Application Performance Tuning: A Practitioner’s Approach

Roger Cornejo
roger.d.cornejo@gsk.com

SEOUG
Raleigh/Durham, NC
November 3, 2011
Tuning from practitioners perspective

- **Goal:** improve query response times
- **Topics:** SQL Tools, Tips, Tricks/Traps
- **Tools:** mining standard V$ views

**Basic Tuning Strategy**
- Seek relevant **evidence** for the case
- Devise tuning **plan** from evidence
- Take **action** / **Measure** benefit

- No “sliver bullet”, so having a well understood / rehearsed bag of tricks will **improve your tuning skills and value**

**Focus on things that cause high I/O e.g.**
- Full scans, poor indexes, i/o events, single row processing vs set processing ... to name a few
What’s inside the bag-of-tricks?

- Ignoring Performance Tuning
  - Trick: Eliminate outer Join
  - High-Water Mark and FULL scans
- Performance Tuning Tools
  - Tool: Active Sessions and SQL
  - Tool: Sql_id Profile
- Trap/Tip: Update of very large table
- Tool: Monitor TEMP space usage
- Tool: Statspack Data Mining
- Tip: Analysis of Wait Events
- Tool: AWR / DBA_HIST Data Mining
- Tip: Ultra-Slow MV Creation
- WITH Clause Sub-query Factoring
- Tip: Single Row vs. Set Processing
Potential Problem Scenarios for performance

- **Architecture**: location of DB server and app server
  - client server over wide area network

- **Design**: moving from logical to physical design
  - partitioning needed?
  - is design over normalized or not normalized?
  - is the row too big?

- **Implementation**: moving from DEV to PROD
  - data volumes scale-up
  - multi-user concurrent access scale-up

- **Upgrading**: Oracle to Oracle

- **Porting**: to another machine

BASELINE METRICS
Trick - Eliminate outer Join

Outer Join is often needed for look-ups in table where value may not be there

```sql
select myTab.code , code.translation , myTab.position
from code , myTab
where code.code (+) = myTab.code
order by position;
```

since Oracle V8: select statements in the select list can be used to replace outer join

```sql
select myTab.code ,
(select code.translation
    from code where code.code = myTab.code) translation
, myTab.position
from myTab
order by position
```

Trap – High-Water Mark and FULL scans

- For FULL scans Oracle reads every block below high-water mark (even if block has no data)
- Often found in tables that expand and contract a lot (e.g. table used as a queue)

**Case Study:** 167 Mb table; few rows; no index
- >99% of the 'buffer busy waits' and
- >90% of the 'physical reads' on the instance.
- ~24K/month UPDATE and DELETE operations on this table no indexes used
- Small number of rows but full scan killing performance

**Solution:** Adding index on key column in WHERE clause
Case Study: 888 mb table; Full Scan; **90% of rows deleted**
- Query not using existing indexes; app. can’t change query
- “high-water mark” is still high;
  confirmed via DBMS_SPACE.UNUSED_SPACE
  0 Unused Blocks [block above HWM; not read in full scans]

Solution: Reorganize table:
ALTER TABLE x MOVE tablespace_a;
**Case Study:** add a column to a 26 gig table (34M rows) and update new column to a particular value

- **Solution 1:** 
  ```sql
  UPDATE t set new_col='val';
  ```
  after many many hours **ORA-01555: snapshot too old**
  undo retention increased many gig added to undo tablespace

- **Solution 2:** PL/SQL updating 5000 rows at a time => **SLOW**
  estimated to take several years to get through all the rows

**Solution 3:** Tried and true method for massive updates

1. CTAS [Create Table As Select] to a new table with update built in
2. Recreate indexes, triggers …
3. Drop the old table
4. Rename new table to old name
5. Rebuild any views, procedures … that may have been invalidated

In this case study, soln. completed in 3 hours elapsed time
Tools – Many Performance Tuning Tools Available

- Oracle Tools
  - TKProf
  - Statspack
  - AWR Automated Workload Repository (extra cost)

- 3rd Party tools and freeware tools

- Specialized testing tools (e.g. LoadRunner)

- Custom Tools
  - Queries against various v$tables
  - Queries against Statspack and AWR data
**Tool – Monitor v$sort_usage (TEMP Space)**

- **Case Study:** TEMP tablespace blowouts
  ORA-1652: unable to extend temp segment by 128 in tablespace TEMP
- User getting 1652 and is a victim of another query using TEMP
- v$sort_usage: temp tablespace usage, showing queries/users
  - transient data and only shows you what is happening now

**Solution:** Persist information on v$sort_usage, v$session, and v$sqlarea so we could get back to the original queries and users causing the blowouts

1. Create a `sort_usage_history` table to hold the data
2. Create a pl/sql procedure that inserts into `sort_usage_history` from join view of v$sort_usage, v$session, and v$sqlarea
3. Submit the PL/SQL as a job to collect on an interval
   --run every 10 minutes starting now -- vary as desired
   ```sql
   dbms_job.submit(v_job, 'persist_sort_usage_history;', sysdate, 'SYSDATE+(10/24/60)');
   ```
4. Query `sort_usage_history` to find high temp space users and their sql
Tool – Active Sessions and SQL

- **Active Sessions and SQL RAC - redux.sql**
- Gives quick idea as to what the active sessions are doing
- lists sessions with status active and their sql execution times and if doing full scan

### Spreadsheet

<table>
<thead>
<tr>
<th>SID</th>
<th>SERIAL</th>
<th>INS</th>
<th>USER NAME</th>
<th>OSUSER</th>
<th>SQL ID</th>
<th>TIME_PER</th>
<th>EVENT</th>
<th>OBJECT_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>567</td>
<td>15786</td>
<td>1</td>
<td>MOSAIC_USER</td>
<td>uzz5629</td>
<td>9mn557vp98k79</td>
<td>00:00:10</td>
<td>SQL*Net message from client</td>
<td>MOSAIC.AUD_EVENT</td>
</tr>
<tr>
<td>832</td>
<td>18416</td>
<td>1</td>
<td>P3I_PMMD_APP</td>
<td>oracle</td>
<td>2gap141ndy0zg</td>
<td>00:00:14</td>
<td>db file sequential read</td>
<td></td>
</tr>
<tr>
<td>840</td>
<td>38488</td>
<td>1</td>
<td>ICARUS_US</td>
<td>oxx66716</td>
<td>8gjus6jxqwcwa</td>
<td>2 22:16:30</td>
<td>db file sequential read</td>
<td>ICARUSCPI.PRODUCT</td>
</tr>
<tr>
<td>1069</td>
<td>62240</td>
<td>1</td>
<td>ORAAPPSS</td>
<td>rdc0208</td>
<td>9mj0tx38rhq4z</td>
<td>00:00:00</td>
<td>SQL*Net more data to client</td>
<td>SYS.1_OBJ1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SID</th>
<th>SERIAL</th>
<th>INS</th>
<th>USER NAME</th>
<th>SCAN</th>
<th>SQL TEXT</th>
<th>STATUS</th>
<th>MACHINE_NAME</th>
<th>LOGON TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>567</td>
<td>15786</td>
<td>1</td>
<td>MOSAIC User</td>
<td>FULL SCAN</td>
<td>select distinct wl.Worklistld, w</td>
<td>ACTIVE</td>
<td>WMSERVICE\DH1M7PJ1</td>
<td>25-OCT 14:14:04</td>
</tr>
<tr>
<td>832</td>
<td>18416</td>
<td>1</td>
<td>P3I_PMMD_APP</td>
<td>FULL SCAN</td>
<td>delete from &quot;P3I_PMMD_APP&quot;</td>
<td>ACTIVE</td>
<td>kopsduxrac36</td>
<td>27-OCT 14:17:41</td>
</tr>
<tr>
<td>840</td>
<td>38488</td>
<td>1</td>
<td>ICARUS_US</td>
<td>FULL SCAN</td>
<td>SELECT /*+ DRIVING_SITE */</td>
<td>ACTIVE</td>
<td>US1SAWN00452</td>
<td>24-OCT 11:18:04</td>
</tr>
<tr>
<td>1069</td>
<td>62240</td>
<td>1</td>
<td>ORAAPPSS</td>
<td>FULL SCAN</td>
<td>with sqlarea as (select inst_id)</td>
<td>ACTIVE</td>
<td>WMSERVICE\RTPWL06</td>
<td>27-OCT 14:18:09</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SID</th>
<th>SERIAL</th>
<th>INS</th>
<th>USER NAME</th>
<th>TERMINAL</th>
<th>PROGRAM</th>
<th>LAST_CALL_ET</th>
</tr>
</thead>
<tbody>
<tr>
<td>567</td>
<td>15786</td>
<td>1</td>
<td>MOSAIC_USER</td>
<td>DH1M7PJ1</td>
<td>genericreadergui.exe</td>
<td>6</td>
</tr>
<tr>
<td>832</td>
<td>18416</td>
<td>1</td>
<td>P3I_PMMD_APP</td>
<td>UNKNOWN</td>
<td>oracle@kopsduxrac36</td>
<td>29</td>
</tr>
<tr>
<td>840</td>
<td>38488</td>
<td>1</td>
<td>ICARUS_US</td>
<td>unknown</td>
<td>JDBC Thin Client</td>
<td>259070</td>
</tr>
<tr>
<td>1069</td>
<td>62240</td>
<td>1</td>
<td>ORAAPPSS</td>
<td>RTPWL06D03647</td>
<td>TOAD.exe</td>
<td>0</td>
</tr>
</tbody>
</table>
Tool – Sql_id Profile

SQL*Plus script producing HTML page including:

- Session info / Execution info
- Sql_Plan
- Table Cardinality / Indexes
- Bind Variables / Wait Summary

sql_id = 8qjusp6jxqcwa

html page example from previous page

Script - sqlid_profile_RAC.sql
Oracle Statistics Package (Statspack) is a set of utilities provided by Oracle for collecting and reporting on performance statistics.

References:
- Oracle Statspack Survival Guide:
- Statspack in Oracle:

Data accessible through the standard Statspack report or through custom queries:
- 67 tables, but will generally query just a few
Tip – Statspack / PERFSTAT Tables

a few of the 67 tables

- stats$sys_time_mode

Image from: http://www.dba-oracle.com/art_statspack.htm
example using statistic 'DB time' to produce `db_time_per_hour`
- Rough measure of how busy the database is
- helps identify where the Oracle database spends its CPU processing time
- Consider some stats from here as a baseline

code is pulling data from `stats$sys_time_model`
which is similar to `v$sys_time_model`
- cumulative times, expressed in microseconds, collected from all non-idle database sessions
  uses **LAG function** to get deltas between periods
Tool – Statspack – DB Time per hour (continue)

<table>
<thead>
<tr>
<th>STAT_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB CPU</td>
</tr>
<tr>
<td>DB time</td>
</tr>
<tr>
<td>Java execution elapsed time</td>
</tr>
<tr>
<td>PL/SQL compilation elapsed time</td>
</tr>
<tr>
<td>PL/SQL execution elapsed time</td>
</tr>
<tr>
<td>RMAN cpu time (backup/restore)</td>
</tr>
<tr>
<td>background cpu time</td>
</tr>
<tr>
<td>background elapsed time</td>
</tr>
<tr>
<td>connection management call elapsed time</td>
</tr>
<tr>
<td>failed parse (out of shared memory) elapsed time</td>
</tr>
<tr>
<td>failed parse elapsed time</td>
</tr>
<tr>
<td>hard parse (bind mismatch) elapsed time</td>
</tr>
<tr>
<td>hard parse (sharing criteria) elapsed time</td>
</tr>
<tr>
<td>hard parse elapsed time</td>
</tr>
<tr>
<td>inbound PL/SQL rpc elapsed time</td>
</tr>
<tr>
<td>parse time elapsed</td>
</tr>
<tr>
<td>repeated bind elapsed time</td>
</tr>
<tr>
<td>sequence load elapsed time</td>
</tr>
</tbody>
</table>

```sql
select stat_name
from
  v$sys_time_model
order by stat_name
;

- 19 statistics in 10g
  (not available in 9i)
### Tool – Statspack – DB Time per hour (continue)

- **stats$sys_time_model.sql**
- various statistics from statspack data over time

<table>
<thead>
<tr>
<th>SNAP_TIME</th>
<th>SNAP_DAY</th>
<th>SNAP_HOUR</th>
<th>db time per hour</th>
<th>parse time per hour</th>
<th>sql exec time per hour</th>
<th>db cpu time per hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011/10/28 22:30</td>
<td>2011/10/28</td>
<td>22:30</td>
<td>947</td>
<td>57</td>
<td>948</td>
<td>714</td>
</tr>
<tr>
<td>2011/10/28 23:30</td>
<td>2011/10/28</td>
<td>23:30</td>
<td>1239</td>
<td>31</td>
<td>1239</td>
<td>965</td>
</tr>
<tr>
<td>2011/10/29 00:30</td>
<td>2011/10/29</td>
<td>00:30</td>
<td>512</td>
<td>17</td>
<td>516</td>
<td>378</td>
</tr>
<tr>
<td>2011/10/29 01:30</td>
<td>2011/10/29</td>
<td>01:30</td>
<td>1097</td>
<td>28</td>
<td>1099</td>
<td>865</td>
</tr>
<tr>
<td>2011/10/29 02:30</td>
<td>2011/10/29</td>
<td>02:30</td>
<td>367</td>
<td>13</td>
<td>371</td>
<td>276</td>
</tr>
<tr>
<td>2011/10/29 03:30</td>
<td>2011/10/29</td>
<td>03:30</td>
<td>245</td>
<td>6</td>
<td>247</td>
<td>188</td>
</tr>
<tr>
<td>2011/10/29 04:30</td>
<td>2011/10/29</td>
<td>04:30</td>
<td>207</td>
<td>4</td>
<td>210</td>
<td>167</td>
</tr>
<tr>
<td>2011/10/29 05:30</td>
<td>2011/10/29</td>
<td>05:30</td>
<td>255</td>
<td>8</td>
<td>258</td>
<td>206</td>
</tr>
<tr>
<td>2011/10/29 06:30</td>
<td>2011/10/29</td>
<td>06:30</td>
<td>321</td>
<td>15</td>
<td>324</td>
<td>252</td>
</tr>
<tr>
<td>2011/10/29 07:30</td>
<td>2011/10/29</td>
<td>07:30</td>
<td>288</td>
<td>17</td>
<td>293</td>
<td>230</td>
</tr>
<tr>
<td>2011/10/29 08:30</td>
<td>2011/10/29</td>
<td>08:30</td>
<td>328</td>
<td>12</td>
<td>331</td>
<td>233</td>
</tr>
<tr>
<td>2011/10/29 09:30</td>
<td>2011/10/29</td>
<td>09:30</td>
<td>2570</td>
<td>27</td>
<td>2574</td>
<td>2461</td>
</tr>
<tr>
<td>2011/10/29 10:30</td>
<td>2011/10/29</td>
<td>10:30</td>
<td>294</td>
<td>22</td>
<td>298</td>
<td>220</td>
</tr>
<tr>
<td>2011/10/29 11:30</td>
<td>2011/10/29</td>
<td>11:30</td>
<td>357</td>
<td>37</td>
<td>361</td>
<td>263</td>
</tr>
<tr>
<td>2011/10/29 12:30</td>
<td>2011/10/29</td>
<td>12:30</td>
<td>227</td>
<td>22</td>
<td>230</td>
<td>173</td>
</tr>
<tr>
<td>2011/10/29 13:30</td>
<td>2011/10/29</td>
<td>13:30</td>
<td>581</td>
<td>77</td>
<td>584</td>
<td>483</td>
</tr>
<tr>
<td>2011/10/29 14:30</td>
<td>2011/10/29</td>
<td>14:30</td>
<td>140</td>
<td>5</td>
<td>144</td>
<td>119</td>
</tr>
<tr>
<td>2011/10/29 15:30</td>
<td>2011/10/29</td>
<td>15:30</td>
<td>140</td>
<td>5</td>
<td>144</td>
<td>119</td>
</tr>
</tbody>
</table>
Tool – Statspack – STATS$SQL_SUMMARY

- analogous to v$sqlstats
- tracks executions, parse calls, data blocks read … and written for each SQL statement.

**Example execution for 9-10 AM interval**

**SQL Code:** stats$sql_summary.sql
Tool – Statspack – `STATS$SYSTEM_EVENT`

- analogous to `v$system_event`
- persists overall wait events during period

**SQL Code:** `PERFSTAT wait events over time.sql`

<table>
<thead>
<tr>
<th>SNAP_ID</th>
<th>SNAP_TIME</th>
<th>EVENT</th>
<th>TIME_WAIT_DELTA</th>
<th>TIME_RANK</th>
</tr>
</thead>
<tbody>
<tr>
<td>40132</td>
<td>10/29/2011 9:30:01 AM</td>
<td>Backup: sbtwrite2</td>
<td>4684571187</td>
<td>1</td>
</tr>
<tr>
<td>40132</td>
<td>10/29/2011 9:30:01 AM</td>
<td>db file sequential read</td>
<td>2531413160</td>
<td>2</td>
</tr>
<tr>
<td>40132</td>
<td>10/29/2011 9:30:01 AM</td>
<td>db file scattered read</td>
<td>2048723389</td>
<td>3</td>
</tr>
<tr>
<td>40132</td>
<td>10/29/2011 9:30:01 AM</td>
<td>log file parallel write</td>
<td>800027239</td>
<td>4</td>
</tr>
<tr>
<td>40132</td>
<td>10/29/2011 9:30:01 AM</td>
<td>db file parallel write</td>
<td>496954005</td>
<td>5</td>
</tr>
<tr>
<td>40133</td>
<td>10/29/2011 10:30:01 AM</td>
<td>db file sequential read</td>
<td>2983951281</td>
<td>1</td>
</tr>
<tr>
<td>40133</td>
<td>10/29/2011 10:30:01 AM</td>
<td>db file scattered read</td>
<td>2253832857</td>
<td>2</td>
</tr>
<tr>
<td>40133</td>
<td>10/29/2011 10:30:01 AM</td>
<td>Backup: sbtwrite2</td>
<td>1820344675</td>
<td>3</td>
</tr>
<tr>
<td>40133</td>
<td>10/29/2011 10:30:01 AM</td>
<td>Backup: sbtclose2</td>
<td>1378197825</td>
<td>4</td>
</tr>
<tr>
<td>40133</td>
<td>10/29/2011 10:30:01 AM</td>
<td>RMAN backup &amp; recovery I/O</td>
<td>327117134</td>
<td>5</td>
</tr>
</tbody>
</table>
Tip – Analysis of Wait Events

- Incredibly complex, but extremely helpful if you can develop an understanding of common events

- `select * from v$system_event;`  
  -- 117 wait events  

- `select distinct wait_class from v$system_event;`  
  -- 10 classes  

- `select * from v$system_event where wait_class = 'Idle';`  
  -- 14 Idle events

  plus other “Idle” events you may want to ignore

- Maintain a record of wait events you have researched and build on that knowledge
Tool – Statspack – $STATS\$SYSSTAT$

- analogous to v$systat
- persists stats on 387 parameters in 10g during period
- statistics are cumulative => need for LAG function to get the DELTAS

Example: parse related stats:
- 'parse time cpu'
- 'parse time elapsed'
- 'parse count (hard)'
- 'CPU used by this session'

Consider some metrics from stats$sysstat as a BASELINE

Sample data

SQL Code: PERFSTAT system statistics.sql
Tip – AWR / DBA_HIST Data Mining

- **Automated Workload Repository (AWR)**
  - cost feature FROM Oracle for collecting and reporting on performance statistics

- **Takes snapshots of various v$ tables – persists in ...**
  - DBA_HIST_SNAPSHOT snapshots every hour
  - v$active_session_history data capture every second
  - DBA_HIST_ACTIVE_SESS_HISTORY rollup from v$ table
  - Many others

- **Data accessible through the standard AWR reports or through custom queries**
  - 78 tables, but will generally query just a few

- **A Tour of the AWR Tables**
Tip – DBA_HIST Tables

- **78 Tables Listing**

- **DBA_HIST_SNAPSHOT** - Starting point
  - time period for each snapshot (1 hour intervals)

- **Resource usage**
  - **DBA_HIST_SYSSTAT** (analogous V$SYSSTAT)

- **Session and event history**
  - **gv$active_session_history**
  - **DBA_HIST_ACTIVE_SESS_HISTORY**
    - 1 sec intervals - captures what’s in the v$ tables at that time => it will miss some events
Tool – *CPU Load from DB* (Third Party Script)

- Time Ranges of load spikes - `aas-exact.sql`
- Uses `DBA_HIST_SYSSTAT` (analogous V$SYSSTAT)

<table>
<thead>
<tr>
<th>BER</th>
<th>SNAP_ID</th>
<th>BEGIN_HOUR</th>
<th>SECONDS_PER_HOUR</th>
<th>AAS</th>
<th>PROBLEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>35152</td>
<td>2011-10-24 15:00</td>
<td>17424</td>
<td>4.8</td>
<td>CPU available</td>
</tr>
<tr>
<td>1</td>
<td>35151</td>
<td>2011-10-24 14:00</td>
<td>13717</td>
<td>3.8</td>
<td>CPU available</td>
</tr>
<tr>
<td>1</td>
<td>35150</td>
<td>2011-10-24 13:00</td>
<td>15718</td>
<td>4.4</td>
<td>CPU available</td>
</tr>
<tr>
<td>1</td>
<td>35149</td>
<td>2011-10-24 12:00</td>
<td>14058</td>
<td>3.9</td>
<td>CPU available</td>
</tr>
<tr>
<td>1</td>
<td>35148</td>
<td>2011-10-24 11:01</td>
<td>58023</td>
<td>16.1</td>
<td>There is a bottleneck</td>
</tr>
<tr>
<td>1</td>
<td>35147</td>
<td>2011-10-24 10:00</td>
<td>18081</td>
<td>5</td>
<td>CPU available</td>
</tr>
<tr>
<td>1</td>
<td>35146</td>
<td>2011-10-24 09:00</td>
<td>38775</td>
<td>10.8</td>
<td>Could have performan</td>
</tr>
<tr>
<td>1</td>
<td>35145</td>
<td>2011-10-24 08:00</td>
<td>19072</td>
<td>5.3</td>
<td>CPU available</td>
</tr>
<tr>
<td>1</td>
<td>35144</td>
<td>2011-10-24 07:00</td>
<td>15956</td>
<td>4.4</td>
<td>CPU available</td>
</tr>
</tbody>
</table>
### Tool – ASH - *all waits for a user name*

1. **ASH - events for a user.sql** *all waits for a username*
2. **Can modify basic query to see events for sql_id; session; object ...**

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>EVENT</th>
<th>OBJECT_ID_NAME</th>
<th>SESSION_STATE</th>
<th>TOT_DURATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0yprthkpr248z</td>
<td></td>
<td></td>
<td>ON CPU</td>
<td>10453168640</td>
</tr>
<tr>
<td>0014d582y0my7</td>
<td>library cache lock</td>
<td></td>
<td>WAITING</td>
<td>5067641075</td>
</tr>
<tr>
<td>93ga7698h0dad</td>
<td>library cache lock</td>
<td></td>
<td>WAITING</td>
<td>4982965507</td>
</tr>
<tr>
<td>2d14yqcc9cc86</td>
<td>DBMS_LDAP: LDAP operation</td>
<td></td>
<td>WAITING</td>
<td>419284648</td>
</tr>
<tr>
<td>gv5sn3ubxq0f6</td>
<td>JS kill job wait</td>
<td></td>
<td>WAITING</td>
<td>311606997</td>
</tr>
<tr>
<td>1qxs1g9m68gr4</td>
<td></td>
<td></td>
<td>ON CPU</td>
<td>65257883</td>
</tr>
<tr>
<td>4far7z6k93jrj</td>
<td></td>
<td></td>
<td>ON CPU</td>
<td>29014641</td>
</tr>
<tr>
<td>6tshctswzutbk</td>
<td></td>
<td></td>
<td>ON CPU</td>
<td>19265292</td>
</tr>
<tr>
<td>a73wbv1yu8x5c</td>
<td></td>
<td></td>
<td>ON CPU</td>
<td>11775473</td>
</tr>
<tr>
<td>2d14yqcc9cc86</td>
<td></td>
<td></td>
<td>ON CPU</td>
<td>10205203</td>
</tr>
<tr>
<td>c0hzbcntqwdmf</td>
<td>log buffer space</td>
<td>573884 MV_CUR_WBS_NODE</td>
<td>WAITING</td>
<td>8358106</td>
</tr>
<tr>
<td>9shsudizr87zti</td>
<td></td>
<td></td>
<td>ON CPU</td>
<td>8232428</td>
</tr>
</tbody>
</table>
**Tool – SQL_TEXT for a Sql_id**

- **AWR - sql_text for a sql_id.sql**

```
select max(DBMS_LOB.SUBSTR(sql_text, 4000))
    sql_text
from dba_hist_sqltext
where sql_id = :sql_id
;
```

- **93ga7698h0dad from previous slide**

---

**SQL_TEXT**

```
truncate table "PDR_REPORT"."MV_PROJECT" purge snapshot log
```
Tool – *All Wait Events for a given SQL*

- **ASH - events for sql_id.sql**
- **all waits for a sql_id**
  
  group by object, sql, event, session_state

- Uses gv$active_session_history
  
  all_objects, all_events

93ga7698h0dad from previous slide

<table>
<thead>
<tr>
<th>EVENT</th>
<th>OBJECT_ID_NAME</th>
<th>state</th>
<th>duration</th>
<th>event cnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>library cache lock</td>
<td></td>
<td>WAITING</td>
<td>5044695342</td>
<td>5111</td>
</tr>
<tr>
<td>db file sequential read</td>
<td>574049 PDR_REPORT.PK_MV_PROJECT</td>
<td>WAITING</td>
<td>92498</td>
<td>3</td>
</tr>
<tr>
<td>enq: RO - fast object reuse</td>
<td></td>
<td>WAITING</td>
<td>20073</td>
<td>3</td>
</tr>
<tr>
<td>local write wait</td>
<td>574049 PDR_REPORT.PK_MV_PROJECT</td>
<td>WAITING</td>
<td>8318</td>
<td>2</td>
</tr>
<tr>
<td>local write wait</td>
<td>573997 PDR_REPORT.MV_PROJECT</td>
<td>WAITING</td>
<td>2654</td>
<td>2</td>
</tr>
</tbody>
</table>
### Tool – Problem SQL Over Time

**ASH - problem sql over time.sql**

93ga7698h0dad from previous slide

<table>
<thead>
<tr>
<th>SAMPLE_MIN</th>
<th>EVENT</th>
<th>TIME_ON_CPU_CENTISEC</th>
<th>TIME_WAITED_CENTISEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-10-31 05:56</td>
<td>library cache lock</td>
<td>0</td>
<td>58790284</td>
</tr>
<tr>
<td>2011-10-31 05:57</td>
<td>library cache lock</td>
<td>0</td>
<td>58045448</td>
</tr>
<tr>
<td>2011-10-31 05:58</td>
<td>library cache lock</td>
<td>0</td>
<td>20576533</td>
</tr>
<tr>
<td>2011-10-31 05:59</td>
<td>library cache lock</td>
<td>0</td>
<td>14697461</td>
</tr>
<tr>
<td>2011-10-31 06:00</td>
<td>library cache lock</td>
<td>0</td>
<td>58790233</td>
</tr>
<tr>
<td>2011-10-31 06:01</td>
<td>library cache lock</td>
<td>0</td>
<td>58790002</td>
</tr>
<tr>
<td>2011-10-31 06:02</td>
<td>enq: RO - fast object reuse</td>
<td>0</td>
<td>6964</td>
</tr>
<tr>
<td>2011-10-31 06:02</td>
<td>library cache lock</td>
<td>0</td>
<td>58790003</td>
</tr>
<tr>
<td>2011-10-31 06:30</td>
<td>db file sequential read</td>
<td>0</td>
<td>9246</td>
</tr>
<tr>
<td>2011-10-31 06:46</td>
<td>local write wait</td>
<td>0</td>
<td>7136</td>
</tr>
<tr>
<td>2011-10-31 07:01</td>
<td>local write wait</td>
<td>0</td>
<td>1128</td>
</tr>
<tr>
<td>2011-10-31 07:16</td>
<td>enq: RO - fast object reuse</td>
<td>0</td>
<td>5848</td>
</tr>
<tr>
<td>2011-10-31 07:31</td>
<td>enq: RO - fast object reuse</td>
<td>0</td>
<td>7261</td>
</tr>
<tr>
<td>2011-10-31 08:31</td>
<td>library cache lock</td>
<td>0</td>
<td>1699159</td>
</tr>
<tr>
<td>2011-10-31 10:45</td>
<td>db file sequential read</td>
<td>0</td>
<td>10428</td>
</tr>
<tr>
<td>2011-10-31 11:00</td>
<td>local write wait</td>
<td>0</td>
<td>1182</td>
</tr>
<tr>
<td>2011-10-31 13:00</td>
<td>db file sequential read</td>
<td>0</td>
<td>72824</td>
</tr>
</tbody>
</table>
Tool – AWR – Find Expensive SQL


- **AWR - find-expensive sql.sql**
  
  *tweaked version from above link*

- **Pulls from** `dba_hist_sqlstat` (mainly), also
  - `dba_hist_snapshot`
  - `dba_hist_sql_plan`
  - `dba_hist_active_sess_history`
  - `dba_hist_sqltext`

- **Spreadsheet**
SQL Performance Tuning
with Oracle AWR & ASH:
Real World Use Cases
  Vlado Barun
  Oracle America

Wrap-up with a few more Tips Tricks and Traps …
**Case Study**: Create a MV from a very large and complex join;
- Took over 24 hours
- MV creation can generate lots of REDO
- MV getting different explain plan than just the SQL
- Data was static read-only, so MV refresh not needed

**Solution**: custom “MV” creation
- CTAS [Create Table As Select] with no logging and parallel options
  
  ```sql
  CREATE TABLE x NOLOGGING PARALLEL 4 AS
  SELECT * FROM view;
  ```

- In this example, completed in 15 minutes elapsed time for 4 million rows
Trap/Tip - Single Row vs. Set Processing

- Problem found in procedural code (any language)
  - Pattern to look for:
    - Loop on a cursor in one table
    - Aggregate some data
    - Update a record in another table for that row
    - Lather, rinse, repeat

- Generic Solution:
  - Use a single UPDATE ... SET ... WHERE ...

- Case Study: 6 page VB program taking 5 hours to update 5000 rows

Solution: replaced with single UPDATE, now 2.5 min.
Trick - Improved performance with WITH

/* Prev SQL Executed - 1 min 8 seconds to execute this */
/* Typical Query Style --------------------------------------- */

SELECT sql_text
  FROM v$sqlarea
  WHERE hash_value IN (SELECT prev_hash_value hash_value
                        FROM v$session
                        WHERE SID = (SELECT DISTINCT SID
                                      FROM v$mystat))

;

/* Prev SQL Executed - 3 seconds to execute this */
/* Subquery Factoring Query Style -------------------------- */

WITH my_sid AS (SELECT DISTINCT SID FROM v$mystat)
  , previous_hash AS (SELECT prev_hash_value hash_value FROM v$session
                      WHERE SID = (SELECT SID FROM my_sid))

SELECT sql_text
  FROM v$sqlarea
  WHERE hash_value IN (SELECT hash_value
                        FROM previous_hash)

;
Tip - Subquery Factoring a.k.a. the WITH clause

WITH <subquery> - SQL keyword introduced with Oracle 9i
- Create any number of “result tables” from nested selects
- Same as Inline views but “result tables” can be used multiple times
- Can be used as a performance optimisation technique
  - access to the data for the “result table” is done once
    [likely to be somewhat slower for very small queries]
  - Best when <subquery> is used more than once in main SELECT
  - Optimization treats the “result tables” as an in-line view or as a temporary table.
    [Can force materialization with the hint /*+ materialize */]
- Secondarily it can make query easier to read
  - helps modularize complex SQL statements to simplify the structure
- Tips:
  - a <subquery> defined in one WITH clause can be used in the subquery of any subsequent WITH clause
  - but cannot be nested WITH subquery within WITH subquery
  - WITH clause can be used anywhere a select stmt can be used i.e. INSERT, UPDATE, DELETE, …
Tip – Testing using SQL

- Often propose alternate tuned SQL to replace existing slow SQL

- Can easily test equivalence using SQL set operators with old as ( <select - - - > )
  , new as ( <select + + + > )
  (select * from old minus select * from new)
  union
  (select * from new minus select * from old)

- Above query returns no rows when the old and new queries return identical results.
GlaxoSmithKline

Roger Cornejo
roger.d.cornejo@gsk.com
Tuning with cardinality hint

- Available since 9i r1, but not so well documented
- Oracle is not always good at estimating cardinality (e.g.
  - global temporary table,
  - complex join;
  - temp materialized tables [including WITH subquery])
- tells the Optimizer how many rows are coming out of a row source
  [gives information to the optimizer but does not tell it what to do;
  gets you close enough to get optimizer to make the right choice]

- Other usage idea:
  Use for scalability testing to predict plans when data sizes change
  - Use cardinality hint to tell Oracle that you have a different number
    of rows as projected with growth.
  - Obtain execution plans with various cardinality values to evaluate
    impacts
Example: Original query took > 24 hours to execute

```sql
SELECT COUNT (UNIQUE t1.aliquot_id), t1.u_gsk_aliquot_status
FROM lims.aliquot_user t1, lims.rack_aliquot t2, lims.rack_location t8
WHERE t1.aliquot_id = t2.aliquot_id
AND t2.rack_usage_id = t8.rack_usage_id
AND t1.u_aliquot_sample_type NOT LIKE 'DNA%'
AND t8.removed_on IS NOT NULL
AND t1.aliquot_id NOT IN (
    SELECT UNIQUE t2.aliquot_id
    FROM lims.rack_aliquot t2,
        lims.aliquot_user t1,
        lims.rack_location t8
    WHERE t2.rack_usage_id = t8.rack_usage_id
        AND t1.aliquot_id = t2.aliquot_id
        AND t1.u_aliquot_sample_type NOT LIKE 'DNA%'
        AND t8.removed_on IS NULL)
GROUP BY t1.u_gsk_aliquot_status
```
Tuning with cardinality hint [Example - Original]
The following SQL executed in 1 min 6 seconds.

```sql
SELECT /*+ cardinality( t8 5 ) */ COUNT (UNIQUE t1.aliquot_id),
        t1.u_gsk_aliquot_status
FROM lims.aliquot_user t1,
     lims.rack_aliquot t2,
     lims.rack_location t8
WHERE t1.aliquot_id = t2.aliquot_id
     AND t2.rack_usage_id = t8.rack_usage_id
     AND t1.u_aliquot_sample_type NOT LIKE 'DNA%
     AND t8.removed_on IS NOT NULL
     and t1.aliquot_id IN ( 
         SELECT t1.aliquot_id
             FROM lims.aliquot_user t1
         minus
         SELECT t2.aliquot_id
             FROM lims.rack_aliquot t2
             where t2.aliquot_id > -999
     )
GROUP BY t1.u_gsk_aliquot_status;
```
Tuning with cardinality hint [Example-Tuned]
Find the top 20 problem SQL statements, again, sorted by your favorite performance metric

Look for:
- Long elapsed times per execution
- Queries being executed often
## Tool - Large Tables with Full Scans

- List large tables involved in full table scans
- **Importance:**
  - Full scan of small tables rarely a performance problem
  - Much i/o and thus elapsed time will be spent here
  - Provides evidence for further drill down
- **Interpreting output:**
- Don’t worry if few full scans
- Take a second look at the two 1.16 million scan queries

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>TABLE_TYPE</th>
<th>SIZE_KB</th>
<th>EXECUTIONS</th>
<th>TOTAL_SCANS</th>
<th>STATEMENT_COUNT</th>
<th>REF</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLELEFREQUENCY</td>
<td>TABLE</td>
<td>1886208</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PROJECT</td>
<td>TABLE</td>
<td>38912</td>
<td>1798</td>
<td>5394</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CORE_SAMPLE_TAP_ORDER</td>
<td>TABLE</td>
<td>3072</td>
<td>580223</td>
<td>1160446</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>BACKGROUND</td>
<td>TABLE</td>
<td>2304</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LOCATION</td>
<td>TABLE</td>
<td>1536</td>
<td>12418</td>
<td>12418</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Tool - Top 20 problem sessions

- Find the top 20 problem sessions, sorted by your favorite performance metric
- I usually sort by Physical I/O
- Can see current and last SQL_ID for further drill down

<table>
<thead>
<tr>
<th>SID</th>
<th>SERIAL#</th>
<th>INS...</th>
<th>USERNAME</th>
<th>STATUS</th>
<th>SQL_ID</th>
<th>PREV_SQL_ID</th>
<th>EXECUTIONS</th>
<th>PHYSICAL_IO</th>
<th>MEMORY_SORTS</th>
<th>LOGON_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>210</td>
<td>153</td>
<td>2</td>
<td>CITRIX4ADM</td>
<td>INACTIVE</td>
<td>2u21pa2ut6y2g</td>
<td></td>
<td>14211</td>
<td>47815</td>
<td></td>
<td>339 30-JAN 15:17:28</td>
</tr>
<tr>
<td>221</td>
<td>31</td>
<td>1</td>
<td>CITRIX4ADM</td>
<td>INACTIVE</td>
<td>a58qsr3kfp6h</td>
<td></td>
<td>21647</td>
<td>30013</td>
<td></td>
<td>151 30-JAN 14:47:28</td>
</tr>
<tr>
<td>232</td>
<td>130</td>
<td>1</td>
<td>CITRIX4ADM</td>
<td>INACTIVE</td>
<td>2gzfaj2ukz64q</td>
<td></td>
<td>7617</td>
<td>28125</td>
<td></td>
<td>82 30-JAN 14:59:46</td>
</tr>
<tr>
<td>227</td>
<td>20</td>
<td>1</td>
<td>CITRIX4ADM</td>
<td>INACTIVE</td>
<td>0zhajkoafdkm</td>
<td>2u21pa2ut6y2g</td>
<td>24243</td>
<td>19485</td>
<td></td>
<td>121 30-JAN 14:47:16</td>
</tr>
<tr>
<td>238</td>
<td>26</td>
<td>2</td>
<td>CITRIX4ADM</td>
<td>INACTIVE</td>
<td>0zhajkoafdkm</td>
<td>2u21pa2ut6y2g</td>
<td>29423</td>
<td>10136</td>
<td></td>
<td>5034 30-JAN 15:06:37</td>
</tr>
<tr>
<td>237</td>
<td>91</td>
<td>1</td>
<td>CITRIX4ADM</td>
<td>INACTIVE</td>
<td>0zhajkoafdkm</td>
<td>2u21pa2ut6y2g</td>
<td>21567</td>
<td>8806</td>
<td></td>
<td>51 30-JAN 14:45:17</td>
</tr>
<tr>
<td>225</td>
<td>29</td>
<td>4</td>
<td>CITRIX4ADM</td>
<td>INACTIVE</td>
<td>0zhajkoafdkm</td>
<td>390zcuyak3fz</td>
<td>7911</td>
<td>6599</td>
<td></td>
<td>255 30-JAN 15:53:54</td>
</tr>
<tr>
<td>227</td>
<td>12</td>
<td>2</td>
<td>CITRIX4ADM</td>
<td>INACTIVE</td>
<td>0zhajkoafdkm</td>
<td>2u21pa2ut6y2g</td>
<td>33012</td>
<td>3316</td>
<td></td>
<td>7114 30-JAN 15:07:25</td>
</tr>
<tr>
<td>192</td>
<td>3114</td>
<td>2</td>
<td>PRIVUSER</td>
<td>INACTIVE</td>
<td>bw0fgx3a6x2tt</td>
<td>bw0fgx3a6x2tt</td>
<td>7777</td>
<td>3051</td>
<td></td>
<td>2449 23-FEB 16:06:46</td>
</tr>
<tr>
<td>238</td>
<td>32241</td>
<td>4</td>
<td>CITRIX4ADM</td>
<td>INACTIVE</td>
<td>2gzfaj2ukz64q</td>
<td></td>
<td>7382</td>
<td>2503</td>
<td></td>
<td>95 15-FEB 18:55:17</td>
</tr>
<tr>
<td>199</td>
<td>939</td>
<td>4</td>
<td>CITRIX4ADM</td>
<td>INACTIVE</td>
<td>2u21pa2ut6y2g</td>
<td></td>
<td>5767</td>
<td>2242</td>
<td></td>
<td>27 18-FEB 01:31:50</td>
</tr>
<tr>
<td>214</td>
<td>6</td>
<td>3</td>
<td>CITRIX4ADM</td>
<td>INACTIVE</td>
<td>2gzfaj2ukz64q</td>
<td></td>
<td>7542</td>
<td>581</td>
<td></td>
<td>69 30-JAN 15:35:20</td>
</tr>
<tr>
<td>220</td>
<td>19</td>
<td>3</td>
<td>CITRIX4ADM</td>
<td>INACTIVE</td>
<td>390zcuyak3fz</td>
<td></td>
<td>9361</td>
<td>62</td>
<td></td>
<td>103 30-JAN 15:30:05</td>
</tr>
<tr>
<td>248</td>
<td>1</td>
<td>1</td>
<td>SYS</td>
<td>ACTIVE</td>
<td>531sc6y5xdd41</td>
<td></td>
<td>255</td>
<td>46</td>
<td></td>
<td>94 30-JAN 14:43:30</td>
</tr>
<tr>
<td>240</td>
<td>192</td>
<td>2</td>
<td>CITRIX4ADM</td>
<td>INACTIVE</td>
<td>390zcuyak3fz</td>
<td></td>
<td>2464</td>
<td>83</td>
<td></td>
<td>103 30-JAN 15:30:05</td>
</tr>
</tbody>
</table>

*Note:* The table provides session details including status, SQL_ID, previous SQL_ID, executions, physical I/O, memory sorts, and logon time.
Tool - Hot Segments

- Top 20 segments with buffer busy waits (i.e. I/O waits)
- Reduce I/O ~=> reduced elapsed times ~=> happy users
- PCT total buffer busy waits shows relative importance
- Evidence to drill down further
- Find if the table is involved in full scans
- Find the SQL
- Get the explain plan
- Work on finding a tuning suggestion

<table>
<thead>
<tr>
<th>SEGMENT_NAME</th>
<th>OBJECT_TYPE</th>
<th>OBJECT_BUFF_BUSY_WAITS</th>
<th>PCT_TOT_BFF_BSY_WAITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAMPLEBANK.SAMPLE</td>
<td>TABLE</td>
<td>6545</td>
<td>39.61</td>
</tr>
<tr>
<td>MIRNA.MIRNA_HIT</td>
<td>TABLE</td>
<td>5050</td>
<td>30.56</td>
</tr>
<tr>
<td>SAMPLEBANK.SAMPLE_IDX01</td>
<td>INDEX</td>
<td>717</td>
<td>4.34</td>
</tr>
<tr>
<td>SNOMEDCT.SNOMEDCONCEPT</td>
<td>TABLE</td>
<td>443</td>
<td>2.68</td>
</tr>
<tr>
<td>SAMPLEBANK.SAMPLE_TRACKING</td>
<td>TABLE</td>
<td>34</td>
<td>0.21</td>
</tr>
<tr>
<td>SAMPLEBANK.DONOR_UK01</td>
<td>INDEX</td>
<td>32</td>
<td>0.19</td>
</tr>
<tr>
<td>SAMPLEBANK.DONOR_PK</td>
<td>INDEX</td>
<td>11</td>
<td>0.07</td>
</tr>
<tr>
<td>GENELOGIC_ANALYSIS_TSCORE_PK</td>
<td>INDEX</td>
<td>7</td>
<td>0.04</td>
</tr>
</tbody>
</table>
### Sample.SQL

```sql
@sample  SCHEMANAME,SQL_HASH_VALUE v$session  SCHEMANAME='PRD_BEAN_01'  100000

<table>
<thead>
<tr>
<th>SCHEMANAME</th>
<th>SQL_HASH_VALUE</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRD_BEAN_01</td>
<td>4074119381</td>
<td>298501</td>
</tr>
<tr>
<td>PRD_BEAN_01</td>
<td>603598186</td>
<td>200000</td>
</tr>
<tr>
<td>PRD_BEAN_01</td>
<td>2930944270</td>
<td>188906</td>
</tr>
<tr>
<td>PRD_BEAN_01</td>
<td>2962744231</td>
<td>100016</td>
</tr>
<tr>
<td>PRD_BEAN_01</td>
<td>554578501</td>
<td>100000</td>
</tr>
<tr>
<td>PRD_BEAN_01</td>
<td>3205572202</td>
<td>100000</td>
</tr>
<tr>
<td>PRD_BEAN_01</td>
<td>3302879190</td>
<td>100000</td>
</tr>
<tr>
<td>PRD_BEAN_01</td>
<td>426433858</td>
<td>80640</td>
</tr>
<tr>
<td>PRD_BEAN_01</td>
<td>4193086889</td>
<td>54009</td>
</tr>
<tr>
<td>PRD_BEAN_01</td>
<td>3795553787</td>
<td>28545</td>
</tr>
<tr>
<td>PRD_BEAN_01</td>
<td>3394574506</td>
<td>9204</td>
</tr>
</tbody>
</table>
```