Back to Basics:
DB Time Performance Tuning: Theory and Practice

John Beresniewicz, Graham Wood
Oracle America

ST Partners June, 2011
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Agenda

• Time
  • Database Time
  • Average Active Sessions

• Techniques
  • The DB Time Method

• Tools
  • ADDM
  • EM User Interface
  • Active Reports
Oracle Tuning Methods: A History

- Prehistoric (v5)
  - Debug code

- Dark Ages (v6)
  - Counters/Ratios
  - BSTAT/ESTAT
  - SQL*Trace

- Renaissance (v7/v8)
  - Introduction of Wait Event instrumentation
  - Move from counters to timers
  - STATSPACK

- Modernity (v10)
  - DB Time Tuning – Tuning using fundamental notion of time spent in database
  - Multiple scoping levels
  - Always on, non-intrusive
  - Built into infrastructure: instrumentation, ASH, AWR, ADDM, EM
Why Do We Care About Time?

• Human time is critical to the enterprise

• Systems performance affects business goals
  • Human time + technology resource time

• “Time is money”

• Performance improvement means doing things faster

*Performance is always and only about time*
Database Time and Average Active Sessions
Database Time (DB Time)

- Total time in database calls by foreground sessions
- Includes CPU time, IO time and non-idle wait time
- DB Time <> response time
- Common currency for Oracle performance analysis

*Database time is total time spent by user processes either actively working or actively waiting in a database call.*
A Single Session

Single session with Database Black Box server

- Browse Books
- Read Reviews For One Book
- Add to Cart
- Checkout

= time spent in database

TIME
Database Time (DB Time) =
Total time session spent in all database calls

Active Session =
Session currently spending time in a database call

Average Activity of the Session (% Activity) =
The ratio of time active to total wall clock time

= time spent in database

TIME
Session Details: 1869 (AFOTHERG)

Collected From Target Oct 30, 2007 9:49:37 AM CDT

Detail for Selected 5 Minute Interval

Start Time Oct 30, 2007 9:17:05 AM

Activity (%) SQL ID SQL Command Plan Hash Value Module Action Client ID

100.00 gkm77ywz1na0 SELECT 64730335 oraclealan@ap103fam (TNS V1-V3) AFOTHERG

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

DB Time = Sum of DB Time Over All Sessions

Avg. Active Sessions = Sum of Avg. Activity Over All Sessions

At time $t$ we have 2 active sessions

$\text{Green} = \text{time spent in database}$
Visualizing DB Time

Avg. Active Sessions = \[ \frac{\text{Total Database Time}}{\text{Wall Clock (Elapsed) Time}} \]

Active Sessions over time
Active Sessions by wait class over time
Colored area = amount of DB time
“Click on the big stuff”
Average active sessions

= DB time / elapsed time

- Time-normalized DB time
- Time units in numerator and denominator must synchronize to produce the proper metric
Average Active Sessions

• Full-time equivalent sessions
  • Not whole sessions
  • How many full-time virtual sessions to do the work?

• Comparable
  • Across systems
  • Across time periods
DB Time and System Performance
System Load and DB Time

- More users
  - => More calls
    - => DB time increases

- Larger transactions
  - => Longer calls
    - => DB time increases

*DB time increases as system load increases.*
System Performance and DB Time

- IO performance degrades
  - => IO time increases
    - => DB time increases

- Application performance degrades
  - => Wait time increases
    - => DB time increases

*DB time increases when performance degrades.*
Host Performance and DB Time

- Host is CPU-bound
  - => foregrounds accumulate active run-queue time
    - => wait event times are artificially inflated
      - => DB time increases

Tune for CPU before waits when CPU constrained
CPU Run-queue and DB Time

Recorded wait time

Db file sequential read
Run-queue
On CPU

Recorded wait time

Db file sequential read
Run-queue
On CPU

Actual wait time

Actual wait time

DB time is inflated when host is CPU-bound
CPU or I/O problem?
Instrumentation: Where to find DB Time?

- **V$SYS_TIME_MODEL, V$SESS_TIME_MODEL**
  - STAT_NAME = ‘DB time’

- **V$SYSEMTRIC_HISTORY**
  - “Database Time Per Second”, “CPU Usage Per Sec”
  - 10g units = centi-secs/sec (100xAvg. Active Sessions)
  - 11g new metric “Average Active Sessions”

- **V$SQL**
  - ELAPSED_TIME and CPU_TIME
  - Wait class times: APPLICATION, CONCURRENCY, CLUSTER, USER_IO

- **V$ACTIVE_SESSION_HISTORY**
Active Session History
Active Session History (ASH)

• All ‘Active’ sessions captured every second
  • Foregrounds and backgrounds are sampled
  • Active foregrounds contribute to DB Time

• In-memory: V$ACTIVE_SESSION_HISTORY
  • Sampling interval = 1 second

• On-disk: DBA_HIST ACTIVE_SESS_HISTORY
  • Sampling interval = 10 second

• ASH is a system-wide record of database activity
COUNT(*) = DB Time

GROUP BY ?
ASH Math: \( \text{COUNT(*)} = \text{DB Time} \)

- ASH is a big fact table
  - Each row represents 1-second of active session time

- \text{V$ACTIVE\_SESSION\_HISTORY}\
  - \text{COUNT(*)} = \text{DB time in seconds}

- \text{DBA\_HIST\_ACTIVE\_SESS\_HISTORY}\
  - \text{COUNT(*)} \times 10 = \text{DB time in seconds}
Estimating DB Time with ASH

- **ASH sample counts = DB Time in seconds**
  - Low sample counts are less reliable

- Enables DB Time analysis over many dimensions
  - Sqlid, session id, instance, service, module, action
  - 10gR2
    - Blocking_sid (10gR2)
    - XID
  - 11g
    - Row source
    - Execution ID
    - Operation type
      - Connect
      - Java/SQL/PLSQL
      - parse, bind, execute/fetch, close
Example: DB Time by SQL ID

```sql
select sql_id,
       count(*) DBTime,
       round(count(*)*100/sum(count(*)) over (), 2) pctload
from v$active_session_history
where sample_time > sysdate - 1/24/60
  and session_type <> 'BACKGROUND'
group by sql_id
order by count(*) desc;
```
**Example: DB Time by SQL ID**

```sql
select sql_id,
       count(*) DBTime,
       round(count(*)*100/sum(count(*)) over (), 2) pctload
from v$active_session_history
where sample_time > sysdate - 1/24/60
  and session_type <> 'BACKGROUND'
group by sql_id
order by count(*) desc;
```

<table>
<thead>
<tr>
<th>SQL_ID</th>
<th>DBTIME</th>
<th>PCTLOAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>6bmxrabnwwsxd</td>
<td>60</td>
<td>63.83</td>
</tr>
<tr>
<td>azzsynmz43nrr</td>
<td>8</td>
<td>8.51</td>
</tr>
<tr>
<td>28pb73sbwhmm8</td>
<td>5</td>
<td>5.32</td>
</tr>
<tr>
<td>58psyvgau23s2</td>
<td>3</td>
<td>3.19</td>
</tr>
<tr>
<td>amrq8hk767tuz</td>
<td>2</td>
<td>2.13</td>
</tr>
<tr>
<td>2r5qhb3fb63vm</td>
<td>1</td>
<td>1.06</td>
</tr>
<tr>
<td>f3919usqp5wj2</td>
<td>1</td>
<td>1.06</td>
</tr>
</tbody>
</table>
The calculus of DB time

• The number of active sessions at any time is the rate of change of the DB time function at that time.

\[ \frac{\Delta DBtime}{\Delta t} = ActiveSessions \]

• DB time is the integral of the Active Session function.

\[ DBtime = \int_{t_0}^{t_1} ActiveSessions \]
Avg Active Sessions and DB Time

Active sessions

ASH sample count is value of active sessions function at sample times

DB time is area under curve

Δt = 1 sec
DB Time: ASH vs Time Model

Top Activity

Drag the shaded box to change the time period for the detail section below.
ASH Timing for Nano-Operations

• Some important operations are still too frequent and short-lived for timing
  • No “bind” wait event
• A session-level bit vector is updated in binary fashion before/after an operation
  • Cheaper than timer call
• The bit vector is sampled into ASH
• ASH math allows us to estimate time spent in these un-timed transient operations
Techniques:
The DB Time Method
Where is DB Time used?

- ADDM
- EM Performance page and drill downs
- ASH report
- AWR and AWR compare periods reports
- SYSMETRICS and Server-generated Alerts
The DB Time Method: Short Course

or

just ask ADDM
The DB Time Method: Process

1. Identify performance issue
2. Scope the issue
3. Set goals
4. Data capture (NO OP)
5. Investigate DB time distribution
   • Identify the largest potential for improvement
6. Modify system to tune for largest gain
7. Evaluate against goals
   • Repeat from step 4 if goals not met

*Performance tuning by removing excess DB time*
Investigate DB Time Distribution

• Identify uneven distributions of DB time (skew)
  • => Largest potential improvement within scope

• System scope:
  • Resource limits – is problem outside the DB?

• Application scope:
  • Service, module, action
  • Resource contention (e.g. latches)
  • SQLID, rowsource

• Session scope:
  • Long running SQL
  • Resource contention (e.g. enqueues)
Identify Potential Solutions

- Session contention issues
  - Kill session
  - Fix application

- SQL issues
  - SQL Tuning Advisor => Indexes, SQL profile
  - Re-write SQL

- Design issues
  - Access Advisor => Indexes, physical layout

- System issues
  - Initialization parameters
  - Add resources
Modify System

• Start with the largest DB time issues first
  • Address root causes, not symptoms

• Match solution scope to problem scope
  • Don’t tweak optimizer parameters before tuning SQL

• Proceed iteratively one fix at a time
  • Concurrent fixes should be orthogonal

• Measure and validate results at each successive step

• Stop when goals are met
The DB Time Method: Advantages

- Tunes the one thing that affects users: Time

- Data capture scoping not necessary
  - ‘Always on’ data collection
  - No requirement to reproduce problem

- Works for concurrency problems such as locking

- Combines best of current methods
  - Less intrusive, more inclusive
Method Summary

• DB time is the fundamental performance metric

• The method allows DB time analysis at many scopes
  • Properly scoped problems and solutions are critical to success

• DB time based diagnosis removes value judgments
  • Scientific method, not sorcerer’s magic

• Performance improvement means doing the same work in less DB Time
Tools:

- ADDM
- Enterprise Manager
- Active Reports
Tools for Applying DB Time Method

Two use-cases, one method:

1. Tuning steady-state performance
   - Improve overall workload throughput or response time
   - Best practice: use ADDM

2. Diagnosing transient performance problems
   - Confirm and investigate reported performance issues
   - Best practice: use EM real-time screens
Best Practice: Use ADDM

- Embedded expert system using the DB time method
  - Identifies root causes behind the symptoms

- Variably scoped:
  - Host to instance to SQL and even database block
  - Scoped to database for RAC (new in 11g)

- Findings prioritized by impact on DB time
  - Finding history allows flexible time scoping
  - Directives can filter findings

- Recommendations by benefit (reduction) to DB time
## Automatic Database Diagnostic Monitor (ADDM)

### Database Activity

The icon selected below the graph identifies the ADDM analysis period. Click on a different icon to select a different analysis period.

![Graph showing Active Sessions over time]

### ADDM Performance Analysis

<table>
<thead>
<tr>
<th>Task Name</th>
<th>ADDM:3132078998_1_1978</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Owner</td>
<td>SYS</td>
</tr>
<tr>
<td>Average Active</td>
<td>10.2</td>
</tr>
<tr>
<td>Sessions</td>
<td></td>
</tr>
<tr>
<td>Period Start Time</td>
<td>Apr 4, 2008 4:00:31 AM PDT</td>
</tr>
<tr>
<td>Period Duration</td>
<td>60</td>
</tr>
<tr>
<td>Instance</td>
<td>emtarget_emtarget1</td>
</tr>
</tbody>
</table>

### Impact (%)

<table>
<thead>
<tr>
<th>Impact (%)</th>
<th>Finding</th>
<th>Occurrences (latest 24 hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>47.9</td>
<td>Top SQL by DB Time</td>
<td>24 of 24</td>
</tr>
<tr>
<td>39.5</td>
<td>Top SQL By I/O</td>
<td>0 of 24</td>
</tr>
<tr>
<td>36.3</td>
<td>Top Segments by I/O</td>
<td>1 of 24</td>
</tr>
<tr>
<td>12.9</td>
<td>Commits and Rollbacks</td>
<td>23 of 24</td>
</tr>
<tr>
<td>7.6</td>
<td>I/O Throughput</td>
<td>1 of 24</td>
</tr>
<tr>
<td>2.5</td>
<td>Undersized RCs</td>
<td>0 of 24</td>
</tr>
</tbody>
</table>
Performance Finding Details: Top SQL by DB Time

Finding
- SQL statements consuming significant database time were found.
  - Impact (Active Sessions): 4.03
  - Impact (%): 52.8
  - Period Start Time: Apr 4, 2008 12:00:04 PM PDT
  - Period Duration (minutes): 60.2
  - Filtered: No

Recommendations

Schedule SQL Tuning Advisor

Select All | Select None | Show All Details | Hide All Details

Select Details Category

- Hide SQL Tuning

Action
- Investigate the SQL statement with SQL_ID "66n44vwsmyknr" for possible performance improvements.
  - SQL Text: `select /* serial_guys */ / p_brand, p_type, p_size, ...
  - SQL ID: 66n44vwsmyknr`

Benefit (%)
- 15%

Rationale: SQL statement with SQL_ID "66n44vwsmyknr" was executed 4 times and had an average elapsed time of 1031 seconds.

- Hide SQL Tuning

Action
- Run SQL Tuning Advisor on the SQL statement with SQL_ID "4sc37x190kp".
  - SQL Text: `select /* big_guys */ /* NO_GBY_PUSHDOWN */ / s_name, s_address ...`
  - SQL ID: 4sc37x190kp

Benefit (%)
- 13.3%

- Show SQL Tuning

Benefit (%)
- 10.2%
- 8%
- 6.6%

Findings Path

Expand All | Collapse All
Finding History: Top SQL by DB Time

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

Detail for Selected 3 Hour Interval

<table>
<thead>
<tr>
<th>Show All Details</th>
<th>Hide All Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details</td>
<td>Finding Details</td>
</tr>
<tr>
<td>(show)</td>
<td>ADDM:3132078998_1_1986</td>
</tr>
<tr>
<td>(show)</td>
<td>ADDM:3132078998_1_1987</td>
</tr>
<tr>
<td>(expand)</td>
<td>ADDM:3132078998_1_1988</td>
</tr>
</tbody>
</table>

Action
- Investigate the SQL statement with SQL_ID "2a6s3wn0nu911w" for possible performance improvements.
  SQL Text: select /* big_guys */ /* pq_distribute(supplier none partition) pq_map(supplie...
  SQL ID: 2a6s3wn0nu911w

Action
- Investigate the SQL statement with SQL_ID "1pzgsfa2jr8m8" for possible performance improvements.
  SQL Text: select /* big_guys */ o_year, sum(case when nation='BRAZIL' then volume...
  SQL ID: 1pzgsfa2jr8m8

Action
- Investigate the SQL statement with SQL_ID "dt7umutdm8p67" for possible performance improvements.
  SQL Text: select /* big_guys */ supp_nation, cust_nation, year, ...
  SQL ID: dt7umutdm8p67

Action
- Investigate the SQL statement with SQL_ID "9sqv60uk9hjzw" for possible performance improvements.
  SQL Text: select /* big_guys */ o_orderpriority, count(*) as order_count from ...
  SQL ID: 9sqv60uk9hjzw

Action
- Investigate the SQL statement with SQL_ID "66n4vwsmyknw" for possible performance improvements.
  SQL Text: select /* big_guys */ /* pq_distribute(supplier none partition) pq_map(supplie...
  SQL ID: 66n4vwsmyknw

Action
- Investigate the SQL statement with SQL_ID "1e566657stfsx3" for possible performance improvements.
  SQL Text: select /* big_guys */ /* pq_distribute(supplier none partition) pq_map(supplie...
  SQL ID: 1e566657stfsx3
Best Practice: EM Real-time Interface

- Transient (sub-hour) or immediate time scope
  - Requires interactivity of UI

- ‘Click on the big stuff’
  - Data visualizations display skew directly

- Takes some expertise to separate symptoms from root causes
```sql
SELECT /*+ OPAQUE_TRANSFORM */
"RPTNO", "RPTDATE", "RPTD_BY", "VERSION", "UTILITY_VERSION", "CATEGORY", "STATUS", "SUBJECT", "UPD_BY", "CUSTOMER"
FROM "BG"."RPTHEAD" "H" WHERE "RPTDATE">:1 AND "RPTD_BY"<>'BATCH' AND "CUSTOMER" LIKE '%WPTG%' AND
```

**Details**

Select the plan hash value to see the details below. Plan Hash Value: 301316116

- **Statistics**
- **Activity**
- **Plan**
- **Tuning Information**

**Summary**

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

**Detail for Selected 5 Minute Interval**

Start Time: Apr 5, 2008 10:29:32 AM

<table>
<thead>
<tr>
<th>Activity (%)</th>
<th>SID</th>
<th>User</th>
<th>Program</th>
<th>Service</th>
<th>Plan Hash Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>51.89</td>
<td>2228</td>
<td>MOONEL</td>
<td>oracle@mlnxie01 (TNS V1-V3)</td>
<td>boracle.com</td>
<td>301316116</td>
</tr>
<tr>
<td>48.11</td>
<td>2203</td>
<td>MOONEL</td>
<td>oracle@moconnel-lnx (TNS V1-V3)</td>
<td>boracle.com</td>
<td>301316116</td>
</tr>
</tbody>
</table>
### SQL Details: bbxb6c4kmgmq

**Text**

```sql
SELECT /*+ OPAQUE_TRANSFORM */
"RPTNO", "RPTDATE", "RPTD_BY", "VERSION", "UTILITY_VERSION", "CATEGORY", "STATUS", "SUBJECT", "UPD_BY", "CUSTOMER" 
FROM "BG"."RPTHEAD" "H" WHERE "RPTDATE">:1 AND "RPTD_BY"><'BATCH' AND "CUSTOMER" LIKE '%%WPTG%%' AND 
```

**Details**

Select the plan hash value to see the details below. **Plan Hash Value:** 301316116

**Statistics**

- **Capture Time:** Apr 5, 2008 10:53:15 AM
- **Optimizer Mode:** ALL_ROWS

**Plan**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Object</th>
<th>Object Type</th>
<th>Order</th>
<th>Rows</th>
<th>Size (KB)</th>
<th>Cost</th>
<th>Time (sec)</th>
<th>CPU Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td>12</td>
<td></td>
<td></td>
<td>71,662</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILTER</td>
<td></td>
<td></td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>BG.RPTHEAD</td>
<td>TABLE</td>
<td>9</td>
<td>1</td>
<td>0.172</td>
<td>71,662</td>
<td>557 5,287,109,561</td>
<td>7</td>
</tr>
<tr>
<td>BITMAP CONVERSION TO ROWIDS</td>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BITMAP AND</td>
<td></td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BITMAP CONVERSION FROM ROWIDS</td>
<td></td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORT ORDER BY INDEX RANGE SCAN</td>
<td>BG.I_RPTHEAD_PRODUCT_ID</td>
<td>INDEX</td>
<td>1</td>
<td>1</td>
<td>1,074</td>
<td>9</td>
<td>74,441,376</td>
<td>1</td>
</tr>
<tr>
<td>BITMAP CONVERSION FROM ROWIDS</td>
<td></td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORT ORDER BY INDEX RANGE SCAN</td>
<td>BG.I_RPTDATE</td>
<td>INDEX</td>
<td>4</td>
<td>1</td>
<td>4,205</td>
<td>3</td>
<td>33 311,071,176</td>
<td>4</td>
</tr>
<tr>
<td>INDEX RANGE SCAN</td>
<td>BG.BG_ACCESS_UNIQ</td>
<td>INDEX (UNIQUE)</td>
<td>10</td>
<td>1</td>
<td>0.016</td>
<td></td>
<td>1</td>
<td>22,364</td>
</tr>
</tbody>
</table>
## Grid Control DB Loadmap

### Total Active Load: 573.2 active sessions

<table>
<thead>
<tr>
<th>Database</th>
<th>Instance</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>160 (active sessions)</td>
<td>GMAIL_ADB1.oracle.com</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>ADB1.oracle.com</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>rmsprod_ap6001r</td>
<td>100.0% SQL statements consuming significant database time were found.</td>
</tr>
<tr>
<td></td>
<td>ms.us.oracle.com</td>
<td>48.8% SQL statements were found waiting for row lock waits.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23.9% individual SQL statements responsible for</td>
</tr>
<tr>
<td>56</td>
<td>csemdb.oracle.com</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>repos.us.oracle.com</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>gmldap1.oraclecorp.com</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>pdfdprod</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>relw.us.oracle.com</td>
<td></td>
</tr>
</tbody>
</table>

### Database Incidents

<table>
<thead>
<tr>
<th>Event Message</th>
<th>DB/Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>The archiver hung at time/line number <em>Sun May 3 09:30:46 2009/247865</em></td>
<td>emgc.us.oracle.com_emgc2</td>
</tr>
<tr>
<td>The archiver hung at time/line number <em>Sun May 3 09:37:26 2009/248046</em></td>
<td>emgc.us.oracle.com_emgc2</td>
</tr>
<tr>
<td>The archiver hung at time/line number <em>Mon Apr 20 09:40:25 2009/324702</em></td>
<td>bulkgc.us.oracle.com_bulkgc1</td>
</tr>
<tr>
<td>The archiver hung at time/line number <em>Tue Apr 28 09:50:19 2009/425128</em></td>
<td>bulkgc.us.oracle.com_bulkgc1</td>
</tr>
<tr>
<td>The archiver hung at time/line number <em>Tue Apr 28 09:57:19 2009/425162</em></td>
<td>bulkgc.us.oracle.com_bulkgc1</td>
</tr>
</tbody>
</table>
```
SELECT EnterpriseApplicationEO.ENTERPRISE_APPLICATION_ID, EnterpriseApplicationEO.ENTERPRISE_APPLICATION_ID, EnterpriseApplicationEO.SOURCE_FILE, EnterpriseApplicationEO.VERSION, EnterpriseApplicationEO.TYPE, EnterpriseApplicationEO.VISIBILITY, EnterpriseApplicationEO.DEPENDENCY, EnterpriseApplicationEO.FOR_SETUP, EnterpriseApplicationEO.GROUP_NAME, EnterpriseApplicationEO.INSTALL_DIR, EnterpriseApplicationEO.NAME, EnterpriseApplicationEO.SHORT_NAME, EnterpriseApplicationEO.DEFAULT_URL
FROM ASR_ENTERPRISE_APPLICATIONS EnterpriseApplicationEO
WHERE EnterpriseApplicationEO.ENTERPRISE_APPLICATION_ID = :Bind_EnterpriseApplicationId
ORDER BY NAME
```

Details

Select the plan hash value to see the details below. Plan Hash Value 3504394492

Summary

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

Show Maximum CPU Line

Detail for Selected 5 Minute Interval

Start Time: Jan 5, 2011 3:26:33 PM

<table>
<thead>
<tr>
<th>Activity (%)</th>
<th>SID</th>
<th>QC SID</th>
<th>User</th>
<th>Program</th>
<th>Service</th>
<th>Plan Hash Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Duration</td>
<td>SQL ID</td>
<td>User</td>
<td>Parallel</td>
<td>Database Time</td>
<td>IO Requests</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>----------</td>
<td>--------</td>
<td>----------</td>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>24.2h</td>
<td>02zhf5u0056</td>
<td>FUSION</td>
<td></td>
<td>4.8h</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td>8.2h</td>
<td>a3pmbz622apn</td>
<td>FUSION</td>
<td></td>
<td>6.2h</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td>7.0h</td>
<td>lg1d1un37a2k</td>
<td>FUSION</td>
<td></td>
<td>7.0h</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td>8.7h</td>
<td>lg1d1un37a2k</td>
<td>FUSION</td>
<td></td>
<td>8.7h</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td>10.6h</td>
<td>cttz7vd00h17</td>
<td>SQLDBT</td>
<td></td>
<td>6.1h</td>
<td>24K</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FUSION</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.6h</td>
<td>lg1d1un37a2k</td>
<td>FUSION</td>
<td></td>
<td>41h</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FUSION</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>48.8h</td>
<td>d0lmbh023g5</td>
<td>FUSION</td>
<td></td>
<td>74.8h</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td>48.8h</td>
<td>6d1ahb9600mc</td>
<td>FUSION</td>
<td></td>
<td>74.8h</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td>44.0s</td>
<td>6d1ahb9600mc</td>
<td>FUSION</td>
<td></td>
<td>9.0s</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td>42.0s</td>
<td>0x537y35r2z</td>
<td>FUSION</td>
<td></td>
<td>30.0s</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td>46.0s</td>
<td>6d1ahb9600mc</td>
<td>FUSION</td>
<td></td>
<td>13.0s</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td>44.0s</td>
<td>6d1ahb9600mc</td>
<td>FUSION</td>
<td></td>
<td>9.0s</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td>3.0s</td>
<td>51b3f99h17v</td>
<td>FUSION</td>
<td></td>
<td>3.0s</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td>3.0s</td>
<td>2ov37y3a0j2s</td>
<td>FUSION</td>
<td></td>
<td>3.0s</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td>3.0s</td>
<td>5d0b86q71l2</td>
<td>FUSION</td>
<td></td>
<td>3.0s</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td>3.0s</td>
<td>5d1hsr92b7wpm</td>
<td>FUSION</td>
<td></td>
<td>3.0s</td>
<td>23K</td>
</tr>
<tr>
<td></td>
<td>3.0s</td>
<td>752d2x353w5</td>
<td>FUSION</td>
<td></td>
<td>3.0s</td>
<td>23K</td>
</tr>
</tbody>
</table>

Mon Nov 15, 2010 3:54:43 PM
ORACLE IS THE INFORMATION COMPANY