3  ...  SID 999

Regular SQL cursor
execution metrics
(CPU, Elapsed, Exec#, LIO/PIO)

Currently this
session is running SQL ID
c13sma6rkr27

Total CPU, Elapsed
Time, IO consumed
by this SQL ID / Child

AWR, Statspack
“TOP SQL” reports
V$SESSION

- An Oracle process populates information in V$SESSION
  - Immediately when something changes, like new SQL ID is executed, a wait ends...

Currently this session is running SQL ID c13sma6rkr27

This session waits for "db file sequential read" event completion

This session is created by a program called ReportRunner.exe

Currently this session happens to be WAITING

This session was created by user JOE

This session was created by user JOE

...
V$SESSION

• V$SESSION is a fixed array (separate in each RAC instance)
  • The sessions parameter controls the number of slots in this array
  • The SID is just a slot number in the V$SESSION array

• When doing work, each Oracle process updates information in its corresponding session slot
  • Constant, immediate updates about current state / task at hand
  • An update (let’s say a new wait event) overwrites the previous value

• Developers, DBAs can query these attributes to find what other sessions are up to
  • From V$SESSION you see only the current state of a session, no history!
Active Session History

**History** of samples of **Sessions** that happened to be **Active** during the sampling
Active Session History and where its data comes from

**SGA**

**V$SQLSTATS**

- 8dq0v1mjngj7
- 7hk2m2702ua
- c13sma6rkr27
- ...
- 0ruh367af7gb

**V$SESSION**

- SID 1
- SID 2
- SID 3
- ...
- SID 999

**ASH**

- MMNL process
- Active session samples

**Oracle Process (thread)**

**Oracle Process (thread)**

**Oracle Process (thread)**

**AWR, Statspack “TOP SQL” reports**

**Total CPU, Elapsed Time, IO consumed by this SQL ID / Child**

**Regular SQL cursor execution metrics (CPU, Elapsed, Exec#, LIO/PIO)**

---

Getting the Most Out of ASH

© Tanel Poder

blog.tanelpoder.com
Sampling session activity details over time – Which SQL?

Sample session activity over time -> (v$session)

Count the number of (active) sessions each sample, group by SQL ID

Getting the Most Out of ASH
© Tanel Poder

blog.tanelpoder.com
Sampling session activity – Working or Waiting?

Time

Session count

CPU
User IO
Locks
Commit

Getting the Most Out of ASH
© Tanel Poder

blog.tanelpoder.com
On-line Active Session History and historical ASH

- **In-memory array**
  - One in each instance
  - Designed to keep at least 1 hour of history in memory
  - MMNL process copies *active* session info from V$SESSION and an "ASHLoc" memory structure there once per second
  - 1 sample = 1 second
  - **V$ACTIVE_SESSION_HISTORY**

- **Long-term history**
  - One per database
  - Retention configured using the AWR *retention* parameter
  - Only 10% of samples saved by default to save space
  - 1 sample = 10 seconds
  - **DBA_HIST_ACTIVE_SESS_HISTORY**

This is used when switching to **Historical** view in OEM
The correct logic for querying ASH

```sql
SELECT
dimension1,
    COUNT(*) seconds
FROM
    v$active_session_history
WHERE
    sample_time BETWEEN ...
ORDER BY
    seconds DESC
```

SQL> SELECT
    2 sql_id,
    3 , COUNT(*) seconds
    4 FROM
    5 v$active_session_history
    6 WHERE
    7 sample_time > SYSDATE-1/24/12
    8 GROUP BY
    9 sql_id
   10 ORDER BY
   11* seconds DESC
SQL> /

```
SQL_ID SECONDS
--------- ------
2vp4k2kgy2wm4 124
    c13sma6rkr27c 50
8dq0v1mjngj7t 27
0w2qpuc6u2zsp 11
bymb3ujkr3ubk 10
...```

ASH is like a fact table in a dimensional data model.
The existence of a row in ASH is a fact of a session taking time in database (DB Time)
- v$active_session_history row = 1 second
- dba_hist_active_sess_history = 10 seconds
The **incorrect** logic for querying ASH!!

```sql
SELECT
    dimension1,
    SUM(time_waited) as seconds
FROM
    v$active_session_history
WHERE
    sample_time BETWEEN ...
ORDER BY
    seconds DESC
```

SQL> ed
Wrote file afiedit.sql

```sql
1  SELECT
2      sql_id
3    , SUM(time_waited) or this_is_wrong
4  FROM
5      v$active_session_history
6  WHERE
7      sample_time > SYSDATE-1/24/12
8  AND sql_id = '2vp4k2kgy2wm4'
9  GROUP BY
10      sql_id
11  ORDER BY
12      this_is_wrong DESC
SQL> /
```

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>THIS_IS_WRONG</th>
</tr>
</thead>
<tbody>
<tr>
<td>2vp4k2kgy2wm4</td>
<td>0</td>
</tr>
</tbody>
</table>

ASH does not **trace** every single wait there is!

Therefore information about many short waits that happened between ASH samples is lost!

However, **counting** the number of samples where a session was active gives a good **approximation** of where was the DB time spent.
A simple ASH drilldown example

SELECT
dimension1
, dimension2
, COUNT(*) seconds
FROM
  v$active_session_history
WHERE
  sample_time BETWEEN ...
AND sql_id = '2vp4k2kgy2wm4'
ORDER BY
  seconds DESC

SQL> SELECT
  2      sql_id
  3    , session_id,session_serial#
  4    , COUNT(*) seconds
  5 FROM
  6      v$active_session_history
  7 WHERE
  8      sample_time > SYSDATE-1/24/12
  9 AND sql_id = '2vp4k2kgy2wm4'
10  GROUP BY
11      sql_id
12    , session_id,session_serial#
13  ORDER BY
14*     seconds DESC
SQL> /

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>SIDSERIAL</th>
<th>SECONDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2vp4k2kgy2wm4</td>
<td>789,14653</td>
<td>62</td>
</tr>
<tr>
<td>2vp4k2kgy2wm4</td>
<td>789,14631</td>
<td>59</td>
</tr>
<tr>
<td>2vp4k2kgy2wm4</td>
<td>803,61957</td>
<td>56</td>
</tr>
</tbody>
</table>

Who’s running this SQL?

You can drill down into a single SQL, single application or a session by just adding WHERE conditions for filtering data.

You can see more detail by just adding ASH dimension fields to the GROUP BY clause.
SELECT COUNT(*)
FROM ash
GROUP BY wait_class
GROUP BY sample_time
GROUP BY SESSION_ID, USERNAME
GROUP BY wait_class
Average Active Sessions (AAS): Why a new metric?

The meaning of AWR "100%" constantly changes based on how many sessions are logged on and how idle/active they are.
Average Active Sessions (ASH report)

ASH records all active sessions every second. 5919 active sessions samples over 9 minutes (540 seconds) On average we had 5919 / 540 = \textbf{10.96 active sessions} during each sample

<table>
<thead>
<tr>
<th>Event</th>
<th>Event Class</th>
<th>% Event</th>
<th>Avg Active Sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>db file sequential read</td>
<td>User I/O</td>
<td>75.67%</td>
<td>8.29</td>
</tr>
<tr>
<td>log file sync</td>
<td>Commit</td>
<td>6.72%</td>
<td>0.74</td>
</tr>
<tr>
<td>db file parallel read</td>
<td>User I/O</td>
<td>1.10%</td>
<td>0.12</td>
</tr>
</tbody>
</table>

8.29 AAS were waiting for db file sequential read (75.67% of total AAS)

Top Background Events

<table>
<thead>
<tr>
<th>Event</th>
<th>Event Class</th>
<th>% Activity</th>
<th>Avg Active Sessions</th>
</tr>
</thead>
<tbody>
<tr>
<td>log file parallel write</td>
<td>System I/O</td>
<td>5.41%</td>
<td>0.59</td>
</tr>
<tr>
<td>db file parallel write</td>
<td>System I/O</td>
<td>3.70%</td>
<td>0.41</td>
</tr>
<tr>
<td>control file sequential read</td>
<td>System I/O</td>
<td>2.72%</td>
<td>0.30</td>
</tr>
<tr>
<td>CPU + Wait for CPU</td>
<td>CPU</td>
<td>2.31%</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Time spent by background processes, it's not a direct indicator of user response time.
Basic Troubleshooting Examples
Something else is consuming all the CPU resource! (some other database instance, application or a monitoring agent, etc)

Load average even higher than the CPU usage (1600% - 16 virtual threads)

You need to run an **OS tool** (top) to find the top CPU users. Your database instance may just be a victim here.
Performance Home: Extra IO and CPU time?

Why the I/O wait and CPU time spike?
Performance Home: Workload Increase

The database is doing more work!

... and more dirty buffer writes and log file writes

... and more disk reads
Performance Home: Extra I/O time and throughput drop?

Why the I/O spike?

Throughput has dropped!
Performance Home: I/O subsystem issue!

The average I/O time has increased significantly!

We are doing less I/Os?!
Two reasons why your workload is running slowly:

1. It's doing too much work

2. It can not work fast enough
IO time jumps up, some users complain

Throughput has dropped!

Average Active Sessions

Throughput has dropped!
I/O latency

I/O latency has increased
Drill down into the I/O waits

"cell smart table scan" is the full table scan wait event
Break the IO waits down by SQL_ID and Session ID

A single SQL consuming 80% of wait time

No single session waiting much more than others
Drill down into the problem SQL

Just click on the SQL_ID (hover mouse over for SQL text preview)
SELECT CUSTOMER_ID, CUST_FIRST_NAME, CUST_LAST_NAME, NLS_TERRITORY, CREDIT_LIMIT, CUST_EMAIL, ACCOUNT_MGR_ID FROM CUSTOMERS WHERE CUSTOMER_ID = :B2 AND ROWNUM < :B1

First impression: This SQL has not been executed or not used much resource in past?

Multiple different execution plans! The SQL view shows only 1 plan_hash_value at a time!
Let's see the resource usage of the other plan

This SQL was using a different plan earlier, before the problem happened!
Comparing Execution Plans

The Statistics pane shows you the number of executions, logical & physical IO counts.
Using ASH Analytics

Select the problem range (with enough "before" time)
ASH Analytics: Drill Down into a SQL_ID

Current breakdown ("group by") is on SQL_ID

Right click on the SQL of interest to show only its data
ASH Analytics: Drill Down into a SQL_ID

Now the chart shows only given SQL_ID.

The black line shows total activity for reference.

Now let's group the ASH data by a different attribute!
ASH Analytics: Drill Down into a SQL_ID

Same data, broken down by plan hash value

All recorded plans used by this SQL_ID are listed on a single chart
ASH Analytics: Drill Down into a SQL_ID

You can break the activity down by any ASH attribute, like even SQL plan line in 11g+
The full power of ASH data dimensions
ASH is a fact table in a dimensional data model

- ASH is a **fact table**

- The mere **existence** of a record in ASH is a fact
  - ...of database time used by some session in past
  - That's why you should just `COUNT(*)` the records

- You can break down the time consumption using **GROUP BY** on ASH **dimension** attribute columns, to get TOP-reports:
  - SQL ID
  - Wait event
  - User ID
  - Session ID
  - Program
Top-activity by ASH dimensions

Basic ASH attributes to group TOP activity by

Getting the Most Out of ASH
© Tanel Poder

blog.tanelpoder.com
### V$ACTIVE_SESSION_HISTORY columns in Oracle 11.2

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAMPLE_ID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SAMPLE_TIME</td>
<td></td>
<td>TIMESTAMP(3)</td>
</tr>
<tr>
<td>IS_AWR_SAMPLE</td>
<td></td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>SESSION_ID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SESSION_SERIAL#</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SESSION_TYPE</td>
<td></td>
<td>VARCHAR2(10)</td>
</tr>
<tr>
<td>FLAGS</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>USER_ID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SQL_ID</td>
<td></td>
<td>VARCHAR2(13)</td>
</tr>
<tr>
<td>IS_SQLID_CURRENT</td>
<td></td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>SQL_CHILD_NUMBER</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SQL_OPCODE</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SQL_OPNAME</td>
<td></td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>FORCE_MATCHING_SIGNATURE</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>TOP_LEVEL_SQL_ID</td>
<td></td>
<td>VARCHAR2(13)</td>
</tr>
<tr>
<td>TOP_LEVEL_SQL_OPCODE</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SQL_PLAN_HASH_VALUE</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SQL_PLAN_LINE_ID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SQL_PLAN_OPERATION</td>
<td></td>
<td>VARCHAR2(30)</td>
</tr>
<tr>
<td>SQL_PLAN_OPTIONS</td>
<td></td>
<td>VARCHAR2(30)</td>
</tr>
<tr>
<td>SQL_EXEC_ID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SQL_EXEC_START</td>
<td></td>
<td>DATE</td>
</tr>
</tbody>
</table>

**ASH sample metadata**

**Session info**

**SQL statement info**

**SQL execution plan info**

**Individual SQL execution info**
### V$ACTIVE_SESSION_HISTORY columns in Oracle 11.2

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLSQL_ENTRY_OBJECT_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>PLSQL_ENTRY_SUBPROGRAM_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>PLSQL_OBJECT_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>PLSQL_SUBPROGRAM_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>QC_INSTANCE_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>QC_SESSION_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>QC_SESSION_SERIAL#</td>
<td>NUMBER</td>
</tr>
<tr>
<td>PX_FLAGS</td>
<td>NUMBER</td>
</tr>
<tr>
<td>EVENT</td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>EVENT_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>EVENT#</td>
<td>NUMBER</td>
</tr>
<tr>
<td>SEQ#</td>
<td>NUMBER</td>
</tr>
<tr>
<td>P1TEXT</td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>P1</td>
<td>NUMBER</td>
</tr>
<tr>
<td>P2TEXT</td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>P2</td>
<td>NUMBER</td>
</tr>
<tr>
<td>P3TEXT</td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>P3</td>
<td>NUMBER</td>
</tr>
<tr>
<td>WAIT_CLASS</td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>WAIT_CLASS_ID</td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>WAIT_TIME</td>
<td>NUMBER</td>
</tr>
<tr>
<td>SESSION_STATE</td>
<td>VARCHAR2(7)</td>
</tr>
<tr>
<td>TIME_WAITED</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

**PL/SQL object info**, join to `dba_procedures / @procid.sql

**Parallel execution info**

**Wait event info**

**Wait event parameters** (extra info)

---

Remember, you should *not* sum any wait columns, use `COUNT(*)` to estimate DB Time.
### V$ACTIVE_SESSION_HISTORY columns in Oracle 11.2

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOCKING_SESSION_STATUS</td>
<td>VARCHAR2(11)</td>
</tr>
<tr>
<td>BLOCKING_SESSION</td>
<td>NUMBER</td>
</tr>
<tr>
<td>BLOCKING_SESSION_SERIAL#</td>
<td>NUMBER</td>
</tr>
<tr>
<td>BLOCKING_INST_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>BLOCKING_HANGCHAIN_INFO</td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>CURRENT_OBJ#</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CURRENT_FILE#</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CURRENT_BLOCK#</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CURRENT_ROW#</td>
<td>NUMBER</td>
</tr>
<tr>
<td>TOP_LEVEL_CALL#</td>
<td>NUMBER</td>
</tr>
<tr>
<td>TOP_LEVEL_CALL_NAME</td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>CONSUMER_GROUP_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>XID</td>
<td>RAW(8)</td>
</tr>
<tr>
<td>REMOTE_INSTANCE#</td>
<td>NUMBER</td>
</tr>
<tr>
<td>TIME_MODEL</td>
<td>NUMBER</td>
</tr>
<tr>
<td>IN_CONNECTION_MGMT</td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>IN_PARSE</td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>IN_HARD_PARSE</td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>IN_SQL_EXECUTION</td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>IN_PLSQL_EXECUTION</td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>IN_PLSQL_RPC</td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>IN_PLSQL_COMPILATION</td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>IN_JAVA_EXECUTION</td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>IN_BIND, IN_CURSOR_CLOSE, IN_SEQUENCE_LOAD...</td>
<td>VARCHAR2(1)</td>
</tr>
</tbody>
</table>

**Blocking session info**

DB object involved in a wait (not populated for all waits, not always cleaned up properly)

**Database call (OPI call) info**

**Current transaction info**

Time model phase info.

These Y / N flags tell in which phase (SQL parse, SQL execute, login, PL/SQL, login) the session happened to be when sampled.
## V$ACTIVE_SESSION_HISTORY columns in Oracle 11.2

<table>
<thead>
<tr>
<th>Column</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPTURE_OVERHEAD</td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>REPLAY_OVERHEAD</td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>IS_CAPTURED</td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>IS_REPLAYED</td>
<td>VARCHAR2(1)</td>
</tr>
<tr>
<td>DBREPLAY_FILE_ID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>DBREPLAY_CALL_COUNTER</td>
<td>NUMBER</td>
</tr>
<tr>
<td>SERVICE_HASH</td>
<td>NUMBER</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>VARCHAR2(48)</td>
</tr>
<tr>
<td>MODULE</td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>ACTION</td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>CLIENT_ID</td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>MACHINE</td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>PORT</td>
<td>NUMBER</td>
</tr>
<tr>
<td>ECID</td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>TM_DELTA_TIME</td>
<td>NUMBER</td>
</tr>
<tr>
<td>TM_DELTA_CPU_TIME</td>
<td>NUMBER</td>
</tr>
<tr>
<td>TM_DELTA_DB_TIME</td>
<td>NUMBER</td>
</tr>
<tr>
<td>DELTA_TIME</td>
<td>NUMBER</td>
</tr>
<tr>
<td>DELTA_READ_IO_REQUESTS</td>
<td>NUMBER</td>
</tr>
<tr>
<td>DELTA_WRITE_IO_REQUESTS</td>
<td>NUMBER</td>
</tr>
<tr>
<td>DELTA_READ_IO_BYTES</td>
<td>NUMBER</td>
</tr>
<tr>
<td>DELTA_WRITE_IO_BYTES</td>
<td>NUMBER</td>
</tr>
<tr>
<td>DELTA_INTERCONNECT_IO_BYTES</td>
<td>NUMBER</td>
</tr>
<tr>
<td>PGA_ALLOCATED</td>
<td>NUMBER</td>
</tr>
<tr>
<td>TEMP_SPACE_ALLOCATED</td>
<td>NUMBER</td>
</tr>
</tbody>
</table>

**DB Replay & workload capture**

**Client application info**

**Execution context identifier** (end-to-end request ID)

**I/O counters**: These can be summed over multiple samples

**Session memory usage when sampled** (use MAX or AVG*)
Ok, how to use all this awesome information?
Profiling! Breaking down session response time

1. Which DB operation?
   - Login, Cursor open/close, Parse, SQL Execute, PL/SQL Execute, LOB access?

1. Which SQL statement?
   - Or PL/SQL procedure
   - Which execution plan version?

1. Working on CPU or Waiting?
   - If waiting, for what?

2. When?
   - sample_time
Getting the Most Out of ASH

-- @ashtop <GROUP_BY> <FILTER> <BEGIN_TIME> <END_TIME>

SQL> @ashtop username,sql_id session_type='FOREGROUND' sysdate-1/24 sysdate

<table>
<thead>
<tr>
<th>USERNAME</th>
<th>SQL_ID</th>
<th>TotalSeconds</th>
<th>CPU</th>
<th>User</th>
<th>I/O</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOE</td>
<td>c13sma6rkr27c</td>
<td>1218</td>
<td>605</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SOE</td>
<td>8dq0v1mjngj7t</td>
<td>1212</td>
<td>206</td>
<td>1001</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>SOE</td>
<td>bymb3ujkr3ubk</td>
<td>965</td>
<td>255</td>
<td>612</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TANEL</td>
<td>5n83a4q202674</td>
<td>724</td>
<td>24</td>
<td>700</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SQL> @ashtop session_state,event sql_id='c13sma6rkr27c' sysdate-1/24 sysdate

<table>
<thead>
<tr>
<th>SESSION EVENT</th>
<th>TotalSeconds</th>
<th>CPU</th>
<th>User</th>
<th>I/O</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON CPU</td>
<td>603</td>
<td>603</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WAITING</td>
<td>313</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WAITING</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>WAITING</td>
<td>87</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### SQL Query

```sql
SQL> @ashtop event,blocking_session_status,blocking_inst_id,blocking_session, blocking_session_serial# "wait_class in ('Concurrency','Cluster')" sysdate-1/24 sysdate
```

<table>
<thead>
<tr>
<th>EVENT</th>
<th>BLOCKING_SESSION</th>
<th>B_INST_ID</th>
<th>B_SESSION</th>
<th>BSERIAL#</th>
<th>TotalSeconds</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>gc cr block 2-way</td>
<td>UNKNOWN</td>
<td></td>
<td></td>
<td></td>
<td>339</td>
<td>339</td>
</tr>
<tr>
<td>gc current block 2-way</td>
<td>UNKNOWN</td>
<td></td>
<td></td>
<td></td>
<td>299</td>
<td>299</td>
</tr>
<tr>
<td>gc current grant busy</td>
<td>GLOBAL</td>
<td></td>
<td></td>
<td></td>
<td>156</td>
<td>156</td>
</tr>
<tr>
<td>gc current grant 2-way</td>
<td>UNKNOWN</td>
<td></td>
<td></td>
<td></td>
<td>117</td>
<td>117</td>
</tr>
<tr>
<td>gc cr grant 2-way</td>
<td>UNKNOWN</td>
<td></td>
<td></td>
<td></td>
<td>34</td>
<td>34</td>
</tr>
<tr>
<td>gc cr multi block request</td>
<td>UNKNOWN</td>
<td></td>
<td></td>
<td></td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>gc current grant busy</td>
<td>UNKNOWN</td>
<td></td>
<td></td>
<td></td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>gc current block busy</td>
<td>UNKNOWN</td>
<td></td>
<td></td>
<td></td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>gc buffer busy acquire</td>
<td>UNKNOWN</td>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>library cache: mutex X</td>
<td>UNKNOWN</td>
<td></td>
<td></td>
<td></td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>gc current grant 2-way</td>
<td>GLOBAL</td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>gc buffer busy acquire</td>
<td>VALID</td>
<td>1</td>
<td>1174</td>
<td>42307</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>gc buffer busy acquire</td>
<td>VALID</td>
<td>1</td>
<td>588</td>
<td>32309</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>gc buffer busy acquire</td>
<td>VALID</td>
<td>1</td>
<td>981</td>
<td>31893</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>gc buffer busy acquire</td>
<td>VALID</td>
<td>1</td>
<td>1173</td>
<td>26117</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>gc buffer busy acquire</td>
<td>VALID</td>
<td>1</td>
<td>1365</td>
<td>15545</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

**Many wait events populate the `blocking_session` columns.**
(RAC still a problem)
ashtop.sql script

SELECT
  &1
  , COUNT(*)
  , SUM(CASE WHEN wait_class IS NULL
             THEN 1 ELSE 0 END) "CPU"
  , SUM(CASE WHEN wait_class = 'User I/O'
             THEN 1 ELSE 0 END) "User I/O"
  , SUM(CASE WHEN wait_class = 'Application'
             THEN 1 ELSE 0 END) "Application"
  , SUM(CASE WHEN wait_class = 'Concurrency'
             THEN 1 ELSE 0 END) "Concurrency"
  , SUM(CASE WHEN wait_class = 'Commit'
             THEN 1 ELSE 0 END) "Commit"
  , SUM(CASE WHEN wait_class = 'Queueing'
             THEN 1 ELSE 0 END) "Queueing"
  , SUM(CASE WHEN wait_class = 'Other'
             THEN 1 ELSE 0 END) "Other"
FROM v$active_session_history a,
     dba_users u
WHERE a.user_id = u.user_id (+)
AND &2
AND sample_time BETWEEN &3 AND &4
GROUP BY &1
ORDER BY "TotalSeconds" DESC
, &1

Very simple query! The query structure remains the same, just the GROUP cols and filters change

ASH analysis is simple!
Parsing issues

SQL> @ashtop sql_id "session_id=201 and session_serial#=2045" sysdate-1/24 sysdate

%This SQL_ID | TotalSeconds | CPU | User | I/O | Application | Concurrency
--------------|--------------|-----|------|-----|-------------|----------------
7% 6mfhuc2m1wwh6 | 15 | 15 | 0 | 0 | 0 | 0
0% 01p50dapj7mn1 | 1 | 1 | 0 | 0 | 0 | 0
0% 08qbgdbcq0zn1 | 1 | 1 | 0 | 0 | 0 | 0

No single SQL takes majority of time. Many different SQL_IDs

SQL> @ashtop sql_plan_hash_value "session_id=201 and session_serial#=2045"

%This SQL_PLAN_HASH_VALUE | TotalSeconds | CPU | User | I/O | Application
--------------------------|--------------|-----|------|-----|-------------
90% 2274240236 | 202 | 202 | 0 | 0 | 0
10% 0 | 23 | 23 | 0 | 0 | 0

SQL> SELECT sql_text FROM v$sql WHERE plan_hash_value = 2274240236;

SQL_TEXT

select count(*) from dual where rownum = -107532095
select count(*) from dual where rownum = 2016837960

Literal values!
Time Model Phases example - Parsing

SQL> @ash/time_model username "session_id=201 and session_serial#=2045" sysdate-1/24 sysdate

<table>
<thead>
<tr>
<th>COUNT(*)</th>
<th>%This</th>
<th>USERNAME</th>
<th>STAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>133</td>
<td>74%</td>
<td>TANEL</td>
<td>PARSE HARD_PARSE SQL_EXECUTION</td>
</tr>
<tr>
<td>37</td>
<td>21%</td>
<td>TANEL</td>
<td>PARSE SQL_EXECUTION</td>
</tr>
<tr>
<td>7</td>
<td>4%</td>
<td>TANEL</td>
<td>SQL_EXECUTION</td>
</tr>
<tr>
<td>3</td>
<td>2%</td>
<td>TANEL</td>
<td>SQL_EXECUTION PLSQL_EXECUTION</td>
</tr>
</tbody>
</table>

There are also V$SESS_TIME_MODEL and V$SYS_TIME_MODEL views (different scope)

Time Model attributes tell in which type of execution the session was in (it complements SQL_ID and wait event info)
### topcur.sql

- List top SQL by distinct SQL_IDs and number of children (versions)
- Queries V$SQL (not ASH)

```
SQL> @topcur
==============================================================================================
== SQLs not using bind variables (check the SQL texts of top offenders)                     ==
==============================================================================================
TOTAL_CHILDREN DISTINCT_SQLIDS PLAN_HASH_VALUE MIN(SQL_ID)   MAX(SQL_ID)   MIN(FIRST_LOAD_TIME
--------------  ---------------  --------------------  ------------  ------------  -------------------
 3517         3517         2274240236  0027x8k49at8f gzrqcr44fr34t  2013-04-26/14:11:26
 33           31          03tuga43v3xkg g7mt7ptq286u7  2013-04-21/11:41:37
 15           15          1388734953 089dbukv1aanh gn9m0mtakrwmc  2013-04-21/11:41:39

==============================================================================================
== SQLs with many child cursors under a parent (use nonshared*.sql to find the reasons)     ==
==============================================================================================
TOTAL_CHILDREN DISTINCT_SQLIDS DISTINCT_PLANS SQL_ID        MIN(PLAN_HASH_VALUE) MAX(PLAN_HASH_VALUE
--------------  ---------------  ---------------  ------------  -------------------  ------------  -------------------
 2994         1             1            axdbo9g9p3ypsa  2966233522      2966233522
```

- Many different SQL texts
- Check SQL text of some SQL_IDs
- Single SQL text used by all children
- Run nonshared.sql to find the reasons
Time Model Phases example – Sequence Load

```
SQL> @ash/time_model_phases sql_id "session_id=201 and session_serial#=2045" sysdate-1/24/60 sysdate

<table>
<thead>
<tr>
<th>COUNT(*)</th>
<th>%This SQL_ID</th>
<th>STAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>85% 0mnrqmmbrk776</td>
<td>SQL_EXECUTION SEQUENCE_LOAD</td>
</tr>
<tr>
<td>9</td>
<td>15% 0mnrqmmbrk776</td>
<td>SQL_EXECUTION</td>
</tr>
</tbody>
</table>
```

```
SQL> @ash/time_model_phases sql_id,session_state,event "session_id=201 and session_serial#=2045" sysdate-1/24/60 sysdate

<table>
<thead>
<tr>
<th>COUNT(*)</th>
<th>%This SQL_ID</th>
<th>SESSION EVENT</th>
<th>STAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>65% 0mnrqmmbrk776</td>
<td>ON CPU</td>
<td>SQL_EXECUTION SEQUENCE_LOAD</td>
</tr>
<tr>
<td>11</td>
<td>18% 0mnrqmmbrk776</td>
<td>WAITING row cache lock</td>
<td>SQL_EXECUTION SEQUENCE_LOAD</td>
</tr>
<tr>
<td>10</td>
<td>17% 0mnrqmmbrk776</td>
<td>ON CPU</td>
<td>SQL_EXECUTION</td>
</tr>
</tbody>
</table>

-- fix the problem
SQL> ALTER SEQUENCE s CACHE 10000;
Sequence altered.
```
Time Model Hierarchy

1) background elapsed time
   2) background cpu time
      3) RMAN cpu time (backup/restore)

1) DB time
   2) DB CPU
   2) connection management call elapsed time
   2) sequence load elapsed time
   2) sql execute elapsed time
   2) parse time elapsed
      3) hard parse elapsed time
         4) hard parse (sharing criteria) elapsed time
            5) hard parse (bind mismatch) elapsed time
      3) failed parse elapsed time
         4) failed parse (out of shared memory) elapsed time
   2) PL/SQL execution elapsed time
   2) inbound PL/SQL rpc elapsed time
   2) PL/SQL compilation elapsed time
   2) Java execution elapsed time
   2) repeated bind elapsed time

DB Time is the total time the DB spent servicing application calls in DB

Use the "parse elapsed time" numbers for estimating parsing overhead.

Check v$sess_time_model in Oracle Reference docs:
Wait Events
### Wait Events

SQL> SELECT wait_class, COUNT(*)
    FROM v$event_name
    GROUP BY wait_class;

<table>
<thead>
<tr>
<th>WAIT_CLASS</th>
<th>COUNT(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrative</td>
<td>55</td>
</tr>
<tr>
<td>Application</td>
<td>17</td>
</tr>
<tr>
<td>Cluster</td>
<td>50</td>
</tr>
<tr>
<td>Commit</td>
<td>2</td>
</tr>
<tr>
<td>Concurrency</td>
<td>33</td>
</tr>
<tr>
<td>Configuration</td>
<td>24</td>
</tr>
<tr>
<td>Idle</td>
<td>95</td>
</tr>
<tr>
<td>Network</td>
<td>35</td>
</tr>
<tr>
<td>Other</td>
<td>745</td>
</tr>
<tr>
<td>Queueing</td>
<td>9</td>
</tr>
<tr>
<td>Scheduler</td>
<td>8</td>
</tr>
<tr>
<td>System I/O</td>
<td>31</td>
</tr>
<tr>
<td>User I/O</td>
<td>48</td>
</tr>
</tbody>
</table>

1152 different wait events in Oracle 11.2.0.3 (@sed.sql)

**Idle** wait events are not accounted in session DB Time – and do not show up in ASH

Similar events organized into wait classes
User I/O Wait Class

- Your query waits for IO for "too much" either because:

  1. The SQL is doing too much IO (bad plan, too much data, etc)
     - Revert back to the good plan
     - Drill down into the SQL query and optimize it

  2. The I/O itself is slow
     - Talk to DBA / sysadmin / storage team 😊
     - Cache more

[Graph: Latency For Synchronous Single Block Reads]
**User I/O** – examine IO latency with ASH data

```sql
SQL> @ash/event_hist db.file.*read
```

<table>
<thead>
<tr>
<th>Wait Event</th>
<th>Wait time bucket ms+</th>
<th>Num ASH Samples</th>
<th>Estimated Total Waits</th>
<th>Estimated Total ms</th>
<th>% Event Time</th>
<th>Awesome Graphic</th>
</tr>
</thead>
<tbody>
<tr>
<td>db file parallel read</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>1</td>
<td>5.1</td>
<td>128</td>
<td>8.3</td>
<td>#</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>17</td>
<td>47.2</td>
<td>4352</td>
<td>17.5</td>
<td>##</td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>18</td>
<td>24.2</td>
<td>9216</td>
<td>17.5</td>
<td>###</td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>30</td>
<td>30</td>
<td>30720</td>
<td>58.4</td>
<td>######</td>
<td></td>
</tr>
<tr>
<td>2048</td>
<td>2</td>
<td>2</td>
<td>4096</td>
<td>7.8</td>
<td>#</td>
<td></td>
</tr>
<tr>
<td>4096</td>
<td>1</td>
<td>1</td>
<td>4096</td>
<td>7.8</td>
<td>#</td>
<td></td>
</tr>
<tr>
<td>db file sequential read</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>894</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>82</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>9</td>
<td>329.3</td>
<td>144</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>34</td>
<td>729.9</td>
<td>1088</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>62</td>
<td>659.3</td>
<td>3968</td>
<td>.2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>105</td>
<td>557.9</td>
<td>13440</td>
<td>.6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>401</td>
<td>1023.3</td>
<td>102656</td>
<td>4.7</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>1071</td>
<td>1485.5</td>
<td>548352</td>
<td>25.1</td>
<td>###</td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>822</td>
<td>822</td>
<td>841728</td>
<td>38.6</td>
<td>######</td>
<td></td>
</tr>
<tr>
<td>2048</td>
<td>189</td>
<td>189</td>
<td>387072</td>
<td>17.7</td>
<td>##</td>
<td></td>
</tr>
</tbody>
</table>

**IO subsystem is slow, as single block reads often take tens to hundreds of ms**
### Commit Wait Class – slow IO

```sql
SQL> @ash/event_hist "log file.*"
```

<table>
<thead>
<tr>
<th>Wait Event</th>
<th>Wait time bucket</th>
<th>Num ASH Samples</th>
<th>Estimated Total Waits</th>
<th>Estimated Total ms</th>
<th>% Event Time</th>
<th>Awesome Graphic</th>
</tr>
</thead>
<tbody>
<tr>
<td>log file parallel write</td>
<td>0</td>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1</td>
<td>72.4</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>4</td>
<td>150.1</td>
<td>64</td>
<td>.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>4</td>
<td>93.7</td>
<td>128</td>
<td>.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>8</td>
<td>90</td>
<td>512</td>
<td>.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>6</td>
<td>30.4</td>
<td>768</td>
<td>.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>41</td>
<td>103.5</td>
<td>10496</td>
<td>8.5</td>
<td>#</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>77</td>
<td>117</td>
<td>39424</td>
<td>32</td>
<td>###</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>60</td>
<td>60</td>
<td>61440</td>
<td>49.9</td>
<td>####</td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>5</td>
<td>5</td>
<td>10240</td>
<td>8.3</td>
<td>#</td>
</tr>
</tbody>
</table>

IO subsystem is slow, as the "log file flush" writes take hundreds or thousands of ms

<table>
<thead>
<tr>
<th>log file sync</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>1</td>
<td>38.5</td>
<td>16</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>1</td>
<td>17</td>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>4</td>
<td>45.3</td>
<td>256</td>
<td>.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>5</td>
<td>26.2</td>
<td>640</td>
<td>.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>24</td>
<td>65.4</td>
<td>6144</td>
<td>2.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>80</td>
<td>102.5</td>
<td>40960</td>
<td>19.5</td>
<td>###</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>72</td>
<td>72</td>
<td>73728</td>
<td>35.1</td>
<td>####</td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>37</td>
<td>37</td>
<td>75776</td>
<td>36.1</td>
<td>####</td>
</tr>
<tr>
<td></td>
<td>4096</td>
<td>3</td>
<td>3</td>
<td>12288</td>
<td>5.9</td>
<td>#</td>
</tr>
</tbody>
</table>
### SQL> `@ash/event_hist "log file"`

<table>
<thead>
<tr>
<th>Wait Event</th>
<th>Wait time</th>
<th>Num ASH</th>
<th>Estimated Total Waits</th>
<th>Estimated % Time</th>
<th>Awesome</th>
<th>Graphic</th>
</tr>
</thead>
<tbody>
<tr>
<td>log file parallel write</td>
<td>0</td>
<td>12</td>
<td>30717.1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>27</td>
<td>40739.8</td>
<td>27</td>
<td>93.1</td>
<td>#######</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>360.8</td>
<td>2</td>
<td>6.9</td>
<td>#</td>
</tr>
<tr>
<td>log file sync</td>
<td>0</td>
<td>5</td>
<td>9337.5</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>12</td>
<td>17268.9</td>
<td>12</td>
<td>75</td>
<td>#######</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>631.5</td>
<td>4</td>
<td>25</td>
<td>###</td>
</tr>
</tbody>
</table>

Commits are fast, every "log file sync" takes 1-2ms

### Find top committers

SQL> `@ashtop username,program "event='log file sync'" sysdate-1/24 sysdate`

<table>
<thead>
<tr>
<th>%This</th>
<th>USERNAME</th>
<th>PROGRAM</th>
<th>TotalSeconds</th>
<th>CPU</th>
<th>User</th>
<th>I/O</th>
<th>Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>SOE</td>
<td>JDBC Thin Client</td>
<td>247</td>
<td>0</td>
<td>0</td>
<td>247</td>
<td></td>
</tr>
</tbody>
</table>
**Application Wait Class – lock contention**

- Locks taken and released at will by the application code (TM,TX) are listed in Application wait class (not Concurrency)
  - If there's a single session causing most of the **blocking**, see what that session was doing while others waited on its lock
  - Just query ASH data for that session only (within the blocking time range)

SQL> @ashtop event,blocking_session_status,blocking_session,blocking_session_serial#
     wait_class='Application' sysdate-1/24 sysdate

%This EVENT                      BLOCKING_SESSION BLOCKING_SESSION_SERIAL#  Seconds
Application                      ----------------- ----------------- ---------------

- 100% enq: TM - contention      VALID              5               14279           5857
- 0% enq: TM - contention         GLOBAL             11              
- 0% enq: KO - fast object checkpoint  VALID          1166             1              8

Concurrency Wait Class

- Concurrency waits are generally various blocking *internal* locks of which the application does not have direct control
  - Buffer pins, latches, mutexes, internal enqueues etc.
  - Some wait events populate the blocking session info, some don't

SQL> @ashtop event,blocking_session_status,blocking_session
    wait_class='Concurrency' sysdate-1/24 sysdate

%This EVENT B_STATUS B_SESSION Seconds CPU Concurrency
----- ------------------ -------------- ------- ------- ---------------
77% library cache: mutex X UNKNOWN 27 0 27
6% buffer busy waits VALID 1754 2 0 2
6% cursor: pin S UNKNOWN 2 0 2
3% buffer busy waits VALID 2334 1 0 1
3% latch: cache buffers chains UNKNOWN 1 0 1
3% latch: cache buffers chains VALID 1171 1 0 1
3% os thread startup UNKNOWN 1 0 1
Wait event parameters

• Every wait can post up to 3 "additional info" parameters with the wait
  • Shown as P1, P2, P3 (or PARAMETER1-3) in various views and traces
  • Contents dependent on the wait type
  • P1 for latch waits is the *latch address*
  • P1 for disk read wait events is usually the *file#* of the read block
  • V$EVENT_NAME and ASH have P1TEXT, P2TEXT, P3TEXT columns with explanation
Using wait event parameters - 1

SQL> @ashtop p1text,p1,p2text,p2,p3text,p3 "event='buffer busy waits'"
sysdate-1/24  sysdate

%This  P1TEXT     P1  P2TEXT     P2  P3TEXT     P3  TotalSeconds
-----  ---------  -----  ---------  -----  ---------  ----  ----------
 67%    file#     7  block#     43120238 class#    1       2
 33%    file#     7  block#     43120039 class#    1       1

SQL> @bclass 1

CLASS  UNDO_SEGMENT_ID
--------------------  ---------------------
data block

--- find which segment this block belongs to
SQL> @dba2 7 43120238

OWNER  SEGMENT_NAME
-------------------  ------------
SOE    LOGON

Find the hot block using P1/P2 and block type with P3
Using wait event parameters - 2

```
SQL> @ashtop p1text,p1,p2text,
   TO_NUMBER(TRIM(SUBSTR(TO_CHAR(p2,'0XXXXXXXXXXXXXXX'),1,8)),'XXXXXXXXXXXXXXXX')
"event='library cache: mutex X'" sysdate-1/24 sysdate

%This P1TEXT             P1 P2TEXT     TO_NUMB TotalSeconds CPU  Concurrency
----- -------------- ---------- ---------- -------------- ----- -----------
13% idn  4138265817 value  170    3       0         3
 8% idn  539807965 value  170    2       0         2
 8% idn  1792366761 value  121    2       0         2
 4% idn  172627511 value  170    1       0         1
 4% idn  421399181 value  73      1
 4% idn  421399181 value  109     1

SQL> SELECT type,owner,name,locked_total,pinned_total
   FROM v$db_object_cache WHERE hash_value = 4138265817;

TYPE                 OWNER      NAME                 LOCKED_TOTAL PINNED_TOTAL
-------------------- ---------- ------------------- ----------- ---------------
PACKAGE BODY         SYS        DBMS_RANDOM                  329    33065680
```

The P1 for this wait event is the hash value of library cache object. P2 is tricky, but contains the blocking SID.
What is the CPU Wait in EM Performance Page?

CPU Wait is an EM derived metric and is the delta between estimated CPU usage from ASH (AAS) and measured CPU usage from time model.
Manually Querying ASH: How much history in memory?

- **11.2+**

```sql
SQL> SELECT inst_id, oldest_sample_time, sysdate - oldest_sample_time ASH_in_memory
    FROM gv$ash_info;
```

<table>
<thead>
<tr>
<th>INST_ID</th>
<th>OLDEST_SAMPLE_TIME</th>
<th>ASH_IN_MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>25-APR-13 08.32.44.255000000 AM</td>
<td>+00000000000 03:53:22.745000000</td>
</tr>
<tr>
<td>1</td>
<td>25-APR-13 08.22.18.382000000 AM</td>
<td>+0000000000 04:03:48.618000000</td>
</tr>
</tbody>
</table>

- **10g+**

```sql
SQL> SELECT inst_id, min(sample_time), sysdate - MIN(sample_time) ASH_in_memory
    FROM gv$active_session_history GROUP BY inst_id;
```

<table>
<thead>
<tr>
<th>INST_ID</th>
<th>MIN(SAMPLE_TIME)</th>
<th>ASH_IN_MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25-APR-13 08.21.37.284 AM</td>
<td>+00000000000 04:03:51.716</td>
</tr>
<tr>
<td>2</td>
<td>25-APR-13 08.32.13.195 AM</td>
<td>+00000000000 03:53:15.805</td>
</tr>
</tbody>
</table>
ASH and background processes?

- Every background process also has a session in V$SESSION
  - Except Dispatchers (D000), Idle Shared Servers (S000) and Idle PX slaves (P000) and DISM

- You can troubleshoot background process activity with ASH and wait events too!
  - Just like with foreground processes
Drilling down into SQL Execution
SQL Monitoring: Example

Most of DB Time is spent on Waiting (blue)

Focus on line where most of waiting took place
A query bottlenecked by data processing, not retrieval

- A query bottlenecked by data processing, not retrieval
- Hash joins and a GROUP BY spilling to TEMP

![Image of database query statistics]

**Overview**

- SQL ID: 7jsk74a561m2x
- Execution Started: Tue Mar 6, 2012 10:57:26 AM
- Last Refresh Time: Tue Mar 6, 2012 11:07:58 AM
- Execution ID: 16777217
- User: TANEL
- Fetch Calls: 0

**Time & Wait Statistics**

- Duration: 10.5m
- Database Time: 10.5m
- PL/SQL & Java: 0.0s
- Wait Activity %: 100

**IO Statistics**

- Buffer Gets: 1,352K
- IO Requests: 86K
- IO Bytes: 26GB
- Cell Offload Efficiency: -14.94%

**Details**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Name</th>
<th>Estim Cost</th>
<th>Cost</th>
<th>Timeline (6s)</th>
<th>Ex...</th>
<th>Act...</th>
<th>Me...</th>
<th>Tem...</th>
<th>IO Bytes</th>
<th>Cell...</th>
<th>CPU Activity %</th>
<th>Wait Activity %</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE TABLE STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOAD AS SELECT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HASH GROUP BY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HASH JOIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HASH JOIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABLE ACCESS STORAGE ...</td>
<td>ORDERS</td>
<td>45M</td>
<td>96K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABLE ACCESS STORAGE ...</td>
<td>CUSTOMERS</td>
<td>40M</td>
<td>105I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABLE ACCESS STORAGE ...</td>
<td>ORDER_ITEMS</td>
<td>135M</td>
<td>134I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TIP: Right mouse click on the table allows to toggle between IO Requests and IO Bytes
Reading a SQL Monitoring report – in 3 steps

1. **Database Time** breakdown
   - The total time actively consumed by *this* execution of the query
   - Activity time of QC + all PX slaves if parallel execution
   - Q: Mostly on CPU or mostly waiting?

2. **Activity %** breakdown
   - If DB Time was spent mostly on CPU then look into **CPU Activity** % column
   - If DB Time was spent mostly on waiting then **Wait Activity** % column
   - This tells you in which execution plan line(s) most of the time was spent

1. **Executions, IO Bytes/Requests, Actual Rows** returned metrics tell you how much work the row sources had to do
SQL Monitoring – Duration vs DB Time?

- **Duration** is the amount of wall-clock time from the *execution start*, up to *closing the cursor* (or reaching end of data)
  - If you open a cursor and don't fetch for a while, the Duration still keeps increasing

- **DB Time** is the amount of database time (activity inside the DB) your session (and all its PX slaves, if any) spent executing this SQL
  - So if you run an 1-hour DML statement with 8 parallel slaves, you may see DB Time up to 9 hours (up to 1h for the QC, up to 8h for all PX slaves)
DB Time much smaller than Duration?!

Duration = 15 seconds (wall-clock time)

DB Time = 1.3 seconds (time spent inside DB)

332 000 rows were returned, fetched 10 rows at a time (33167 fetches).
Most time spent on network roundtrips outside DB
DB Time much bigger than Duration?

**General**
- SQL Text: `SELECT /*+ PARALLEL(4) */ COUNT(distinct PROD_I` ...
- Execution Plan: 4
- Execution ID: 16777216
- User: SYS
- Fetch Calls: 1

**Time & Wait Statistics**
- Duration: 9.0s
- Database Time: 35.9s
- PL/SQL & Java: 0s
- Activity %: 100%

**Plan Statistics**

<table>
<thead>
<tr>
<th>Parallel Server</th>
<th>Database Time</th>
<th>Activity %</th>
<th>IO Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Set 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Server 1 (p000)</td>
<td>8.4ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Server 2 (p001)</td>
<td>6.0ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Server 3 (p002)</td>
<td>7.2ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Server 4 (p003)</td>
<td>5.3ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Set 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Server 1 (p004)</td>
<td>9.0s</td>
<td>25</td>
<td>2,408</td>
</tr>
<tr>
<td>Parallel Server 2 (p005)</td>
<td>9.0s</td>
<td>25</td>
<td>2,411</td>
</tr>
<tr>
<td>Parallel Server 3 (p006)</td>
<td>8.8s</td>
<td>25</td>
<td>2,377</td>
</tr>
<tr>
<td>Parallel Server 4 (p007)</td>
<td>9.1s</td>
<td>25</td>
<td>2,409</td>
</tr>
</tbody>
</table>

**Parallel Execution!**
Each slave can use up to 1 second of DB Time per wall-clock second (plus QC)
DB Time much bigger than Duration?

Break down this SQL execution activity (across all PX slaves) by wait event or plan line.
# ASH-based “SQL monitoring” report: asqlmon.sql

- Deliberately wide output – highlight a row and scroll left/right:

```sql
SQL> @ash/asqlmon 8dq0v1mjngj7t 1
```

<table>
<thead>
<tr>
<th>SEC Activity</th>
<th>Visual ID</th>
<th>OPERATION</th>
<th>STATE</th>
<th>EVENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>.3 %</td>
<td>0 SELECT STATEMENT</td>
<td>ON CPU</td>
<td></td>
</tr>
<tr>
<td>%</td>
<td>1 COUNT STOPKEY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1257</td>
<td>58.4 %</td>
<td>2 TABLE ACCESS BY INDEX ROWID [CUSTOMERS]</td>
<td>WAITING</td>
<td>db file sequential read</td>
</tr>
<tr>
<td>2</td>
<td>.1 %</td>
<td>2</td>
<td>ON CPU</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>.5 %</td>
<td>2</td>
<td>WAITING</td>
<td>db file scattered read</td>
</tr>
<tr>
<td>2</td>
<td>.1 %</td>
<td>3</td>
<td>INDEX UNIQUE SCAN [CUSTOMERS_PK]</td>
<td>ON CPU</td>
</tr>
<tr>
<td>28</td>
<td>1.3 %</td>
<td>3</td>
<td>WAITING</td>
<td>db file scattered read</td>
</tr>
<tr>
<td>845</td>
<td>39.3 %</td>
<td>3</td>
<td></td>
<td>db file sequential read</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OBJ_ALIAS_QBC_NAME</th>
<th>ASQLMON_PREDICATES</th>
<th>PROJECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>[SEL$1]</td>
<td>[F:]ROWNUM&lt;:B1</td>
<td>&quot;CUSTOMER_ID&quot;[NUMBER,22], &quot;CUST_FIRST_NAME&quot;[VARCHAR2,30],</td>
</tr>
<tr>
<td>CUSTOMERS@SEL$1</td>
<td>[SEL$1]</td>
<td>&quot;CUSTOMER_ID&quot;[NUMBER,22], &quot;CUST_FIRST_NAME&quot;[VARCHAR2,30],</td>
</tr>
<tr>
<td>CUSTOMERS@SEL$1</td>
<td>[SEL$1]</td>
<td>&quot;CUSTOMER_ID&quot;[NUMBER,22], &quot;CUST_FIRST_NAME&quot;[VARCHAR2,30],</td>
</tr>
<tr>
<td>CUSTOMERS@SEL$1</td>
<td>[SEL$1]</td>
<td>&quot;CUSTOMER_ID&quot;[NUMBER,22], &quot;CUST_FIRST_NAME&quot;[VARCHAR2,30],</td>
</tr>
<tr>
<td>CUSTOMERS@SEL$1</td>
<td>[SEL$1]</td>
<td>&quot;CUSTOMER_ID&quot;[NUMBER,22], &quot;CUST_FIRST_NAME&quot;[VARCHAR2,30],</td>
</tr>
<tr>
<td>CUSTOMERS@SEL$1</td>
<td>[SEL$1]</td>
<td>&quot;CUSTOMER_ID&quot;[NUMBER,22], &quot;CUST_FIRST_NAME&quot;[VARCHAR2,30],</td>
</tr>
</tbody>
</table>

*Deliberately wide output – highlight a row and scroll left/right:*
Other stuff, if there's time
What does ASH have that V$SESSION doesn’t?

• SQL Plan Hash Value
  • It’s retrievable from X$KSUSE.KSUSESPH though

• PLAN_LINE columns (11g+)
  • There are less awesome workarounds (like using statistics_level = all for select queries)

• Time Model columns
  • Use V$SESS_TIME_MODEL to get session level breakdown
  • Snapper shows session time model metrics in “TIME” section

• Query Coordinator SID & instance info
  • It’s retrievable from V$PX_SESSION
  • use px.sql & pxs.sql scripts
What does SQL Trace have that ASH doesn’t?

- Full chronologically ordered trace of all SQL executions
  - Regardless of how fast they completed
  - Regardless of recursive depth
  - Row-counts returned from any step in the execution plan!

- Every single bind variable value put in place during the tracing
  - Some exotic datatypes can’t be dumped human-readably though

- Every single wait event
  - P1/P2/P3 values
  - And their exact measured durations

- Noticeable overhead, when enabled
  - In addition to tracefile writing, it enables the rowsource-level profiling for cursors executed in the traced session (except the SQL already running)
SQL Tracing a single SQL ID (11g+)

- Use ASH data to identify problem SQL and drill down with SQL trace
  - If needed...
  - Either use a login trigger or ALTER SYSTEM

```sql
SQL> ALTER SESSION SET EVENTS 'sql_trace[SQL: 32cqz71gd8wy3] - plan_stat=all_executions,wait=true,bind=true';
Session altered.

SQL> ALTER SESSION SET EVENTS 'sql_trace[SQL: 32cqz71gd8wy3] off';
Session altered.
```

PLAN_STAT - when to dump SQL plan line level statistics

NEVER, FIRST_EXECUTION, ALL_EXECUTIONS

You can Enable other traces by SQL_ID too

```sql
ALTER SESSION SET EVENTS 'trace [RDBMS.SQL_Optimizer.*] [SQL: 32cqz71gd8wy3]';
Session altered.
```
Further Reading

• **ASH Architecture and Advanced Usage** presentation
  • By Graham Wood, Uri Shaft, John Beresniewicz of Oracle Corp
  • About ASH motivation, design and internals

  • Or just google for *ASH Architecture and Advanced Usage*

• ASH-simulation without the Diagnostics Pack license (*v*$session)
  • Oracle ASH Simulation – OraSASH
  • ASH Viewer
    • [http://jonathanlewis.wordpress.com/2011/03/06/free-ash/](http://jonathanlewis.wordpress.com/2011/03/06/free-ash/)
Further Reading – Reference

- **Wait Events reference in Oracle Documentation**
  - Oracle Database Reference – Appendix C Oracle Wait events
  - [http://docs.oracle.com/cd/E11882_01/server.112/e17110/waitevents.htm](http://docs.oracle.com/cd/E11882_01/server.112/e17110/waitevents.htm)

- **Another wait event reference by Kyle Hailey**
  - [https://sites.google.com/site/embtdbo/wait-event-documentation](https://sites.google.com/site/embtdbo/wait-event-documentation)

- **V$ACTIVE_SESSION_HISTORY column reference**
  - [http://docs.oracle.com/cd/E18283_01/server.112/e17110/dynviews_1007.htm](http://docs.oracle.com/cd/E18283_01/server.112/e17110/dynviews_1007.htm)

- **V$EVENT_NAME**
  - Includes wait_class, P1,P2,P3 meanings
  - Just query it or use @sed.sql
Thanks!

My blog & scripts:

http://blog.tanelpoder.com

http://gluent.com