Sifting through the ASHes
Performance Analysis with the Oracle 10g
Active Session History

Graham Wood
Graham.Wood@oracle.com

Oracle Corporation
Agenda

- Introduction
- What is ASH
- Querying ASH data
- Comparison of ASH and Statspack/AWR
- Comparison of ASH and SQL trace/tkprof
- EM use of ASH data
- Conclusions
Oracle Statistics

- **Instance level statistics (AWR, Statspack)**
  - Too little detail $\Rightarrow$ Stop short of complete diagnosis
  - Can be collected automatically

- **Trace level statistics (sql_trace)**
  - Too much detail $\Rightarrow$ Intrusive. Hard to see big-picture
  - Must be enabled manually
  - Need prior knowledge that problem exists
Oracle Statistics

- **Solution: Active Session History**
  - Sample session activity in the system including:
    - Session id
    - Wait event
    - SQL id
    - Object
  - Always on for first fault analysis
  - Just right!
Active Session History

- Sampled, detailed, non-intrusive activity data
- Part of Oracle 10g
- On by default
- Licensed as part of the Diagnostic pack
Active Session History (ASH)

- Samples ‘Active’ sessions every second
  - Like doing “select * from v$session_wait” w/o SQL
- Writes into ASH buffer in SGA memory
  - 2MB per CPU, ≤ 5% shared_pool, 2% sga_target
- ‘Active’ == Non-idle sessions
  - Waiting on non-idle event or on CPU
- Data volume based on activity
  - 10,000 sessions => 200 active sessions
  - Design goal: one hour activity held in memory
Active Session History (ASH)

SQL> select * from v$sgastat where name like 'ASH buffers';

<table>
<thead>
<tr>
<th>POOL</th>
<th>NAME</th>
<th>.GetBytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>shared pool</td>
<td>ASH buffers</td>
<td>65011712</td>
</tr>
</tbody>
</table>

SQL> select min(sample_time), max(sample_time) from v$active_session_history;

<table>
<thead>
<tr>
<th>MIN(SAMPLE_TIME)</th>
<th>MAX(SAMPLE_TIME)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20-FEB-05 10.31.38.615 PM</td>
<td>21-FEB-05 02.39.28.950 AM</td>
</tr>
</tbody>
</table>
Active Session History (ASH)

Query for Melanie Craft Novels

Browse and Read Reviews

Add item to cart

Checkout using 'one-click'

DB: Time
## Active Session History (ASH)

### Timeline
- **Query for Melanie Craft Novels** (7:38:26)
- **Browse and Read Reviews** (7:42:35)
- **Add item to cart** (7:50:59)
- **Checkout using ‘one-click’** (7:52:33)

### Table
<table>
<thead>
<tr>
<th>Time</th>
<th>SID</th>
<th>Module</th>
<th>SQL ID</th>
<th>State</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:38:26</td>
<td>213</td>
<td>Book by author</td>
<td>qa324jffritcf</td>
<td>WAITING</td>
<td>db file sequential read</td>
</tr>
<tr>
<td>7:42:35</td>
<td>213</td>
<td>Get review id</td>
<td>aferv5desfzs5</td>
<td>CPU</td>
<td></td>
</tr>
<tr>
<td>7:50:59</td>
<td>213</td>
<td>Add to cart</td>
<td>hk32pekpcbdf</td>
<td>WAITING</td>
<td>buffer busy wait</td>
</tr>
<tr>
<td>7:52:33</td>
<td>213</td>
<td>One click</td>
<td>abngldf95f4de</td>
<td>WAITING</td>
<td>log file sync</td>
</tr>
</tbody>
</table>

*Oracle*
ASH: On disk

- Captured as part of AWR snapshots
  - DBA_HISTACTIVE_SESS_HISTORY
- Takes samples from in-memory ASH
  - 10 second samples
- On-demand flush if required
  - Whenever circular buffer is 66% full
  - No missed data
- Seven days history by default
  - Table is partitioned for easy purging
Active Session History

- **V$SESSION**
  - **V$SESSION_WAIT**
- **Session state objects**
- **MMON Lite (MMNL)**

**Circular buffer in SGA**
- (2MB per CPU)

**AWR**
- Every 1 hour
- or out-of-space

**DBA_HIST_ACTIVE_SESS_HISTORY**

**WRH$ACTIVE_SESSION_HISTORY**
ASH: Challenges – Space

- Memory Usage
  - Module, Action, Client_id (~50%)
  - Variable length rows
- Disk Usage
  - Write 1 out of every 10 samples
- Log generation
  - Direct-path INSERTS
Active Session History

Session state objects

V$SESSION
V$SESSION_WAIT

Session
state
objects

MMON Lite (MMNL)

V$ACTIVE_SESSION_HISTORY

Circular buffer in SGA
(2MB per CPU)

Write 1 out of 10 samples

Every 1 hour
or out-of-space

Variable length rows

DBA_HIST_ACTIVE_SESS_HISTORY

AWR

Direct-path INSERTS
ASH: Challenges – Time

- Reader-Writer Concurrency
  - No Consistent-Read requirement
  - 1 Writer – Multiple Readers
  - Readers go unlatched
- Indexed on time
  - Both V$ view and DBA_HIST view
Active Session History

- V$SESSION
- V$SESSION_WAIT
- MMON Lite (MMNL)
- V$ACTIVE_SESSION_HISTORY
- DBA_HIST_ACTIVE_SESS_HISTORY

Session state objects

Readers go unlatched
Indexed on time
Indexed on time

Circular buffer in SGA
(2MB per CPU)

Every 1 hour or out-of-space

Writer goes one direction
Readers go the opposite way

ORACLE
What you can do with it

• STATISTICAL analysis of where time was being spent by many different dimension.
  – What events were taking most time?
  – What was a session doing?
  – What does a SQL statement wait for?
• Can decide on dimension after the event!
ASH: Dimensions

- Session
- Waits
  - Event, P1, P2, P3
- SQL
  - Sql_id, Opcode, Plan_hash
- Objects
  - Object#, File#, Block#
- Application
  - Program, Module, Action, Client_id, Service
- Combinations of the above, CUBEs, ROLLUPs, …
Accessing ASH data

- Dump to trace file
- V$ACTIVE_SESSION_HISTORY
- DBA_HIST_ACTIVE_SESS_HISTORY
- ASH report
- EM Diagnostic Pack
Dumping ASH to file

> oradebug setmypid
> oradebug dump ashdump 10

> alter session set events 'immediate trace name ashdump level 10';

- 10 ==> minutes of history you want to dump
- Generated file can be loaded into database using supplied control file
  rdbms/demo/ashldr.ctl
V$ACTIVE_SESSION_HISTORY

- Gives most recent data first
- Control C or ‘set pause on’ is your friend
- Simpleash.sql
## ASH: desc v$active_session_history

<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>SESSION_ID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SESSION_SERIAL#</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>USER_ID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SESSION_TYPE</td>
<td></td>
<td>VARCHAR2(10)</td>
</tr>
<tr>
<td>SESSION_STATE</td>
<td></td>
<td>VARCHAR2(7)</td>
</tr>
<tr>
<td>QC_SESSION_ID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>QC_INSTANCE_ID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>EVENT</td>
<td></td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>EVENT_ID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>EVENT#</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SEQ#</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>P1</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>P2</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>P3</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SQL_ID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SQL_CHILD_NUMBER</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SQL_PLAN_HASH_VALUE</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SQL_OPCODE</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>CURRENT_OBJ#</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>CURRENT_FILE#</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>CURRENT_BLOCK#</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>PROGRAM</td>
<td></td>
<td>VARCHAR2(48)</td>
</tr>
<tr>
<td>MODULE</td>
<td></td>
<td>VARCHAR2(48)</td>
</tr>
<tr>
<td>ACTION</td>
<td></td>
<td>VARCHAR2(32)</td>
</tr>
<tr>
<td>CLIENT_ID</td>
<td></td>
<td>VARCHAR2(64)</td>
</tr>
<tr>
<td>SERVICE_HASH</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>WAIT_TIME</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>TIME_WAITED</td>
<td></td>
<td>NUMBER</td>
</tr>
</tbody>
</table>
How to Sift the ASHes

- “group by”s and “count(*)”s
  - Proxy for non-idle elapsed time
  - Proportions of actual time spent
- Can analyze any time slice
- More samples ⇒ More accurate results
ASH: Top SQL

- `select sql_id, count(*), round(count(*) / sum(count(*)) over (), 2) pctload` from `v$active_session_history`
  where `sample_time > sysdate - 1/24/60`
  and `session_type <> 'BACKGROUND'`
  group by `sql_id`
  order by `count(*) desc`;

- Returns most active SQL in the past minute
## ASH: Top SQL

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>COUNT(*)</th>
<th>PCTLOAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>25wtt4ycbtkyz</td>
<td>456</td>
<td>32.95</td>
</tr>
<tr>
<td>7umwqvvcy7tusf</td>
<td>123</td>
<td>8.89</td>
</tr>
<tr>
<td>01vunx6d35khz</td>
<td>119</td>
<td>8.6</td>
</tr>
<tr>
<td>bdyq2uph07cmp</td>
<td>102</td>
<td>7.37</td>
</tr>
<tr>
<td>9y4f9n5hr23yr</td>
<td>73</td>
<td>5.27</td>
</tr>
<tr>
<td>0bnc9a5k5f4wn</td>
<td>57</td>
<td>4.12</td>
</tr>
<tr>
<td>bv1gn48hgxpk</td>
<td>57</td>
<td>4.12</td>
</tr>
<tr>
<td>gq82c5361nxbq</td>
<td>57</td>
<td>4.12</td>
</tr>
<tr>
<td>djzkbxr7cm122</td>
<td>57</td>
<td>4.12</td>
</tr>
<tr>
<td>b2bakhq4w7rbv</td>
<td>57</td>
<td>4.12</td>
</tr>
<tr>
<td>8jydryyvdwcpp</td>
<td>57</td>
<td>4.12</td>
</tr>
<tr>
<td>69x6zf5myht7s</td>
<td>57</td>
<td>4.12</td>
</tr>
<tr>
<td>2ccwhzy8b7ua</td>
<td>57</td>
<td>4.12</td>
</tr>
<tr>
<td>4z5z7xb2g04m6</td>
<td>55</td>
<td>3.97</td>
</tr>
</tbody>
</table>
ASH: Top IO SQL

- select ash.sql_id, count(*)
  from v$active_session_history ash,
       v$event_name evt
  where ash.sample_time > sysdate - 1/24/60
    and ash.session_state = 'WAITING'
    and ash.event_id = evt.event_id
    and evt.wait_class = 'User I/O'
  group by sql_id
  order by count(*) desc;

- Returns SQL spending most time doing I/Os
- Similarly, can do Top Sessions, Top Files, Top Objects
DBA_HIST_ACTIVE_SESS_HISTORY

- Similar to in-memory ASH but adds
  - DB_ID
  - INSTANCE_NUMBER
  - SNAP_ID
- One sample every 10 seconds
ASH data gotcha’s

- Samples are a proxy for time not for counts
- Times are sampled times, not statistically valid for avg, min, max
- Beware of Obj#, File#, Block# (not cleared)
- Temp file numbers
- Wait time vs Time waited
- SQL*Forms RPC bug# 4137362
- Time period of data available in V$ACTIVE_SESSION_HISTORY is variable
ASH: Bad SQL

select
e.event,
e.total_waits - nvl(b.total_waits,0) total_waits,
e.time_waited - nvl(b.time_waited,0) time_waited
from
v$active_session_history b,
v$active_session_history e,
stats$snapshot sn
Where snap_time > sysdate-1
And e.event not like '%timer'
And e.event not like '%message%
And e.event not like '%slave wait%
And e.snap_id = sn.snap_id
And b.snap_id = e.snap_id-1
And b.event = e.event
And e.total_timeouts > 100
And (e.total_waits - b.total_waits > 100
  or e.time_waited - b.time_waited > 100)
;
ASH: Bad SQL

select sum(a.time_waited) total_time
from v$active_session_history a,
     v$event_name b
where a.event# = b.event# and
    sample_time > '21-NOV-04 12:00:00 AM' and
    sample_time < '21-NOV-04 05:00:00 AM' and
    b.wait_class = 'User I/O'
ASH: Bad SQL

```
select sum(a.time_waited) total_time
from   v$active_session_history a,
       v$event_name b
where  a.event# = b.event# and
       sample_time > '21-NOV-04 12:00:00 AM' and
       sample_time < '21-NOV-04 05:00:00 AM' and
       b.wait_class = 'User I/O'
```

- Total time spent waiting on IO?
ASH: Bad SQL

```sql
select sum(a.time_waited) total_time
from v$active_session_history a,
     v$event_name b
where a.event# = b.event# and
    sample_time > '21-NOV-04 12:00:00 AM' and
    sample_time < '21-NOV-04 05:00:00 AM' and
    b.wait_class = 'User I/O'
```

- Total time spent waiting on IO?
- Totals **sampled** IO times
ASH: Bad SQL

```sql
select sum(a.time_waited) total_time
from v$active_session_history a, v$event_name b
where a.event# = b.event# and
  sample_time > '21-NOV-04 12:00:00 AM' and
  sample_time < '21-NOV-04 05:00:00 AM' and
  b.wait_class = 'User I/O'
```

- Total time spent waiting on IO?
- Totals sampled IO times
- Assumes that 5 hours history in memory
ASH: Bad SQL

```
select sess_id, username, program, wait_event, sess_time,
      round(100*(sess_time/total_time), 2) pct_time_waited
from
  (select a.session_id sess_id,
         decode(session_type, 'background', session_type, c.username) username,
         a.program program,
         b.name wait_event,
         sum(a.time_waited) sess_time
  from sys.v_$active_session_history a,
       sys.v_$event_name b,
       sys.dba_users c
  where a.event# = b.event# and
        a.user_id = c.user_id and
        sample_time > '21-NOV-04 12:00:00 AM' and
        sample_time < '21-NOV-04 05:00:00 AM' and
        b.wait_class = 'User I/O'
  group by a.session_id,
         decode(session_type, 'background', session_type, c.username),
         a.program,
         b.name),
  (select sum(a.time_waited) total_time
  from sys.v_$active_session_history a)
```
ASH: WAIT_TIME vs TIME_WAITED

- **WAIT_TIME**
  - Same as V$SESSION_WAIT
  - 0 ⇒ ‘WAITING’
    any other value ⇒ ‘ON CPU’

- **TIME_WAITED**
  - Actual time waited for that event
  - Updated later upon event completion
**ASH: TIME_WAITED**

User Session (SID = 3)

**Time**

**Sample 1**

<table>
<thead>
<tr>
<th>Sample</th>
<th>Session</th>
<th>State</th>
<th>Event</th>
<th>Wait_time</th>
<th>Time_waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>WAITING</td>
<td>db file scattered read</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Sample 2**

After Sample 2, Sample 1 is updated

<table>
<thead>
<tr>
<th>Sample</th>
<th>Session</th>
<th>State</th>
<th>Event</th>
<th>Wait_time</th>
<th>Time_waited</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>WAITING</td>
<td>db file scattered read</td>
<td>0</td>
<td>5ms</td>
</tr>
</tbody>
</table>
## ASH vs AWR/Statpack

<table>
<thead>
<tr>
<th>Feature</th>
<th>ASH</th>
<th>AWR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance wide data</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Time based data</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Counts/occurrence data</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Analyze any time period</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Detailed session level data</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Individual Wait event data</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Sampled data</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Time based analysis</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
ASH vs AWR

Top 5 Timed Events
~~~~~~~~~~~~~~~~~~

<table>
<thead>
<tr>
<th>Event</th>
<th>Waits</th>
<th>Time (s)</th>
<th>DB Time</th>
<th>Wait Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>log file sync</td>
<td>990,495</td>
<td>233,649</td>
<td>43.79</td>
<td>Commit</td>
</tr>
<tr>
<td>latch: library cache</td>
<td>642,247</td>
<td>157,188</td>
<td>29.46</td>
<td>Concurrency</td>
</tr>
<tr>
<td>latch: cache buffers chains</td>
<td>133,136</td>
<td>39,747</td>
<td>7.45</td>
<td>Concurrency</td>
</tr>
<tr>
<td>latch: library cache pin</td>
<td>84,638</td>
<td>22,998</td>
<td>4.31</td>
<td>Concurrency</td>
</tr>
<tr>
<td>latch free</td>
<td>61,709</td>
<td>20,079</td>
<td>3.76</td>
<td>Other</td>
</tr>
</tbody>
</table>

Top Foreground Events
~~~~~~~~~~~~~~~~~~~~~

<table>
<thead>
<tr>
<th>Event</th>
<th>% Total</th>
<th>Wait Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>log file sync</td>
<td>46.01%</td>
<td>Commit</td>
</tr>
<tr>
<td>latch: library cache</td>
<td>23.13%</td>
<td>Concurrency</td>
</tr>
<tr>
<td>latch: cache buffers chains</td>
<td>6.50%</td>
<td>Concurrency</td>
</tr>
<tr>
<td>latch free</td>
<td>4.63%</td>
<td>Other</td>
</tr>
<tr>
<td>latch: library cache pin</td>
<td>2.99%</td>
<td>Concurrency</td>
</tr>
<tr>
<td>Feature</td>
<td>ASH</td>
<td>SQLtrace</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>Parse/Exec/Fetch breakdown</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Time based data</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Counts/occurrence data</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Detailed session level data</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Individual Wait event data</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Complete trace of operations</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Always on</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Bind variables available</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Top SQL

#### Spot SQL vs. Period SQL
- **Spot SQL** shows all the SQL statements that have been active in the last 5 minutes interval.
- **Period SQL** displays SQL statements based on a specified time period.

#### Spot Interval Selection
- Drag the shaded box to select the 5-minute interval for which you want to view details in the section below.

#### Detail for Selected 5 minute Interval

**Start Time**: Mar 8, 2004 12:00:16 PM

**Top SQL (ordered by Activity)**

<table>
<thead>
<tr>
<th>SQL ID</th>
<th>SQL Type</th>
<th>Activity (%)</th>
<th>CPU (%)</th>
<th>Wait (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>368</td>
<td>SELECT</td>
<td>75.89</td>
<td>32.44</td>
<td>0.00</td>
</tr>
<tr>
<td>42</td>
<td>FLUSH_CACHE</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>22</td>
<td>SELECT</td>
<td>93.75</td>
<td>4.45</td>
<td>0.00</td>
</tr>
<tr>
<td>257</td>
<td>SELECT</td>
<td>70.83</td>
<td>1.21</td>
<td>0.00</td>
</tr>
<tr>
<td>238</td>
<td>SELECT</td>
<td>94.17</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>269</td>
<td>SELECT</td>
<td>41.18</td>
<td>7.5</td>
<td>0.00</td>
</tr>
<tr>
<td>27</td>
<td>SELECT</td>
<td>85.71</td>
<td>2.5</td>
<td>0.00</td>
</tr>
<tr>
<td>2597</td>
<td>SELECT</td>
<td>99.99</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2017</td>
<td>SELECT</td>
<td>80.00</td>
<td>1.21</td>
<td>0.00</td>
</tr>
<tr>
<td>2589</td>
<td>SELECT</td>
<td>96.23</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2647</td>
<td>SELECT</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2380</td>
<td>SELECT</td>
<td>99.99</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2381</td>
<td>SELECT</td>
<td>99.99</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

#### Run SQL Tuning Advisor
- Use this to run SQL Tuning Advisor and analyze the SQL performance.

#### Create SQL Tuning Set
- Create a SQL Tuning Set for better performance monitoring.

---

**ORACLE Enterprise Manager 12c**

Source: [Oracle Documentation](https://docs.oracle.com)
**Active Sessions Waiting: User I/O**

Drag the shaded box to change the time period for the detail section below.

**Detail for Selected 5 minute Interval**

Start Time: Jul 26, 2004 11:23:01 AM

**Top Waiting Sessions**

Total Sample Count: 654

- JMillUS(1564)(41.6)
- JMillUS(2111)(11.1)
- ODEBALA(1402)(17.1)
- AVIAN(1571)(1.4%)
- QA(1834)(2.3%)
- Other(13%)

**Wait Events for Top Sessions**

<table>
<thead>
<tr>
<th>User Name</th>
<th>Session ID</th>
<th>Wait Event</th>
<th>Activity (%)</th>
<th>Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMillUS</td>
<td>2111</td>
<td>db file parallel read</td>
<td>29.97%</td>
<td>JDBC Thin Client</td>
</tr>
<tr>
<td>JMillUS</td>
<td>1564</td>
<td>read by other session</td>
<td>28.29%</td>
<td>JDBC Thin Client</td>
</tr>
<tr>
<td>JMillUS</td>
<td>1564</td>
<td>db file sequential read</td>
<td>13.30%</td>
<td>JDBC Thin Client</td>
</tr>
<tr>
<td>JMillUS</td>
<td>2111</td>
<td>db file sequential read</td>
<td>7.49%</td>
<td>JDBC Thin Client</td>
</tr>
<tr>
<td>JMillUS</td>
<td>2111</td>
<td>read by other session</td>
<td>3.36%</td>
<td>JDBC Thin Client</td>
</tr>
<tr>
<td>ODEBALA</td>
<td>1402</td>
<td>db file sequential read</td>
<td>1.86%</td>
<td>http@web37 [TNS V1-V3]</td>
</tr>
<tr>
<td>AVIAN</td>
<td>1571</td>
<td>db file sequential read</td>
<td>1.38%</td>
<td>http@web37 [TNS V1-V3]</td>
</tr>
<tr>
<td>QA</td>
<td>1834</td>
<td>db file sequential read</td>
<td>1.22%</td>
<td>JDBC Thin Client</td>
</tr>
<tr>
<td>JMillUS</td>
<td>2111</td>
<td>db file scattered read</td>
<td>0.31%</td>
<td>JDBC Thin Client</td>
</tr>
</tbody>
</table>
Active Sessions Working: CPU Used

Drag the shaded box to change the time period for the detail section below.

Detail for Selected 30 Minute Interval
Start Time: Jan 17, 2005 7:50:27 AM

Top Working SQL

<table>
<thead>
<tr>
<th>Activity (%)</th>
<th>SQL Hash Value</th>
<th>SQL Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>61.11</td>
<td>2k9rkn75unj61</td>
<td>SELECT</td>
</tr>
<tr>
<td>18.22</td>
<td>4w188h93rmbg65</td>
<td>PL/SQL EXECUTE</td>
</tr>
<tr>
<td>4.07</td>
<td>4dqasq3r3udon</td>
<td>PL/SQL EXECUTE</td>
</tr>
<tr>
<td>3.04</td>
<td>f5b19k0906vac1</td>
<td>SELECT</td>
</tr>
<tr>
<td>2.65</td>
<td>7gpmgwwtrwv1n6</td>
<td>SELECT</td>
</tr>
<tr>
<td>2.52</td>
<td>3p7gtv3swskw0</td>
<td>SELECT</td>
</tr>
<tr>
<td>2.38</td>
<td>ftmma773x08wh0</td>
<td>SELECT</td>
</tr>
<tr>
<td>2.33</td>
<td>2nklfl8x9u3n13</td>
<td>SELECT</td>
</tr>
<tr>
<td>2.13</td>
<td>d9ghiab5y8pu</td>
<td>PL/SQL EXECUTE</td>
</tr>
<tr>
<td>1.55</td>
<td>1ztr5njbov60</td>
<td>UNKOWN</td>
</tr>
</tbody>
</table>

Total Sample Count: 1,549

Top Working Sessions

<table>
<thead>
<tr>
<th>Activity (%)</th>
<th>Session ID</th>
<th>User Name</th>
<th>Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.32</td>
<td>1175</td>
<td>V8HAVE</td>
<td>oracle@ap603rt (TNS V1-V3)</td>
</tr>
<tr>
<td>14.32</td>
<td>1367</td>
<td>V8HAVE</td>
<td>oracle@ap603rt (TNS V1-V3)</td>
</tr>
<tr>
<td>14.32</td>
<td>1535</td>
<td>V8HAVE</td>
<td>oracle@ap603rt (TNS V1-V3)</td>
</tr>
<tr>
<td>14.23</td>
<td>1622</td>
<td>V8HAVE</td>
<td>oracle@ap603rt (TNS V1-V3)</td>
</tr>
<tr>
<td>14.14</td>
<td>1273</td>
<td>V8HAVE</td>
<td>oracle@ap603rt (TNS V1-V3)</td>
</tr>
<tr>
<td>13.87</td>
<td>1167</td>
<td>V8HAVE</td>
<td>oracle@ap603rt (TNS V1-V3)</td>
</tr>
<tr>
<td>13.87</td>
<td>1843</td>
<td>NITGUPPT</td>
<td></td>
</tr>
<tr>
<td>12.39</td>
<td>2178</td>
<td>ASRAJ</td>
<td></td>
</tr>
<tr>
<td>12.39</td>
<td>2178</td>
<td>ASRAJ</td>
<td></td>
</tr>
<tr>
<td>12.39</td>
<td>886</td>
<td>VSHASTRY</td>
<td></td>
</tr>
<tr>
<td>3.06</td>
<td>2263</td>
<td>PCSADMIN</td>
<td>@ap801 utl (TNS V1-V3)</td>
</tr>
</tbody>
</table>

Total Sample Count: 1,110

Additional Monitoring Links
- Period SQL
- Instance Activity
- Snapshots
SELECT h.rptno || '-xx-' || h.cs_priority || '-xx-' || h.status || '-xx-' || h.CATEGORY || '-xx-' || replace(h.subject, chr(10)) || '-xx-...

Details

Select the plan hash value to see the details below. Plan Hash Value 4047790200

Execution Activity

The following chart displays the users executing of the wait events and CPU of the SQL statement in the last hour. Move the slider to view the session activity detail information.

Detail for Selected 5 Minute Interval

Start Time Oct 8, 2004 10:30:08 AM

<table>
<thead>
<tr>
<th>Activity (%)</th>
<th>SID</th>
<th>User</th>
<th>Program</th>
<th>Service</th>
<th>Plan Hash Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>34.21</td>
<td>1286</td>
<td>JPAL</td>
<td>soplus@ap672wgs (TNS V1-V3)</td>
<td>SYS$USERS</td>
<td>4047790200</td>
</tr>
<tr>
<td>32.71</td>
<td>1489</td>
<td>JPAL</td>
<td>soplus@ap672wgs (TNS V1-V3)</td>
<td>SYS$USERS</td>
<td>4047790200</td>
</tr>
<tr>
<td>28.57</td>
<td>1795</td>
<td>JPAL</td>
<td>soplus@ap672wgs (TNS V1-V3)</td>
<td>SYS$USERS</td>
<td>4047790200</td>
</tr>
</tbody>
</table>
ASH: What new in 10gR2

- Blocking sid (maybe in 10.1.0.5)
- XID
Conclusion

- ASH data always available
- Allows instance wide performance analysis
- Allows detailed session level performance analysis
- But it is sampled data, so use statistical analysis techniques