Novices

SQL Tuning for Smarties, Dummies and Everyone in Between

Jagan Athreya
Director, Database Manageability, Oracle

Arup Nanda
Senior Director, Database Architecture, Starwood Hotels and Resorts
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.
Outline

- SQL Tuning Challenges
- SQL Tuning Solutions – New Feature Overview
- Problem Root Causes and their Solutions
- Preventing SQL Problems
- Q & A
SQL Tuning Challenges
Real-world DBA and Development Teams

• DBA team
  – Mostly average, some superstars
  – Superstars take most of the burden – over-stretched

• Development staff
  – Mostly non-Oracle skills – Java, C++
  – Usually considers the DB as a “black box”
  – Writing efficient queries, troubleshooting performance issues is delegated to DBAs
SQL Tuning Challenges
Production Performance

• Situation:
  – Query from hell pops up
  – Brings the database to its knees
  – DBA is blamed for the failure

• Response
  – DBA: “Developer should be taking care of this.”
  – Developer: “Why is the DBA not aware of this problem?”
  – Manager: “DBA will review all queries and approve them.”

• Challenge
  – What is the most efficient way to manage this process?
SQL Tuning Challenges

Change Causing Problems

• Situation
  – New SQL statements added as part of application patch deployment
  – Database upgrades
  – Database patching

• Response
  – Users: “How will the application perform after the changes?”
  – DBA: “How do I ensure that our SLA remains intact after the changes are rolled out?”

• Challenge
  – How to reduce business risk while absorbing new technologies?
SQL Tuning Challenges
Optimizer Statistics Management

• Situation
  – Data in Production has evolved over time. Have the optimizer statistics stayed current?

• Response
  DBA:
  – Will statistics refresh break something?
  – What will happen if we don’t collect?
  – How often should I collect the statistics?
  – What happens when you collect a new set?

• Challenge
  – What is the recommended strategy for managing optimizer statistics to ensure the best performance?
SQL Tuning Challenges
Bad Plans – Diagnosis and Resolution

• No time to find the root cause. How to prevent this from recurring?
• Bind variables: How do you prevent bad plans based on choice of bind variables?
• How to diagnose a bad plan
  – 10053 trace, endless pouring over traces
  – Wrongly constructed predicates
• How to fix a bad plan
  – Hints? change of code?
  – Baselines vs. SQL Profiles
  – Pick out a single SQL or a bunch from the shared pool
Outline

• SQL Tuning Challenges

• SQL Tuning Solutions – New Feature Overview

• Problem Root Causes and their Solutions

• Preventing SQL Problems

• Q & A
Real-Time SQL Monitoring
Looking Inside SQL Execution

- Automatically monitors long running SQL
- Enabled out-of-the-box with no performance overhead
- Monitors each SQL execution
- Exposes monitoring statistics
  - Global execution level
  - Plan operation level
  - Parallel Execution level
- Guides tuning efforts
New capabilities in SQL Monitoring
New in Oracle Database 11g Release 2

- PL/SQL monitoring including associated high load SQL monitored recursively
- Exadata aware I/O performance monitoring and associated metric data
- Capture rich metadata such as bind values, session details e.g. user, program, client_id and error codes and error messages
- Save as Active Report for rich interactive offline analysis
Application Tuning
Automatic SQL Tuning

- Automatic SQL Tuning
  - Identifies high-load SQL from AWR
  - Tunes SQL using SQL Profiles
  - Implements greatly improved SQL plans (optional)
- Performance benefit of advice provided
- SQL Profiling tunes execution plan without changing SQL text
- Enables transparent tuning for packaged applications
Automatic SQL Tuning
New in Oracle Database 11g Release 2

- SQL Tuning Advisor
  - NEW: Identifies alternate execution plans using real-time and historical performance data
  - NEW: Recommends parallel profile if it will improve SQL performance significantly (2x or more)
SQL Tuning for Developers
Integration with Visual Studio

- Introduced in Oracle Developer Tools for Visual Studio Release 11.1.0.7.20
- Oracle Performance Analyzer
  - Tune running applications with the help of ADDM
- Query Window
  - Tune individual SQL statements with STA
- Server Explorer
  - Manage AWR snapshots and ADDM tasks
Agenda

• SQL Tuning Challenges

• SQL Tuning Solutions – New Feature Overview

• Problem Root Causes and their Solutions

• Preventing SQL Problems

• Q & A
What makes SQL go bad?

Root Causes of Poor SQL Performance

1. **Optimizer statistics issues**
   - a. Stale/Missing statistics
   - b. Incomplete statistics
   - c. Improper optimizer configuration
   - d. Upgraded database: new optimizer
   - e. Changing statistics
   - f. Rapidly changing data

2. **Application Issues**
   - a. Missing access structures
   - b. Poorly written SQL statements

3. **Cursor sharing issues**
   - a. Bind-sensitive SQL with bind peeking
   - b. Literal usage

4. **Resource and contention issues**
   - a. Hardware resource crunch
   - b. Contention (row lock contention, block update contention)
   - c. Data fragmentation

5. **Parallelism issues**
   - a. Not parallelized (no scaling to large data)
   - b. Improperly parallelized (partially parallelized, skews)
What makes SQL go bad?
Root Causes of Poor SQL Performance

1. Optimizer statistics issues
   a. Stale/Missing statistics
   b. Incomplete statistics
   c. Improper optimizer configuration
   d. Upgraded database: new optimizer
   e. Changing statistics
   f. Rapidly changing data

2. Application Issues
3. Cursor sharing issues
4. Resource and contention issues
5. Parallelism issues
Oracle Optimizer Statistics

Inaccurate statistics

Suboptimal Plans

Optimizer Statistics

- Table Statistics
- Column Statistics
- Index Statistics
- Partition Statistics
- System Statistics
Oracle Optimizer Statistics
Preventing SQL Regressions

- Automatic Statistics Collection Job (stale or missing)
  - Out-of-the box, runs in maintenance window
  - Configuration can be changed (at table level)
  - Gathers statistics on user and dictionary objects
- Uses new collection algorithm with accuracy of compute and speed faster than sampling of 10%
- Incrementally maintains statistics for partitioned tables – very efficient
  - Set DBMS_STATS.SET_GLOBAL_PREFS
Oracle Optimizer Statistics
Preventing SQL Regressions

• Extended Statistics
  • Extended Optimizer Statistics provides a mechanism to collect statistics on a group of related columns:
    • Function-Based Statistics
    • Multi-Column Statistics
  • Full integration into existing statistics framework
    • Automatically maintained with column statistics
      DBMS_STATS.CREATE_EXTENDED_STATS

• Pending Statistics
  • Allows validation of statistics before publishing
  • Disabled by default
  • To enable, set table/schema PUBLISH setting to FALSE
    DBMS_STATS.SET_TABLE_PREFS('SH','CUSTOMERS','PUBLISH','false')
  • To use for validation
    ALTER SESSION SET optimizer_pending_statistics = TRUE;
  • Publish after successful verification
What makes SQL go bad?

Root Causes of Poor SQL Performance

1. Optimizer statistics issues
2. Application Issues
   a. Missing access structures
   b. Poorly written SQL statements
3. Cursor sharing issues
4. Resource and contention issues
5. Parallelism issues
Identify performance problems using ADDM
Automatic Database Diagnostic Monitor

- Provides database and cluster-wide performance diagnostic
- Throughput centric - Focus on reducing time ‘DB time’
- Identifies top SQL:
  - Shows SQL impact
  - Frequency of occurrence
- Pinpoints root cause:
  - SQL stmts waiting for Row Lock waits
  - SQL stmts not shared
Identify High Load SQL Using Top Activity

Performance Page

• Identify Top SQL by DB Time:
  • CPU
  • I/O
  • Non-idle waits

• Different Levels of Analysis
  • Historical analysis
    • AWR data
    • Performance Page
  • Real-time analysis
    • ASH data
    • More granular analysis
    • Enables identification of transient problem SQL
  • Top Activity Page

• Tune using SQL Tuning Advisor
Advanced SQL Tuning
Universe of 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**: Primary Key materialized views, Object materialized views, ROWID materialized views, Complex materialized views

- **Partitioned Tables**: Range partitioning, Hash partitioning, List partitioning, Composite partitioning, Interval Partitioning, REF partitioning, Virtual Column Based partitioning
SQL Access Advisor: Partition Advisor

Representative Workload

SQL Access Advisor

Automatic Tuning
Optimizer

Access Path
Analysis

Indexes

Materialized views

Materialized views logs

Partitioned objects

Novice+ Mode
**SQL Access Advisor**

**Advanced Options**

- Workload filtering
- Limited vs. advanced mode
- Tablespaces for access structures
- Hypothetical workload tuning
- Factoring in the cost of creation
- Space limitations for indexes and MVs
What makes SQL go bad?
Root Causes of Poor SQL Performance

1. Optimizer statistics issues
2. Application Issues
3. Cursor sharing issues
   a. Literal usage
   b. Bind-sensitive SQL with bind peeking
4. Resource and contention issues
5. Parallelism issues
What makes SQL go bad?
a. Literal Usage Issue

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

cursor_sharing = {exact, force, similar}

Library Cache

Sharing Cursors is good!
What makes SQL go bad?

b. Bind Peeking Issue

Problem: Binds will affect optimality in any subsequent uses of the stored plan.
Fixing problems with Adaptive Cursor Sharing

Adaptive Cursor Sharing

```
SELECT * FROM emp WHERE wage := wage_value;
```

Selectivity Ranges:

1. 20 to 25
2. 22 to 24
3. 30 to 35
4. 34 to 43

Same Plan
Different Plan
Same Plan, Expand Interval
Agenda

• SQL Tuning Challenges

• SQL Tuning Solutions – New Feature Overview

• Problem Root Causes and their Solutions

• Preventing SQL Problems

• Q & A
Preventing problems with SQL Plan Management

• Problem: changes in the environment cause plans to change

• Plan baseline is established

• SQL statement is parsed again and a different plan is generated

• New plan is not executed but marked for verification
SQL Plan Management
Migration of Stored Outlines to Plan Baselines

1. Begin with
   CREATE_STORED_OUTLINES=true

2. Run all SQL in the Application and automatically create a Stored Outline for each one

3. After Store Outlines are captured
   CREATE_STORED_OUTLINES=false

4. Upgrade to 11g

5. Migrate Stored Outlines into SPM
SQL Performance Analyzer (SPA)

Validate statistics refresh with SPA

• Steps:
  1. Capture SQL workload in STS using automatic cursor cache capture capability
  2. Execute SPA pre-change trial
  3. Refresh statistics using PENDING option
  4. Execute SPA post-change trial
  5. Run SPA report comparing SQL execution statistics

• Before PUBLISHing stats:
  • Remediate individual few SQL for plan regressions: SPM, STA
  • Revert to old statistics if too many regressions observed
Conclusion
Identify, Resolve, Prevent

1. Production Performance
2. Change Causing Problems
3. Optimizer Statistics Management
4. Bad plans – Diagnosis and Resolution

Prevent
- SPA
- SPM

Resolve
- Tuning Advisor
- Access Advisor
- Auto Stat Collection

Identify
- Top Activity
- ADDM
- SQL Monitoring