Reducing CPU and Maximizing Db2 Application Performance

This seminar will cover many best practices for our applications in order to achieve performance and reduce CPU in Db2 z/OS. Many techniques to minimize overhead and locking will be explored. We will take a look at techniques for coding efficient SQL and minimizing the amount of data we bring back to process in our applications via optimal predicates, less repetition and better use of indexes. We will also look at how to best utilize the dynamic statement cache and to use it in performance tuning efforts along with several Explain tables.

Course Outline:

High performing applications overview
- What makes an application perform well
- What kills application performance

SQL for call reduction
- SELECT from INSERT
- SELECT from UPDATE/DELETE
- Common table expressions
- Recursion usage to reduce calls and joins
- Order by and fetch first in subselect
- Multi-row fetch
- Multi-row insert
- MERGE for replication
- LISTAGG function

Keeping the database calls in Db2
- Using Db2 referential Integrity
- Caching common code values
- Using triggers
- Basic vs advanced triggers
- Non-deterministic expressions for auditing
- Trigger performance trade offs
- User Defined Functions
- Stored procedures

Minimizing application overhead
- Number of columns retrieved
- Number of columns/rows sorted
- Performance trace for sorting

SQL filtering, index exploitation and predicate evaluation
- SQL performance objectives
- Filtering
- Predicate comparison
- Predicate pushdown
- Residual predicate processing
- Index on expression
- Indexable predicate conversion
- Boolean term predicates
- Join predicates
- Function evaluation
- CASE expressions
- Row expressions

SQL choices for performance
- Subquery performance issues
- Non-correlated subquery performance
- Correlated subquery performance
- Merge vs materialization
- Table expressions
- UNION ALL join distribution
- Searching on multiple conditions

Filter factors and access path influence
- Filter factors
- Distribution statistics
- Histogram statistics
- Statistics feedback
- Filter factor influencing
- Reoptimization
- OPTIMIZE FOR n ROWS

Repeat processing efficiency/elimination
- Random I/O causes and resolution
- Excessive sequential processing
- Excessive index screening
- Programmatic joins conversion to SQL
- Cursor in cursor conversion to SQL
- Putting work in SQL
- Redundant SQL issue/cost
- Using MQTs to avoid repetitive processing

Reducing CPU and Maximizing Db2 Application Performance

Explain and the Dynamic Statement Cache
- What's missing in Explain
- Explain tables
- PLAN_TABLE
- DSN_FILTER_TABLE
- DSN_PREDICATE_TABLE
- DSN_DETCOST_TABLE
- Advanced Explain queries
- Dynamic statement cache
- Coding to use statement cache
- DSN_STATEMENT_CACHE_TABLE
- Analyzing the dynamic statement cache
- Dynamic cache and literal replacement

Binds and Rebinds
- DEGREE and parallelism usage
- ISOLATION options for performance
- REOPT impact and usage
- RELEASE option for performance
- DEFER distributed impact
- Plan stability
- APCOMPARE usage and interpretation

Locking and Concurrency
- Cost of locking
- Locking statistics to monitor
- Locking reduction/avoidance
- Row level locking
- Deadlocks and timeouts
- Lock wait time/escalation
- Optimistic locking
- Skip locked data
- Currently committed data
- Coding for best concurrency

Miscellaneous application performance topics
- Next-key generation
- Using sequence objects
- Sequence objects for key propagation
- Fetch first usage

Batch restart cost
Commit issues/detection
Using Savepoints