Active Session History: Advanced Analysis

David Kurtz
Who Am I?

- Accenture Enkitec Group
- Performance tuning
  - Oracle RDBMS
  - PeopleSoft ERP
- Book
  - [www.psftdba.com](http://www.psftdba.com)
- Oak Table
- [Oracle ACE Director](https://www.oracle.com/us/corporate/awards/ace-director/index.html)
Who Am I?

- Accenture Enkitec Group
- Performance tuning
  - Oracle RDBMS
  - PeopleSoft ERP
- Book
  - www.psftdba.com
- Oak Table
- Ace Director

ASH: Advanced Analysis ©2016
Agenda: Active Session History

- There are a lot of things that I am not going to talk about.
  - Because I want to get to the fun stuff at the end
- What is it? What does it do?
  - How does it work
  - Theory of DB Time
  - Enterprise Manager & ASH Report
  - Compare and contrast SQL*Trace
ASH Fundamentals & Theory of DB Time

- https://www.youtube.com/watch?v=rxQkvXIY7X0
Agenda: Active Session History

- Using SQL to Analyse ASH
  - Top SQL
  - Locking,
  - Changing Plans
  - I/O
  - Temporary Usage
  - Index Usage
  - Limitations & Pitfalls
If you can’t hear me say so now.

Please feel free to ask questions as we go along.

The presentation will be available from

- Conference website
- http://www.go-faster.co.uk
Further Reading

- This presentation started out as a paper about ASH, you might find it easier to work with that than this presentation.
  - Not everything in the paper appears in the presentation.
  - On the conference website, but also at
    - [http://www.go-faster.co.uk/Practical_ASH.pdf](http://www.go-faster.co.uk/Practical_ASH.pdf)
Background Reading

- **Graham Wood**
  - Sifting through the ASHes of (DB) Time
  - ASH Architecture and Advanced Usage
    - [www.youtube.com/watch?v=rxQkvXIY7X0](https://www.youtube.com/watch?v=rxQkvXIY7X0)
- **Doug Burns’ Oracle Blog**
- **ASH Masters**
- **Introduction to DBMS_XPLAN**
  - [http://www.go-faster.co.uk/Intro_DBMS_XPLAN.ppt](http://www.go-faster.co.uk/Intro_DBMS_XPLAN.ppt)
Your Mileage May Vary

- Throughout this presentation I will be showing you examples mostly from PeopleSoft systems.
  - If you have a different package or you own application, you are likely to face similar challenges.
  - Don’t worry about the PeopleSoft specifics.
  - Focus on the kind of information I am using to filter my ASH data.
ASH is a part of the Diagnostics Pack

- That’s means it costs money.
  - I don’t like it either, but that is how it is!
- Only available on Enterprise Edition

- S-ASH: Simulated ASH
  - OraSASH: http://pioro.github.io/orasash/
ASH data in OEM

![Graph showing ASH data in OEM](image)

### Detail for Selected 5 Minute Interval

**Start Time:** 17-May-2010 09:34:13  
**View:** Show Aggregated Data

<table>
<thead>
<tr>
<th>Activity (%)</th>
<th>SQL ID</th>
<th>SQL Command</th>
<th>Plan Hash Value</th>
<th>Module</th>
<th>Action</th>
<th>Client ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.46</td>
<td>7a629083370d7</td>
<td>TRUNCATE TABLE</td>
<td>1077798166</td>
<td>SCRTY_SJTDY</td>
<td>PL-2775767</td>
<td>Processing</td>
</tr>
<tr>
<td>1.09</td>
<td>8q8m0z2oq7p</td>
<td>TRUNCATE TABLE</td>
<td>3392330812</td>
<td>SCRTY_SJTDY</td>
<td>PL-2775767</td>
<td>Processing</td>
</tr>
<tr>
<td>0.73</td>
<td>3cf74swzdgo</td>
<td>UPDATE</td>
<td>1018177566</td>
<td>SCRTY_SJTDY</td>
<td>PL-2775767</td>
<td>Processing</td>
</tr>
<tr>
<td>0.73</td>
<td>6q6ufg9c100</td>
<td>DELETE</td>
<td>2123567390</td>
<td>SCRTY_SJTDY</td>
<td>PL-2775767</td>
<td>Processing</td>
</tr>
<tr>
<td>0.73</td>
<td>8m616jhy50m68</td>
<td>UPDATE</td>
<td>1104492064</td>
<td>SCRTY_SJTDY</td>
<td>PL-2775767</td>
<td>Processing</td>
</tr>
<tr>
<td>0.36</td>
<td>3y24wzgbdv0</td>
<td>INSERT</td>
<td>2973194850</td>
<td>SCRTY_SJTDY</td>
<td>PL-2775767</td>
<td>Processing</td>
</tr>
<tr>
<td>0.36</td>
<td>8h64jbf8w4u</td>
<td>INSERT</td>
<td>2973194850</td>
<td>SCRTY_SJTDY</td>
<td>PL-2775767</td>
<td>Processing</td>
</tr>
<tr>
<td>0.36</td>
<td>3l06f2mg2ud3</td>
<td>INSERT</td>
<td>3422340867</td>
<td>SCRTY_SJTDY</td>
<td>PL-2775767</td>
<td>Processing</td>
</tr>
<tr>
<td>0.36</td>
<td>3l06f2mg2ud3</td>
<td>INSERT</td>
<td>3422340867</td>
<td>SCRTY_SJTDY</td>
<td>PL-2775767</td>
<td>Processing</td>
</tr>
</tbody>
</table>
ASH – v – SQL*Trace

• ASH
  – Licensed
  – Always there
  – No marginal cost
  – Real Time
  – Who is blocking me?
  – Statistical data
  – Plan if captured by AWR
  – Estimate of duration
    • Per wait event
    • Operation in 11gR2

• SQL*Trace
  – Free
  – Enable, File, Profile
  – Run-time overhead
  – Reactive
  – Being Blocked
  – Every SQL & event
  – Actual execution plan
  – Exact duration
    • Operations in Plan
ASH – v- SQL*Trace

- ASH can be used to resolve many of your performance issues.
- Sometimes, you will still need SQL*Trace
Application Instrumentation

• It is essential to be able to match
  – database sessions
  – application processes

• DBMS_APPLICATION_INFO
  – set_module, set_action
  – Calls in application
    • Few packaged application vendors do this
k_module CONSTANT VARCHAR2(48) := $$PLSQL_UNIT;

PROCEDURE my_procedure IS
  l_module VARCHAR2(48);
  l_action VARCHAR2(32);
BEGIN
  dbms_application_info.read_module(module_name=>l_module,
                                     action_name=>l_action);
  dbms_application_info.set_module(module_name=>k_module,
                                     action_name=>'MY PROCEDURE');

  dbms_application_info.set_module(module_name=>l_module,
                                     action_name=>l_action);
EXCEPTION
  WHEN ... THEN
    dbms_application_info.set_module(module_name=>l_module,
                                     action_name=>l_action);
    RAISE / EXIT
END my_procedure;
DIY Instrumentation

- You may need to be creative!
  - PeopleTools <8.51: When a process starts (and sets its own status), I have a trigger that sets module and action

```
CREATE OR REPLACE TRIGGER sysadm.psftapi_store_prcsinstance
BEFORE UPDATE OF runstatus ON sysadm.psprcsrqst FOR EACH ROW
WHEN ((new.runstatus IN('3','7','8','9','10') OR old.runstatus IN('7','8')) AND new.prcstype != 'PSJob')
BEGIN
...
  psftapi.set_action(p_prcsinstance=>:new.prcsinstance
                   ,p_runstatus=>:new.runstatus, p_prcsname=>:new.prcsname);
...
EXCEPTION WHEN OTHERS THEN NULL; --do not crash the scheduler
END;
/
```
No Instrumentation!

- JDBC Thin Client: 97.8%
- Others
- EM Realtime Connection
- DBMS_SCHEDULER
- GATHER_STATS
- DBMS_SCHEDULER ORA
- $AT_SA_SPC_SY_6108
- Streams QMON Coordinator
- DBMS_SCHEDULER ORA
- $AT_OS_OPT_SY_6107
With Instrumentation!
What are you looking for?

- You need a clear idea of the question you are asking of the ASH data.

- What are you interested in?
  - Time Window
    - Recent –v- Historical
  - Single Session / Group of Sessions / Whole Database
  - All ASH Data / One Event / One SQL ID / One Plan
  - Related ASH data (sessions blocked by lock)
Statistical Analysis of ASH data

• Recent
SELECT ... , SUM(1) ash_secs FROM v$active_session_history WHERE ... GROUP BY ...

• Historical
SELECT ... , SUM(10) ash_secs FROM dba_hist_active_sess_history WHERE ... GROUP BY ...
Querying ASH Repository

- **DBA_HIST_ACTIVE_SESS_HISTORY**
  - WRH$_ACTIVE_SESSION_HISTORY$
    partitioned on DBID and SNAP_ID

- **DBA_HIST_SNAPSHOT**
  - WRM$_SNAPSHOT$
Querying ASH Repository

```sql
SELECT /*+LEADING(x) USE_NL(h)*/
  ...
, SUM(10) ash_secs
FROM dba_hist_active_sess_history h
  , dba_hist_snapshot x
WHERE x.snap_id = h.snap_id
AND x.dbid = h.dbid
AND x.instance_number = h.instance_number
AND x.end_interval_time >= ...
AND x.begin_interval_time <= ...
AND h.sample_time BETWEEN ... AND ...
AND ...
GROUP BY ...
```
Example: On-Line

```
SELECT /*+LEADING(x h) USE_NL(h)*/
    h.sql_id
  ,
    h.sql_plan_hash_value
  ,
    SUM(10) ash_secs
FROM dba_hist_snapshot x
  ,
    dba_hist_active_sess_history h
WHERE x.end_interval_time
    >= TO_DATE('201402010730','yyyymmddhh24mi')
AND x.begin_interval_time
    <= TO_DATE('201402010830','yyyymmddhh24mi')
AND h.sample_time BETWEEN TO_DATE('201402010730','yyyymmddhh24mi')
    AND TO_DATE('201402010830','yyyymmddhh24mi')
AND h.SNAP_id = X.SNAP_id
AND h.dbid = x.dbid
AND h.instance_number = x.instance_number
AND h.module like 'PSAPPSRV%'
GROUP BY h.sql_id, h.sql_plan_hash_value
ORDER BY ash_secs DESC
/
```

Snapshots for Period for which process was running

ASH Data for period for which process was running

Application not instrumented Can Only Filter by Module
## Top SQL

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>Hash Value</th>
<th>ASH SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>7hvaxp65s70qw</td>
<td>1051046890</td>
<td>1360</td>
</tr>
<tr>
<td>fdudyw87n6prc</td>
<td>313261966</td>
<td>760</td>
</tr>
<tr>
<td>8d56bz2qxyw6j</td>
<td>2399544943</td>
<td>720</td>
</tr>
<tr>
<td>876mfmryd8yv7</td>
<td>1569761114</td>
<td>710</td>
</tr>
<tr>
<td>bphpwrud1q83t</td>
<td>3575267335</td>
<td>690</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Get the Execution Plan from AWR

SELECT * from table(
dbms_xplan.display_awr(
    '7hvaxp65s70qw', 1051046890, NULL,
    'ADVANCED'));
Example: A Batch Process

- In PeopleSoft: a process request table
  - one row per scheduled process.

- Process Attributes
  - Process Instance
  - Operator ID
  - Process Type & Name
  - Begin Date/Time
  - End Date/Time
Example: A Batch Process

SELECT /*+LEADING(r x h) USE_NL(h)*/
  r.prcsinstance,
  h.sql_id, h.sql_plan_hash_value,
  (r.enddttm-r.begindttm)*86400 exec_secs,
  SUM(10) ash_secs
FROM dba_hist_snapshot x,
     dba_hist_active_sess_history h,
     sysadm.pspcrsrqst r
WHERE x.end_interval_time >= r.begindttm
  AND x.begin_interval_time <= r.enddttm
  AND h.sample_time BETWEEN r.begindttm AND r.enddttm
  AND h.snap_id = x.snap_id
  AND h.dbid = x.dbid
  AND h.instance_number = x.instance_number
  AND h.module = r.prcsname
  AND h.action LIKE 'PI='||r.prcsinstance||'\%'
  AND r.prcsinstance = 1956338
GROUP BY r.prcsinstance, r.prcsname, r.begindttm,
  h.sql_id, h.sql_plan_hash_value
ORDER BY ash_secs DESC
Example: A Running Batch Process

• The process has been running for a long time.
  - “It appears to have ground to a halt.”
### Example: A Running Batch Process

```sql
SELECT /*+LEADING(r)*/
    r.prcsinstance
  , h.sql_id
  , h.sql_child_number
  , h.sql_plan_hash_value
  , (NVL(r.enddttm,SYSDATE) - r.begindttm) * 86400 AS exec_secs
  , SUM(1) AS ash_secs
  , max(sample_time) AS max_sample_time
FROM v$active_session_history h
  , sysadm.pspcrsqst r
WHERE h.sample_time BETWEEN r.begindttm AND NVL(r.enddttm, SYSDATE)
AND h.module = r.prcsname
AND h.action LIKE 'PI=' || r.prcsinstance || '%'
AND r.prcsinstance = 1561519
GROUP BY r.prcsinstance, r.prcsname, r.begindttm,
        h.sql_id, h.sql_plan_hash_value, h.sql_child_number
ORDER BY max_sample_time DESC
```

**Current Data rather than Historical ASH repository**

**Process Request Table**
- One Row per Process

**Specific Process by ID**
- Latest ASH sample for statement

**ASH Data for period for which process was running**

**Filter ASH data by MODULE and ACTION**
## Example: A Running Batch Process

<table>
<thead>
<tr>
<th>Process Instance</th>
<th>SQL_ID</th>
<th>Child No.</th>
<th>SQL Plan Hash Value</th>
<th>Exec Secs</th>
<th>ASH Secs</th>
<th>Last Running</th>
</tr>
</thead>
<tbody>
<tr>
<td>1561509</td>
<td>9yj020x2762a9</td>
<td>0</td>
<td>3972644945</td>
<td>18366</td>
<td>17688</td>
<td>19-FEB-14 04.24.41.392 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>9yj020x2762a9</td>
<td>0</td>
<td>799518913</td>
<td>18366</td>
<td>1</td>
<td>19-FEB-14 11.26.29.096 AM</td>
</tr>
<tr>
<td>1561509</td>
<td>b5r9c04ck29zb</td>
<td>1</td>
<td>149088295</td>
<td>18366</td>
<td>1</td>
<td>19-FEB-14 11.26.28.085 AM</td>
</tr>
<tr>
<td>1561509</td>
<td>5vdhh2m8skh86</td>
<td>1</td>
<td>0</td>
<td>18366</td>
<td>1</td>
<td>19-FEB-14 11.26.27.075 AM</td>
</tr>
<tr>
<td>1561509</td>
<td>gyuq5arbj7yxx</td>
<td>0</td>
<td>3708596767</td>
<td>18366</td>
<td>1</td>
<td>19-FEB-14 11.26.26.065 AM</td>
</tr>
<tr>
<td>1561509</td>
<td></td>
<td>0</td>
<td>0</td>
<td>18366</td>
<td>1</td>
<td>19-FEB-14 11.26.25.055 AM</td>
</tr>
<tr>
<td>1561509</td>
<td>5jkh8knvxw7k2</td>
<td>0</td>
<td>1549543016</td>
<td>18366</td>
<td>1</td>
<td>19-FEB-14 11.26.24.043 AM</td>
</tr>
<tr>
<td>1561509</td>
<td>9pz262n5gbhmk</td>
<td>0</td>
<td>193554713</td>
<td>18366</td>
<td>1</td>
<td>19-FEB-14 11.26.23.033 AM</td>
</tr>
<tr>
<td>1561509</td>
<td>6qg99cfg26kwb</td>
<td>1</td>
<td>3610545376</td>
<td>18366</td>
<td>1</td>
<td>19-FEB-14 11.26.22.035 AM</td>
</tr>
<tr>
<td>1561509</td>
<td>gpdwr389mg61h</td>
<td>0</td>
<td>211018888</td>
<td>18366</td>
<td>422</td>
<td>19-FEB-14 11.26.21.014 AM</td>
</tr>
<tr>
<td>1561509</td>
<td>gpdwr389mg61h</td>
<td>0</td>
<td>1518</td>
<td>18366</td>
<td>1</td>
<td>19-FEB-14 11.19.13.931 AM</td>
</tr>
<tr>
<td>1561509</td>
<td>fmbbqjm351p05q</td>
<td>0</td>
<td>4875690</td>
<td>18366</td>
<td>1</td>
<td>19-FEB-14 11.19.12.916 AM</td>
</tr>
<tr>
<td>1561509</td>
<td></td>
<td>0</td>
<td>791</td>
<td>18366</td>
<td>14</td>
<td>19-FEB-14 11.18.57.771 AM</td>
</tr>
<tr>
<td>1561509</td>
<td></td>
<td>0</td>
<td>903</td>
<td>18366</td>
<td>9</td>
<td>19-FEB-14 11.18.48.679 AM</td>
</tr>
<tr>
<td>1561509</td>
<td></td>
<td>0</td>
<td>731</td>
<td>18366</td>
<td>10</td>
<td>04.24.41.392 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>cbppam9ph5bu8</td>
<td>0</td>
<td>3488560417</td>
<td>18366</td>
<td>19</td>
<td>19-FEB-14 11.18.38.571 AM</td>
</tr>
<tr>
<td>1561509</td>
<td>3cswz2x9ubjm3</td>
<td>0</td>
<td>504495601</td>
<td>18366</td>
<td>19</td>
<td>19-FEB-14 11.18.36.541 AM</td>
</tr>
</tbody>
</table>

*This Statement has been running for a while*
Same Process, a little later

<table>
<thead>
<tr>
<th>Process Instance</th>
<th>SQL_ID</th>
<th>Child No.</th>
<th>SQL Plan Hash Value</th>
<th>Exec Secs</th>
<th>ASH Secs</th>
<th>Last Running</th>
</tr>
</thead>
<tbody>
<tr>
<td>1561509</td>
<td>5zq8mtxp0nfn8</td>
<td>0</td>
<td>1505304026</td>
<td>38153</td>
<td>1</td>
<td>19-FEB-14 09.28.52.628 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>b023ph16myv5d</td>
<td>0</td>
<td>1416307094</td>
<td>38153</td>
<td>1</td>
<td>19-FEB-14 09.28.21.300 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>14k7bqan2vh8</td>
<td>0</td>
<td>1620828024</td>
<td>38153</td>
<td>1</td>
<td>19-FEB-14 09.28.20.280 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>d2498j5x025rq</td>
<td>0</td>
<td>3746253366</td>
<td>38153</td>
<td>82</td>
<td>19-FEB-14 09.28.19.270 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>fsvwg5xnp66nf</td>
<td>0</td>
<td>3224283327</td>
<td>38153</td>
<td>43</td>
<td>19-FEB-14 09.26.54.280 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>5zq8mtxp0nfn8</td>
<td>0</td>
<td>1505304026</td>
<td>38153</td>
<td>14</td>
<td>19-FEB-14 09.24.54.853 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>495b094we2632</td>
<td>0</td>
<td>47220797</td>
<td>38153</td>
<td>1</td>
<td>19-FEB-14 09.24.27.533 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>5zq8mtxp0nfn8</td>
<td>0</td>
<td>1505304026</td>
<td>38153</td>
<td>78</td>
<td>19-FEB-14 09.24.26.523 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>5zq8mtxp0nfn8</td>
<td>0</td>
<td>1505304026</td>
<td>38153</td>
<td>42</td>
<td>19-FEB-14 09.23.07.685 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>5zq8mtxp0nfn8</td>
<td>0</td>
<td>1505304026</td>
<td>38153</td>
<td>3</td>
<td>19-FEB-14 09.22.25.207 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>5zq8mtxp0nfn8</td>
<td>0</td>
<td>1505304026</td>
<td>38153</td>
<td>13296</td>
<td>19-FEB-14 09.22.21.167 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>8za7232u5pnrnf</td>
<td>0</td>
<td>177741215</td>
<td>38153</td>
<td>1</td>
<td>19-FEB-14 05.38.13.085 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>8msvfudz3bc1w</td>
<td>0</td>
<td>450255751</td>
<td>38153</td>
<td>24</td>
<td>19-FEB-14 05.38.11.939 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>5fvtbncfpkbv</td>
<td>0</td>
<td>144493251</td>
<td>38153</td>
<td>32</td>
<td>19-FEB-14 05.37.47.615 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>59sdxn718fs8w</td>
<td>0</td>
<td>1746491243</td>
<td>38153</td>
<td>11</td>
<td>19-FEB-14 05.37.13.236 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>g0by0mjld6dy2</td>
<td>0</td>
<td>2128929267</td>
<td>38153</td>
<td>1</td>
<td>19-FEB-14 05.37.02.049 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>7sx5plug5aqi2</td>
<td>1</td>
<td>2873308018</td>
<td>38153</td>
<td>1</td>
<td>19-FEB-14 05.37.01.033 PM</td>
</tr>
<tr>
<td>1561509</td>
<td>9yj020x2762a9</td>
<td>0</td>
<td>3972644945</td>
<td>38153</td>
<td>13295</td>
<td>19-FEB-14 05.36.59.620 PM</td>
</tr>
</tbody>
</table>
Obtain Execution Plan from Library Cache

```
SELECT * from table(
  dbms_xplan.display_cursor('9yj020x2762a9', 0, 'ADVANCED'));
```
Obtain Execution Plan from AWR

SELECT * from table(
    dbms_xplan.display_awr(
        '9yj020x2762a9', NULL, 'ADVANCED'));
Use SQL Query to Generate Code to Obtain Execution Plan

```sql
SELECT DISTINCT 'SELECT * FROM table(dbms_xplan.display_cursor(''' || sql_id || '''','' || sql_child_number || '','ADVANCED'))';
FROM v$active_session_history
...
```
Which Part of Execution Plan Consumed the Most Time?

```
SELECT ...
, h.sql_plan_line_id
, sum(10) ash_secs
FROM dba_hist_snapshot x
, dba_hist_active_sess_history h
...
WHERE ...
AND h.sql_id = 'a47fb0x1b23jn'
GROUP BY ...
, h.sql_plan_line_id
ORDER BY prcsinstance, ASH_SECS DESC
```
### Which Part of Execution Plan Consumed the Most Time?

<table>
<thead>
<tr>
<th>PRCSINSTANCE</th>
<th>SQL_PLAN_HASH_VALUE</th>
<th>SQL_PLAN_LINE_ID</th>
<th>ASH_SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4945802</td>
<td>483167840</td>
<td>25</td>
<td>2410</td>
</tr>
<tr>
<td>483167840</td>
<td></td>
<td>24</td>
<td>1190</td>
</tr>
<tr>
<td>483167840</td>
<td></td>
<td>26</td>
<td>210</td>
</tr>
<tr>
<td>483167840</td>
<td></td>
<td>20</td>
<td>190</td>
</tr>
<tr>
<td>483167840</td>
<td></td>
<td>21</td>
<td>30</td>
</tr>
<tr>
<td>483167840</td>
<td></td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>483167840</td>
<td></td>
<td>23</td>
<td>10</td>
</tr>
<tr>
<td>483167840</td>
<td></td>
<td>22</td>
<td>10</td>
</tr>
<tr>
<td>483167840</td>
<td></td>
<td>18</td>
<td>10</td>
</tr>
<tr>
<td>483167840</td>
<td></td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>

ASH: Advanced Analysis

©2016
### Which Part of Execution Plan Consumed the Most Time?

Plan hash value: 483167840

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
<th>TQ</th>
<th>IN-OUT</th>
<th>P Q Distrib</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>NESTED LOOPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>NESTED LOOPS</td>
<td></td>
<td>3988</td>
<td>669K</td>
<td>113K (1)</td>
<td>00:06:08</td>
<td>Q1,04</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>HASH JOIN SEMI</td>
<td></td>
<td>3851</td>
<td>481K</td>
<td>112K (1)</td>
<td>00:06:05</td>
<td>Q1,04</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>PX RECEIVE</td>
<td>:TQ10003</td>
<td>3771K</td>
<td>233M</td>
<td>61175 (1)</td>
<td>00:03:19</td>
<td>Q1,04</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>PX SEND HASH</td>
<td>:TQ10003</td>
<td>3771K</td>
<td>233M</td>
<td>61175 (1)</td>
<td>00:03:19</td>
<td>Q1,03</td>
<td>P-&gt;P</td>
<td>HASH</td>
</tr>
<tr>
<td>19</td>
<td>PX BLOCK ITERATOR</td>
<td></td>
<td>3771K</td>
<td>233M</td>
<td>61175 (1)</td>
<td>00:03:19</td>
<td>Q1,03</td>
<td>PCWC</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>TABLE ACCESS FULL</td>
<td>PS_CM_DEPLETE</td>
<td>3771K</td>
<td>233M</td>
<td>61175 (1)</td>
<td>00:03:19</td>
<td>Q1,03</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>BUFFER SORT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>PX RECEIVE</td>
<td>:TQ10003</td>
<td>6058K</td>
<td>364M</td>
<td>50906 (1)</td>
<td>00:02:46</td>
<td>Q1,04</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>PX SEND HASH</td>
<td>:TQ10003</td>
<td>6058K</td>
<td>364M</td>
<td>50906 (1)</td>
<td>00:02:46</td>
<td>Q1,04</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>INDEX FULL SCAN</td>
<td>PS_CM_DEPLETE_COST</td>
<td>6058K</td>
<td>364M</td>
<td>50906 (1)</td>
<td>00:02:46</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>INDEX UNIQUE SCAN</td>
<td>PS_TRANSACTION_INV</td>
<td>1</td>
<td></td>
<td>1 (0)</td>
<td>00:00:01</td>
<td>Q1,04</td>
<td>PCWP</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>PS_TRANSACTION_INV</td>
<td>1</td>
<td></td>
<td>1 (0)</td>
<td>00:00:01</td>
<td>Q1,04</td>
<td>PCWP</td>
<td></td>
</tr>
</tbody>
</table>

...
Profile by Line and Event

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>INSERT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td>318 (100)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>LOAD TABLE CONVENTIONAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>FILTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>PS_PROJ_RESOURCE</td>
<td>724</td>
<td>337K</td>
<td>314 (0)</td>
<td>00:00:04</td>
</tr>
<tr>
<td>4</td>
<td>INDEX RANGE SCAN</td>
<td>PS_PROJ_PROJECT</td>
<td>724</td>
<td>9</td>
<td>2800 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>5</td>
<td>INDEX RANGE SCAN</td>
<td>PS_PRJ_RE_SBF_TMP</td>
<td>1</td>
<td>6</td>
<td>4 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

ASH

<table>
<thead>
<tr>
<th>SQL_PLAN_LINE_ID EVENT</th>
<th>ASH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Secs</td>
</tr>
<tr>
<td>3 db file sequential read</td>
<td>40320</td>
</tr>
<tr>
<td>3 read by other session</td>
<td>12280</td>
</tr>
<tr>
<td>1 db file sequential read</td>
<td>10680</td>
</tr>
<tr>
<td>3 db file sequential read</td>
<td>4320</td>
</tr>
<tr>
<td>3 db file sequential read</td>
<td>2800</td>
</tr>
<tr>
<td>4 db file sequential read</td>
<td>1560</td>
</tr>
<tr>
<td>4 db file sequential read</td>
<td>1160</td>
</tr>
<tr>
<td>4 db file sequential read</td>
<td>520</td>
</tr>
<tr>
<td>4 db file sequential read</td>
<td>240</td>
</tr>
<tr>
<td>4 read by other session</td>
<td>200</td>
</tr>
<tr>
<td>4 read by other session</td>
<td>80</td>
</tr>
<tr>
<td>5 db file sequential read</td>
<td>40</td>
</tr>
</tbody>
</table>

---

sum 74200

ASH: Advanced Analysis

©2016
Adaptive Execution Plan

```
SELECT sql_full_plan_hash_value, sql_plan_hash_value,
       sql_adaptive_plan_resolved,
       COUNT(DISTINCT sql_exec_id) execs,
       sum(10) db_secs
FROM dba_hist_active_Sess_history
WHERE sql_id = '4dszd9dysry0c'
AND sql_plan_hash_value > 0
GROUP BY dbid, sql_plan_hash_value, sql_full_plan_hash_value,
        sql_adaptive_plan_resolved
```

<table>
<thead>
<tr>
<th>SQL_FULL_PLAN_HASH_VALUE</th>
<th>SQL_PLAN_HASH_VALUE</th>
<th>SQL_ADAPTIVE_PLAN_RESOLVED</th>
<th>EXECS</th>
<th>DB_SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4059585501</td>
<td>4114868852</td>
<td>0</td>
<td>1</td>
<td>387</td>
</tr>
<tr>
<td>4059585501</td>
<td>4114868852</td>
<td>1</td>
<td>274</td>
<td>3050</td>
</tr>
<tr>
<td>4059585501</td>
<td>3412983073</td>
<td>1</td>
<td>387</td>
<td>3980</td>
</tr>
</tbody>
</table>

One SQL statement, two plan hash values, same full plan hash value
Adaptive Execution Plan

select * from table(dbms_xplan.display('ASH_PLAN_TABLE','4dszd9dysry0c',null,'dbid=2783210685 and plan_hash_value = 3412983073'));

Plan hash value: 3412983073

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td>13 (100)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SORT ORDER BY</td>
<td></td>
<td>9</td>
<td>747</td>
<td>13 (8)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>2</td>
<td>HASH JOIN</td>
<td></td>
<td>9</td>
<td>747</td>
<td>12 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>3</td>
<td>JOIN FILTER CREATE</td>
<td>:BF0000</td>
<td>9</td>
<td>747</td>
<td>12 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>4</td>
<td>TABLE ACCESS BY INDEX ROWID BATCHED</td>
<td>PGRELS</td>
<td>9</td>
<td>585</td>
<td>5 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>5</td>
<td>INDEX RANGE SCAN</td>
<td>LINKSOURCE_201</td>
<td>9</td>
<td></td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>6</td>
<td>JOIN FILTER USE</td>
<td>:BF0000</td>
<td>122</td>
<td>2196</td>
<td>7 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS STORAGE FULL</td>
<td>USERGROUPS</td>
<td>122</td>
<td>2196</td>
<td>7 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Note

- this is an adaptive plan
Adaptive Execution Plan

```sql
select * from table(dbms_xplan.display('ASH_PLAN_TABLE','4dszd9dysry0c' 'ADAPTIVE',dbid=2783210685 and plan_hash_value = 3412983073'));
```

Plan hash value: 3412983073

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td>13 (100)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SORT ORDER BY</td>
<td></td>
<td>9</td>
<td>747</td>
<td>13 (8)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>2</td>
<td>HASH JOIN</td>
<td></td>
<td>9</td>
<td>747</td>
<td>12 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>3</td>
<td>JOIN FILTER CREATE</td>
<td>:BF0000</td>
<td>9</td>
<td>747</td>
<td>12 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>-</td>
<td>NESTED LOOPS</td>
<td></td>
<td>9</td>
<td>747</td>
<td>12 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>-</td>
<td>NESTED LOOPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>STATISTICS COLLECTOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS BY INDEX ROWID BATCHED</td>
<td>PGRELS</td>
<td>9</td>
<td>585</td>
<td>5 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>8</td>
<td>INDEX RANGE SCAN</td>
<td>LINKSOURCE_201</td>
<td>9</td>
<td></td>
<td>3 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>-</td>
<td>INDEX UNIQUE SCAN</td>
<td>SYS_C008784</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>USERGROUPS</td>
<td>1</td>
<td>18</td>
<td>7 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>11</td>
<td>JOIN FILTER USE</td>
<td>:BF0000</td>
<td>122</td>
<td>2196</td>
<td>7 (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>12</td>
<td>TABLE ACCESS STORAGE FULL</td>
<td>USERGROUPS</td>
<td>122</td>
<td>2196</td>
<td>7 (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Note
----
- this is an adaptive plan (rows marked '-' are inactive)
Adaptive Execution Plan

```sql
select * from table(dbms_xplan.display('ASH_PLAN_TABLE', '4dszd9dysry0c', '+ADAPTIVE', 'dbid=2783210685 and plan_hash_value = 4114868852'));
```

Plan hash value: 4114868852

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td>13 (100)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SORT ORDER BY</td>
<td></td>
<td>8</td>
<td>664</td>
<td>13   (8)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>-</td>
<td>HASH JOIN</td>
<td></td>
<td>8</td>
<td>664</td>
<td>12   (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>3</td>
<td>JOIN FILTER CREATE</td>
<td>:BF0000</td>
<td>8</td>
<td>664</td>
<td>12   (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>4</td>
<td>NESTED LOOPS</td>
<td></td>
<td>8</td>
<td>664</td>
<td>12   (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>5</td>
<td>NESTED LOOPS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>STATISTICS COLLECTOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>TABLE ACCESS BY INDEX ROWID BATCHED</td>
<td>PGRELS</td>
<td>8</td>
<td>520</td>
<td>5   (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>8</td>
<td>INDEX RANGE SCAN</td>
<td>LINKSOURCE_201</td>
<td>9</td>
<td></td>
<td>3   (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>9</td>
<td>INDEX UNIQUE SCAN</td>
<td>SYS_C008784</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>USERGROUPS</td>
<td>1</td>
<td>18</td>
<td>7   (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>-</td>
<td>JOIN FILTER USE</td>
<td>:BF0000</td>
<td>122</td>
<td>2196</td>
<td>7   (0)</td>
<td>00:00:01</td>
</tr>
<tr>
<td>-</td>
<td>TABLE ACCESS STORAGE FULL</td>
<td>USERGROUPS</td>
<td>122</td>
<td>2196</td>
<td>7   (0)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Note
----
- this is an adaptive plan (rows marked '-' are inactive)
profiling adaptive plans

```
select sql_plan_hash_value,
       sql_full_plan_hash_value,
       sql_plan_line_id,
       sql_adaptive_plan_resolved,
       sum(10) db_secs
from dba_hist_active_Sess_history
where sql_id = '4dszd9dysry0c'
group by
       sql_plan_hash_value,
       sql_full_plan_hash_value,
       sql_plan_line_id,
       sql_adaptive_plan_resolved
order by 1,2,4
/```

ash: advanced analysis ©2016
### Profiling Adaptive Plans

<table>
<thead>
<tr>
<th>SQL_PLAN_HASH_VALUE</th>
<th>SQL_FULL_PLAN_HASH_VALUE</th>
<th>SQL_PLAN_LINE_ID</th>
<th>SQL_ADAPTIVE_PLAN_RESOLVED</th>
<th>DB_SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3412983073</td>
<td>4059585501</td>
<td>1</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>3412983073</td>
<td>4059585501</td>
<td>2</td>
<td>1</td>
<td>1540</td>
</tr>
<tr>
<td>3412983073</td>
<td>4059585501</td>
<td>3</td>
<td>1</td>
<td>80</td>
</tr>
<tr>
<td>3412983073</td>
<td>4059585501</td>
<td>7</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>3412983073</td>
<td>4059585501</td>
<td>8</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>3412983073</td>
<td>4059585501</td>
<td>12</td>
<td>1</td>
<td>750</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4114868852</td>
<td>4059585501</td>
<td>1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>4114868852</td>
<td>4059585501</td>
<td>2</td>
<td>1</td>
<td>230</td>
</tr>
<tr>
<td>4114868852</td>
<td>4059585501</td>
<td>3</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>4114868852</td>
<td>4059585501</td>
<td>4</td>
<td>1</td>
<td>200</td>
</tr>
<tr>
<td>4114868852</td>
<td>4059585501</td>
<td>7</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>4114868852</td>
<td>4059585501</td>
<td>8</td>
<td>1</td>
<td>140</td>
</tr>
<tr>
<td>4114868852</td>
<td>4059585501</td>
<td>9</td>
<td>1</td>
<td>150</td>
</tr>
<tr>
<td>4114868852</td>
<td>4059585501</td>
<td>10</td>
<td>1</td>
<td>70</td>
</tr>
<tr>
<td>4114868852</td>
<td>4059585501</td>
<td></td>
<td></td>
<td>800</td>
</tr>
</tbody>
</table>

**SQL Plan Line IDs** correspond to +ADAPTIVE plan.
Bind Variables

• Developers Should Use Them!

• Unfortunately...
## Different SQL, Same Plan

<table>
<thead>
<tr>
<th>P.I.</th>
<th>SQL_ID</th>
<th>SQL_PLAN_HASH_VALUE</th>
<th>EXEC_SECS</th>
<th>ASH_SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1949129</td>
<td>0uj7k70z1s76y</td>
<td>2239378934</td>
<td>619</td>
<td>210</td>
</tr>
<tr>
<td>1949819</td>
<td>0sd03jvun7us6</td>
<td>2239378934</td>
<td>336</td>
<td>20</td>
</tr>
<tr>
<td>1953197</td>
<td>22kn2sb7vttnp</td>
<td>2239378934</td>
<td>753</td>
<td>150</td>
</tr>
<tr>
<td>1956338</td>
<td>0xkjtywub2861</td>
<td>2602481067</td>
<td>19283</td>
<td>18550</td>
</tr>
<tr>
<td>1956338</td>
<td>998wf4g84dk8z</td>
<td>1041940423</td>
<td>19283</td>
<td>10</td>
</tr>
<tr>
<td>1956805</td>
<td>7c7dzavm70yku</td>
<td>2602481067</td>
<td>16350</td>
<td>15690</td>
</tr>
<tr>
<td>1956925</td>
<td>1knvx57dnrz29</td>
<td>2602481067</td>
<td>15654</td>
<td>15010</td>
</tr>
<tr>
<td>1956925</td>
<td>a9mw8hjxfwczm</td>
<td>338220129</td>
<td>15654</td>
<td>10</td>
</tr>
<tr>
<td>1957008</td>
<td>9s2jct0jfmwgy</td>
<td>2602481067</td>
<td>15077</td>
<td>14430</td>
</tr>
<tr>
<td>1957008</td>
<td>9s2jct0jfmwgy</td>
<td>3265949623</td>
<td>15077</td>
<td>10</td>
</tr>
<tr>
<td>1957087</td>
<td>cwarnq7kv4d84</td>
<td>2602481067</td>
<td>14638</td>
<td>14000</td>
</tr>
<tr>
<td>1957691</td>
<td>9nv93p134xjb0</td>
<td>2602481067</td>
<td>13477</td>
<td>12980</td>
</tr>
<tr>
<td>1958659</td>
<td>9s2jct0jfmwgy</td>
<td>2602481067</td>
<td>9354</td>
<td>9140</td>
</tr>
<tr>
<td>1958697</td>
<td>1bd0fg0fvsfyp</td>
<td>2602481067</td>
<td>9176</td>
<td>8950</td>
</tr>
</tbody>
</table>
Non-Shareable SQL

SQL_ID djqf1zcypm5fm
____________________
SELECT ... 
FROM PS_TL_EXCEPTION A,
    PS_PERSONAL_DATA B,
    PS_PERALL_SEC_QRY B1,
... 
WHERE B.EMPLID = B1.EMPLID 
AND B1.OPRID = '12345678'
...

ASH: Advanced Analysis
**Same Problem in One Process**

<table>
<thead>
<tr>
<th>PRCSINSTANCE</th>
<th>SQL_ID</th>
<th>SQL_PLAN_HASH_VALUE</th>
<th>EXEC_SECS</th>
<th>ASH_SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>50002824</td>
<td>2ybtak62vmx58</td>
<td>2262951047</td>
<td>10306</td>
<td>50</td>
</tr>
<tr>
<td>50002824</td>
<td>ck3av6cnquwfc</td>
<td>2262951047</td>
<td>10306</td>
<td>20</td>
</tr>
<tr>
<td>50002824</td>
<td>gvys6kd9fqn7u</td>
<td>2262951047</td>
<td>10306</td>
<td>20</td>
</tr>
<tr>
<td>50002824</td>
<td>7ymcbn6q8utj8</td>
<td>2262951047</td>
<td>10306</td>
<td>10</td>
</tr>
<tr>
<td>50002824</td>
<td>9qud2n3qq7nzr</td>
<td>2262951047</td>
<td>10306</td>
<td>10</td>
</tr>
<tr>
<td>50002824</td>
<td>6pxvns97mlfua</td>
<td>2262951047</td>
<td>10306</td>
<td>10</td>
</tr>
<tr>
<td>50002824</td>
<td>5ngqj5zg8vbz8</td>
<td>2262951047</td>
<td>10306</td>
<td>10</td>
</tr>
<tr>
<td>50002824</td>
<td>9zp6nddfvn66b</td>
<td>2262951047</td>
<td>10306</td>
<td>10</td>
</tr>
<tr>
<td>50002824</td>
<td>15kfs3c3005xm</td>
<td>2262951047</td>
<td>10306</td>
<td>10</td>
</tr>
<tr>
<td>50002824</td>
<td>4qvhpygc7cq2t</td>
<td>2262951047</td>
<td>10306</td>
<td>10</td>
</tr>
<tr>
<td>50002824</td>
<td>23yc8dcz9z4yj</td>
<td>2262951047</td>
<td>10306</td>
<td>10</td>
</tr>
<tr>
<td>50002824</td>
<td>bn8xczrvs2hpr</td>
<td>2262951047</td>
<td>10306</td>
<td>10</td>
</tr>
<tr>
<td>50002824</td>
<td>9g6k9dnrjap08</td>
<td>2262951047</td>
<td>10306</td>
<td>10</td>
</tr>
<tr>
<td>50002824</td>
<td>1art8dhzbvwpwt</td>
<td>2262951047</td>
<td>10306</td>
<td>10</td>
</tr>
<tr>
<td>50002824</td>
<td>6gqj337xnr5y4</td>
<td>2262951047</td>
<td>10306</td>
<td>10</td>
</tr>
</tbody>
</table>

...
<table>
<thead>
<tr>
<th>PRCSINSTANCE</th>
<th>SQL_PLAN_HASH_VALUE</th>
<th>EXEC_SECS</th>
<th>ASH_SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>50002824</td>
<td>2262951047</td>
<td>10306</td>
<td>2300</td>
</tr>
<tr>
<td>50002824</td>
<td>0</td>
<td>10306</td>
<td>60</td>
</tr>
<tr>
<td>50002824</td>
<td>3085938243</td>
<td>10306</td>
<td>20</td>
</tr>
<tr>
<td>50002824</td>
<td>563410926</td>
<td>10306</td>
<td>10</td>
</tr>
<tr>
<td>50002824</td>
<td>1068931976</td>
<td>10306</td>
<td>10</td>
</tr>
</tbody>
</table>
Can you find the SQL in AWR?

- Now Find the SQL with that plan.
- If it was captured by AWR
  - Lots of parsing causes statements to be aged out of library cache before they get stored in AWR by a snapshot
  - Only Top-n statements are captured.
Can you find the SQL in AWR?

• Outer Join SQLTEXT

```
SELECT h.sql_id, h.sql_plan_hash_value,
       SUM(10) ash_secs,
       10*COUNT(t.sql_id) awr_secs
FROM   dba_hist_snapshot X,
       dba_hist_active_sess_history h,
       dba_hist_sqltext t
LEFT OUTER JOIN dba_hist_sqltext t
ON     t.sql_id = h.sql_id
WHERE ...
GROUP BY h.sql_id, h.sql_plan_hash_value
```
Can you find the SQL in AWR?

```
SELECT * FROM (
    SELECT ROW_NUMBER() OVER (PARTITION BY x.sql_plan_hash_value
                                ORDER BY x.awr_secs desc) ranking
                   ,x.sql_id, x.sql_plan_hash_value
                   ,SUM(x.ash_secs) OVER (PARTITION BY x.sql_plan_hash_value) ash
                   ,SUM(x.awr_secs) OVER (PARTITION BY x.sql_plan_hash_value) awr
                   ,COUNT(DISTINCT sql_id) OVER
                       (PARTITION BY x.sql_plan_hash_value) sql_ids
    FROM ( SELECT h.sql_id, h.sql_plan_hash_value
                   , SUM(10) ash_secs
                   , 10*COUNT(t.sql_id) awr_secs
    FROM dba_hist_snapshot X
           , dba_hist_active_sess_history h
           LEFT OUTER JOIN dba_hist_sqltext t ON t.sql_id = h.sql_id
    WHERE ... GROUP BY h.sql_id, h.sql_plan_hash_value ) x ) y
WHERE y.ranking = 1
ORDER BY tot_ash_secs desc, ranking
```
Can you find the SQL in AWR?

<table>
<thead>
<tr>
<th>RNK</th>
<th>SQL_ID</th>
<th>SQL_PLAN</th>
<th>TOTAL</th>
<th>TOTAL</th>
<th>SQL_IDS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>HASH_VALUE</td>
<td>ASH_SECS</td>
<td>AWR_SECS</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8mkvraydrxvycn</td>
<td>0</td>
<td>38270</td>
<td>480</td>
<td>74^1</td>
</tr>
<tr>
<td>1</td>
<td>027qsfj7n71cy</td>
<td>1499159071</td>
<td>4230</td>
<td>4230</td>
<td>1^2</td>
</tr>
<tr>
<td>1</td>
<td>cxwz9m3auk4y7</td>
<td>1898065720</td>
<td>4190</td>
<td>4190</td>
<td>198^3</td>
</tr>
<tr>
<td>1</td>
<td>9513hhulvucxz</td>
<td>2044891559</td>
<td>3590</td>
<td>3590</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>95dx0mkjq38v5</td>
<td>1043916244</td>
<td>3450</td>
<td>3450</td>
<td>23</td>
</tr>
</tbody>
</table>

1. Special case. There is no plan because it’s the `dbms_stats` function. There were 74 statements, but in reality they were all totally different.

2. One SQL, one plan, this is a shareable SQL_ID, or it did just execute once.

3. This is many statements with the same plan, at least 198.
ASH for Single Wait Event

```
SELECT /*+LEADING(x h) USE_NL(h)*/
    h.sql_id,
    h.sql_plan_hash_value,
    SUM(10) ash_secs
FROM   dba_hist_snapshot x,
        dba_hist_active_sess_history h
WHERE  x.end_interval_time   <= TO_DATE('201402010830','yyyymmddhh24mi')
AND    x.begin_interval_time >= TO_DATE('201402010730','yyyymmddhh24mi')
AND    h.sample_time BETWEEN   TO_DATE('201401261100','yyyymmddhh24mi')
    AND   TO_DATE('201401261300','yyyymmddhh24mi')
AND    h.SNAP_id = x.SNAP_id
AND    h.dbid = x.dbid
AND    h.instance_number = x.instance_number
AND    h.event = 'db file sequential read'
GROUP BY h.sql_id, h.sql_plan_hash_value
ORDER BY ash_secs desc
/```
### Statements with Highest I/O

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>Hash Value</th>
<th>ASH_SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>90pp7bcnmz68r</td>
<td>2961772154</td>
<td>2490</td>
</tr>
<tr>
<td>81gz2ratabaa8n</td>
<td>1919624473</td>
<td>2450</td>
</tr>
<tr>
<td>7hvaxp65s70qw</td>
<td>1051046890</td>
<td>1320</td>
</tr>
<tr>
<td>7fk8raq16ch0u</td>
<td>3950826368</td>
<td>890</td>
</tr>
<tr>
<td>9dzpwkff7zytcg</td>
<td>2020614776</td>
<td>840</td>
</tr>
</tbody>
</table>

...
What Kind of Single Block Read?

• For I/O wait events ASH reports
  – File number
  – Block number
  – Object number
  – Row number (from 11g)

• Only valid on DB File events.
  – Invalid on other events because simply not cleared from previous operation.
Categorise Tablespaces/Data Files

CREATE TABLE dmk_data_files as
SELECT tablespace_name,
       file_id,
       CASE
           WHEN f.tablespace_name LIKE 'SYS%' THEN 'SYSTEM'
           WHEN f.tablespace_name LIKE 'UNDO%' THEN 'UNDO'
           WHEN f.tablespace_name LIKE '%IDX%' THEN 'INDEX'
           WHEN f.tablespace_name LIKE '%INDEX%' THEN 'INDEX'
           ELSE 'TABLE'
       END AS tablespace_type
FROM dba_data_files f
ORDER BY tablespace_name
/

• Working storage table performs better than DBA_DATA_FILES
ASH Data by Tablespace Type

SELECT /*+LEADING(x h) USE_NL(h f)*/
    f.tablespace_type,
    SUM(10) ash_secs
FROM dba_hist_snapshot x,
    dba_hist_active_sess_history h,
    dmk_data_files f
WHERE x.end_interval_time <= TO_DATE('201402161300', 'yyyymmddhh24mi')
    AND x.begin_interval_time
        >= TO_DATE('201402161100', 'yyyymmddhh24mi')
    AND h.sample_time BETWEEN TO_DATE('201401261100', 'yyyymmddhh24mi')
        AND TO_DATE('201401261300', 'yyyymmddhh24mi')
    AND h.SNAP_id = X.SNAP_id
    AND h.dbid = x.dbid
    AND h.instance_number = x.instance_number
    AND h.event LIKE 'db file%'
    AND h.p1text = 'file#'
    AND h.p2text = 'block#'
    AND f.file_id = h.p1
GROUP BY f.tablespace_type
ORDER BY ash_secs desc
### ASH Data by Tablespace Type

<table>
<thead>
<tr>
<th>Type</th>
<th>ASH_SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDEX</td>
<td>30860</td>
</tr>
<tr>
<td>TABLE</td>
<td>26970</td>
</tr>
<tr>
<td>UNDO</td>
<td>1370</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>490</td>
</tr>
</tbody>
</table>

- Most time spent on index read
  - Includes index maintenance during DML
- Not much undo, so not much consistent read.
Which Tables Account for the I/O?

- Need own copy of DBA_OBJECTS

```sql
CREATE TABLE dmk_objects
  (object_id NUMBER NOT NULL,
   owner VARCHAR2(30) NOT NULL,
   object_name VARCHAR2(128) NOT NULL,
   subobject_name VARCHAR2(30),
   PRIMARY KEY (OBJECT_ID))
/

INSERT INTO dmk_objects
SELECT object_id, owner,
     object_name, subobject_name
FROM dba_objects
WHERE object_type LIKE 'TABLE'
UNION ALL
SELECT o.object_id, i.table_owner,
    i.table_name, o.subobject_name
FROM dba_objects o, dba_indexes i
WHERE o.object_type like 'INDEX'
AND i.owner = o.owner
AND i.index_name = o.object_name
/
```
Which Objects are Used?

```sql
SELECT /*+LEADING(x h) USE_NL(h)*/
    o.owner, o.object_name
, SUM(10) ash_secs
FROM dba_hist_snapshot x,
    dba_hist_active_sess_history h,
    dmk_objects o
WHERE x.end_interval_time >= SYSDATE - 7
AND x.begin_interval_time <= SYSDATE
AND h.sample_time >= SYSDATE - 7
AND h.sample_time <= SYSDATE
AND h.Snap_id = X.snap_id
AND h.dbid = x.dbid
AND h.instance_number = x.instance_number
AND h.event LIKE 'db file%'
AND h.current_obj# = o.object_id
GROUP BY o.owner, o.object_name
HAVING SUM(10) >= 3600
ORDER BY ash_secs DESC
```
### Which Objects are Used?

<table>
<thead>
<tr>
<th>OWNER</th>
<th>OBJECT_NAME</th>
<th>Secs</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSADM</td>
<td>PS_TL_RPTD_TIME</td>
<td>800510</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_TL_PAYABLE_TIME</td>
<td>327280</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_GP_RSLT_ACUM</td>
<td>287870</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_SCH_DEFN_DTL</td>
<td>161690</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_SCH_DEFN_TBL</td>
<td>128070</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_GP_RSLT_PIN</td>
<td>124560</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_GP_PYE_PRC_STAT</td>
<td>92410</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_SCH_ADHOC_DTL</td>
<td>88810</td>
</tr>
</tbody>
</table>
Which Processes Read This Table?

```
SELECT /*+LEADING(x) USE_NL(h)*/
    o.owner, o.object_name
,    h.module
,    SUM(10) ash_secs
FROM dba_hist_snapshot x
,    dba_hist_active_sess_history h
,    dmk_objects o
WHERE x.end_interval_time   >= SYSDATE - 7
AND x.begin_interval_time <= SYSDATE
AND h.sample_time         >= SYSDATE - 7
AND h.sample_time         <= SYSDATE
AND h.Snap_id = X.snap_id
AND h.dbid = x.dbid
AND h.instance_number = x.instance_number
AND h.event LIKE 'db file%'
AND h.current_obj# = o.object_id
AND o.object_name = 'PS_GP_RSLT_ACUM'
GROUP BY o.owner, o.object_name
,    h.module
HAVING SUM(10) >= 900
ORDER BY ash_secs desc
```
### Which Processes Read This Table?

<table>
<thead>
<tr>
<th>OWNER</th>
<th>OBJECT_NAME</th>
<th>MODULE</th>
<th>Secs</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSADM</td>
<td>PS_GP_RSLT_ACUM</td>
<td>XXX_HOL_MGMT</td>
<td>79680</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_GP_RSLT_ACUM</td>
<td>DBMS_SCHEDULER</td>
<td>37810</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_GP_RSLT_ACUM</td>
<td>SQL*Plus</td>
<td>37060</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_GP_RSLT_ACUM</td>
<td>GPGBHLE</td>
<td>30710</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_GP_RSLT_ACUM</td>
<td>GPPDPRUN</td>
<td>27440</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_GP_RSLT_ACUM</td>
<td>XXX_AE_AB007</td>
<td>21440</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_GP_RSLT_ACUM</td>
<td>SQL Developer</td>
<td>11210</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_GP_RSLT_ACUM</td>
<td>GPGBEPTD</td>
<td>7240</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_GP_RSLT_ACUM</td>
<td>XXX_CAPITA</td>
<td>5850</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_GP_RSLT_ACUM</td>
<td>GPGB_PSLIP_X</td>
<td>5030</td>
</tr>
<tr>
<td>SYSADM</td>
<td>PS_GP_RSLT_ACUM</td>
<td>GPGB_EDI</td>
<td>4880</td>
</tr>
</tbody>
</table>

- **Top Application Process**
- **Statistics Gathering**
- **People are writing ad-hoc queries in SQL*Plus**
Who is using this index?

- CURRENT_OBJ# has been suggested as a way to identify index usage.
  - It only identifies index physical read
  - So it also includes index maintenance during DML
  - Doesn’t work if the object has been rebuilt and has a new object number
SQL Plans captured by AWR

- SQL statements and plans captured during AWR snapshot
  - Top N by Elapsed Time, CPU Time, Parse Calls, Shareable Memory, Version Count

- **DBA_HIST_SQL_PLAN**
  - OBJECT_OWNER
  - OBJECT_TYPE
  - OBJECT_NAME
Who is using this index?

• Join plans that reference index to ASH data by SQL_PLAN_HASH_VALUE
  – Do not join by SQL_ID

• Filter out
  – SQL*Plus, Toad, Ad-Hoc query tools
  – Statistics collection
CREATE TABLE my_ash COMPRESS AS
WITH p AS (  
    SELECT DISTINCT p.plan_hash_value, p.object#,  
    , p.object_owner, p.object_type, p.object_name  
    FROM dba_hist_sql_plan p  
    WHERE p.object_name like 'PS_PROJ_Resource'  
        AND p.object_type LIKE 'INDEX%'  
        AND p.object_owner = 'SYSADM')  
SELECT p.object# object_id  
    , p.object_owner, p.object_type, p.object_name  
    , h.*  
FROM dba_hist_active_sess_history h  
    , p  
WHERE h.sql_plan_hash_value = p.plan_hash_value

Extract ASH for statements that use specified indexes
Profile the ASH extracted

WITH h AS (  
SELECT object_name,  
      CASE WHEN h.module IS NULL THEN REGEXP_SUBSTR(h.program,'[^@]+',1,1)  
           WHEN h.module LIKE 'PSAE.%' THEN REGEXP_SUBSTR(h.module,'[^.]+',1,2)  
           ELSE REGEXP_SUBSTR(h.program,'[^@]+',1,1)  
      END as module,  
      CASE WHEN h.action LIKE 'PI=%' THEN NULL  
           ELSE h.action  
      END as action,  
      CAST(sample_time AS DATE) sample_time,  
FROM my_ash h  
)  
SELECT object_name, module, action,  
      sum(10) ash_secs,  
      COUNT(DISTINCT sql_plan_hash_value) sql_plans,  
      COUNT(DISTINCT sql_id||sql_plan_hash_value||sql_exec_id) sql_execs,  
      MAX(sample_time) max_sample_time  
WHERE NOT lower(module) IN('oracle','toad','sqlplus','sqlplusw')  
AND NOT lower(module) LIKE 'sql%'  
ORDER BY object_name, module, action  
ORDER BY SUBSTR(object_name,4), object_name, ash_secs desc
Profile the ASH extracted

<table>
<thead>
<tr>
<th>OBJECT_NAME</th>
<th>MODULE</th>
<th>ACTION</th>
<th>Secs</th>
<th>Plans</th>
<th>Execs</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum</td>
<td></td>
<td></td>
<td>60</td>
<td>2</td>
<td>6</td>
<td>18:35:18 20/08/2014</td>
</tr>
</tbody>
</table>

This index used widely. Probably can't drop it.

<table>
<thead>
<tr>
<th>OBJECT_NAME</th>
<th>MODULE</th>
<th>ACTION</th>
<th>Secs</th>
<th>Plans</th>
<th>Execs</th>
<th>Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC_TL_TO_PC</td>
<td>GF_PBINT_AE.CallmeA.Step26.S</td>
<td></td>
<td>7800</td>
<td>1</td>
<td>49</td>
<td>18:53:50 26/08/2014</td>
</tr>
<tr>
<td>sum</td>
<td></td>
<td></td>
<td>60</td>
<td>2</td>
<td>6</td>
<td>18:35:18 20/08/2014</td>
</tr>
</tbody>
</table>

This index used lightly. Perhaps we don’t really need it.

Indexes that do not appear are probably not used.

Bit stream

©2016

This index used widely. Probably can’t drop it.
Limitations of Method

- AWR doesn’t capture all SQLs
  - A very effective index that is only used occasionally might not be captured.
  - Results are only indicative, not absolute.
- ASH data purged after 31 days (by default)
  - An index only be used for annual process might not be detected, but it might be essential for that process
  - Consider establishing longer term repository, retaining perhaps 400 days.
- EM 12c Rel4 AWR Warehouse
Before I can drop an index...

- Need to look at SQL found to reference it.
- Might prefer to make index invisible and drop later if no issue.
AWR Data indicates locking
Where did we wait on a lock?

```sql
SELECT /*+LEADING(x h) USE_NL(h)*/
    h.sql_id,
    h.sql_plan_hash_value,
    SUM(10) ash_secs
FROM dba_hist_snapshot x,
     dba_hist_active_sess_history h
WHERE x.end_interval_time >= TO_DATE('201401261100', 'yyyymmddhh24mi')
AND x.begin_interval_time <= TO_DATE('201401261300', 'yyyymmddhh24mi')
AND h.sample_time BETWEEN TO_DATE('201401261100', 'yyyymmddhh24mi')
    AND TO_DATE('201401261300', 'yyyymmddhh24mi')
AND h.snap_id = x.snap_id
AND h.dbid = x.dbid
AND h.instance_number = x.instance_number
AND h.event = 'enq: TX - row lock contention'
GROUP BY h.sql_id, h.sql_plan_hash_value
ORDER BY ash_secs desc
/```
### Where did we wait on a lock?

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>Hash Value</th>
<th>ASH_SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>7qxdrwcn4yzzh</td>
<td>3723363341</td>
<td>26030</td>
</tr>
<tr>
<td>652mx4tfq415</td>
<td>1888029394</td>
<td>11230</td>
</tr>
<tr>
<td>c9jjtvk0qf649</td>
<td>3605988889</td>
<td>6090</td>
</tr>
<tr>
<td>artqgxug4z0f1</td>
<td>8450529</td>
<td>240</td>
</tr>
<tr>
<td>gtj7zuzy2b4g6</td>
<td>2565837323</td>
<td>100</td>
</tr>
</tbody>
</table>
Statements Blocked by TX Locks

SQL_ID 7qxdrwcn4yzhh
--------------------
UPDATE PSIBQUEUEINST SET QUEUESEQID=QUEUESEQID+:1
WHERE QUEUENAME=:2

SQL_ID 652mx4tffq415
---------------------
UPDATE PSAPMSGPUBSYNC SET LASTUPDDTTM=SYSDATE
WHERE QUEUENAME=:1

SQL_ID c9jjtvk0qf649
---------------------
UPDATE PSAPMSGSUBCSYNC SET LASTUPDDTTM=SYSDATE
WHERE QUEUENAME=:1
The real question about locking:

- What is the session that is holding the lock doing while it is holding the lock?  
  - and can I do something about that?

- Home-made sequences are not scalable. Should really be using an Oracle Sequence. 
  - Not possible in a PeopleSoft Application
Resolve the Lock Chain

- Navigating the lock chain works across RAC instances from 11g.
- There may not be any ASH data for session C because it is not active on the database.
Extract ASH data for period in question

CREATE TABLE my_ash AS
SELECT /*+LEADING(x) USE_NL(h)*/ h.*
FROM dba_hist_snapshot x,
     dba_hist_active_sess_history h
WHERE x.end_interval_time >= TO_DATE('201401261100','yyyymmddhh24mi')
AND x.begin_interval_time <= TO_DATE('201401261300','yyyymmddhh24mi')
AND h.sample_time BETWEEN TO_DATE('201401261100','yyyymmddhh24mi')
    AND TO_DATE('201401261300','yyyymmddhh24mi')
AND h.snap_id = X.snap_id
AND h.dbid = x.dbid
AND h.instance_number = x.instance_number;

CREATE UNIQUE INDEX my_ash ON my_ash (dbid, instance_number, snap_id, sample_id, session_id, sample_time, session_serial#) COMPRESS 4;
CREATE INDEX my_ash2 ON my_ash (event, dbid, instance_number, snap_id) COMPRESS 3;
What are the blockers doing?
(on a single instance database)

```
SELECT /*+LEADING(x w) USE_NL(h w)*/
    h.sql_id,
    h.sql_plan_hash_value,
    SUM(10) ash_secs
FROM my_ash w
    LEFT OUTER JOIN my_ash h
    ON h.snap_id = w.snap_id
    AND h.dbid = w.dbid
    AND h.instance_number = w.instance_number
    AND h.sample_id = w.sample_id
    AND h.sample_time = w.sample_time
    AND h.session_id = w.blocking_session
    AND h.session_serial# = w.blocking_session_serial#
WHERE w.event = 'enq: TX - row lock contention'
GROUP BY h.sql_id, h.sql_plan_hash_value
ORDER BY ash_secs desc
```
What are the blockers doing? (works on RAC)

WITH h AS (  
SELECT /*+LEADING(x w) USE_NL(h w)*/  
  CASE WHEN h.dbid IS NULL THEN 'idle holder'  
    WHEN h.event IS NULL THEN 'CPU + CPU wait'  
    ELSE h.event END as event  
,   h.module, h.sql_id, h.sql_plan_hash_value  
FROM my_ash w  
  LEFT OUTER JOIN my_ash h  
  ON h.snap_id = w.snap_id  
  AND h.dbid = w.dbid  
  AND h.session_id = w.blocking_session  
  AND h.session_serial# = w.blocking_session_serial#  
  AND h.instance_number = w.blocking_inst_id  
  AND (h.sample_id = w.sample_id OR w.blocking_inst_id != w.instance_number)  
  AND h.sample_time >= w.sample_time-5/86400  
  AND h.sample_time <  w.sample_time+5/86400  
WHERE w.blocking_Session_status = 'VALID')  
SELECT  
  h.event, h.module, h.sql_id, h.sql_plan_hash_value, SUM(10) ash_secs  
FROM h  
GROUP BY  
  h.event, h.module,  
  h.sql_id, h.sql_plan_hash_value  
ORDER BY ash_secs DESC
What are the blocking sessions doing?

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>SQL_PLAN_HASH_VALUE</th>
<th>ASH_SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>5st32un4a2y92</td>
<td>2494504609</td>
<td>10670</td>
</tr>
<tr>
<td>652mx4ttffq415</td>
<td>1888029394</td>
<td>7030</td>
</tr>
<tr>
<td>artqgxug4z0f1</td>
<td>8450529</td>
<td>580</td>
</tr>
<tr>
<td>7qxdrcwn4yzhh</td>
<td>3723363341</td>
<td>270</td>
</tr>
</tbody>
</table>

1. This SQL_ID is blank. May not be able to find ASH sample for blocking session because it is idle – busy on the client not the database.
2. This statement is running while the session holds a lock that is blocking another session.
EXECUTION PLAN captured by AWR: Correct Plan, Old Costs, Old Binds

- DISPLAY_AWR()

SQL_ID 5st32un4a2y92
----------------------------------
SELECT 'X' FROM PS_CDM_LIST WHERE CONTENTID = :1

Plan hash value: 2494504609

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td>22 (100)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>INDEX FAST FULL SCAN</td>
<td>PS_CDM_LIST</td>
<td>1</td>
<td>5</td>
<td>22 (10)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

Query Block Name / Object Alias (identified by operation id):

1 - SEL$1 / PS_CDM_LIST@SEL$1

Peeked Binds (identified by position):

1 - :1 (NUMBER): 17776
Fresh Execution Plan generated by `execute explain plan`

- Note increase of cost of full scan.

### Plan hash value: 2494504609

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td>1</td>
<td>6</td>
<td>3178 (9)</td>
<td>00:00:05</td>
</tr>
<tr>
<td>* 1</td>
<td>INDEX FAST FULL SCAN</td>
<td>PS_CDM_LIST</td>
<td>1</td>
<td>6</td>
<td>3178 (9)</td>
<td>00:00:05</td>
</tr>
</tbody>
</table>

Predicate Information (identified by operation id):

1 - filter("CONTENTID"=TO_NUMBER(:1))
Did My Execution Plan Change?

- Can see change in execution plan and performance
  - (same SQL, different literals, same force matching signature)

<table>
<thead>
<tr>
<th>PRCSINSTANCE</th>
<th>BEGINDTMM</th>
<th>SQL_ID</th>
<th>HASH_VALUE</th>
<th>EXEC_SECS</th>
<th>ASH_SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1964975</td>
<td>08:30:52</td>
<td>46smbgcfcriebd</td>
<td>2602481067</td>
<td>20379</td>
<td>20080</td>
</tr>
<tr>
<td>1965250</td>
<td>09:08:51</td>
<td>fpftdx2405zyq</td>
<td>2602481067</td>
<td>20983</td>
<td>20690</td>
</tr>
<tr>
<td>1968443</td>
<td>16:42:51</td>
<td>3rxad5z3ccusv</td>
<td>3398716340</td>
<td>105</td>
<td>80</td>
</tr>
<tr>
<td>1968469</td>
<td>16:47:21</td>
<td>3rxad5z3ccusv</td>
<td>3398716340</td>
<td>90</td>
<td>70</td>
</tr>
<tr>
<td>1968485</td>
<td>16:50:19</td>
<td>3rxad5z3ccusv</td>
<td>3398716340</td>
<td>62</td>
<td>40</td>
</tr>
<tr>
<td>1968698</td>
<td>17:40:01</td>
<td>0ku8f514k3nt0</td>
<td>3398716340</td>
<td>76</td>
<td>50</td>
</tr>
<tr>
<td>1968866</td>
<td>18:19:19</td>
<td>cbmyvpsxzyf5n</td>
<td>3398716340</td>
<td>139</td>
<td>120</td>
</tr>
<tr>
<td>1968966</td>
<td>18:34:24</td>
<td>5jb1sgmjc7436</td>
<td>3398716340</td>
<td>187</td>
<td>170</td>
</tr>
</tbody>
</table>
Temporary Tablespace Usage

SELECT /*+leading(r x h) use_nl(h)*/
  h.sql_id
, h.sql_plan_hash_value
, COUNT(DISTINCT sql_exec_id) num_execs
, SUM(10) ash_secs
, 10*COUNT(DISTINCT sample_id) elap_secs
, ROUND(MAX(temp_space_allocated)/1024/1024,0) tempMb
, COUNT(distinct r.prcsinstance) PIs
FROM dba_hist_snapshot x
, dba_hist_active_sess_history h
, sysadm.psprcsrqst r
WHERE ...
ORDER BY ash_secs DESC
**Temporary Tablespace Usage**

- Can see temporary usage of individual SQL statements

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>SQL_PLAN_HASH_VALUE</th>
<th>NUM_EXECS</th>
<th>ASH_SECS</th>
<th>ELAP_SECS</th>
<th>TEMPMB</th>
<th>PIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>a47fb0x1b23jn</td>
<td>483167840</td>
<td>3</td>
<td>6280</td>
<td>910</td>
<td>132</td>
<td>3</td>
</tr>
<tr>
<td>cbw2hztjyztnq</td>
<td>544286790</td>
<td>4</td>
<td>5920</td>
<td>390</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>fcrxxp8f0c8cg</td>
<td>2119221636</td>
<td>2</td>
<td>4480</td>
<td>280</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>8h7ga9g761naj</td>
<td>4127129594</td>
<td>1</td>
<td>3980</td>
<td>3980</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>8cfyfzadglob4k</td>
<td>4127129594</td>
<td>1</td>
<td>3450</td>
<td>3450</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>3gz46jhw7b5x8</td>
<td>3643021188</td>
<td>8</td>
<td></td>
<td>290</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>a47fb0x1b23jn</td>
<td>3805993318</td>
<td>1</td>
<td>2610</td>
<td>1120</td>
<td>132</td>
<td>1</td>
</tr>
<tr>
<td>dxqkbuynhpgq09</td>
<td>2119221636</td>
<td>1</td>
<td>2240</td>
<td>140</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>c75jcr5s71s2h</td>
<td>2119221636</td>
<td>1</td>
<td>2240</td>
<td>140</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Effect of Plan Stability

- Three scenarios
  1. Large payroll – collecting profiles
  2. Small payroll – no profiles
  3. Small payroll – with profiles applied
Effect of Plan Stability

```sql
SELECT /*+ LEADING(@q1 r1@q1 x1@q1 h1@q1) USE_NL(h1@q1)
    LEADING(@q2 r2@q2 x2@q2 h2@q2) USE_NL(h2@q2)
    LEADING(@q3 r3@q3 x3@q3 h3@q3) USE_NL(h3@q3) */
    q1.sql_id,
    q1.sql_plan_hash_value, q1.ash_secs,
    DECODE(q1.sql_plan_hash_value,q2.sql_plan_hash_value,'**SAME**',
        q2.sql_plan_hash_value)   sql_plan_hash_value2,
    q2.ash_secs,
    DECODE(q1.sql_plan_hash_value,q3.sql_plan_hash_value,'**SAME**',
        q3.sql_plan_hash_value)   sql_plan_hash_value3,
    q3.ash_secs
FROM (...
    ) Q1
LEFT OUTER JOIN (...
    ) Q2 ON q1.sql_id = q2.sql_id
INNER JOIN (...
    ) Q3 ON q1.sql_id = q3.sql_id
ORDER BY q3.ash_secs desc, q1.sql_id
```

Usual Query in each of three in-line views
## Effect of Plan Stability

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>SCENARIO 1 ASH_SECS</th>
<th>SCENARIO 2 ASH_SECS</th>
<th>SCENARIO 3 ASH_SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4uzmzh74rdzn</td>
<td>2514155560</td>
<td>&gt;28750</td>
<td><strong>SAME</strong></td>
</tr>
<tr>
<td>4n482cm%f9qyn</td>
<td>1595742310</td>
<td>140 <strong>SAME</strong></td>
<td>889 <strong>SAME</strong></td>
</tr>
<tr>
<td>2186y2u9r4luV</td>
<td>1145973676</td>
<td>630 <strong>SAME</strong></td>
<td>531 <strong>SAME</strong></td>
</tr>
<tr>
<td>1n2dfvb3jrn2m</td>
<td>1293172177</td>
<td>150 <strong>SAME</strong></td>
<td>150 <strong>SAME</strong></td>
</tr>
<tr>
<td>652y9682bqqvp</td>
<td>3325291917</td>
<td>30 <strong>SAME</strong></td>
<td>110 <strong>SAME</strong></td>
</tr>
<tr>
<td>d8gxmqp2zydta</td>
<td>1716202706</td>
<td>10 <strong>SAME</strong></td>
<td>32 <strong>SAME</strong></td>
</tr>
<tr>
<td>2np47twhd5nga</td>
<td>3496258537</td>
<td>10 <strong>SAME</strong></td>
<td>27 <strong>SAME</strong></td>
</tr>
<tr>
<td>4ru0618dswz3y</td>
<td>2621940820</td>
<td>100 <strong>SAME</strong></td>
<td>22 <strong>SAME</strong></td>
</tr>
<tr>
<td>4ru0618dswz3y</td>
<td>539127764</td>
<td><strong>SAME</strong></td>
<td>22 <strong>SAME</strong></td>
</tr>
<tr>
<td>4ru0618dswz3y</td>
<td>539127764</td>
<td>22 <strong>SAME</strong></td>
<td>22 <strong>SAME</strong></td>
</tr>
<tr>
<td>4ru0618dswz3y</td>
<td>1403673054</td>
<td>539127764</td>
<td>22 <strong>SAME</strong></td>
</tr>
<tr>
<td>gnunu2hfkjm2yd</td>
<td>1559321680</td>
<td>80 <strong>SAME</strong></td>
<td>19 <strong>SAME</strong></td>
</tr>
</tbody>
</table>

1. Better with profile, but not great, but it did run
2. A little worse
3. 4 execution plans, now just 1, much better.
SELECT /*+LEADING(x h) USE_NL(h)*/ h.program,
       h.sql_id, h.sql_plan_hash_value,
       SUM(10) ash_secs,
       COUNT(DISTINCT h.sql_exec_id) execs,
       COUNT(DISTINCT xid) XIDs
FROM DBA_HIST_SNAPSHOT x,
     DBA_HIST_ACTIVE_SESS_HISTORY h
WHERE x.END_INTERVAL_TIME   >= …
   AND x.begin_interval_time <= …
   AND h.sample_TIME >= …
   AND h.sample_time <= …
   AND h.SNAP_id = X.SNAP_id
   AND h.dbid = x.dbid
   AND h.instance_number = x.instance_number
GROUP BY h.program, h.sql_id, h.sql_plan_hash_value
ORDER BY ash_secs desc
/

How Many Executions (from 11g)
### How Many Executions?

<table>
<thead>
<tr>
<th>PROGRAM</th>
<th>SQL_ID</th>
<th>Hash Value</th>
<th>ASH</th>
<th>Secs</th>
<th>EXECs</th>
<th>XIDS</th>
<th>USERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_async.exe</td>
<td>7q90ra0vmd9xx</td>
<td>2723153562</td>
<td></td>
<td>3020</td>
<td>297</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>t_async.exe</td>
<td>6mw25bgbh1stj</td>
<td>1229059401</td>
<td></td>
<td>320</td>
<td>32</td>
<td>0</td>
<td>17</td>
</tr>
</tbody>
</table>

- **Samples ≈ Executions**
  - Based on DBA_HIST_ACTIVE_SESS_HISTORY
  - 1 sample / 10 seconds.
  - Each sample is worth 10 seconds.
  - Probably underestimates number of executions.
SELECT /*+leading(r q x h) use_nl(h)*/
  NULLIF(h.top_level_sql_id, h.sql_id) top_level_sql_id,
  h.sql_id
...
FROM dba_hist_snapshot x
  , dba_hist_active_sess_history h
...

SQL_ID – v -
TOP_LEVEL_SQL_ID
<table>
<thead>
<tr>
<th>TOPLEVEL_SQL</th>
<th>SQL_ID</th>
<th>Hash Value</th>
<th>SQL_IDS</th>
<th>PLAN_EXECS</th>
<th>PLAN_ASH_SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>210</td>
<td></td>
</tr>
<tr>
<td>6np8gdbrmj8s4</td>
<td>2609910643</td>
<td>8</td>
<td>12</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>105xa4pfkv2jz</td>
<td>1dtnz2z7ujv23</td>
<td>3901024798</td>
<td>2</td>
<td>14</td>
<td>140</td>
</tr>
<tr>
<td>3m3ubmf7529mh</td>
<td>2188542943</td>
<td>2</td>
<td>13</td>
<td>140</td>
<td></td>
</tr>
<tr>
<td>g21xv51r09w4j</td>
<td>2905535923</td>
<td>1</td>
<td>10</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
Things That Can Go Wrong

- **DISPLAY_AWR**
  - Correct plan, old costs & binds
  - ORA-6502 – very large SQL
  - ORA-44002 – short-lived objects(?)
  - ORA-1422 – duplicate SQL from cloning
- **Statement not in Library Cache**
  - Only Some Statements in Library Cache
- **Lots of short-lived non-shareable SQL**
Statement not in Library Cache

SELECT * FROM table(dbms_xplan.display_cursor('gpdwr389mg61h', 0, 'ADVANCED'));

PLAN_TABLE_OUTPUT

-----------------------------------------------
SQL_ID: gpdwr389mg61h, child number: 0 cannot be found
### Statements not in AWR Repository

<table>
<thead>
<tr>
<th>RANKING</th>
<th>SQL_ID</th>
<th>Hash Value</th>
<th>TOT_ASH_SECS</th>
<th>TOT_AWR_SECS</th>
<th>SQL_IDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1wfhpn9k2x3hq</td>
<td>0</td>
<td>7960</td>
<td>4600</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>2wsan9j1pk3j2</td>
<td>1061502179</td>
<td>4230</td>
<td>4230</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>bnxddum0rrvyh</td>
<td>918066299</td>
<td>2640</td>
<td>1200</td>
<td>179</td>
</tr>
<tr>
<td>1</td>
<td>02cymzmyt4mdh</td>
<td>508527075</td>
<td>2070</td>
<td>0</td>
<td>45</td>
</tr>
<tr>
<td>1</td>
<td>5m0xbf7vn8490</td>
<td>2783301143</td>
<td>1700</td>
<td>0</td>
<td>49</td>
</tr>
<tr>
<td>1</td>
<td>0jfp0g054cb3n</td>
<td>4135405048</td>
<td>1500</td>
<td>0</td>
<td>47</td>
</tr>
<tr>
<td>1</td>
<td>11bygm2nyqh0s</td>
<td>3700906241</td>
<td>1300</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>1</td>
<td>6ggg99cfg26kwb</td>
<td>3058602782</td>
<td>1300</td>
<td>1300</td>
<td>1</td>
</tr>
</tbody>
</table>

- 207 samples, representing 2070 seconds of SQL
- 45 distinct SQL_IDs, we don’t know how many executions
  - probably one per SQL_ID.
- Often associated with non-shareable SQL
### Lots of Short-lived SQL Statements

A table of SQL statements with their execution and ASH seconds:

<table>
<thead>
<tr>
<th>PRCSINSTANCE</th>
<th>NUM_SQL_ID</th>
<th>HASH_VALUE</th>
<th>EXEC_SECS</th>
<th>ASH_SECS</th>
</tr>
</thead>
<tbody>
<tr>
<td>50007687</td>
<td>169</td>
<td>953836181</td>
<td>3170</td>
<td>1690</td>
</tr>
<tr>
<td>50007687</td>
<td>50</td>
<td>807301148</td>
<td>3170</td>
<td>500</td>
</tr>
<tr>
<td>50007687</td>
<td>22</td>
<td>4034059499</td>
<td>3170</td>
<td>220</td>
</tr>
<tr>
<td>50007687</td>
<td>14</td>
<td>2504475139</td>
<td>3170</td>
<td>140</td>
</tr>
<tr>
<td>50007687</td>
<td>2</td>
<td>0</td>
<td>3170</td>
<td>70</td>
</tr>
<tr>
<td>50007687</td>
<td>1</td>
<td>1309703960</td>
<td>3170</td>
<td>20</td>
</tr>
<tr>
<td>50007687</td>
<td>1</td>
<td>3230852326</td>
<td>3170</td>
<td>10</td>
</tr>
</tbody>
</table>

...  

- Probably more than 169 statements that took about 1690 seconds, but we only sampled 169.
### Application Reports: Lots of Compiles

![Batch Timings - Summary]

<table>
<thead>
<tr>
<th>Process</th>
<th>Type: Application Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance: 50007687</td>
<td></td>
</tr>
<tr>
<td>Name: AR_CNDMON</td>
<td></td>
</tr>
<tr>
<td>Description: Receivables Condition Monitor</td>
<td></td>
</tr>
</tbody>
</table>

#### Time (in milliseconds)

- Elapsed: 3164410
- In PeopleCode: 90500
- In SQL: 2940090

#### Trace Level

- Application Engine: 1159
- SQL & PeopleCode: 128

<table>
<thead>
<tr>
<th>Program</th>
<th>Detail line identifier</th>
<th>Compile Count</th>
<th>Compile Time</th>
<th>Execute Count</th>
<th>Execute Time</th>
<th>Fetch Count</th>
<th>Fetch Time</th>
<th>PC Count</th>
<th>PC Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR_CNDMON</td>
<td>CHK_USER.INSPRCS2.S</td>
<td>64224</td>
<td>30960</td>
<td>64224</td>
<td>2566340</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AR_CNDMON</td>
<td>CHK_USER.LDSQL.S</td>
<td>64224</td>
<td>6230</td>
<td>64224</td>
<td>230220</td>
<td>64224</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AR_CNDMON</td>
<td>CANCLACT.CANSLST3.S</td>
<td>64224</td>
<td>6230</td>
<td>64224</td>
<td>230220</td>
<td>64224</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
• This seems to be associated with very large SQL statements

SQL_ID 9vnan5kqsh1aq

An uncaught error happened in prepare_sql_statement:
ORA-06502: PL/SQL: numeric or value error

Plan hash value: 2262951047

<table>
<thead>
<tr>
<th>Id</th>
<th>Operation</th>
<th>Name</th>
<th>Rows</th>
<th>Bytes</th>
<th>Cost (%CPU)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>HASH GROUP BY</td>
<td></td>
<td>1</td>
<td>164</td>
<td>1 (100)</td>
<td>00:00:01</td>
</tr>
</tbody>
</table>

...
I have seen this with Global Temporary Tables and with direct path mode (the APPEND hint).

---

ERROR: cannot get definition for table 'BZTNCMUX31XP5'
ORA-44002: invalid object name
This happens on a database that has been cloned, often from production to test.

- Consider recreating AWR repository on clone?

An uncaught error happened in
prepare_sql_statement : ORA-01422: exact fetch returns more than requested number of rows
• Workaround

DELETE FROM sys.wrh$_sqltext t1
WHERE t1.dbid != (
    SELECT d.dbid FROM v$database d)
AND EXISTS(
    SELECT 'x'
    FROM sys.wrh$_sqltext t2
    WHERE t2.dbid = (  
        SELECT d.dbid FROM v$database d)
    AND t2.sql_id = t1.sql_id)
Conclusion

• ASH data
  – Consider longer term retention in a central AWR repository

• Application Instrumentation is essential

• Lots of ways to query the data
  – Be imaginative!

• Understand the pitfalls.
Questions?
Questions?
Conclusion

- ASH data
  - Consider longer term retention in a central AWR repository
- Application Instrumentation is essential
- Lots of ways to query the data
  - Be imaginative!
- Understand the pitfalls.