Mughees A. Minhas
Director of Product Management
Database and Systems Management

Take the Guesswork out of SQL Performance with SPA
Outline

• Motivation
• Overview
• Use Cases
• Real-world Deployments
• Conclusion
Motivation

• Businesses want systems that are performant and meet SLA’s
• SQL performance regressions are #1 cause of poor system performance
• Solution for proactively detecting all SQL regressions resulting from changes not available
• DBA’s use ineffective and time-consuming manual scripts to identify problems

SPA identifies all changes in SQL performance before end-users can be impacted
SQL Performance Analyzer (SPA)

- Test impact of change on SQL query performance

- Capture SQL workload in production including statistics & bind variables

- Re-execute SQL queries in test environment

- Automatically remediate regressed SQL
  - Integrated with SQL Plan Baselines and SQL Tuning Advisor to provide end-to-end solution
SPA: Supported Changes

Changes Supported

- Database Upgrades, Patches
- Schema, Parameters
- Optimizer statistics refresh
- Tuning actions
- I/O subsystem changes
- Etc.

Recording of External Client Requests
SQL Performance Analyzer
Upgrading from Oracle Database 9.2 to 10.2

Capture from 9.2 or higher

Clients
Mid-Tier
Storage

Test in 10.2 or higher

Copy to Test
Remote Execute SQL
Compare Performance

Capture (SQL*Trace)

11.1
Send SQL to execute

9.2

10.2
Send execution stats
Step 1: Capture SQL Workload

- Capture workload using
  - SQL*Trace (Oracle 9i or 10.1)
  - SQL Tuning Set (Oracle 10.2 or 11.1)
- Covert SQL*Trace workload into STS (SQL Tuning Set)
- Incremental capture used to populate STS from cursor cache over a time period
- STS includes:
  - SQL Text
  - Bind variables
  - Execution plans
  - Execution statistics
- STS’s filtering and ranking capabilities filters out undesirable SQL
Step 2: Move SQL Workload to Test System

- Copy SQL*Trace file(s) to test system and convert into STS
- Copy STS to staging table ("pack")
- Transport staging table to test system (datapump, db link, etc.)
- Copy STS from staging table ("unpack")
Step 3: Execute SQL Before Making Change

- Establishes SQL workload performance baseline
- SQL execution plan and statistics captured
- SQL executed serially (no concurrency)
- Each SQL executed only once
- DDL/DML skipped
- Option to do Explain Plan only analysis
- SQL*Trace capture does not require this step as the trace file has the necessary execution stats
Step 4: Execute SQL After Making Change

- Manually implement the planned change
  - Database upgrade, patches
  - Optimizer statistics refresh
  - Schema changes
  - Database parameter changes
  - Tuning actions, e.g., SQL Profile creation

- Re-execute SQL after change
  - Gathers new SQL execution plans and statistics
Step 5: Compare & Analyze Performance

- Compare performance using different metrics, e.g.,
  - Elapsed Time
  - CPU Time
  - Optimizer Cost
  - Buffer Gets

- SPA Report shows impact of change for each SQL
  - Improved SQL
  - Regressed SQL
  - Unchanged SQL

- Fix regressed SQL using SQL Tuning Advisor or SQL Plan Baselines
SPA Report

SQL Performance Analyzer Task Result: SYS.UPGRADE_10G11G

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Task Owner</th>
<th>Task Description</th>
<th>SQL Tuning Set Name</th>
<th>STS Owner</th>
<th>Total SQL Statements</th>
<th>SQL Statements With Errors</th>
<th>Replay Trial 1</th>
<th>Replay Trial 2</th>
<th>Comparison Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPGRADE_10G11G</td>
<td>SYS</td>
<td>test upgrade to 11g</td>
<td>OOW_54G</td>
<td>SYS</td>
<td>54</td>
<td>0</td>
<td>10g_data</td>
<td>11g_data</td>
<td>Buffer Gets</td>
</tr>
</tbody>
</table>

**Global Statistics**

- **Projected Workload Buffer Gets**
  - **Buffer Gets**: 0 - 80,000,000
  - **Improvement Impact**: 24%
  - **Regression Impact**: -2%
  - **Overall Impact**: 22%

- **SQL Statement Count**
  - **Count**: 2
  - **Change in Buffer Gets**
    - Improved: 75
    - Regressed: 25
    - Unchanged: 0

**Recommendations**

Oracle offers two options to fix regressed SQL resulting from plan changes:

1. Use the better execution plan from SQL Trial 1 by creating SQL Plan Baselines.
2. Explore alternate execution plans using SQL Tuning Advisor.

**Top 10 SQL Statements Based on Impact on Workload**

<table>
<thead>
<tr>
<th>SQL ID</th>
<th>Net Impact on Workload (%)</th>
<th>Buffer Gets</th>
<th>Net Impact on SQL (%)</th>
<th>% of Workload</th>
<th>Plan Changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>g4df44ak4rus2</td>
<td>12.000</td>
<td>12,973,052.000</td>
<td>33.550</td>
<td>35.780</td>
<td>30.670 Y</td>
</tr>
<tr>
<td>gfacm5r3r39rj</td>
<td>11.990</td>
<td>6,990,541.000</td>
<td>97.420</td>
<td>12.310</td>
<td>0.410 Y</td>
</tr>
<tr>
<td>2ny75l6at2vd9</td>
<td>-0.820</td>
<td>12,740,524.000</td>
<td>-3.610</td>
<td>22.850</td>
<td>30.530 Y</td>
</tr>
<tr>
<td>c2fbo3q5p7d4p</td>
<td>-0.750</td>
<td>12,740,524.000</td>
<td>-3.340</td>
<td>22.440</td>
<td>29.910 Y</td>
</tr>
<tr>
<td>2wtgxbjg6u2by</td>
<td>0.050</td>
<td>244,678.000</td>
<td>10.690</td>
<td>0.430</td>
<td>0.500 Y</td>
</tr>
</tbody>
</table>
# SPA Report

## Regressed SQL Statements

<table>
<thead>
<tr>
<th>SQL ID</th>
<th>Net Impact on Workload (%)</th>
<th>Buffer Gets</th>
<th>Net Impact on SQL (%)</th>
<th>% of Workload</th>
<th>Plan Changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>2ny751aat2vd9</td>
<td>-0.820</td>
<td>12,973,952,000</td>
<td>13,440,825,000</td>
<td>-3.610</td>
<td>22.850</td>
</tr>
<tr>
<td>c2fb0uc3yd4p</td>
<td>-0.750</td>
<td>12,973,952,000</td>
<td>13,440,825,000</td>
<td>-3.610</td>
<td>30.530 Y</td>
</tr>
</tbody>
</table>

### SQL Details: 2ny751aat2vd9

**Parsing Schema**: DWH_TEST  
**Execution Frequency**: 1

#### Single Execution Statistics

<table>
<thead>
<tr>
<th>Execution Statistic Name</th>
<th>Net Impact on Workload (%)</th>
<th>Execution Statistic Collected</th>
<th>Net Impact on SQL (%)</th>
<th>% of Workload</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elapsed Time</td>
<td>-4.340</td>
<td>70.518</td>
<td>89.593</td>
<td>-27.050</td>
</tr>
<tr>
<td>Parse Time</td>
<td>-13.830</td>
<td>0.207</td>
<td>0.312</td>
<td>-50.720</td>
</tr>
<tr>
<td>CPU Time</td>
<td>-5.700</td>
<td>64.704</td>
<td>85.188</td>
<td>-31.660</td>
</tr>
<tr>
<td>Buffer Gets</td>
<td>-0.820</td>
<td>12,973,952,000</td>
<td>13,440,825,000</td>
<td>-3.610</td>
</tr>
<tr>
<td>Optimizer Cost</td>
<td>0.170</td>
<td>982.000</td>
<td>638.000</td>
<td>32.990</td>
</tr>
<tr>
<td>Disk Reads</td>
<td>10.800</td>
<td>7,011.000</td>
<td>5.000</td>
<td>99.930</td>
</tr>
<tr>
<td>Direct Writes</td>
<td>10.950</td>
<td>6,968.000</td>
<td>0.000</td>
<td>100.000</td>
</tr>
<tr>
<td>Rows Processed</td>
<td>0.000</td>
<td>0.111.000</td>
<td>0.111.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

### Plan Comparison

**10g_data**  
**Plan Hash Value**: 3935030222

---

**Operation**

- SELECT STATEMENT
- HASH
- TABLE ACCESS
- NESTED LOOPS
- MERGE JOIN
- SORT
- TABLE ACCESS
- NESTED LOOPS
Use Cases
Database Upgrade: 9.2/10.1* → 10.2

**Scenario 1:**

I have heard premier DB support for 9.2 has ended, I want to upgrade 10.2 database release. How can I use 11g SPA functionality to accomplish the upgrade?

**Goal:**

Assess impact of upgrade on SQL workload performance on a test system using SPA so that there are no surprises after upgrade.

* For 10.1, the solution is similar to 9.2
Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

System Setup

No patches needed

11g Test System: No application schema/data, stores SPA results
11.1.0.7 or 11.1.0.6 + patch

Metalink Note: 560977.1
Scenario 1: Database Upgrade: 9.2/10.1 → 10.2
SPA Enhancements

1. Consume 9i SQL Trace files and generate STS

2. Remote Test Execute Workload over DB Link
Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

1. Capture SQL workload using SQL*Trace

2. Generate STS from trace file

3. Generate Pre-Change Trial

4. Send SQL to 10.2 DB to execute

5. Compare performance and generate SPA report

Production Database

Test Database

Prod DB (9.2/10.1)

Test DB (10.2)

11g SPA System

Collect execution stats

Send SQL for Remote Execution

Move trace file to 11g SPA system
Scenario 1: Database Upgrade: 9.2/10.1 ➞ 10.2

1. Capture SQL workload using SQL*Trace

Move trace file to 11g SPA system

11g SPA System

Send SQL for Remote Execution

Collect execution stats

2. Generate STS from trace file

3. Send SQL to 10.2 DB to execute

4. Compare performance and generate SPA report
Scenario 1: Database Upgrade: 9.2/10.1 → 10.2
Step 1: Capture SQL workload using SQL Trace

• Identify all interesting workloads such as month-end, daily peak, etc.
• Capture SQL trace for the workload, few sessions at a time
• Use dbms_support/dbms_monitor package, these support
  • bind value capture
  • tracing other running sessions
• SQL trace considerations
  • time_statistics=true: Important for performance data
  • user_dump_dest
  • max_dump_file_size
  • trace_file_identifier
  • Performance overhead: 10-15% for traced sessions
Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

Step 1: Capture SQL workload using SQL Trace (contd.)

- SQL Trace files only have object identifiers
- Create mapping table to map object identifiers in trace files to schema names

<table>
<thead>
<tr>
<th>Id</th>
<th>Owner</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>SCOTT</td>
<td>EMP</td>
</tr>
<tr>
<td>124</td>
<td>SCOTT</td>
<td>DEPT</td>
</tr>
</tbody>
</table>

* SQL in Note pages/OTN
Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

Step 2: Transport Workload and Create STS

- Transport SQL trace files, mapping table using ftp/expdp/impdp.., etc.
- Create STS from trace files using dbms_sqltune API (SQL in Notes page/OTN)
  - Specify directory object containing trace files, mapping table, STS name as input
**Scenario 1: Database Upgrade: 9.2/10.1 → 10.2**

Step 3a: Create SPA Task

- Create SPA task on 11g SPA System

---

**Guided Workflow**

The following guided workflow contains the sequence of steps necessary to execute a successful two-trial SQL Performance Analyzer test.

*Note:* Be sure that the Trial environment matches the tests you want to conduct.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Executed</th>
<th>Status</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create SQL Performance Analyzer Task based on SQL Tuning Set</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>2</td>
<td>Replay SQL Tuning Set in Initial Environment</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>3</td>
<td>Replay SQL Tuning Set in Changed Environment</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>4</td>
<td>Compare Step 2 and Step 3</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>5</td>
<td>View Trial Comparison Report</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

Create SPA Task

Create SQL Performance Analyzer Task

The SQL Performance Analyzer Task is a container for the execution of trial experiments designed to test the effects of changes in execution environment on the SQL performance of an STS.

* Name: UPG_TEST

Owner: SYS

Description: TIP Use the description to characterize the intended SQL Performance Analyzer investigations.

SQL Tuning Set

The SQL Tuning Set is the basis for SQL Performance Analyzer Task experiments. The STS should represent a coherent set of SQL for the changes being investigated (e.g. full workload for an upgrade test).

* Name: HR_WORKLOAD

TIP: You can create a new STS here: Link to STS Creation Wizard
Scenario 1: Database Upgrade: 9.2/10.1 → 10.2
Step 3b: Establish “Before Change” Trial

- Trial Creation Method: Select “Build From SQL Tuning Set” option to use plans and statistics from 9i

| Database Instance: v11gk > Advisor Central > SQL Performance Analyzer > Guided Workflow: |
| Create SQL Trial |

SQL Trials capture execution performance of the SQL Tuning Set under a given optimizer environment.

SQL Performance Analyzer Task: SYS.UPGRADE_TEST
.SQL Tuning Set: APPS.HR_WORKLOAD

* SQL Trial Name: BEFORE_TRIAL

SQL Trial Description: 9i TRIAL

Creation Method: Build From SQL Tuning Set

- Execute SQLs Locally
- Execute SQLs Remotely
- Generate Plans Locally
- Generate Plans Remotely

Schedule:
- Time Zone: America/Los_Angeles
- Immediately
Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

Step 4: Establish “After Change” Trial

Guided Workflow: SYS.Upgrade_Test

Create SQL Trial

SQL Trials capture execution performance of the SQL Tuning Set under a given optimizer environment.

SQL Performance Analyzer Task SYS.T2

SQL Tuning Set **APPS.HR_WORKLOAD**

- **SQL Trial Name**: AFTER_TRIAL_UPGRADE_10204
- **SQL Trial Description**: Trial after upgrade to 10204
- **Creation Method**: Execute SQLs Remotely
- **Per-SQL Time Limit**: Unlimited

**TIP** Time limit is on elapsed time of test execution of SQL.

- **Database Link**: DBLINK_10204

**TIP** Provide a PUBLIC database link connecting to a remote user with privileges to execute the Tuning Set SQL.
Scenario 1: Database Upgrade: 9.2/10.1 → 10.2

1. Capture SQL workload using SQL*Trace

Move trace file to 11g SPA system

2. Generate STS from trace file

3. Send SQL to 10.2 DB to execute

4. Compare performance and generate SPA report
Scenario 1: Database Upgrade: 9.2/10.1 → 10.2
Step 5: Compare and Generate Report

• Compare Pre-Change and After-Change Trials based on a performance metric
  • Oracle recommends using CPU_TIME and BUFFER_GETS metrics
  • Use multiple metrics that provide repeatable and comprehensive statistics
Fixing Regressed SQL

- Systematic problems
  - Check un-analyzed tables, PGA memory, statistics collection, system statistics
  - Refer “Upgrading from Oracle 9i to 10g: What to expect from the Optimizer” on OTN

- For statements suffering from isolated problems use one of the following fixes
  - SQL Profiles: Implement Profiles recommended by SQL Tuning Advisor (STA)
  - Stored Outlines**: If no profile was recommended by STA, then capture Stored Outlines in 9i for the targeted SQL statements. Import stored outline into 10g.
Database Upgrade: 10.2.0.x → 10.2.0.y

Scenario 2:

One of the database I’m managing is on 10.2.0.2. How can I use 11g SPA functionality to accomplish an 10.2.0.4 patchset upgrade?

Goal:

Assess impact of upgrade on SQL workload performance using SPA so that there are no surprises after upgrade.
Scenario 2: Database Upgrade: 10.2.0.x → 10.2.0.y

System Setup

Production Database

No patches needed

Test Database

10.2.0.x: Clone of Prod
10.2 systems + Test Exec
patch installed

11g Test System: No application schema/data, stores SPA results
11.1.0.7 or 11.1.0.6 + patch

Metalink Note: 560977.1
Scenario 1: Database Upgrade: 10.2.0.x → 10.2.0.y

**Workflow**

1. Capture SQL workload to STS using Increment Cursor Cache Capture
2. Import STS
3. Send SQL for test execution on 10.2 before and after upgrade
4. Compare performance and generate SPA report
Scenario 3: Using SPA Functionality for 9i/10g → 11g Upgrades

- Similar workflow as Scenario 2
- Use 11g SPA system and test execute on 10g/11g source and destination target databases
  - Stores results of experiments separately
  - Allows use of latest releases for 11g SPA system
Evaluating Optimizer Statistics Refresh

**Scenario 4:**

Can I use SPA to check if any SQL statements regressed due to optimizer statistics refresh on my 10.2 production databases. If so, how can I evaluate the refreshed optimizer statistics?

**Goal:**

Assess impact of optimizer statistics gathering on SQL workload performance on production system & make sure are no negative effects of the change.
Evaluating Optimizer Statistics Refresh

- Assumptions
  - Optimizer has already gathered statistics on the database
  - Statistics refreshed periodically
  - No prod copy is available on test

- Use “11g SPA system” to evaluate optimizer statistics on 10.2 production database
  - Remote test execute before/after statistics refresh

- Analyze SPA report and take appropriate action
  - Overall improvement but few SQL regressions
    - Solution: Use SQL Profiles for regressed SQL
  - No improvement and many regressions
    - Solution: Revert to old statistics: Use optimizer statistics retention/history feature
  - For Oracle Database 11g, use publish pending statistics feature to publish statistics after evaluation of statistics
Evaluating Optimizer Statistics Refresh for 10.2

1. Capture SQL workload to STS

2. Import STS

3. Use SPA to detect plan changes

4. Compare performance and generate SPA report

5. Retain new statistics or revert to old statistics or use SQL Profiles for regression tuning
Real-World Deployments
# Large International Hotel Chain

| **Challenge** | • Upgrade critical customer-facing application providing rates for room reservations from Oracle Database 10.2.0.4 to 11.1  
|               | • Highly volatile data where plan stability is critical  
|               | • Unsuccessfully used synthetic queries to test previous upgrades |
| **Solution Approach** | • SQL Performance Analyzer to identify SQL regressions  
|               | • SQL Profiles to tune SQL transparently  
|               | • SQL Plan Baselines for plan stability |
| **Benefit** | • Very successful upgrade. No surprises!  
|               | • Predictable performance and SLAs  
|               | • Reduced testing time from 5 months to 10 days |
# E-Business Suite (EBS) Certification and Testing

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Solution Approach</th>
<th>Benefit</th>
</tr>
</thead>
</table>
| • Certify EBS release 11i, R12 against Oracle Database 11g  
• Complex & large workload: More than 650K unique SQL statements need to be validated  
• Ensure application optimized for Oracle Database 11g  
• Difficult to perform realistic and efficient testing with previous (home-grown) tools | • SQL Performance Analyzer to run regression tests and identify performance deviations  
• Regressions reported to base development for fixes | • Reduced testing time from 21 to 2 days for each release  
• Faster and higher quality testing  
• Faster adoption and certification of newer features |
## Real Application Testing Applicable for Pre-11g Database Releases

<table>
<thead>
<tr>
<th>Feature</th>
<th>Capture From</th>
<th>Test Changes In</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Performance Analyzer</td>
<td>9i R2</td>
<td>10g R2 or 11g</td>
</tr>
<tr>
<td></td>
<td>10g R1</td>
<td>10g R2 or 11g</td>
</tr>
<tr>
<td></td>
<td>10g R2</td>
<td>10g R2 or 11g</td>
</tr>
<tr>
<td>Database Replay</td>
<td>9i R2</td>
<td>11g</td>
</tr>
<tr>
<td></td>
<td>10g R2</td>
<td>11g</td>
</tr>
</tbody>
</table>

- **SQL Performance Analyzer**
  - Capture on 9i, 10.1, 10.2 database releases
  - Test changes in 10.2 & above

- **Database Replay**
  - Capture on 9i, 10.2 database releases
  - Test changes in 11.1 & above