2010 IDUG North America

DB2 10 for z/OS Performance Preview

Akiko Hoshikawa
IBM Silicon Valley Lab,
akiko@us.ibm.com

Session Code: <A03>

May 11, 2010 : 3pm
Platform: <DB2 for z/OS>

Disclaimer

DB2 10 Performance Preview

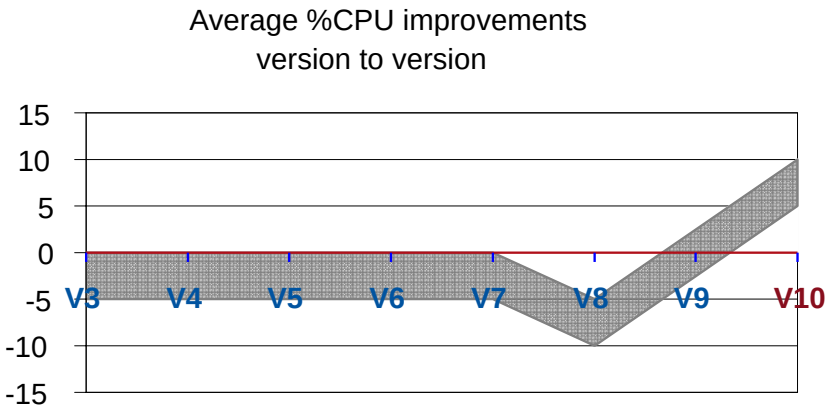
- Abstract

  This session offers a preliminary look at performance impact of DB2 X for z/OS, especially performance scalability of online transactions and CPU and elapsed time reduction based on early measurements done in Silicon Valley Lab.

- Agenda
  - DB2 10 for z/OS Performance goal and expectation
  - Scalability and Buffer pool enhancement
  - Insert improvement
  - Fetch / Select improvement
  - LOB, XML and SQL Procedure performance
  - JDBC and DDF performance

DB2 10 Performance Objective

Historical goal of <5 % version-to-version performance regression
Goal of 5% -10% performance improvement for DB2 10

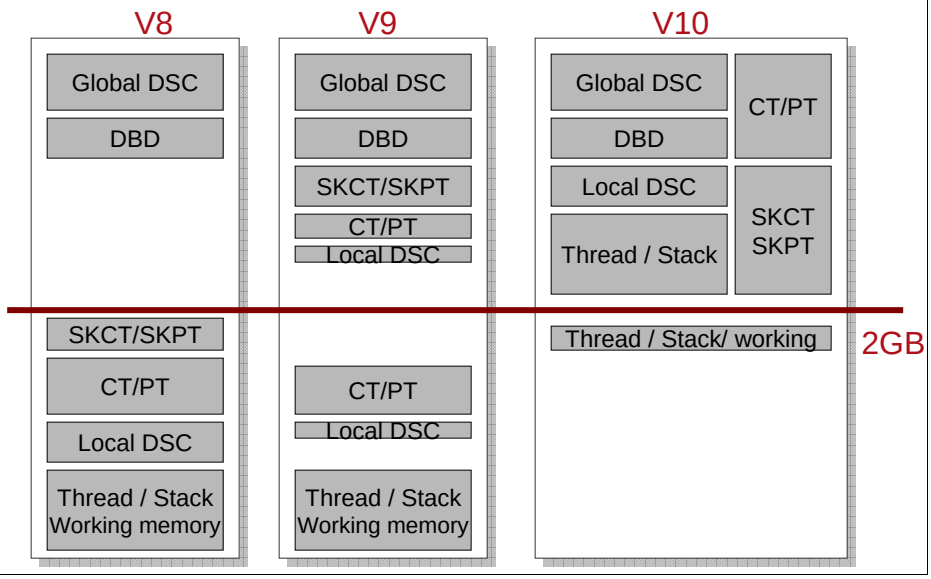


DB2 10 Performance Expectation

- Most of workloads : 0-10% CPU reduction after REBIND packages
  - Higher improvement with workload with scalability issues in V8/V9 or accessed thru DRDA

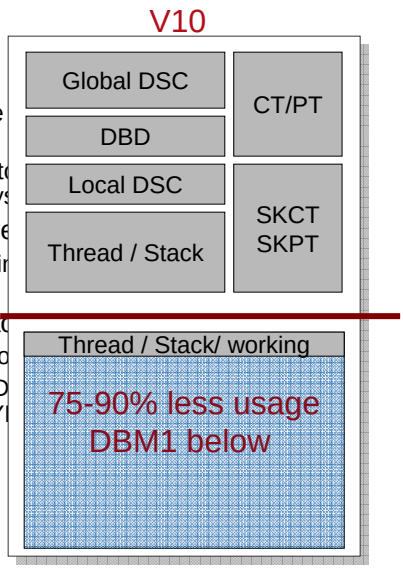
- Workload using native SQL procedures : up to 20% CPU reduction after DROP/CREATE or REGENERATE the procedures
- Concurrent sequential insert :5-40% CPU reduction depends on table space type
- Query : up to 20% CPU reduction without access path change
  - Higher Improvement with positive access path change

DBM1 Virtual Storage Constraint Relief

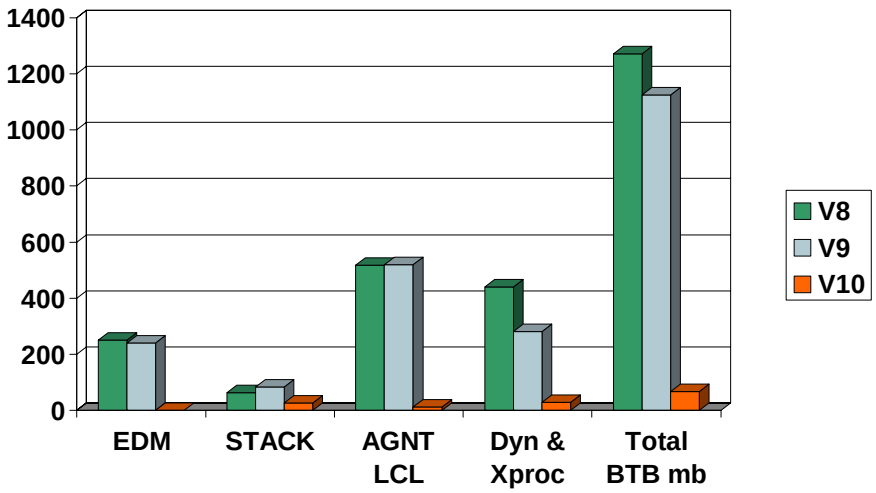


DBM1 VSCR

- DBM1 below 2GB
  - 70-90% less usage compared to V9
  - Some of working st xproc storage) stays
- Larger number of thre
  - Possible data sharin consolidation
- Improve CPU with st
  - More release deallo
  - Larger MAXKEEPD KEEPDYNAMIC=Y



Preliminary Measurement: 320 'fat' threads BTB storage

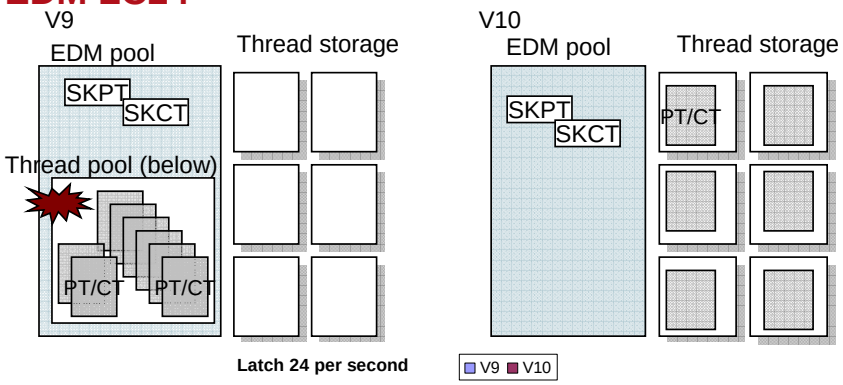


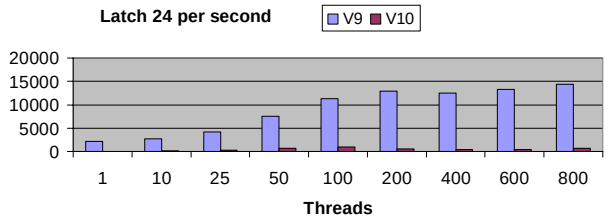
Performance Scalability - DB2 Latches (CM)

Most of DB2 latches from 64 cp scalability evaluation will have a relief

- LC12 : Global Transaction ID serialization
- LC14 : Buffer Manager serialization
- LC19 : Log write in both data sharing and non data sharing
- LC24 : EDM thread storage serialization (Latch 24)
- LC24 : Buffer Manager serialization (Latch 56)
- LC25 : EDM hash serialization
- LC27 : WLM serialization latch for stored proc/UDF
- LC32 : Storage Manager serialization
- IRLM : IRLM hash contention
- CML : z/OS Cross Memory Local suspend lock
- UTSERIAL : Utility serialization lock for SYSLGRNG *need to be in NFM

EDM LC24


V9
EDM pool
Thread storage
SKPT
SKCT
Thread pool (below)
PT/CT
PT/CT
V10
EDM pool
Thread storage
SKPT
SKCT
PT/CT


Latch 24 per second
V9
V10
20000
15000
10000
5000
0
1
10
25
50
100
200
400
600
800
Threads

Performance Scalability - H/W synergy

- Exploitation of z10 features
  - CPU improvement using z10 prefetch instructions
  - Large fixed page frames for buffer pool
    - Buffer pools with PGFIX=YES
    - Define IEASYSxx LFAREA 1MB page frames
    - Reduction of hit miss in TLB (translation lookaside buffer)
  - Observed 1-4% CPU reduction
- Avoidance of sequential BP scan in page set p-lock negotiation
  - Performance impact with large buffer pools in data sharing
- In memory buffer pool with large real
  - DB2 managed in memory buffer pool
    - PGSTEAL = NONE
    - Pre-load the data at the first open or at ALTER BPOOL
    - Avoid unnecessary prefetch request (similar to VPSEQT=0)
    - Avoid LRU maintenance -> no LRU latch (LC14)

Preliminary Measurements

- IBM Relational Warehouse Workload (IRWW) Data Sharing
- V9 NFM REBIND with PLANMGMT(EXTENDED)
- V9 NFM -> V10 CM without REBIND
  - Measured 3.7% CPU reduction from V9
- V10 CM REBIND with APREUSE (YES)
  - Measured 7.4% CPU reduction from V9
- V10 NFM
  - Measured same 7.4% CPU reduction from V9
- V10 NFM with RELEASE (DEALLOCATE)
  - Measured additional 10% CPU reduction from V10 NFM RELEASE(COMMIT)

Insert Performance Improvement

V9
- Large index pages
- Asymmetric index split
- Data sharing Log latch contention and LRSN spin loop reduction
- More index look aside
- Support APPEND option
- RTS LASTUSED support
- Remove log force write at new page (Segmented and UTS) via PK83735

V10 CM
- Space search improvement
- Index I/O parallelism
- Log latch contention reduction and faster commit process
- Additional index look aside

V10 NFM
- INCLUDE index
- Support Member Cluster in UTS
- Complete LRSN spin avoidance

General Insert Enhancements

- Log latch reduction in both data sharing and non data sharing
  - Complete LRSN spin avoidance (NFM)
- New IFCID 359 to record index split
- Eliminate Mass Delete Locks from UTS
- Referential integrity check performance
  - Sequential detection and index look aside for RI
  - Avoid RI check for each insert of a child under the same parent
- Performance improvement for sequential inserts into the middle of a cluster index
  - Significant space search improvement in sequential insert
- Member Cluster option now available with UTS (PBG/PBR)

Universal Table Space (UTS) – Member Cluster (NFM)

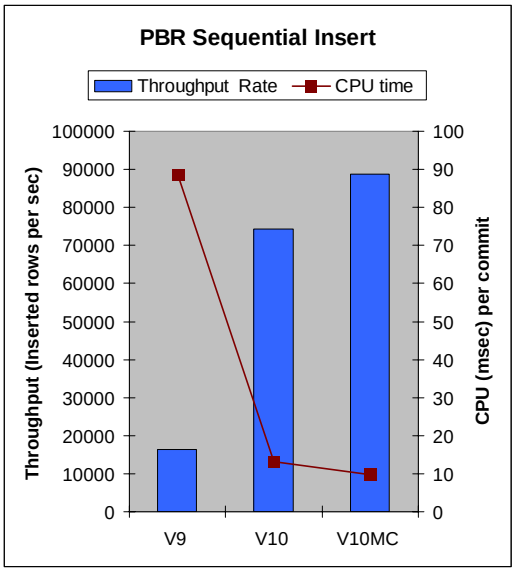
- Member Cluster option in create table space
  - Assigns a set of pages and associated space map page to each member
  - Remove the “hot spots” in concurrent sequential insert in data sharing
  - It does not maintain data cluster during the INSERT
  - Data cluster needs to be restored via REORG
  - Each space map contains 10 segments
- Altering to MEMBER CLUSTER

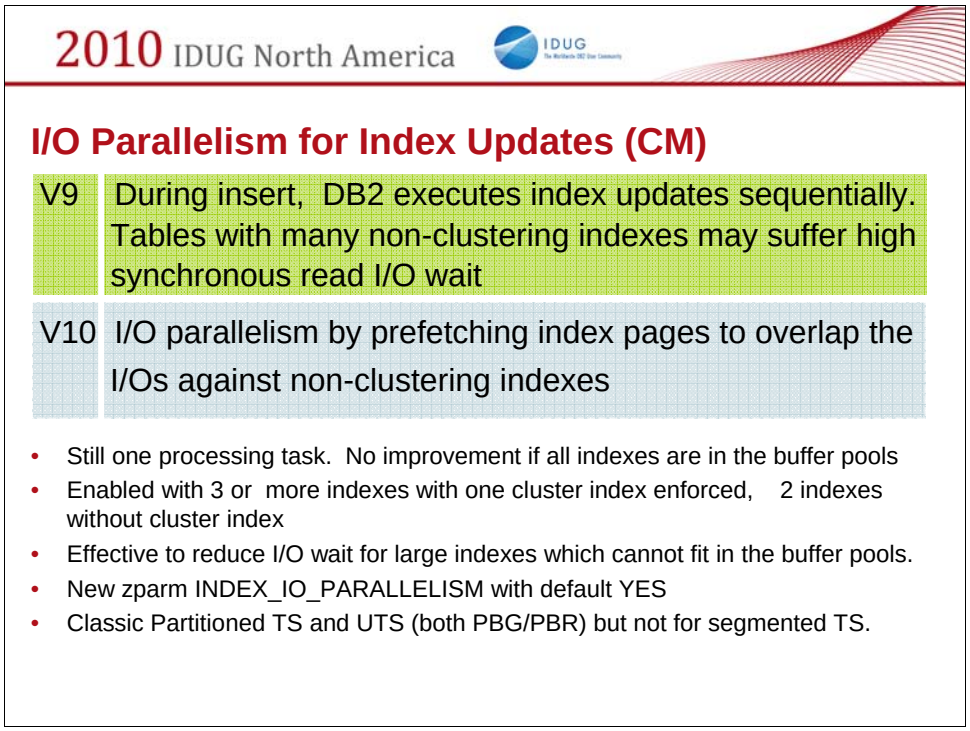
  ALTER TABLESPACE MyTableSp

  MEMBER CLUSTER YES/NO;
  - REORG/LOAD on the table space level to materialize the pending alter
  - RECOVER needs image copy taken from REORG that materialized the pending ALTER

Sequential Inserts

- Optimize when index manager picks the candidate RID during sequential insert
- Member Cluster to distribute space map/data page
- Result: Higher chance to find the space and avoiding a space search
- Less page latch contention with MC
- Test case: Sequential key insert into 3 UTS Partitioned By Range TSs from jdbc 240 clients in 2way data sharing. Multi row insert.





NOTE:

Preliminary measurements

For a table with 6 indexes and 2000 inserts, elapsed time improvements of up to 50%

Some CPU overhead for scheduling prefetch

zIIP offload for prefetch read SRB time

Buffer pool hit ratio affects the measured elapsed time improvement.

No elapsed time improvement if all indexes are in the buffer pool

Additional Non-key Columns in a unique index (NFM)

V9	Multiple indexes per table An index is used to enforce uniqueness constraint. Additional indexes are necessary to achieve index only access on columns not part of the unique constraint during queries. Higher Insert / Delete CPU time, Increased storage requirements
V10	Additional Non-key Columns in an unique indexes Reduce index maintenance cost during insert, DASD space saving

Preliminary measurement

2 Index vs 1 index with INCLUDE columns shows 30% cpu reduction in insert with same query performance using the indexes.

Additional Non-key columns in a unique index

- V9 definition
  CREATE UNIQUE INDEX i1 ON t1(c1,c2,c3)
  CREATE INDEX i2 ON t1(c1,c2,c3,c4,c5)

- Possible V10 definition
  CREATE UNIQUE INDEX i1 ON t1(c1,c2,c3) INCLUDE (c4,c5)
  or
  ALTER INDEX i1 ADD INCLUDE (c4,c5) and DROP INDEX i2
  (note: i1 becomes rebuild pending status)

- The following restrictions will apply:
  - INCLUDE columns are not allowed in non-unique indexes
  - Indexes on Expression will not support INCLUDE columns
  - Indexes with INCLUDEd columns can not have additional unique columns ALTER ADDed to the index

More on TS Management and UTS

- Eliminate Mass Delete Locks from UTS
- Default non-partitioned table space stays as segmented TS
- Default partitioned table space is Partition by Range UTS
  - SEGSIZE 0 needs to be defined for traditional partitioned table space
- Default UTS SEGSIZE is changed to 32
- TRACKMODE zparm (IMPTKMOD) to specify default
- New functions on UTS
  - Hash access
  - Inline LOB
  - Access currently commited data, etc.

Select/Fetch Performance Improvement

V9	Sort performance improvement , In memory workfile/Sparse index Index on Expression Many access path related improvements • Plan Stability for static SQL statements • Histgram stats, etc.
V10	CPU reduction on index predicate evaluation Better performance using a disorganized index Row Level Sequential Detection Group by using Hash, More in memory workfile usage Sproc improvement by removing column size limitation Dynamic statement cache support for literal constants Many access path related enhancements • Plan stability for both static and dynamic statements • Parallelism improvement • IN list access improvement • Auto stats…and more

CPU reduction in Predicate Evaluation (CM)

- Optimize in index predicate evaluation process
  - Applicable in any workload but query with many predicate shows higher improvement
- Performance improvement
  - Preliminary measurements shows average 20% CPU reduction (1% thru 50%) from TPC-H like workload using 150 home made queries.

Improvement in using Disorganized Index (CM)

- Index scan using disorganized index causes high sync I/O wait
- Disorganized index detection at execution
- Use List Prefetch on index leaf pages with range scan
  - Reduce Synchronous I/O waits for queries accessing disorganized indexes.
  - Reduce the need of REORG Index
  - Throughput improvement in Reorg, Runstats, Check Index
  - Limited to forward index scan
- Preliminary Performance results
  - Observed 2 to 6 times faster with simple SQL statements with small key size using list prefetch compared to Sync I/Os

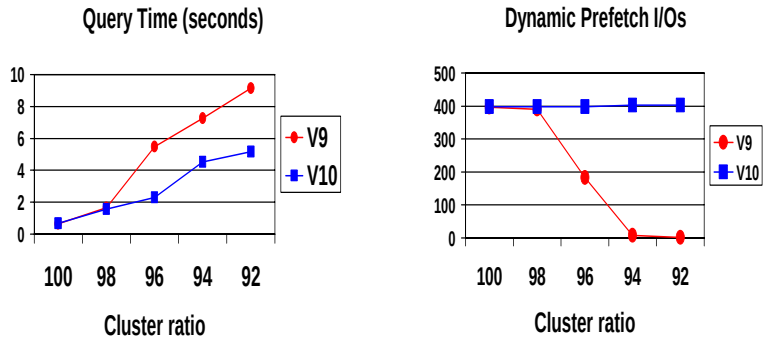
Row Level Sequential Detection (CM)

- Problem : Dynamic prefetch sequential works poorly when the number of rows per page is large
- Solution: Row Level Sequential Detection (RLSD)
  - Count rows, not pages to track the sequential detection
- Since DB2 10 will trigger prefetch more quickly, it will use progressive prefetch quantity:
  - For example, with 4K pages the first prefetch I/O reads 8 pages, then 16 pages, then all subsequent I/Os will prefetch 32 pages (as today).
  - Also applies to indexes.

Index—>Data Range Scan

Row size = 49 bytes, page size = 4K (81 rows per page)

Read 10% of the rows in key sequential order

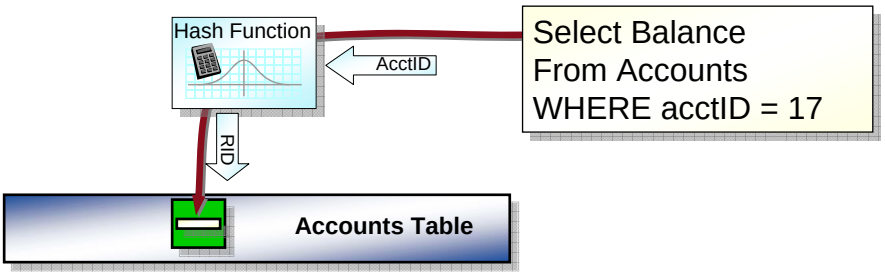


- Row level sequential detection (RLSD) preserves good sequential performance for the clustered pages

Test	Cluster ratio	Cardinality	NPAGES
1	100%	20,000,000	253167
2	98%	20,200,000	256024
3	96%	20,400,000	258882
4	94%	20,600,000	261740
5	92%	20,800,000	264598

Hash Access (NFM)

Hash Access (NFM)



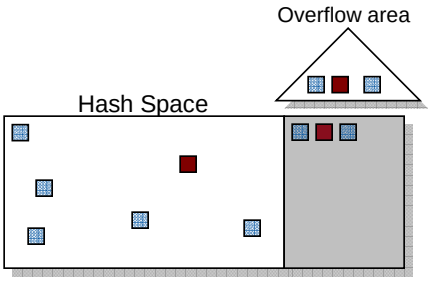
… ORGANIZE BY HASH UNIQUE (AcctID)

- A hash key is defined on the table
  - The key must be unique – no duplicates are allowed
  - A hashed table can have only a single hash key
- Benefits the singleton Select/Update or a row fetch with **equal predicates on all hash key columns**
  - Better with larger tables with small rows

Using Hash Access

- CREATE TABLE or
  ALTER TABLE table ADD ORGANIZATION SET HASH SPACE x G;
  - UTS
  - Object will be Advisory Reorg Pending
  - No new inserts are allowed (can delete/update)
  - CREATE will format entire hash space
- REORG AUTOESTSPACE (YES|NO)
  - Materialize HASH organization, build overflow index
- REBIND applications
  - New hash access “H”
- Monitor performance and Real Time Statistics and potentially drop original unique index

Hash Access and Hash Space

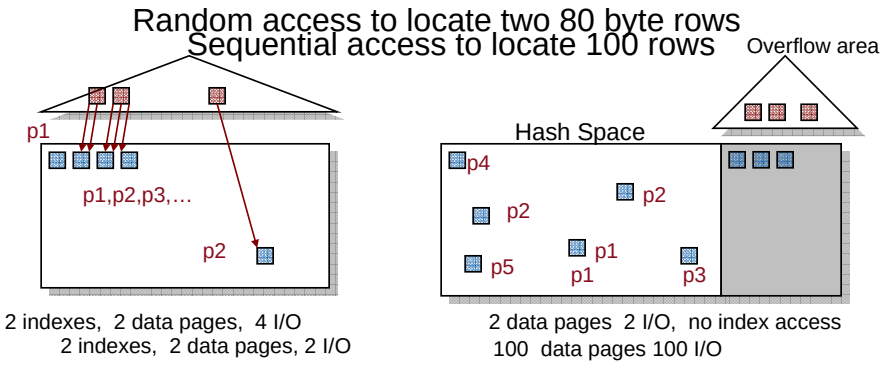


- From base hash space - 1 I/O
- If overflow - 3 I/O

- Need enough space to maintain performance
  - REORG with **AUTOESTSPACE YES**
- Real Time Statistics
  - Number of Overflows
    - **TOTALENTRIES**
    - Overflow Index only has keys for Overflows
  - Percentage of Overflows
    - **TOTALENTRIES / TOTALROWS**
  - Average Row Size
    - **DATASIZE / TOTALROWS**

Hash Access – Candidate Tables

- Large tables with random access thru an unique index
- Potential degradation if used in sequential access



Hash Access Summary

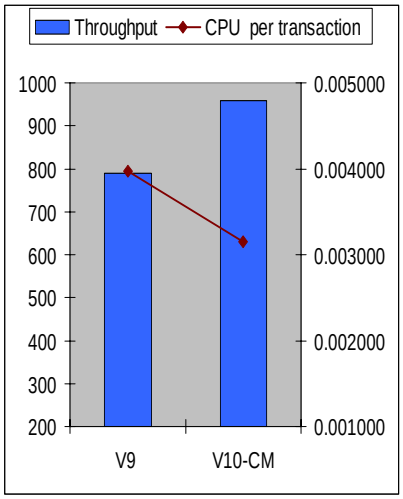
- Performance benefit :
  - Preliminary measurement up to 30% DB2 CPU reduction
  - Faster Probes on Hash Key
    - Fewer page access, I/Os
  - Savings in Index Maintenance
    - Faster Insertion and Deletion
  - Possible reduction in Hotspots
    - Rows are randomly distributed
- Performance concern :
  - Possible INCREASE in I/O or BP space in some cases
    - If small 'active' working set, may need more Getpages, I/Os
  - Not for sequential Insert (No Member Cluster support)
  - Performance will slowly degrade as space becomes over-utilized
    - Use PCTFREE to allow for growth
    - Monitor Utilization of space and numbers of overflows

SQLPL, JDBC and DDF

SQL Procedure Performance (CM)

V9	Introduced native SQL Procedure Improvement by executing procedures in DBM1 instead of WLM address space
V10	Native SQL Procedures Further performance optimization Specific CPU reduction in commonly used area • Section load avoidance with SET statements with function • Pathlength reduction in IF statement • Optimization in SELECT x from SYSDUMMY1 Chained SET statement support (NFM)

Preliminary Measurements – SQLPL (CM)



- OLTP using SQLPL
  - 20% CPU reduction with V10 CM
  - 89% DBM1 Below the Bar usage reduction
  - 5% resp time improvement due to latch contention relief

Local JDBC and ODBC Application Performance

- **Local Java and ODBC applications did not always perform faster compared to the same application called remotely**
  - DDF optimized processing with DBM1 that was not available to local ODBC and JDBC application.
  - zIIP offload significantly reduced chargeable CP consumption
- Open support of DDF optimization in DBM1 to local JCC type 2 and ODBC z/OS driver
  - Limited block fetch
  - LOB progressive streaming
  - Implicit CLOSE
- Expect significant performance improvement for applications with
  - Queries that return more than 1 row
  - Queries that return LOBs

High Performance DBATs

- Re-introducing RELEASE(DEALLOCATE) in distributed packages
  - Could not break in to do DDL, BIND
  - V6 PQ63185 to disable RELEASE(DEALLOACTE) on DRDA DBATs
- High Performance DBATs reduce CPU consumption by
  - RELEASE(DEALLOCATE) to avoid repeated package allocation/deallocation
  - Avoids processing to go inactive and then back to active
  - Bigger CPU reduction for short transactions
- Using High Performance DBATs
  - Stay active if there is at least one RELEASE(DEALLOCATE) package exists
  - Connections will turn inactive after 200 times (not changeable) to free up DBAT
  - Normal idle thread time-out detection will be applied to these DBATs.
  - Good match with JCC packages
  - Not for KEEPDYNAMIC YES users

High Performance DBAT…

- New -MODIFY DDF PKGREL command
  - To alter DDF's inactive connection processing (CMSTATS=INACTIVE)
  - Options
    - PKGREL(BNDOPT) honors package bind option
    - PKGREL(COMMIT) forces package bind option RELEASE(COMMIT)
      - Same as V9 inactive connection behavior
      - Will allow BIND and DDL to run concurrently with distributed work
    - PKGREL(DEALLOC) forces package bind option RELEASE(DEALLOCATE)
      - Provides better performance behavior
      - BIND and DDL can not break in when concurrent distributed work runs

LOB and XML

Inline LOBs (NFM)

- CREATE or ATLTER TABLE INLINE LENGTH on UTS
  - INLINE to base table up to 32K bytes
- Completely Inline LOBs
  - Reduce DASD space
    - No more one LOB per page, Compression
  - CPU and I/O saving
    - Avoid LOB aux indexes overhead
    - Small inline LOBs uses 5-10% more than VARCHAR
  - Potential impact on SQLs which does not touch LOBs
- Split LOBs
  - A part of LOB resides in base and other part in LOB TS
  - Incur the cost of both inline and out of line
  - Index on expression can be used for INLINE portion

XML performance improvement

- Significant Performance improvement in V9 service stream
- DB2 10 performance improvement
  - Binary XML support
    - Avoid the cost of XML parsing during insert
    - Reduce the XML size
    - Measured 10-30% CPU and elapsed time improvement
  - Schema Validation in engine
    - No more UDF call for validation
    - Utilize XML System Service Parser
      - 100% zIIP / zAAP eligible for validation parser cost
  - XML Update
    - No more full document replace

DB2 10 Monitoring Enhancements and Changes

1. New Monitor class 29 for statement detail level monitoring
   - IFCID 318, 400, 316, 401
2. Index split IFCID 359
3. Separate accounting with lock and latch suspension in class 3
4. Package LAST USED
5. IFCID 225 for DIST address space
6. Accounting : zIIP SECP values
   - Possible redirection value is no longer supported, always zero in DB2 X
   - SE CPU (actual offloaded CPU time) continues to be available
7. Statistics trace interval
   - Always 1 minute interval in V10 no matter what you use in STATIME

DB2 10 Performance Expectation

- Most of workloads : 0-10% CPU reduction after REBIND packages
- Higher improvement if hit the sweet spots
  - Workloads with scalability issues (DBM1 storage, DB2 latches)
  - SQL PL
  - DDF using high perf DBAT
  - Simple queries evaluating multiple rows thru indexes
  - Concurrent sequential insert
  - Small LOBs
- DBM1 Storage constraint relief
  - 80-90% of DBM1 Below the Bar storage reduction
    - More concurrent threads
    - CPU saving using more storage

Thank you !

Session Code: A03
DB2 10 for z/OS Performance Preview

Akiko Hoshikawa (akiko@us.ibm.com)