Oracle Optimizer, Statistics and Workloads on Partitioned Tables

All courses are built and delivered by Tanel Poder.

This course applies both to data warehouses and OLTP systems with partitioned tables, where consistent SQL performance and good optimizer behavior are important. We will jump straight into Oracle 19c DBMS_STATS, statistics usage and optimizer behavior on large partitioned tables.

Contents & topics:

- New scripts for examining table, partition-level stats and histograms
- Optimizer behavior with partitioned tables - partition-level stats usage, cardinality estimates

- Choosing between table level vs. partition level statistics
- When to use partition level histograms instead of global ones
- Partition level statistics, cardinality estimates with bind variables

- Efficient incremental statistics maintenance with or without histograms
- DBMS_STATS preferences for incremental stats gathering (and pitfalls)
- Ensuring modern (HyperLogLog-based) approximate NDV calculation over partitions

- Understanding statistics staleness, *incremental_staleness*, table modification monitoring
- Monitoring DBMS_STATS gathering activity, completion success and changes

- Handling stats for empty partitions and new data loads with concurrent queries
- When to use DBMS_STATS.COPY_TABLE_STATS
- How do the global table-level stats and histograms work with partition exchange loads?
- Partition maintenance DDL, cursor invalidation and concurrent query issues *(ORA-8103)*

- Dealing with optimizer bugs and known bugfixes (*v$fix_control*) with partitioned tables
- Dealing with DBMS_STATS gathering bugs and known bugfixes with partitioned tables

Example symptoms, questions and problems we will address:

These are some of the "bad SQL performance" related keywords that you may already be familiar with:

- Wrong E-Rows (cardinality = 1) for some queries using partition level stats
- Wrong cardinality estimates as *"selectivity of pred having unreasonable low value"* in CBO trace
- *"Ignoring histogram of column (COL_NAME) used only for incremental stats maintenance"*
- *dba_tab_part_col_statistics.notes* field shows *HIST_FOR_INCREM_STATS* - why?
- Stats on some partitions are not updated, despite having changed data in them - why?
- DBMS_STATS is doing too much work, re-gathering stats on all partitions or running out of time
- Queries touching freshly loaded partitions (without stats) pick a wrong plan and are very slow

*The goal of this course is to help you make the **optimizer work for you**, efficiently and predictably. This gives you better performance and fewer surprises, even on very large & busy partitioned databases.*

Adaptive Behavior in Oracle SQL Optimizer and Plan Execution

All courses are built and delivered by Tanel Poder.

In this course, we will dive deep into how important it is for the Optimizer to calculate correct enough cardinality (row-count) estimates for SQL plan optimization. All the features you see here are tied to cardinality estimation in one way or another. Let's help the optimizer get it right!

Contents & topics:

- Data types - datatype conversion can disable access paths and mess up *cardinality estimates*
- Bind peeking - a reminder of why it's even needed and what problems it brings
- Adaptive Cursor Sharing - pick a child cursor based on estimated *selectivity* of new binds

- Statistics Feedback (cardinality feedback) - learn the real row-counts from previous execution
- SQL plan auto-reoptimization and V$SQL_REOPTIMIZATION_HINTS

- Adaptive Joins - switch from nested loop join to hash join if too many rows in driving rowsource
- Adaptive serial direct path reads - nowadays a partition-level decision, derived from statistics
- parallel_degree_policy=AUTO - can cause PX table scans via buffer cache (Exadata!)

- Examining automatic column group stats gathering (dbms_stats.auto_stats_extensions)
- Manually creating and examining multi-column (column group) stats

- Controlling and strategic use of dynamic sampling
- Exploring SQL Plan Directives
- Should I ever enable the full Optimizer Adaptive Statistics feature?

- How does all this work together, with previously created SQL Patches, Baselines, Profiles?

Example symptoms, questions and problems we will address:

These are some of the "bad SQL performance" related keywords that you may already be familiar with:

- Bad SQL performance due to cardinality misestimates, due to datatype mismatch in SQL or binds
- Bad SQL performance due to optimizing for unlucky values in statistics distribution
- Unstable SQL execution performance due to unlucky bind variable peeking or stats
- Unstable SQL execution performance due to adaptive join decision overhead
- Unstable SQL execution performance due to unlucky direct path scanning logic
- Bad cardinality estimates due to missing (or counter-productive) histogram choices
- Bad system concurrency due to too many adaptive features and their interactions
- I/O overhead due to excessive dynamic sampling and/or SQL plan directives (and how to fix it)
- Bugs: Chasing down which optimizer bug (and fix_control) might be affecting your workloads

*The goal of this course is to help you make the **optimizer work for you**, efficiently and predictably. This gives you better performance and fewer surprises, on diverse workloads that your databases are running.*

Running I/O-Intensive Oracle Databases On-Premises & in the Cloud

All courses are built and delivered by Tanel Poder.

In this course, we will learn how to approach, measure and understand the Oracle Database + I/O + Linux topics listed below, to be equipped for systematically improving your databases' performance and stability. We will use built-in Oracle instrumentation and Tanel's new scripts & tools for workload level I/O and DB activity analysis. We will also use OS tools for validating any new platform's I/O capabilities in advance.

Contents & topics:

- Workload level data-access and I/O analysis (not just a single SQL at a time)
- In a complex system, it's not enough to just look into top-5 individual SQL statements in AWR
- What kinds of operation types, reasons and access patterns are causing the most I/O and load

- Concurrency of batch jobs and their effect to buffer cache overload and excessive I/O
- What does the *read by other session* wait event really indicate
- Measuring block re-reads, buffer caching efficiency at table, index level

- Using *fast_start_mttr_target* for smoothing out the spikes of DBWR writes and checkpointing
- Using *_time_based_rcv_ckpt_target* for smoothing out checkpoints on DataGuard instances

- Isolating redo logging and other low latency writes from I/O spikes from other activity
- Why put redo logs on a separate Linux block device, if it's backed by the same storage array
- Why put controlfiles on that separate block device, next to redo logs, too

- Measuring Linux OS level I/O latency & filesystem bottlenecks under concurrency
- Validating that your (cloud) platform can even provide the required throughput, IOPS and latency
- Configuring and using the Linux *fio* tool for generating Oracle-like I/O patterns
- Running synthetic Oracle Database workloads for stress testing the servers and I/O subsystem

Example symptoms, questions and problems we will address:

- Database I/O waits have gone up, is it because of the database, OS or the storage?
- Random I/O wait event spikes and query, transaction slowdowns
- *log file switch (checkpoint incomplete)* wait events and warnings in alert.log
- *log file switch (private strand flush incomplete)* wait events
- High I/O wait event durations in Oracle, but the OS tools show that everything is OK
- High I/O wait event durations in Oracle and the OS tools also show I/O problems
- High I/O waits and the Linux system load goes through the roof
- The *log file parallel write* events are OK, but the application commits are slow
- The database is not able to drive I/O nearly at the rate of what the storage vendor promised

The goal of this course is to help you get a thorough understanding of your Oracle database and Linux I/O capabilities - and tools for gathering evidence. In addition to the Oracle metrics, we'll use Linux "fio" and Tanel's troubleshooting tools for isolating where an I/O bottleneck comes from, on-prem and in the cloud.