IBM Information On Demand 2007

DB2 9 for z/OS Select, Insert, Update, Delete SQL Application Performance

Akira Shibamiya, IBM, shibamiy@us.ibm.com

Session: 1227, Track: Data Servers – System z – DB2 and Tools

Act.Right.Now.

IBM INFORMATION ON DEMAND 2007
October 14 – 19, 2007
Mandalay Bay
Las Vegas, Nevada

Abstract

- **This presentation provides a look at the performance impact of DB2 9 for z/OS from SQL performance viewpoint, primarily for Select, Open/Fetch, Insert, Update, and Delete.**

Acknowledgment and Disclaimer

- Measurement data included in this presentation are obtained by the members of the DB2 performance groups at the IBM Silicon Valley and Poughkeepsie Laboratory.
- Powerpoint notes are provided by Roger Miller.
- The materials in this presentation are subject to
  - enhancements at some future date,
  - a new release of DB2, or
  - a Programming Temporary Fix
- The information contained in this presentation has not been submitted to any formal IBM review and is distributed on an "As Is" basis without any warranty either expressed or implied. The use of this information is a customer responsibility.

Agenda

1. SELECT, OPEN/FETCH Performance
2. INSERT, UPDATE, DELETE Performance
3. Topics Common to Most SQL Calls

- For each new V9 performance feature, (CM) or (NFM) is shown to indicate if it is supported in Compatibility Mode or New Function Mode.

SELECT, OPEN/FETCH Performance

SELECT, OPEN/FETCH Performance

- Sort (CM)
- Cross-query block optimization (CM)
- Pair-wise join in star schema (CM)
- Sparse index or in-memory data extension (CM)
- More parallelism (CM)
- DPSI improvement (CM)
- Histogram statistics (NFM)
- Index on expression (NFM)
- Automatic reoptimization (NFM)
- Concurrency improvement (NFM)

Sort Performance Improvement

- Group By sort and Distinct sort (CM)
  - Group By sort improvement when no column function and no available index
    - Eg SELECT a1 FROM A GROUP BY a1
  - Distinct sort improvement when no index or only non unique index available
    - Eg SELECT DISTINCT a1 FROM A
  - Up to 2 times improvement

Sort Performance - continued

- Fetch First N Rows with Order By (CM)
  - Example: Get top 10 Americans in income tax paid
  - Avoid tournament tree sort for small N
  - Up to -50% cpu in one measurement test
  - Supported in Subselect also in V9 (NFM)
- In-memory work file for small sorts (CM)
  - 10 to 30% cpu reduction for short-running SQL calls with small sorts
  - Beneficial for online transaction with relatively short-running SQL calls in which the number of rows sorted can be small.

Sort Performance - continued

- **Heavier use of 32K work file BP to help large work file record performance (CM)**
  - **V8 uses 4K BP if less than 4KB row**
  - **V9 uses 32K BP for more records to gain improved performance, especially I/O time**
    - **Less work file space and faster I/O, for example 15 2050byte records on one 32K page vs 8 records on 8 4K pages**
  - **Some measurement example**
    - **<10% difference for 50 and 85byte records as 4K BP continues to be used**
    - **10-50% improvement for 95byte and bigger records because of 32K BP**

NOTES

- For smaller records, 4K BP better because of 255 rows per page limit
  - eg over 90% wasted space on 32K page for 10byte records
- Recommendation
  - Assign bigger 32K work file BP
  - Allocate bigger/more 32K work file datasets
  - If 4K work file BP activity is significantly less, corresponding BP size and work file datasets can be reduced.
- New statistics on how often more optimal 32K workfiles ran out and 4K workfiles used instead, or vice versa

Access Path Enhancement

- Histogram statistics over a range of column values (NFM)
  - For more precise filter factor estimates and better access path selection
  - Useful in range as well as equal predicates with high cardinality, eg NAME in contrast to STATE
  - Equal-depth (each interval with roughly same number of rows)

- Cross query block optimization in complex query (CM)
  - Optimization across, rather than within, query blocks
  - More predicate transitive closure across query blocks

Access path - continued

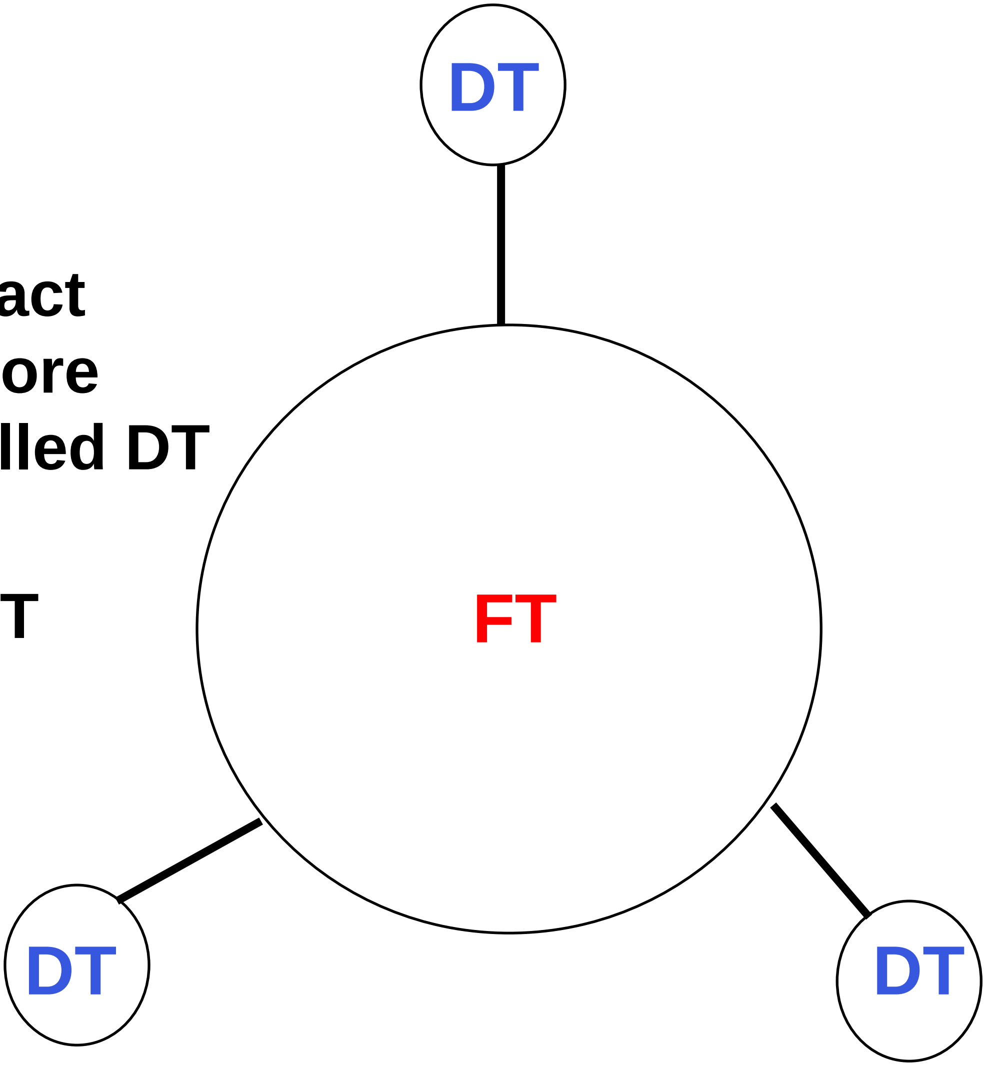
- Index on expression (NFM)
  - Example: Create Index xxx on Employee(salary+bonus, bonus*100/salary)
  - Orders of magnitude improvement if a predicate using such an index
  - Extra cost in Load, Update on key value, Rebuild Index, Check Index, Reorg tablespace but not Reorg index, and Insert as expressions are evaluated in Insert or index rebuild
  - Not eligible for zIIP offload as index expression evaluation done at load or unload rather than build index phase

Access Path - continued

- Pair-wise join in star schema queries (CM)
  - Each dimension table joins with fact table separately
  - Using single column indexes, instead of difficult-to-tune multi-column index(es), in parallel
  - Each pair of candidate ridlists are AND’d together to produce a final result (Dynamic Index Anding)
  - Especially effective when indexes are not well designed and tuned for performance

NOTES

- Star schema query
  - Large table called Fact Table FT and 2 or more Dimension Table called DT
  - Equi-join between FT and DT but no join between DTs



Access Path - continued

- Pair-wise join in star join queries - continued
  - More parallelism and more zIIP offload
    - For 100 star join query example, 37% average offload in V8 and 62% in V9
    - These figures are very sensitive to queries themselves as well as hardware environment, eg zIIP utilization
  - Optimization at runtime for further performance improvement
    - Ridpool overflow to DASD via work file
      - Supported in star schema only for now
    - Some parallelism even when degree 1

NOTES

- When STARJOIN is enabled and the number of tables joined meets SJTABLES threshold, star join and pair-wise join are considered along with nested loop join, merge join, and hybrid join based on cost estimates.

- Significant bind time reduction also possible for star join

Access Path - continued

- **Optimizer cost model update (CM), eg**
  - Cluster ratio
  - Parallel access path selection separate from sequential
  - Comprehensive Runstats data recommended to keep any performance regression to a minimum
    - Statistics Advisor can help identify the useful statistics to collect
      - For a given query in V8 (CM)
      - For an entire workload in V9 (CM)

NOTES

- Access path APARs
  - PK41165 7/07 Star join query performance not optimal
  - PK46082 8/07 Improved clone table access path by using corresponding base table stats
  - PK48500 8/07 Multi-index access path not considered in some OR predicates

Access Path - continued

- REOPT AUTO (NFM)
  - REOPT None, Once, or Always in V8
  - Automatic REOPT based on parameter marker change
    - Only for dynamic statements that can be cached

- Generalized sparse index or in-memory data for all accesses, not just star join (CM)

- More parallelism leading to more zIIP offload (CM)
  - Especially in star schema queries
  - Access path selection for parallelism separate from sequential

NOTES

- REOPT AUTO apars
  - PK47318 Not reoptimizing when it should
  - PK49348 to reduce the number of short prepares
- For more Optimizer details, see Terry Purcell's "What's New with the Optimizer in DB2 9 for z/OS" session 1299
- For more details on the following, see Gene Fuh's "New Technology for the Top Query Performance Issues in V8, V9, and Beyond" session 1229
  - OSC Optimizer Service Center
    - Tools to monitor and tune SQL workloads
  - OE Optimizer Expert
    - OSC + Index/Query/Statistics Advisors

Data Partitioned Secondary Index (DPSI) Improvement (CM)

- Enhanced page range screening
  – To avoid having to visit all partitions when DPSI is used wherever possible
- More parallelism with DPSI
  – Parallel support for DPSI when DPSI is used as index access to data and ordering is expected
  – Example: SELECT * FROM T1 ORDER BY C1 with DPSI on C1
- More index lookaside
- Index-only access with DPSI and ORDER BY
- Unique DPSI support when DPSI columns are superset of partitioning columns
  – E.g. partitioning key on c1.c2 and DPSI key on c1.c2.c3

Concurrency

- Default bind option change (NFM)
  - Isolation RR to CS and Currentdata Yes to No
  - For reduced locking cpu and better concurrency

- Skip locked data option in SELECT (NFM)
  - This option skips locked row or page, while Uncommitted Read isolation mode does not
    - For CS or RS, not UR or RR
  - Conditional instead of unconditional lock request issued
    - Instant lock with immediate unlock for each row or page

Insert/Update/Delete Performance

- Scalability
  - Log latch contention
  - Index page P-lock contention
  - Index tree latch contention
- CPU time
  - Index lookaside
  - Table space append option
  - Index usage history

NOTES

- Insert/Update/Delete performance is, and has always been, one of the most challenging issues in any database management system.
  - V9 adds dramatic performance/scalability improvement in this area.

- LCxx = Latch Class xx

- V8 PK30160 9/06 non segmented, PK36717 1/07 segmented, to avoid excessive conditional lock failures with many concurrent inserters using page locking
  - V7/V8 PK47840 7/07 and V9 PK51099 Also for P-lock failures for row locking

Log-related enhancements

- LC19 Log latch contention relief in data sharing (NFM)

- No Log table space option where appropriate (NFM)
  - No difference in accounting CPU
  - Significant accounting CPU time reduction possible if high log latch contention
  - If log I/O-bound, then good elapsed time reduction

- Archive log to use BSAM, enabling (NFM)
  - I/O striping
  - Compression if Extended Format data set

Index lookaside for additional indexes (CM)

- In V8, for clustering index only in Insert, none for Delete
- In V9, possible for more indexes in both Insert and Delete
- Big reduction in the number of index Getpages possible
  - In one benchmark of heavy insert into a large table with 3 indexes, all in ascending index key sequence,
    - 0+6+6=12 index Getpages per average insert in V8
    - 0+1+1=2 in V9

Randomized index key to avoid hot spots (NFM)

- Can be beneficial for data sharing because of index page P-lock contention
- CREATE/ALTER INDEX … column-name RANDOM, instead of ASC or DESC
- Trade-off between contention relief and additional Getpage, read/write I/O, and lock request
  - Better for indexes resident in buffer pool

Tablespace append option in Insert (CM)

- CREATE/ALTER TABLE … APPEND YES
- To reduce longer chain of spacemap page search as tablespace keeps getting bigger

Bigger preformatting quantity and trigger ahead (CM)

- From 2 (V8) to 16 (V9) cylinders if >16cyl allocation
- 27% faster Insert with ESS 800 in one measurement
- 47% faster with DS8300 turbo
- Wait for asynchronous preformat shows up in x’09’ Lock wait

Index page split reduction

- Bigger index page (NFM)
  - 4K, 8K, 16K, or 32K page
    - Up to 8 times less index split
  - Good for heavy inserts to reduce index splits
    - Especially recommended if high LC6 contention in data sharing
      - 2 forced log writes per split in data sharing
    - Or high LC254 contention in non data sharing shown in IFCID57
  - Trade-off with possibly more index page P-lock contention in random access

Index page split - continued

- Asymmetric index page split depending on an insert pattern when inserting in the middle of key range (NFM)
  - Instead of 50-50 split
  - Up to 50% reduction in index split
  - 20% class 2 cpu and 31% elapsed time improvement in one data sharing measurement
  - 10% cpu and 18% elapsed time improvement in one non data sharing measurement

Real Time Stats SYSINDEXSPACESTATS.LASTUSED (NFM)

- **Indicates when index used in SELECT/FETCH, searched UPDATE/DELETE, and Referential Integrity check, but not INSERT, LOAD, etc. (also in V9 catalog)**
- **Useful in getting rid of unnecessary indexes**
- **PK44579 8/07 to support the use of index in Referential Integrity, Rid list processing, set functions, and XML values index**

Very Heavy Insert Performance Measurement Example

- **Up to -18% cpu in non data sharing**
- **Up to -65% cpu in data sharing**
  - **-3x Getpage and index page P-lock**
  - **-25x index page P-lock suspension/negotiation**

Topics Common to Most SQL Calls

Act.Right.Now.

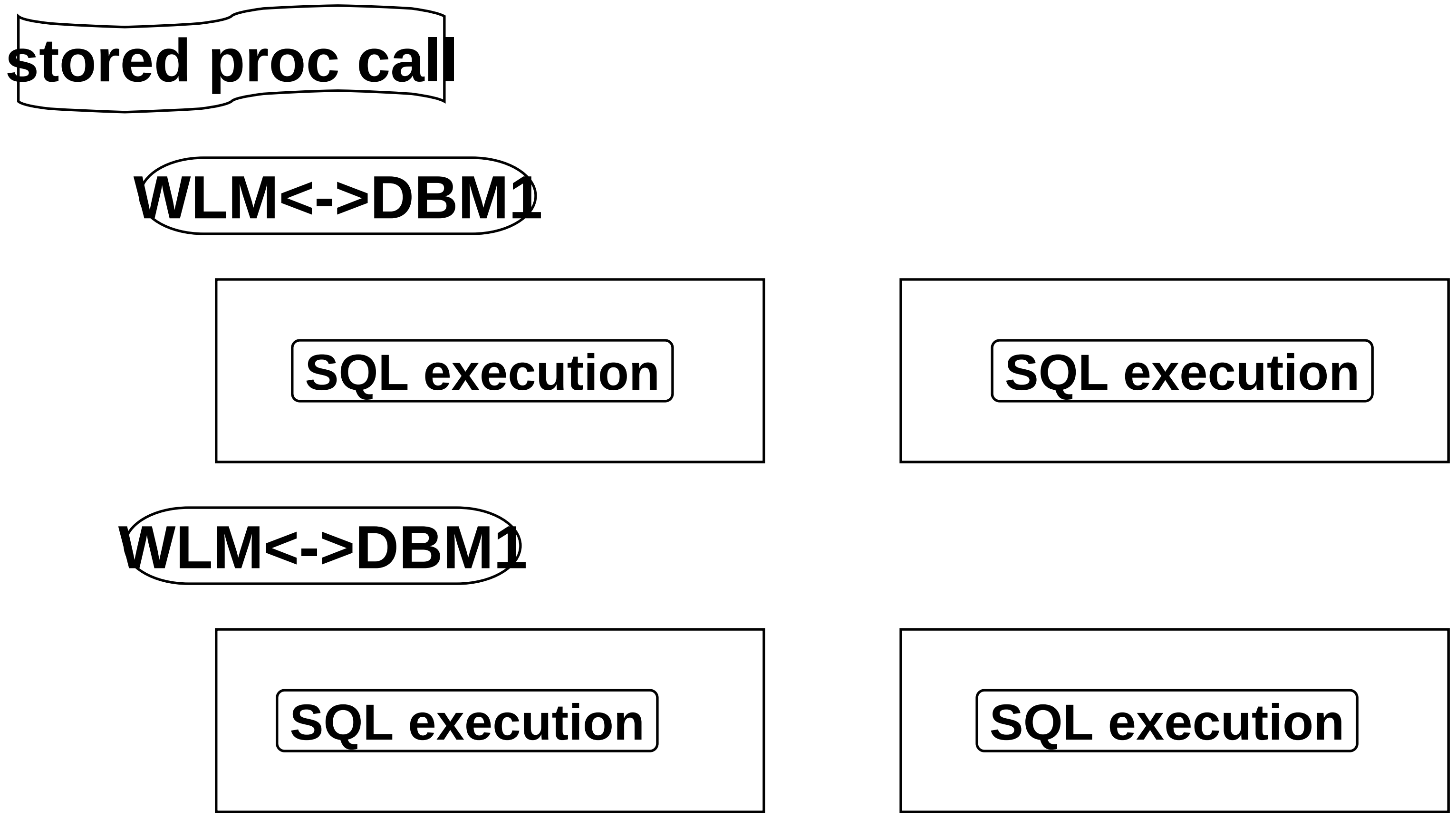
This page is intentionally left blank.

Native SQL Procedure (NFM)

- **Avoid the stored proc invocation overhead and roundtrip between Work Load Manager and DBM1 address spaces for each SQL call**
  - **0 to 40% ITR improvement compared to external SQL procedure observed**
  - **No difference if long-running SQL**
- **zIIP-eligible if DRDA as it runs in DBM1, not WLM, address space under DDF enclave SRB**
- **V9 PK45265 8/07 (also V8 PK28046) to reduce LC24, EDM pool full, and unavailable storage when same external stored procedure or native SQL procedure is called multiple times without an intervening commit or close result sets**

External Stored Procedure

Native SQL procedure



Not zIIP-eligible except stored proc call,

result set, and commit processing

zIIP-eligible

LOB

- Reorg LOB to reclaim space (CM)
  - In V8, LOB Reorg did not reclaim free space, leading often to a bigger table space as a result of Reorg.
  - In V9, free space is reclaimed. A general recommendation is to Reorg when the free space is bigger than the used space; i.e. SYSTABLESPACESTATS.SPACE>2*DATASIZE/1024 in Real Time Statistics.
  - Another LOB Reorg indicator is REORGDISORGLOB/TOTALROWS>50% in Real Time Statistics to keep pages of a given LOB together for an efficient read/write performance.

LOB - continued

- LOB read/write I/O performance improvement (CM)
  - From doubling of prefetch and deferred write quantity
  - From 8 times increase in preformat quantity
  - Described in Buffer Manager enhancement section of “DB2 9 for z/OS System Performance” session 1228

LOB - continued

- LOB lock avoidance (NFM)
  - Up to 100% reduction in Lock and Unlock requests in Fetch
  - One measurement with SAP optimized LOB streaming
    - -67% IRLM requests
    - -26% class2 accounting CPU time
    - -14% elapsed time
- LOB/XML flow optimization by size (NFM)
  - V9 DRDA LOB handling instead of SAP optimized LOB handling
    - Additional 11% elapsed time and 2% CPU time improvement
- LOB insert with -14% CPU time (CM)
  - Despite +50% IRLM requests to avoid lock escalation

NOTES

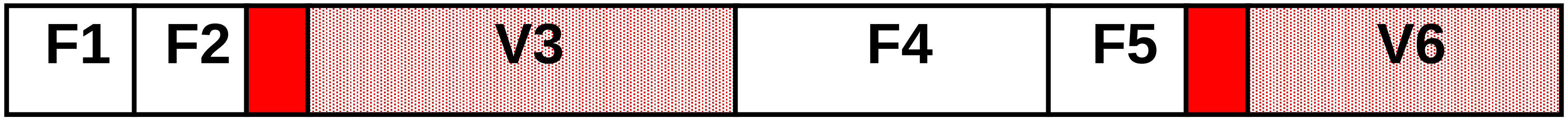
- Also file reference variable for Insert/Select of LOB data to/from sequential file (NFM)

- LOB CPU time reduction also from DDF/DBM1 shared memory above 2GB and more efficient space search (CM)

- Recent LOB performance improvement apars
  - V7/V8 PK10278 3/06, PK22910 4/06 Improved LOB support using file reference variable in Load/Reorg
  - V8 PK22887 6/06 Fix increasing LOB update time as more LOBs updated without commit
  - V8 PK25241 9/06 Improved LOB insert/update performance by reducing exhaustive spacemap scan

Varchar Performance Improvement (NFM)

- Remember the tuning recommendation for rows with many columns with any varchar present?



  – V9 DB2 internally executes this recommendation and more

  – V8 and prior: any column after the first varchar requires offset calculation
  – V9: all columns directly accessible

NOTES

- 2 times or more improvement observed when many rows with many varchars are scanned and/or fetched using many predicates

- <5% improvement for a typical online transaction

- Reorg and Load Replace override Keepdictionary when migrating first time to V9 with data compression of variable-length rows
  - Use of old compression dictionary results in ineffective compression ratio

- Improvement applies for vargraphic also
  - No difference if no varchar nor vargraphic

DGTT – Declared Global Temp Table

- Workfile and temp database are now stored in workfile database (CM)
- PK43765 6/07 to reduce LC24 DGTT prefetch latch contention
  - Prefetch quantity increased from 8 to segsize with 16 default and 64 maximum
- 30 to 60% faster for SELECT COUNT
  - Bigger prefetch quantity and 32K workfile
- 5 to 15% faster and less CPU for INSERT
  - Bigger preformat quantity and asynchronous preformat in V9 but not V8

Additional V9 Performance-Sensitive Apars

- **PK41878 4/07 Slow query on a partitioned table space that uses Data Partitioned Secondary Index**

➲ **PK42008 4/07 Excessive locking on each and every partition even with lock avoidance**

- **PK41323 5/07 Improve performance of implicit Create/Drop of table space**

- **PK46972 8/07 System hang or deadlock when degree any with DPSI**

Reference

- Redbooks at www.redbooks.ibm.com
  - DB2 9 for z/OS Technical Overview SG24-7330
  - DB2 9 for z/OS Performance Topics SG24-7473

- DB2 for z/OS home page at www.ibm.com/software/db2zos
  - E-support (presentations and papers) at www.ibm.com/software/db2zos/support.html