TD Performance Tuning

Agenda

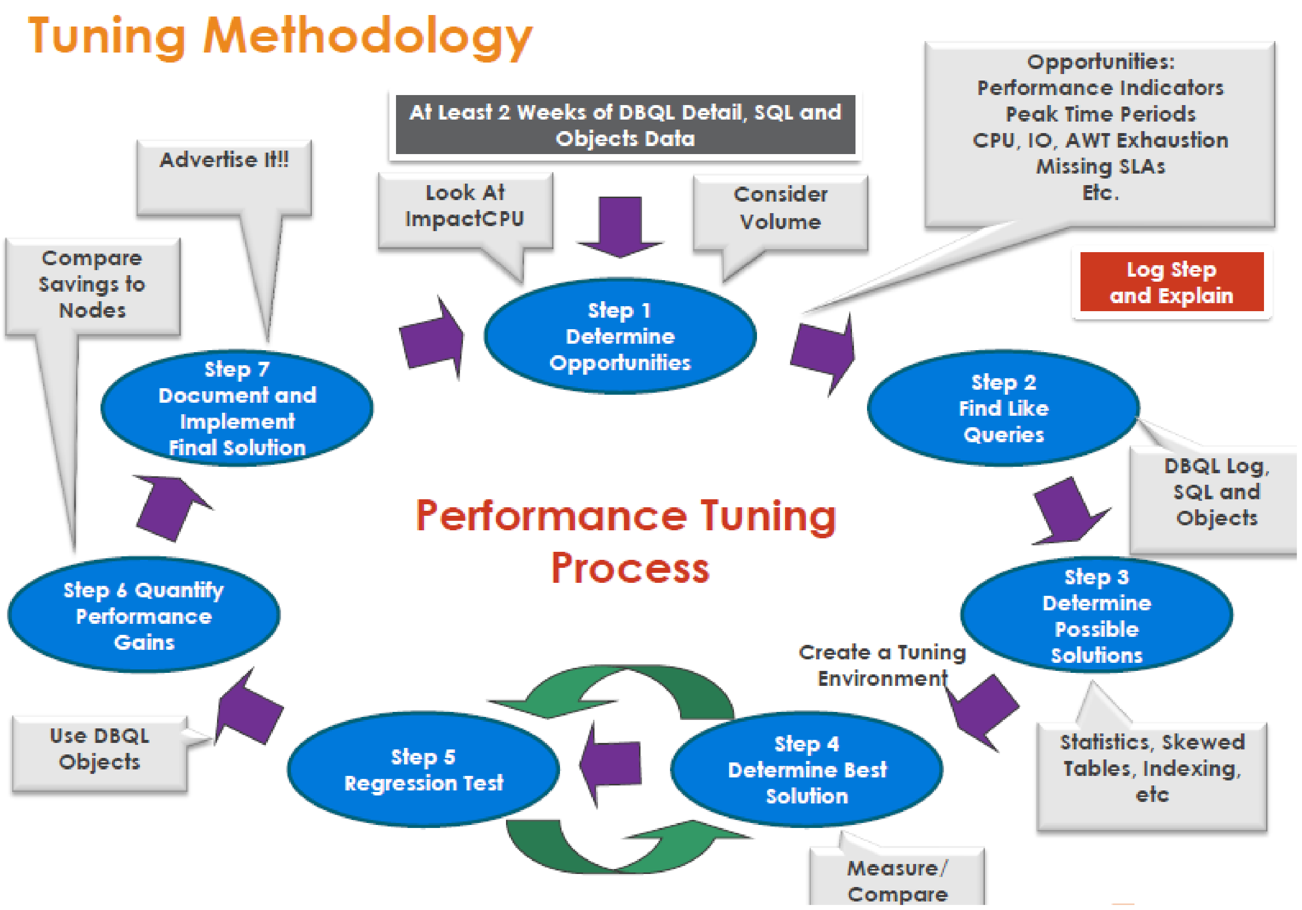
- Performance Tuning Methodology
- Finding Tuning Opportunities
- Tuning Tips and Techniques

Introduction

- Performance tuning efforts aim at identifying the high impact performance problems and to fix them as fast as possible. To do that, it includes:

- Identifying queries with the greatest impact on system resources.
    - High resource usage – Impact CPU, High CPU, High IO consumers.
    - Skewed queries.
    - Long running queries.

Tuning Methodology



Tuning Methodology

At Least 2 Weeks of DBQL Detail, SQL and Objects Data

Look At ImpactCPU

Consider Volume

Opportunities:
Performance Indicators
Peak Time Periods
CPU, IO, AWT Exhaustion
Missing SLAs
Etc.

Log Step and Explain

Step 1 Determine Opportunities

Step 2 Find Like Queries

DBQL Log, SQL and Objects

Step 3 Determine Possible Solutions

Statistics, Skewed Tables, Indexing, etc

Create a Tuning Environment

Step 4 Determine Best Solution

Measure/ Compare

Step 5 Regression Test

Use DBQL Objects

Step 6 Quantify Performance Gains

Compare Savings to Nodes

Step 7 Document and Implement Final Solution

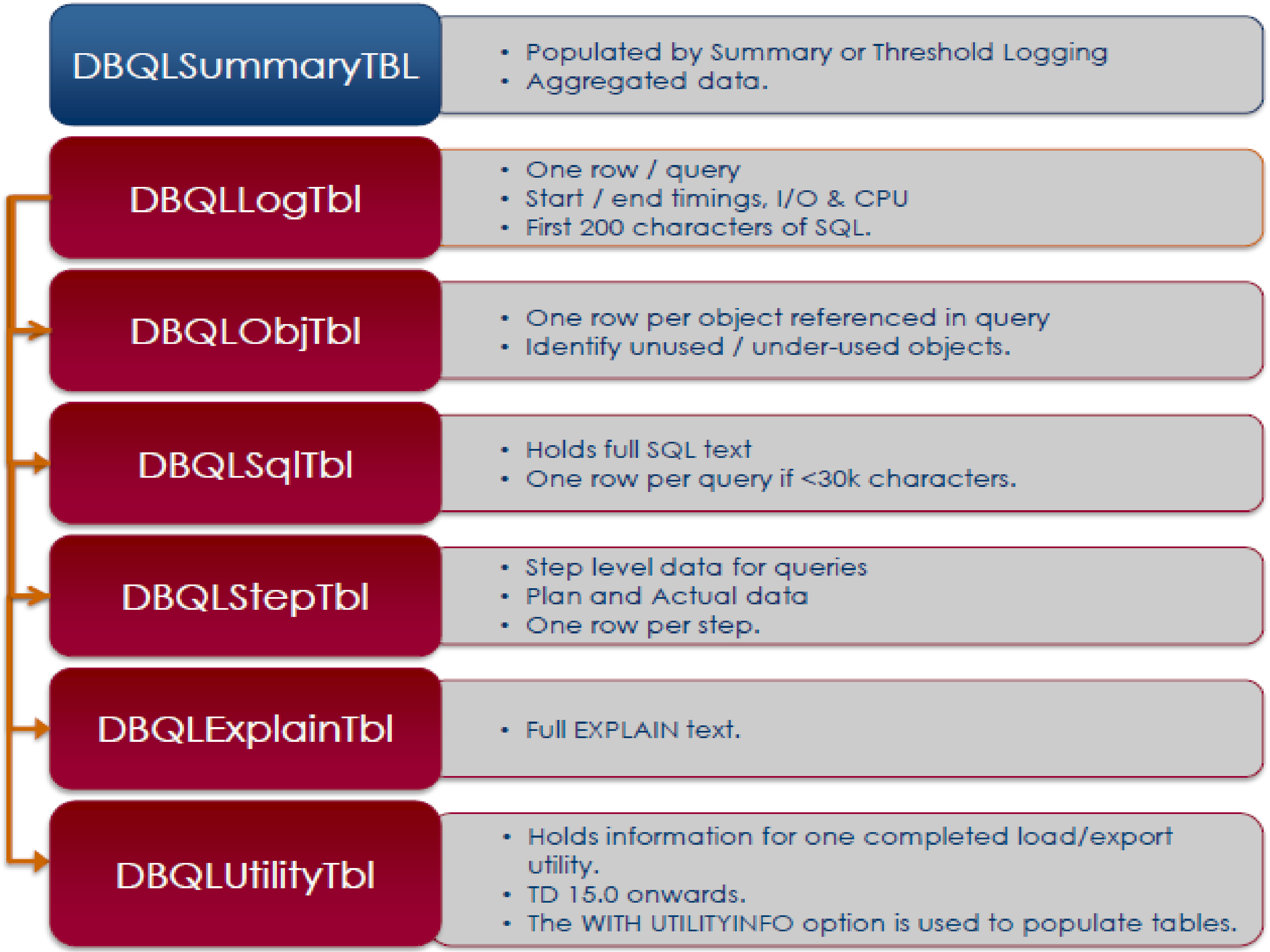
Advertise It!!

Performance Tuning Process

Performance Tuning Process Step 1 – Determine Opportunities

- By doing trend analysis of system for at least 2-3 weeks of data using TD perf tables
- Look for Groups of Usage with High Skewed Query CPU, Product Join Query CPU or Large Scan Query CPU
- Prioritize based frequency, high resource consumption

DBQL Trend Analysis



DBQLSummaryTBL

- Populated by Summary or Threshold Logging
- Aggregated data.

DBQLLogTbl

- One row / query
- Start / end timings, I/O & CPU
- First 200 characters of SQL.

DBQLObjTbl

- One row per object referenced in query
- Identify unused / under-used objects.

DBQLSqlTbl

- Holds full SQL text
- One row per query if <30k characters.

DBQLStepTbl

- Step level data for queries
- Plan and Actual data
- One row per step.

DBQLExplainTbl

- Full EXPLAIN text.

DBQLUtilityTbl

- Holds information for one completed load/export utility.
- TD 15.0 onwards.
- The WITH UTILITYINFO option is used to populate tables.

Performance Indicators in DBQL

- Product Join Indicator (PJI)
- Unnecessary IO Indicator (UII)
- CPU or IO Skew.
- ImpactCPU (a measure of the impact of skewing).
- High CPU.
- High IO.
- Significant Response Time.

Tuning Tips and Techniques

- Review the selected queries to identify potential tuning solutions

–Analysis Includes Reviewing

-Missing or Stale Statistics
-Query Step Data
-Index Review
-Primary Indexes
-Partition Primary Indexes
-Secondary Indexes
-Join Indexes
-Correct SQL
-General Query Rewrite

Checklist to tune Queries

Task	Details
DBQL analysis	Use DBQL Information available to determine what Queries perform poorly.
Statistics	Check stats for joined columns - single and multi-column, PARTITION
Explain	Check explain plan for problem areas and whether the plan is created as expected.
Join conditions	Evaluate join paths.
Where clause	Review hard coded conditions in where clause.
PI changes	Based on join conditions above, check whether any table PI needs to change to facilitate local amp joins. Check skew.
PPI for where clause columns	Based on where clause conditions, check whether any columns can be used as a PPI so as to eliminate partitions.
MLPPI if possible	Based on where clause conditions, check whether any columns can be made as MLPPI so as to eliminate partitions.
MVC, BLC	Check for compression on large tables.
SOFT RI	Check for candidates for SOFT RI

Continued….

Task	Details
Fallback	Check if tables inserted to are FALLBACK. If yes, make them NOT FALLBACK if not needed.
Character set mismatch	Checks if the columns with character data type have the same name but have different character sets.
PI For Volatile Tables	Check for PI of volatile tables based on joined columns. Check skew.
Stats on Volatile Tables	Check for stats on joined columns and PARTITION of volatile tables.
Query Rewrite	Check if the query can be re-written to make it more efficient. E.g. Volatile tables, WITH clause for repeated derived tables, SQL changes, etc.
NUSI Possibility	rarely use NUSIs, use this as one of the final options; and check on its usage
Join Index	Should be a last resort. Choose proper PI and PPIs for skew issue and add rowid. This includes STJI, AJI.