Oracle Database Performance Management: Theory and Practice

Mughees A. Minhas
VP, Product Management
Oracle Corp.
Safe Harbor Statement

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.
Historical Perspective
Brief History of Oracle Diagnostics Instrumentation

• Prehistoric (V5)
  – Debug Code

• Dark Ages, (V6)
  – Counters/ ratios , BSTAT/ESTAT

• Renaissance (V7)
  – Introduction of wait events ,
  – Moving from counters to timers

• Modernity (V10)
  – DB Time, ASH, AWR, ADDM, EM

• Post-modernity (V11 and V12)
  – ASH Analytics, Real Time ADDM, Real-Time SQL Monitoring , Active Reports, SPA Quick Check, Performance Hub
Key Concepts
What is 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 metric for Oracle performance analysis

*Database time is total time spent by user processes either actively working or actively waiting in a database call.*
System Variables 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.*
Quiz

Question: What happens to DB Time when performance improves?

Answer: DB Time decreases
Quiz

Question: On a four CPU system, for one hour elapsed time, what is the maximum amount of CPU time that can be accumulated?

Answer: Four hours of CPU time maximum.
Quiz

Question: On a four CPU system, for one hour elapsed time, what is the maximum amount of DB Time that can be accumulated?

Hint: How many sessions can wait in a queue?

Answer: Number of ‘sessions’ * elapsed time

The maximum number of sessions that are on CPU or in a non-idle wait is the init.ora parameter ‘sessions’. If ‘sessions’ is 300 then:

300 * 1 hour = 300 hours of DB Time
Active Session =  
Session currently spending time in a database call

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

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

= time spent in database
Multiple Sessions

DB Time = Sum of DB Time Over All Sessions

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

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

At time \( t \) we have 2 active sessions

= time spent in database
Visualizing DB Time

Avg. Active Sessions =

Total Database Time

Wall Clock (Elapsed) Time

Active Sessions over time

TIME

User 1
User 2
User 3
User n
• Active Sessions by wait class over time
• Colored area = Amount of DB time;
• To find root cause of problem: “Click on the Big Stuff” -- CBS Method
CPU Run-queue and DB Time

**DB time is inflated when host is CPU-bound**
Quiz: What is the problem with this database?

- All wait times are inflated when CPU is maxed out
Instrumentation: 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”
  – 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 (ASH)

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

• 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 Session History (ASH)

<table>
<thead>
<tr>
<th>Time</th>
<th>SID</th>
<th>Module</th>
<th>SQL ID</th>
<th>State</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:38:26</td>
<td>213</td>
<td>Book by author</td>
<td>qa324jffritcf</td>
<td>WAITING</td>
<td>db file sequential read</td>
</tr>
<tr>
<td>7:42:35</td>
<td>213</td>
<td>Get review id</td>
<td>aferv5desfzs5</td>
<td>CPU</td>
<td></td>
</tr>
<tr>
<td>7:50:59</td>
<td>213</td>
<td>Add to cart</td>
<td>hk32pekfbdbfr</td>
<td>WAITING</td>
<td>buffer busy wait</td>
</tr>
<tr>
<td>7:52:33</td>
<td>213</td>
<td>One click</td>
<td>abngldf95f4de</td>
<td>WAITING</td>
<td>log file sync</td>
</tr>
</tbody>
</table>
ASH Math

COUNT(*) = DB Time

GROUP BY ?
ASH Math: COUNT(*)=DB Time

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

• V$ACTIVE_SESSION_HISTORY
  – COUNT(*) = DB time in seconds

• DBA_HIST_ACTIVE_SESS_HISTORY
  – COUNT(*) * 10 = DB time in seconds
Example: DB Time by SQL ID

```sql
select sql_id, count(*) as DBTime
, round(count(*)*100/sum(count(*))
over (), 2) as 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>6bmxrabnwwxsxd</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>
User Session Performance Dimensions

- **SQL**
  - SQL ID
  - Plan Hash
  - Operation
  - OpCode

- **PL/SQL**
  - PL/SQL
  - Top Level PL/SQL

- **Resources**
  - Wait Class
  - Wait Event
  - Object
  - Blocking Session

- **Identifiers**
  - Instance
  - Services
  - User Session
  - Parallel Process
  - Program
  - Session Type

- **Attributes**
  - Consumer Group
  - Module
  - Action
  - Client
  - Trans. ID

**Session Attributes**

**Session Identifiers**

**Resources**

**ASH**

**PL/SQL**
Quiz: Why do Perf AAS Graph and Top Activity don’t look identical
Quiz: Why do Perf AAS Graph and Top Activity don’t look identical
Performance Tuning Methodology

Proactive Performance Management

Reactive Performance Management
Proactive Performance Management: SPA Quick Check

- Helps users predict the impact of routine system changes on production SQL workload
- Low overhead capture of SQL workload to SQL Tuning Set (STS)
- Build different SQL trials (experiments) of SQL statements performance by test execution or explain plan
- Day to day use cases integrated with SPA Quick Check, SQL Plan Baselines, & SQL Tuning Advisor to form an end-to-end solution
Proactive Performance Management
Predict the impact of routine system changes on SQL workload response time

• Optimized
  – Optimized for use on prod systems
  – Optimal Trial or Explain Plan mode
  – Disable multi-executions, full DML execute disabled

• Controlled
  – Per SQL time limits
  – Testing scoped to private session
  – Associate with Resource Consumer Group

• Change-Aware
  – Context-aware change testing workflows, such as,
  – Optimizer gather statistics
  – Init.ora parameter changes
Proactive Performance Management

Predict the impact of system changes on SQL workload response time
Performance Tuning Methodology

Reactive Performance Management

• Analyzing transient performance problems
  – ASH Analytics

• Diagnose systemic and significant performance issues
  – ADDM

• In-depth SQL performance analysis
  – Real-Time SQL Monitoring

• Optimizing top SQL’s with sub-optimal plans
  – SQL Tuning Advisor
Reactive Performance Management

Identify performance issues using ASH Analytics

- Graphical ASH report for advanced analysis
- Provides visual filtering for recursive drill-downs
- Select any time period for analysis
- Analyze performance across many dimensions
Reactive Performance Management

• I am a CDBA and asked to investigate transient performance issues reported by one application owner. I need to diagnose the cause of these issues and address them

• AWR report indicates some unusual issues on the system
  – But I don’t get a PDB specific report...
  – What to do next?
Reactive Performance Management

Analyzing transient performance problems using ASH Analytics
ASH Analytics identifies User I/O as the problem

Identify SQL’s that are subject to User I/O
Which Database Performance Diagnostics Tool to Use?

- **Automatic Workload Repository – AWR Reports**
  - Reports about performance and workload data from AWR

- **Active Session History – ASH**
  - Gathers fine-grain data about every active database session every second

- **Automatic Database Diagnostics Monitor - ADDM**
  - Data Analysis and Problem Identification
  - Findings and Advise on how best to resolve bottlenecks

- **Real-time SQL and Database Operations Monitoring**
  - Provides in-depth diagnostics about SQL execution at row source level

Database Performance Hub provides holistic performance management
Database Performance Hub

**Holistic Performance Management**

- Single view of DB performance
  - ADDM, SQL Tuning, Real-Time SQL Monitoring, ASH Analytics
- Switch between ASH analytics, workload view, ADDM findings and SQL monitoring seamlessly
- Supports both real-time & historical mode
- Historical view of SQL Monitoring reports
Performance Hub Report

- New interactive report for analyzing AWR data
- Performance Hub report generated from SQL*Plus
  - @$ORACLE_HOME/rdbms/admin/perfhubrpt.sql
  - OR calling dbms_perf.report_perfhub(....) function
- Single view of DB performance
  - ADDM, SQL Tuning, Real-Time SQL Monitoring, ASH Analytics
- Switch between ASH analytics, workload view, ADDM findings and SQL monitoring seamlessly
- Supports both real-time & historical mode
- Historical view of SQL Monitoring reports
Automatic Performance Diagnostics
Continuous Evolution in Database Performance Management

- Diagnose persistent performance issues
- Uses AWR snapshots
- Regular interval
- Automatic / Manual

- In-depth performance comparison across two periods
- Relies on AWR data
- Manual

- Hung or extremely slow databases
- Uses a normal and diagnostic mode connection
- Manual

- Proactively detect & diagnose transient high-impact problems
- Built inside the DB
- Automatically runs every 3 seconds
Automatic Performance Diagnostics Monitor (ADDM)

- Runs proactively out of the box, reactively when required
- Top-down analysis using Automatic Workload Repository snapshots
- Real-time performance and historic analysis with Automatic Database Diagnostic Monitor
- Resolve performance issues faster with drill-down root-cause analysis
- Classification tree based on Oracle performance tuning expertise
- Performance expert; now a RAC specialist too in Oracle Database 11g
Proactive Performance Management: ADDM
Compare Period ADDM

• Full ADDM analysis across two AWR snapshot periods
• Detects *causes*, measure *effects*, then *correlates* them
  – Causes: workload changes, configuration changes
  – Effects: regressed SQL, reach resource limits (CPU, I/O, memory, interconnect)
• Makes actionable recommendations along with quantified impact
Compare Period ADDM: Method

**STEP 1:**
- Identify what changed
- DB configurations, workload changes

**STEP 2:**
Did the Buffer cache get smaller?
- Why is there 10% new SQL?
- Uses DB Time as basis for measuring performance

**STEP 3:**
Why did Top SQL impact increased by 45%?
- Read I/O are up by 55%, why?
- Correlate performance differences with changes

Did a buffer cache reduction cause a read I/O increase?
Real-Time ADDM—Architecture

- Makes a lightweight connection without acquiring additional locks and resources, bypassing the SQL layer through the agent
- Also attempts to initiate standard JDBC connection
- Data returned by either connection is analyzed by ADDM
Real-Time ADDM Enhancements

**Oracle Database 12c**

- Automatic real-time problem detection and analysis
- Database self-monitors for serious performance issues
- Recognize bad performance trends and trigger analysis:
  - High CPU, I/O spikes, memory, interconnect, hangs, deadlocks
  - Identify a problem before it threatens application performance
- Short duration (5 min spikes) ADDM analysis
  - Actionable advice for critical issues
  - Richer data set available for analysis
- Reports (analysis and data) stored in AWR for historical analysis
  - ADDM, SQL Monitoring reports
# Triggering Conditions

<table>
<thead>
<tr>
<th>#</th>
<th>Rule</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High Load</td>
<td>Average active sessions greater than 3 times the number of CPU cores</td>
</tr>
<tr>
<td>2</td>
<td>I/O bound</td>
<td>Impact on active sessions based on single block read performance</td>
</tr>
<tr>
<td>3</td>
<td>CPU bound</td>
<td>Active sessions greater than 10% of total load and CPU utilization greater than 50%</td>
</tr>
<tr>
<td>4</td>
<td>Over-allocated memory</td>
<td>Allocation over 95% of physical memory</td>
</tr>
<tr>
<td>5</td>
<td>Interconnect bound</td>
<td>Single block interconnect transfer time based</td>
</tr>
<tr>
<td>6</td>
<td>Session Limit</td>
<td>Session limit close to 100%</td>
</tr>
<tr>
<td>7</td>
<td>Process Limit</td>
<td>Process limit close to 100%</td>
</tr>
<tr>
<td>8</td>
<td>Hung Session</td>
<td>Significant number of hung sessions. If this number is greater than 10% of total sessions</td>
</tr>
<tr>
<td>9</td>
<td>Deadlock Detected</td>
<td>Any deadlock detected by hang analyzer</td>
</tr>
</tbody>
</table>
Find: Performance Diagnostics Summary

Topics Covered

• Database Time
• AWR
• Enterprise Manager ASH: Transient / Targeted Performance Analysis
• ADDM: Proactive Performance Management and Diagnosis
• The above toolset helps identify the potential problems and recommends appropriate solutions
• For SQL related issues, ADDM recommends SQL Tuning Advisors...

Next Steps: Tune the identified SQL problems
What makes SQL go bad?

Optimizer Issues

Inaccurate statistics  \rightarrow  Suboptimal Plans

<table>
<thead>
<tr>
<th>Optimizer Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Table Statistics</td>
</tr>
<tr>
<td>• Column Statistics</td>
</tr>
<tr>
<td>• Index Statistics</td>
</tr>
<tr>
<td>• System Statistics</td>
</tr>
<tr>
<td>• Exadata System Statistics</td>
</tr>
</tbody>
</table>
What makes SQL go bad?
Parallelism Issues

Not parallelized (no scaling to large data)

Improperly parallelized (skews, RAC, etc)
What makes SQL go bad?

Missing Access Structures

- **Indexes**: B-tree indexes, B-tree cluster indexes, Hash cluster indexes, Global and local indexes, Reverse key indexes, Bitmap indexes, Function-based indexes, Domain indexes

- **Materialized Views**

- **Materialized View Logs**

- **Partitioned Tables**
What makes SQL go bad?
Poorly written SQL statements

- Convert literal once and not the whole column
- If you know there are no rows in common
What makes SQL go bad?

Resource and Contention Issues

• Hardware Resource Contention
  • CPU
  • memory
  • I/O

• Logical Contention
  • row lock contention
  • block update contention
What makes SQL go bad?
a. Bind Peeking Issue

Problem: Binds will affect optimality in any subsequent uses of the stored plan.
What makes SQL go bad?

b. Literal Usage Issue

```
SELECT * FROM jobs WHERE min_salary > 12000;
SELECT * FROM jobs WHERE min_salary > 15000;
SELECT * FROM jobs WHERE min_salary > 10000;
```

Library Cache

Cursor Sharing
SQL Performance Issues: Broad Categories

Optimizer
- Stale/Missing statistics
- Incomplete statistics
- Improper optimizer configuration
- Upgraded Database: new optimizer
- Changing statistics
- Rapidly changing data

Resources
- Hardware resource crunch
- Contention (row lock contention, block contention)
- Data fragmentation
- Not parallelized (no scaling to large data)
- Improperly parallelized (partially parallelized, skews)
- Rapidly changing data

Applications
- Missing access structures
- Poorly written SQL statements
- Bind-sensitive SQL with bind peeking (Cursor Sharing)
- Literal usage
- Application Issues

Copyright © 2014 Oracle and/or its affiliates. All rights reserved.
Automatic SQL Tuning

- **SQL Tuning Advisor**
  - Gives suggestions on the various problems identified during the diagnosis phase
  - Uses the same CBO but has more time budget to perform comprehensive analysis
  - Identifies alternate execution plans using real-time and historical performance data
  - Recommends parallel profile if it will improve SQL performance significantly (2x or more)
What is SQL Profile?

• Contains auxiliary information collected by the ATO for a SQL statement
  – Customized optimizer settings
• Based on past execution history (e.g., first_rows vs. all_rows)
  – Compensation for missing or stale statistics
  – Compensation for errors in optimizer estimates
• Estimation errors occur due to data skews and correlations, complex filters and joins
• Doesn’t require any change to the SQL text
  – Ideal for Packaged Apps
• Persistent: Works across shutdowns & upgrades
• Transportable across databases (10.2)
• Force Matching for Literals: This setting may be useful for queries that use only literal values
• SQLTUNE_CATEGORY: Enables you to test a profile in a restricted environment before making it available to other sessions. Can be enforced at session level or system level, this defaults to system level
SQL Profiling Flow

SQL Profiling

SQL Tuning Advisor

submit

Optimizer (Tuning Mode)

create

SQL Profile

After ...

Database Users

submit

Optimizer (Normal Mode)

output

Well-Tuned Plan
SQL Tuning in Oracle Database 10g

End-to-End Workflow

Workload → AWR → ADDM → SQL Tuning Candidates

Implement

Evaluate Recommendations

Generate Recommendations

Invoke Advisor

SQL Tuning Advisor

A good end-to-end solution, but manual intervention required
Automatic SQL Tuning in Oracle Database 11g/12c

- Complete automation of SQL tuning
- Automatically captures high-load SQL
- Automatically tunes SQL without changing application by creating SQL Profiles
- Automatically implements (optional) greatly improved SQL plans
- Automatically reports analysis
- Automatically runs during maintenance window
Automatic SQL Tuning in Oracle 11g/12c

1. Workload
2. SQL
3. Choose Candidate SQL
4. Test SQL Profiles
5. Generate Recommendations
6. Implement SQL Profiles
7. View Reports / Control Process
8. It’s Automatic!
Automatic SQL Tuning Result Summary

The Automatic SQL Tuning runs during system maintenance windows as an automated maintenance task, searching for ways to improve the execution plans of high-load SQL statements.

Task Status

Automatic SQL Tuning (SYS.AUTO_SQL_TUNING_TASK) is currently Enabled Configure
Automatic Implementation of SQL Profiles is currently Disabled Configure

Key SQL Profiles: 0
TIP Key SQL Profiles were verified to yield at least a 3X performance improvement and would have been implemented automatically had auto-implementation been enabled.

Summary Time Period

Choose a time period to focus the graphs and statistics below on a specific range of tuning results. Drill down to view focused results or see the results for all SQLs by clicking the "View Report" button.

Time Period: Last 30 Days

Begin Date: Jun 21, 2014 06:32 AM GMT +00:00
End Date: Jul 22, 2014 06:32 AM GMT +00:00

Overall Task Statistics

Executions: 1  Candidates: 39  Distinct SQL Examined: 39

SQL Examined Status

- 0% SQL Examined With Findings (19)
- 47% SQL Examined Without Findings (17)
- 53% SQL Skipped Due To Error (19)

Breakdown by Finding Type

- Number of SQLs: 12
- SQL Profile: 4
- Index: 3
- Statistics: 4
- Restructure SQL: 0
- Alternative Plan: 0

Profile Effect Statistics

Workload Potential DB Time Benefit (seconds per week)
- Implemented (sec) 0  Potential Benefit (sec) 11
Workload Tuning: SQL Access Advisor

Partition Advisor:
- Range Partition
- Interval Partition
- Hash Partition
- List Partition

- Provides List Partition recommendations for 12c Database
- Analyzes already partitioned tables for further optimization

Recommendations
- B*-tree indexes
- Bit-map indexes
- Partitions (11g +)
- MV and MV Logs

Significant reduction in analysis time for very large workloads (DB12c)
Partition Advisor analyses large workloads 40x faster than before.

### SQL Access Advisor: Partition Advisor

- **Analysis Run Times Oracle Database 11.2 versus 12.1**

  ![Graph showing run times对比](chart.png)

  - **DB 11.2**
  - **DB 12.1**

<table>
<thead>
<tr>
<th>SQL Workload Size (in thousands)</th>
<th>Run Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

  - **2X**
  - **20X**
  - **40X**
SQL Access Advisor Features

• Recommends indexes, partitions (hash and interval only) materialized views, and materialized view logs to create and/or drop for faster performance
• Analyzes entire workload and not just independent SQL statements
• Takes into account impact of new access structures on DML operations
• Considers storage, creation and maintenance costs
• Simultaneously considers
  – index solutions
  – materialized view solutions
  – combinations of both
• Optimizes materialized views for
  – maximum query rewrite usage
  – fast refresh
Real-Time SQL Monitoring
Looking Inside SQL Execution

• Automatically monitors long running SQL
• Enabled out-of-the-box with no performance impact
• Monitors each SQL execution
• Exposes monitoring statistics
  – Global execution level
  – Plan operation level
  – Parallel Execution level
• Guides tuning efforts
• Bind values shown
• SQL level metrics
  – CPU, I/O requests, throughput, PGA, temp space
• Graphical explain plan
• I/O statistics for each operation
Real-Time **PL/SQL** Monitoring

- PL/SQL execution no longer a “black box”
  - Answers questions like “why did my DBMS_STATS job take twice as long this time?”
- Shows global (PL/SQL) and SQL level statistics
- Each SQL called by PL/SQL recursively monitored
- Drill-down to slow SQL for diagnosing unexpected PL/SQL behavior
SQL Monitoring Use Cases

Big Plans

Plan Hash Value: 493342630

<table>
<thead>
<tr>
<th>Operation</th>
<th>Name</th>
<th>Estimate</th>
<th>Cost</th>
<th>Time</th>
<th>Actu.</th>
<th>Memor.</th>
<th>Temp.</th>
<th>CPU Activity %</th>
<th>Wait Activity %</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUNT STOPKEY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIEW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORT GROUP BY STOPKEY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HASH JOIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABLE ACCESS FULL</td>
<td>LINGTEM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORT AGGREGATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NESTED LOOPS OUTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HASH JOIN RIGHT OUTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDEX FULL SCAN</td>
<td>L_USER2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NESTED LOOPS OUTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HASH JOIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NESTED LOOPS OUTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HASH JOIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HASH JOIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABLE ACCESS FULL</td>
<td>USER1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HASH JOIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HASH JOIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## SQL Monitoring Use Cases

### Big Plans

<table>
<thead>
<tr>
<th>Operation</th>
<th>User1</th>
<th>User2</th>
<th>User3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sort Aggregate</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Nested Loops Outer</td>
<td>33</td>
<td>730</td>
<td>1</td>
</tr>
<tr>
<td>Hash Join Right Outer</td>
<td>31</td>
<td>675</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Index Full Scan</td>
<td>1_USER2</td>
<td>1</td>
</tr>
<tr>
<td>Nested Loops Outer</td>
<td>31</td>
<td>688</td>
<td>1</td>
</tr>
<tr>
<td>Nested Loops Outer</td>
<td>31</td>
<td>671</td>
<td>1</td>
</tr>
<tr>
<td>Hash Join</td>
<td>31</td>
<td>660</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Table Access Full</td>
<td>USER1</td>
<td>1</td>
</tr>
<tr>
<td>Hash Join</td>
<td>1322</td>
<td>445</td>
<td>1</td>
</tr>
<tr>
<td>Hash Join</td>
<td>1322</td>
<td>227</td>
<td>1</td>
</tr>
<tr>
<td>Merge Join Cart...</td>
<td>10</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Hash Join</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Fixed Table F...</td>
<td>X</td>
<td>KSPMV</td>
</tr>
<tr>
<td></td>
<td>Fixed Table F...</td>
<td>X</td>
<td>KSPPOV</td>
</tr>
<tr>
<td>Buffer Sort</td>
<td>10</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Table Access ...</td>
<td>USER1</td>
<td>1</td>
</tr>
<tr>
<td>Table Access Full</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Union-All</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Filter</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Hash Join</td>
<td>61K</td>
<td>219</td>
<td>1</td>
</tr>
<tr>
<td>Table Access F...</td>
<td>61K</td>
<td>219</td>
<td>1</td>
</tr>
<tr>
<td>Hash Join</td>
<td>61K</td>
<td>219</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Index Full Scan</td>
<td>1_USER2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Table Access ...</td>
<td>USER1</td>
<td>1</td>
</tr>
<tr>
<td>Table Access By...</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Index Unique B...</td>
<td>USER1</td>
<td>1</td>
</tr>
<tr>
<td>Nested Loops</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Index Full Scan</td>
<td>1_USER2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Index Range Scan</td>
<td>1_0B4</td>
<td>1</td>
</tr>
<tr>
<td>Hash Join</td>
<td>14</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Index Full Scan</td>
<td>USER1</td>
<td>1</td>
</tr>
</tbody>
</table>
### SQL Monitoring Use Cases

#### Big Plans

<table>
<thead>
<tr>
<th>Operation</th>
<th>CPU (%)</th>
<th>I/O (%)</th>
<th>CPU(M)</th>
<th>I/O(M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash Join</td>
<td>1505</td>
<td>4452</td>
<td>1535</td>
<td>645k</td>
</tr>
<tr>
<td>Table Access Full</td>
<td>71</td>
<td>2</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>Hash Join</td>
<td>1505</td>
<td>4452</td>
<td>1535</td>
<td>645k</td>
</tr>
<tr>
<td>Merge Join Cart...</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Hash Join</td>
<td>11</td>
<td>1</td>
<td>11</td>
<td>198k</td>
</tr>
<tr>
<td>Fixed Table F...</td>
<td>100</td>
<td>6</td>
<td>100</td>
<td>1944</td>
</tr>
<tr>
<td>Buffer Sort</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>Table Access...</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>Table Access Full</td>
<td>1323</td>
<td>226</td>
<td>1329</td>
<td></td>
</tr>
<tr>
<td>View</td>
<td>611</td>
<td>207</td>
<td>611</td>
<td></td>
</tr>
<tr>
<td>Union All</td>
<td>160</td>
<td></td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>Filter</td>
<td></td>
<td></td>
<td>60k</td>
<td></td>
</tr>
<tr>
<td>Hash Join</td>
<td>611</td>
<td>214</td>
<td>611</td>
<td>630k</td>
</tr>
<tr>
<td>Table Access F...</td>
<td>71</td>
<td>2</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>Hash Join</td>
<td>611</td>
<td>214</td>
<td>611</td>
<td>630k</td>
</tr>
<tr>
<td>Index Full Scan</td>
<td>14</td>
<td>1</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Table Access...</td>
<td>611</td>
<td>207</td>
<td>611</td>
<td></td>
</tr>
<tr>
<td>Table Access SV...</td>
<td>14</td>
<td>1</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Index Unique S...</td>
<td>1</td>
<td>2077</td>
<td>1</td>
<td>2077</td>
</tr>
<tr>
<td>Nested Loops</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index Full Scan</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index Range Scan</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hash Join</td>
<td>14</td>
<td>1</td>
<td>14</td>
<td>180k</td>
</tr>
<tr>
<td>Index Full Scan</td>
<td>1</td>
<td>1</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Table Access Full</td>
<td>71</td>
<td>2</td>
<td>72</td>
<td>72</td>
</tr>
<tr>
<td>Index Range Scan</td>
<td>1</td>
<td>1</td>
<td>1776</td>
<td>93</td>
</tr>
<tr>
<td>Index Range Scan</td>
<td>1</td>
<td>1</td>
<td>1776</td>
<td>1067</td>
</tr>
<tr>
<td>Table Access Cluster</td>
<td>1</td>
<td>1</td>
<td>1776</td>
<td>1107</td>
</tr>
<tr>
<td>Index Unique Scan</td>
<td>1</td>
<td>1</td>
<td>1776</td>
<td>1107</td>
</tr>
<tr>
<td>Table Access Full</td>
<td>604</td>
<td>248</td>
<td>857k</td>
<td>0.85</td>
</tr>
</tbody>
</table>

---

Copyright © 2014 Oracle and/or its affiliates. All rights reserved.
SQL Monitoring Use Cases

Expensive Hash Join
SQL Monitoring Use Cases

Poor Indexing

Plan Hash Value: 967776149

<table>
<thead>
<tr>
<th>Operation</th>
<th>Name</th>
<th>Estimated</th>
<th>Cost</th>
<th>Timeline(462s)</th>
<th>Ex...</th>
<th>Actual R...</th>
<th>Hm...</th>
<th>Tm...</th>
<th>CPU Activity %</th>
<th>Wait Activity %</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT STATEMENT</td>
<td>HASH GROUP BY</td>
<td>1</td>
<td>69L</td>
<td>1</td>
<td>78</td>
<td>1200%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VIEW</td>
<td>1</td>
<td>66L</td>
<td>1</td>
<td>78</td>
<td>1200%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HASH GROUP BY</td>
<td>1</td>
<td>64L</td>
<td>1</td>
<td>78</td>
<td>1198%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>VIEW</td>
<td>1</td>
<td>65L</td>
<td>1</td>
<td>78</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WINDOW HOSORT</td>
<td>1</td>
<td>65L</td>
<td>1</td>
<td>78</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SORT GROUP BY</td>
<td>1</td>
<td>65L</td>
<td>1</td>
<td>78</td>
<td>0192</td>
<td>9.63</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NESTED LOOPS</td>
<td>1</td>
<td>65L</td>
<td>1</td>
<td>66K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NESTED LOOPS</td>
<td>1</td>
<td>65L</td>
<td>1</td>
<td>60K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NESTED LOOPS</td>
<td>1</td>
<td>65L</td>
<td>1</td>
<td>60K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NESTED LOOPS</td>
<td>1</td>
<td>63L</td>
<td>1</td>
<td>34K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>HASH JOIN</td>
<td>1</td>
<td>63L</td>
<td>1</td>
<td>34K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TABLE ACCESS BY INDEX ROWID</td>
<td>1</td>
<td>8</td>
<td>204K</td>
<td>60K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDEX RANGE SCAN</td>
<td>LU_ELEMENTGROUP_REL</td>
<td>47</td>
<td>6</td>
<td>204K</td>
<td>882K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Copyright © 2014 Oracle and/or its affiliates. All rights reserved.
## SQL Monitoring Use Cases

**FORCE PARALLEL QUERY PARALLEL 4**

### Monitored SQL Execution Detail

#### Overview
- **SQL ID**: abd0jhwry9ja
- **Execution Started**: Thu Feb 14 2008 02:17:11 PM
- **Last Refresh Time**: Thu Feb 14 2008 02:20:18 PM
- **Execution ID**: 15777216
- **Session**: 70
- **Fetch Calls**: 0

#### Time
- **Duration**: 3.1m
- **Database Time**: 6.2m
- **PL/SQL & Java**: 0.0s

#### IO & Wait Statistics
- **IO Count**: 7,093
- **Buffer Gets**: 7,162
- **Wait Activity %**: 100

### Operation Details

<table>
<thead>
<tr>
<th>Operation</th>
<th>Name</th>
<th>Estimate Cost</th>
<th>Timeline (18s)</th>
<th>Exe Time</th>
<th>Actual Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT STATEMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOAD AS SELECT</td>
<td></td>
<td>62K</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>COUNT STOPKEY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PX COORDINATOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PX SEND QC (RANDOM)</td>
<td>TQ10000</td>
<td>48M</td>
<td></td>
<td>4</td>
<td>20M</td>
</tr>
<tr>
<td>COUNT STOPKEY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PX BLOCK ITERATOR</td>
<td></td>
<td>48M</td>
<td></td>
<td>4</td>
<td>20M</td>
</tr>
<tr>
<td>TABLE ACCESS FULL</td>
<td>LINEITEM</td>
<td>48M</td>
<td></td>
<td>20</td>
<td>20M</td>
</tr>
</tbody>
</table>
Real-Time Database Monitoring
What’s Really Happening in the Database

Challenge:

• SQL & PL/SQL Monitoring only monitors a single execution
• How does a DBA monitor a composite operation such as a batch job?

Solution:

• Real-Time Database Operations Monitoring
• Benefit: Allows DBAs to look into complex composite DB operations for analysis and tuning purposes
Monitor Complex Database Operations

- **Oracle Database 11g**: Support for simple DB operations
  - SQL statements (e.g., SQL for DSS, batch/report SQL, runaway SQL)
  - PL/SQL procedures/functions

- **Oracle Database 12c**: Support for composite operations
  - Session(s) activity between 2 points of time defined by application code / DBA
  - For example; SQL*Plus script, batch job, or ETL processing
  - At most one DBOP per DB session
SPA Overview

- Helps users predict the impact of system changes on SQL workload response time
- Low overhead capture of SQL workload to SQL Tuning Set (STS) on production system
- Build different SQL trials (experiments) of SQL statements performance by test execution
- Analyzes performance differences
- Offers fine-grained performance analysis on individual SQL
- Integrated with STS, SQL Plan Baselines, & SQL Tuning Advisor to form an end-to-end solution
When to use SPA?

• SPA can be used with Oracle Database Releases 10.2, 11g and 12c
  – Optimizer statistics refresh
  – Validate and implement of tuning recommendations from ATO
  – Database parameter changes, schema changes (e.g., add/drop indexes)
  – I/O subsystem changes (e.g., ASM, Database Machine)
• SPA handles trials in a manner that does not change database data
  – Hence can be used for testing in production/standby environments
• For supported releases, test database upgrades and patch-set releases*

*MOS Note: 560977.1

SPA Provides Broad Testing Coverage

• Across many releases of Oracle and for upgrades
• On test, standby, and production environments
• Extended to home-grown scripts, third-party testing tools, etc.
• Supports most applications - EBS, SAP, Siebel, home-grown, etc.
SPA: Enterprise Manager Interface

- Rich GUI through Enterprise Manager
- DBMS_SQLPA package PL/SQL API
# SPA Report: Regressed SQL Statements

## Regressed SQL Statements

<table>
<thead>
<tr>
<th>SQL ID</th>
<th>SQL Trial 1</th>
<th>SQL Trial 2</th>
<th>Net Impact on SQL (%)</th>
<th>New Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>3Fv28gfUy0aq</td>
<td>25,504</td>
<td>29,573</td>
<td>-11.580</td>
<td>Y</td>
</tr>
<tr>
<td>czzzuh8fzg96</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SQL Details: czzzuh8fzg96

```
SELECT /* my_query_21 */ '/*+ ORDERED INDEX(t1) USE_HASH(t1) /*+ 'B' */ t2.take_02 take_02, 'B' */ t2.take_15 take_15, 'B' */ t2.take_08 take_08, 'r' */ t3.record_nr price_eur_id...
```

### Single Execution Statistics

<table>
<thead>
<tr>
<th>Execution Statistic Name</th>
<th>SQL Trial 1</th>
<th>SQL Trial 2</th>
<th>Net Impact on SQL (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Time (sec)</td>
<td>0.220</td>
<td>0.001</td>
<td>0.018</td>
</tr>
<tr>
<td>User I/O Time (sec)</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>Buffer Gets</td>
<td>-0.030</td>
<td>1.419</td>
<td>1.981</td>
</tr>
<tr>
<td>Optimizer Cost</td>
<td>-3.340</td>
<td>9,839</td>
<td>9,970</td>
</tr>
</tbody>
</table>

### Plan Comparison

**SQL_TRIAL_1353942463446**

Plan Hash Value: 1165613724

### Execution Statistics Collected

- **Execution Statistic Name**: CPU Time (sec)
  - **Net Impact on SQL (%)**: -0.018
- **Execution Statistic Name**: User I/O Time (sec)
  - **Net Impact on SQL (%)**: 0.000
- **Execution Statistic Name**: Buffer Gets
  - **Net Impact on SQL (%)**: 1.981
- **Execution Statistic Name**: Optimizer Cost
  - **Net Impact on SQL (%)**: 9.970

### Operation Details

<table>
<thead>
<tr>
<th>Operation</th>
<th>Line ID</th>
<th>Object</th>
<th>Rows</th>
<th>Cost</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT STATEMENT</td>
<td>0</td>
<td></td>
<td>9,830</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HASH GROUP BY</td>
<td>1</td>
<td>1</td>
<td>9,830</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MERGE JOIN</td>
<td>2</td>
<td>1</td>
<td>9,829</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SORT JOIN</td>
<td>3</td>
<td>8</td>
<td>9,795</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HASH JOIN</td>
<td>4</td>
<td>8</td>
<td>9,794</td>
<td>&quot;T1.PERIOD_CODE&quot;=&quot;T4&quot;, &quot;FLYER...</td>
<td></td>
</tr>
<tr>
<td>INDEX RANGE SCAN</td>
<td>5</td>
<td>APPS.ID$$_080F0004</td>
<td>8,792</td>
<td>2</td>
<td>&quot;T4.EXPORT_LIC_NR&quot;=&quot;19650&quot;</td>
</tr>
<tr>
<td>HASH JOIN</td>
<td>6</td>
<td>14,210</td>
<td>9,792</td>
<td>&quot;T1.SKU_NR&quot;=&quot;T2&quot;, &quot;SKU_NR&quot;=&quot;AN...</td>
<td></td>
</tr>
</tbody>
</table>
Find → Fix → Validate

STEP 1 - Find Issues

STEP 2 - Fix Issues

STEP 3 - Validate Fix
Summary
Conclusion: Find-Fix-Validate  ➔  Improved Quality Of Service

Manage and Automate
• **MORE** databases
• **MORE** risk free administration on production
• **MORE** effort less testing on production
• **MORE** applications support: OLTP, DW, COTS
• **MORE** users, larger databases
• **MORE** mission-critical applications

Get
• **MORE** uptime
• **MORE** Quality Of Service
• **MORE** sleep at night!
• **MORE** weekends off!

Become
• **MORE** proactive and strategic
• **MORE** important and valuable!
Hardware and Software
Engineered to Work Together