Performance Fundamentals for Oracle Database 10g and 11g

Graham Wood, Uri Shaft, John Beresniewicz
Oracle Corporation

Sept 2008
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.
Oracle’s Complete Enterprise Software Stack

Built-in & Integrated Manageability

- Leader in the complete enterprise application stack
- Built-in manageability in every tier
- Integrated manageability across the entire stack
Oracle Enterprise Manager
*Increases Business Efficiency*

- Manage applications top-down, from the business perspective by understanding user experiences and business impact of IT issues.

- Manage entire application lifecycle to increase business agility with comprehensive application quality management and compliance solutions.

- Reduce operational costs through intelligent diagnostics and automated IT processes.
Agenda

• Time
  • Database Time and Average Active Sessions

• Techniques
  • The DB Time Method

• Tools
  • ADDM
  • EM Performance User Interface
  • 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
- New lingua franca 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 = time spent in database
**Fundamental Concepts**

**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

---

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

= time spent in database

TIME
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

- User 1
- User 2
- User 3
- User $n$

$\vdash$ = time spent in database
Visualizing DB Time

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

Active Sessions over time
EM Performance Page

- Active Sessions by wait class over time
- Colored area = amount of DB time
- “Click on the big stuff”
DB Time and System Performance
System Load and DB Time

- More users
  - $\Rightarrow$ More calls
    - $\Rightarrow$ DB time increases

- Larger transactions
  - $\Rightarrow$ Longer calls
    - $\Rightarrow$ 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

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

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

- **V$SYSMETRIC_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
Active Sessions and DB Time

Active sessions

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

Δt = 1 sec

DB time is area under curve

ORACLE
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

```
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

```
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>azzsynnz43nrr</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>
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
Techniques:
The DB Time Method
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
  - Proper scoping of problems and solutions is 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

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.

TIP For an explanation of the icons and symbols used in this page, see the Icon Key

ADDM Performance Analysis

Task Name: ADDM:3132078998_1_1978

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

Impact (%)

- **Finding**: Top SQL by D8 Time
  - Occurrences (latest 24 hrs): 24 of 24
- Top SQL by I/O
  - Occurrences (latest 24 hrs): 0 of 24
- Top Segments by I/O
  - Occurrences (latest 24 hrs): 1 of 24
- Commits and Rollbacks
  - Occurrences (latest 24 hrs): 23 of 24
- I/O Throughput
  - Occurrences (latest 24 hrs): 1 of 24
- Unhandled IPCA
  - Occurrences (latest 24 hrs): 0 of 24
Performance Finding Details: Top SQL by DB Time

Finding Impact (Active Sessions) 52.8
Impact (%) 4.03
Period Start Time Apr 4, 2008 12:00:04 PM PDT
Period Duration (minutes) 60.2

Recommendations

Schedule SQL Tuning Advisor

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

Select Details | Category | Benefit (%)

SQL Tuning | Investigate the SQL statement with SQL_ID "66n44vvsymknr" for possible performance improvements.

Action: View Tuning History

Rationale: SQL statement with SQL_ID "66n44vvsymknr" was executed 4 times and had an average elapsed time of 1031 seconds.

Action: Run SQL Tuning Advisor on the SQL statement with SQL_ID "45c37xx190kp".

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

Show All Details | Hide All Details

<table>
<thead>
<tr>
<th>Details</th>
<th>Finding Details</th>
<th>Impact (Active Sessions)</th>
<th>Start Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>► Show</td>
<td>ADDM:312079996_1_1986</td>
<td>4.03</td>
<td>Apr 4, 2008 12:00:04 PM PDT</td>
</tr>
<tr>
<td>► Show</td>
<td>ADDM:312079996_1_1987</td>
<td>4.55</td>
<td>Apr 4, 2008 1:00:18 PM PDT</td>
</tr>
<tr>
<td>► Show</td>
<td>ADDM:312079996_1_1988</td>
<td>6.24</td>
<td>Apr 4, 2008 2:00:45 PM PDT</td>
</tr>
</tbody>
</table>

Action

Investigate the SQL statement with SQL_ID "2a6c34w0nu91l" for possible performance improvements.
SQL Text: select /* big_guys */ * from pses;
SQL ID: 2a6c34w0nu91l

Investigate the SQL statement with SQL_ID "1pqsfsba2yrm8" for possible performance improvements.
SQL Text: select /* big_guys */ o_year, sum(case when nation="BRAZIL" then volume end) from sales where country="MEXICO" and shipper=109;
SQL ID: 1pqsfsba2yrm8

Investigate the SQL statement with SQL_ID "dt7umudm8p67" for possible performance improvements.
SQL Text: select /* big_guys */ l_supp_nation, c_nation, year, ... from lineitem where order_id=1234567890;
SQL ID: dt7umudm8p67

Investigate the SQL statement with SQL_ID "9sqy60u9k9hnw" for possible performance improvements.
SQL Text: select /* big_guys */ o_orderpriority, count(*) as order_count from orders where ...
SQL ID: 9sqy60u9k9hnw

Investigate the SQL statement with SQL_ID "66n14vwsny0kn" for possible performance improvements.
SQL Text: select /* big_guys */ some_column from table1 where condition;
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
```
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

**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>MOCONNEL</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>MOCONNEL</td>
<td>oracle@moconnel-lnx (TNS V1 V3)</td>
<td>boracle.com</td>
<td>301316116</td>
</tr>
</tbody>
</table>
```
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%'
```

### Details

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

<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>71,662</td>
<td></td>
<td></td>
<td>557,628,109.661</td>
<td>71</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></td>
<td></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>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDEX RANGE SCAN</td>
<td>BG.BG_ACCESS_UNQ</td>
<td>INDEX (UNIQUE)</td>
<td>10</td>
<td>1</td>
<td>0.016</td>
<td>3</td>
<td></td>
<td>22,364</td>
</tr>
</tbody>
</table>
```
Selected Additional Enterprise Manager Sessions

• Tuesday Sept 23
  • 11:30 a.m. *Advanced Performance Diagnostics: What the GUI Doesn't Tell You* Moscone West Rm 2003
  • 1:00 p.m. *Demystifying SQL Tuning: Tips and Techniques for SQL Experts* Moscone South Rm 303
  • 1:00 p.m. *Oracle Enterprise Manager Hands-on Lab: - Database Performance Diagnostics and Tuning* Marriott Golden Gate B3

• Wednesday Sept 24
  • 11:30 a.m. *Oracle Enterprise Manager Hands-on Lab: - Database Performance Diagnostics and Tuning* Marriott Golden Gate B3
  • 1:00 p.m. *SQL Tuning Roundtable with the Experts* Moscone West Rm 2001

• Thursday Sept 25
  • 1:30 p.m. *Proactive Performance Monitoring with Baselines and Adaptive Thresholds* Moscone South Rm 303
ORACLE IS THE INFORMATION COMPANY