IBM System z Technology Summit

Gain insight into DB2 9 and DB2 10 for z/OS performance updates and save costs

John Iczkovits – DB2 ATS

iczkovit@us.ibm.com

April 14, 2011

Miami

Disclaimer: Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The Information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance Disclaimer

This document contains performance information based on measurements done in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the numbers stated here.

DB2 10 for z/OS: Out-of-the-Box Savings

CPU reductions for transactions, queries, and batch

- Out-of-the-box CPU reductions of 5-10% for traditional workloads
- Up to additional 10% CPU savings using new functions or avoiding constraints
- Out-of-the box CPU reductions of up to 20% for new workloads

Scales with less complexity and cost

- 5-10x more concurrent users – up to 20,000 per subsystem
- Significant scale-up capabilities in addition to existing scale-out support
- Consolidate to fewer LPARs and subsystems

Improved operational efficiencies and lower administration cost

- Automatic diagnostics, tuning, and compression

Even better performance

- Elapsed time improvement for small LOBS and Complex Queries, and HASH access

Objective

This session offers what we have done in DB2 9 and 10 to improve the performance and provides a little more detail than “it depends..” for expected improvement

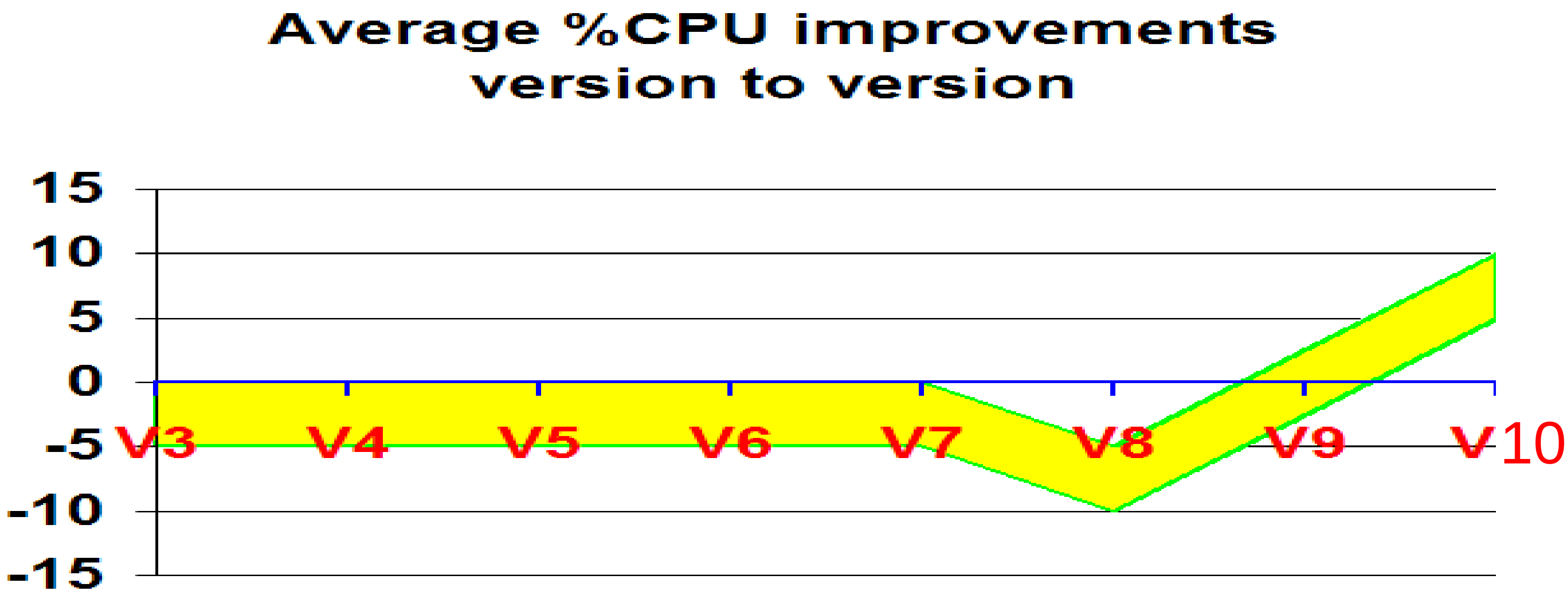
Agenda : Why you may see improvement ?

1. System level performance without changing schema or applications
2. Application level performance
   - Without changing applications nor schema
   - With schema or application change
3. Monitoring improvement
4. Summary

DB2 10 Performance

Constant Cost Pressures

- Performance improvements in key workloads: Transactions, Batch, Insert, …
- Lower CPU usage for large & small DB2 subsystems
- DB2 10 Most customers 5% - 10% CPU reduction out of the box with rebind
- Some workloads and customer situations can reduce CPU time up to 20%



1. System Level Performance and Scalability

   Migration performance without changing schema, applications

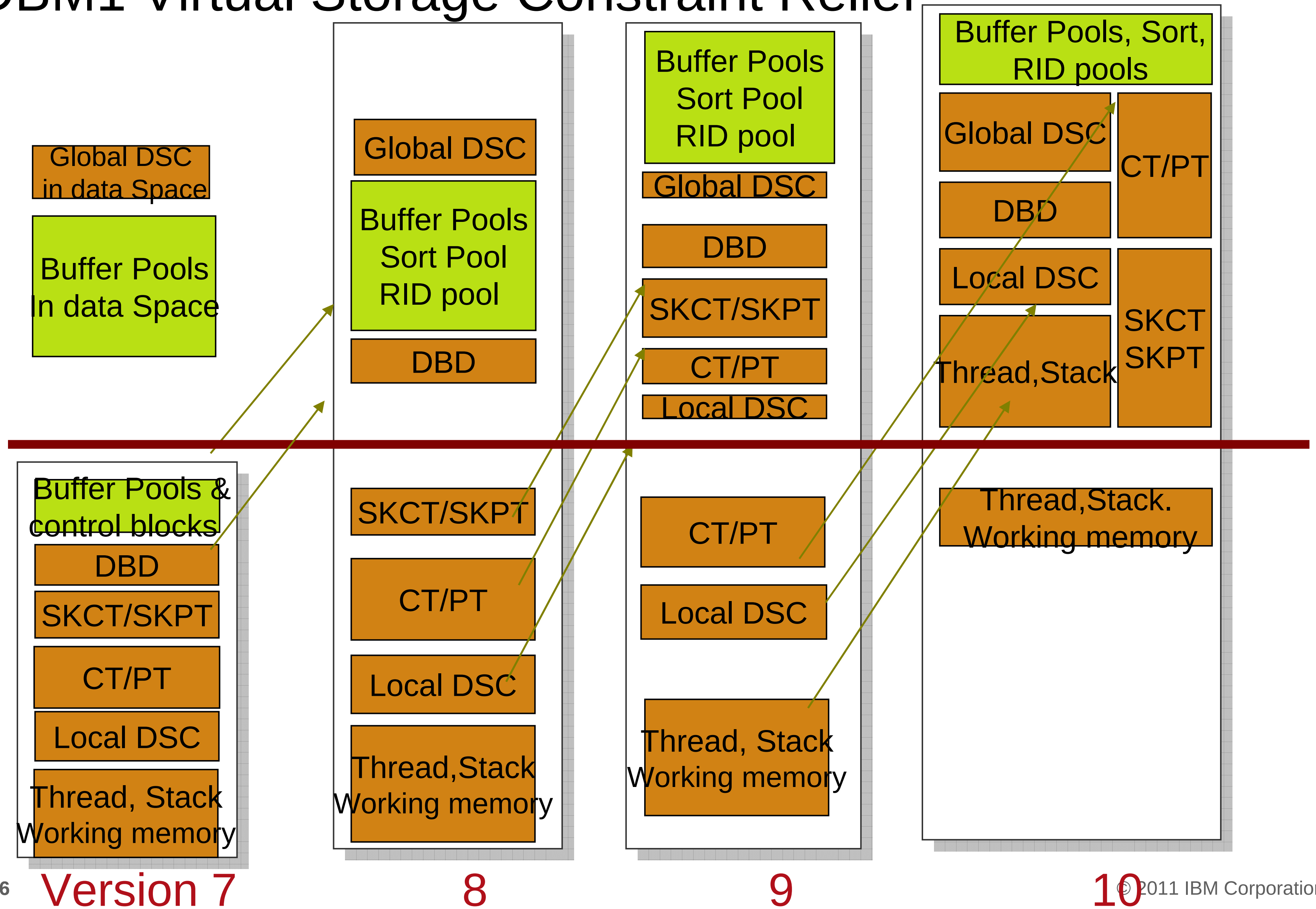
   - **Virtual Storage Constraint Relief**
   - **Reduce CPU by utilizing more real storage**
   - **Latch Reduction**
   - **System z Synergy and Buffer Pool**
   - **Migration Story**
   - **Utilities and zIIP support**

2. Application Level Performance

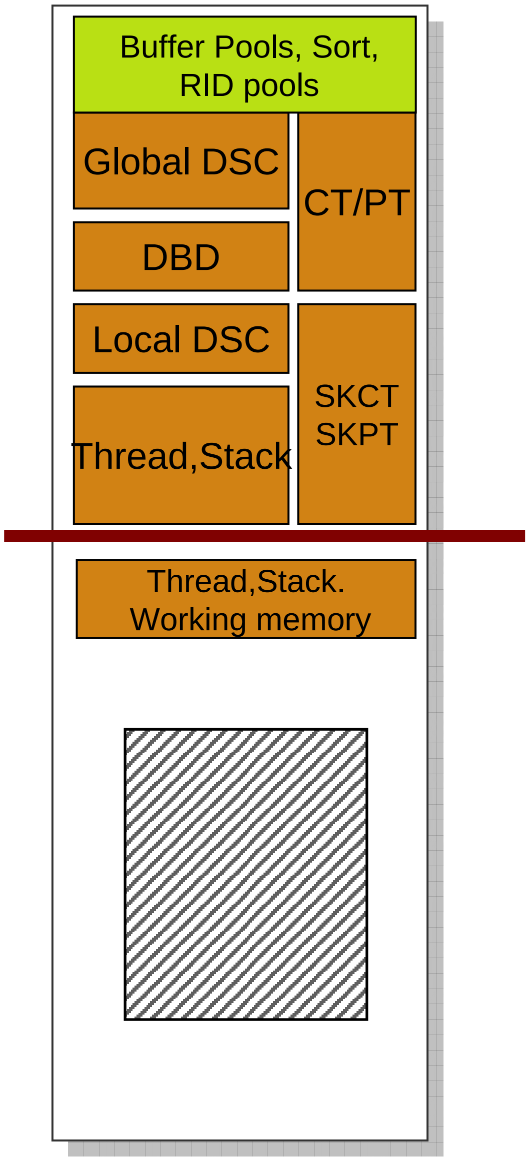
3. Monitoring Support

4. Summary

DBM1 Virtual Storage Constraint Relief



DB2 10



Relief in DBM1 Below The Bar

- **DBM1 below 2GB**
  - EDM storage - All above
  - Thread + Stack - 70-90% less usage in DB2 10 compared to DB2 9
    - xPROC (SPROC, IPROC, UPROC, etc) loaded in below the 2GB bar
    - Built in BIND time, shared at runtime
- **More number of threads**
- **Reduce CPU time by expense of storage**
  - More thread reuse to avoid allocate/deallocate
  - Wider usage for bind option RELEASE(DEALLOCATE)
  - High Performance DBATs
  - Larger MAXKEEPD values for KEEPDYNAMIC=YES users to avoid short prepare

High Performance DBATs

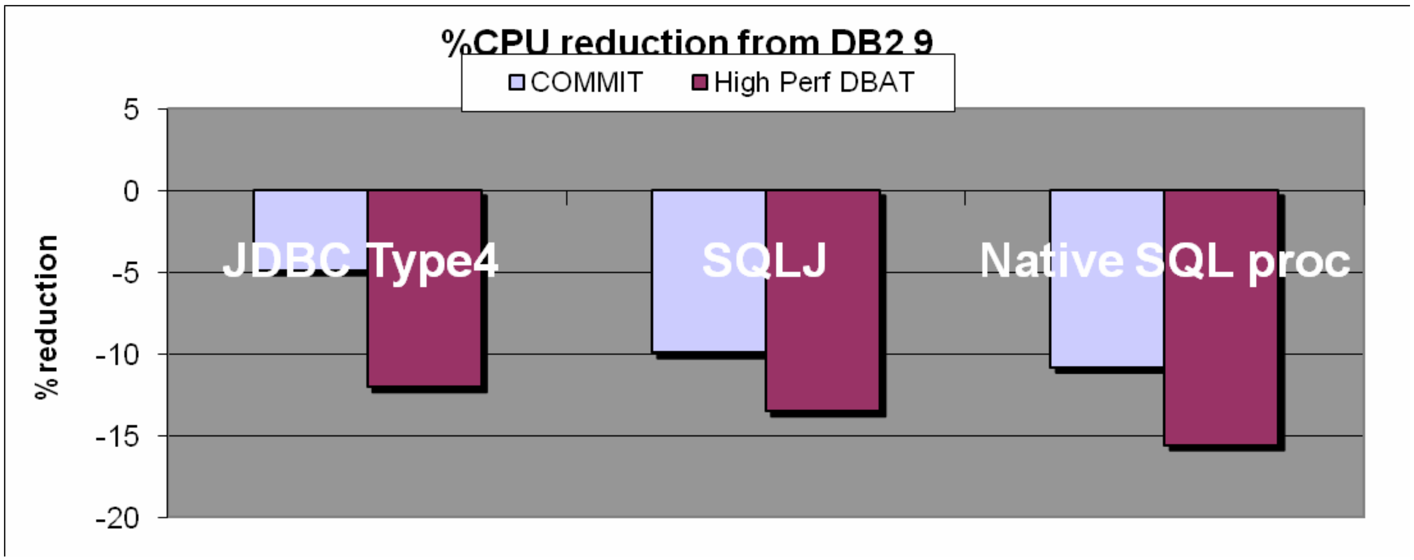
- **Re-introducing RELEASE(DEALLOCATE) in distributed packages**
  - Could not break in to do system maintenance - Utility or DDL
  - V6 PQ63185 to disable RELEASE(DEALLOACTE) on DRDA DBATs
- **High Performance DBATs reduce CPU consumption by**
  - RELEASE(DEALLOCATE) to avoid repeated package allocation/deallocation
  - Avoids processing to go inactive and then back to active
  - Bigger CPU reduction for short transactions commit often
- **Using High Performance DBATs**
  - Stay active if there is at least one RELEASE(DEALLOCATE) package exists
  - Connections will turn inactive after 200 commits to free up resources
  - Normal idle thread time-out detection will be applied to these DBATs.
  - Good match with JCC packages (handful packages)
  - No benefit and not support for ACTIVE threads (CMSTATS=ACTIVE)
  - No benefit for KEEPDYNAMIC YES users

Enable High Performance DBATs

- **Two steps to enable High Performance DBAT**
  1. REBIND with RELEASE(DEALLOCATE)
     - Default BIND option in DB2 client driver will be RELEASE (DEALLOCATE) for the client matching with DB2 10 (DB2 connect and JCC 9.7 FP3a)
  2. Then command –MODIFY DDF PKGREL (BNDOPT)
     - No more support on PKGREL(DEALLOC) which was available early beta code
     - -DISPLAY DDF shows the option currently used
- **To disable,**
  – -MODIFY DDF PKGREL (COMMIT) to overlaid BNDOPT option
- **To monitor,**
  – Statistics GLOBAL DDF activity report

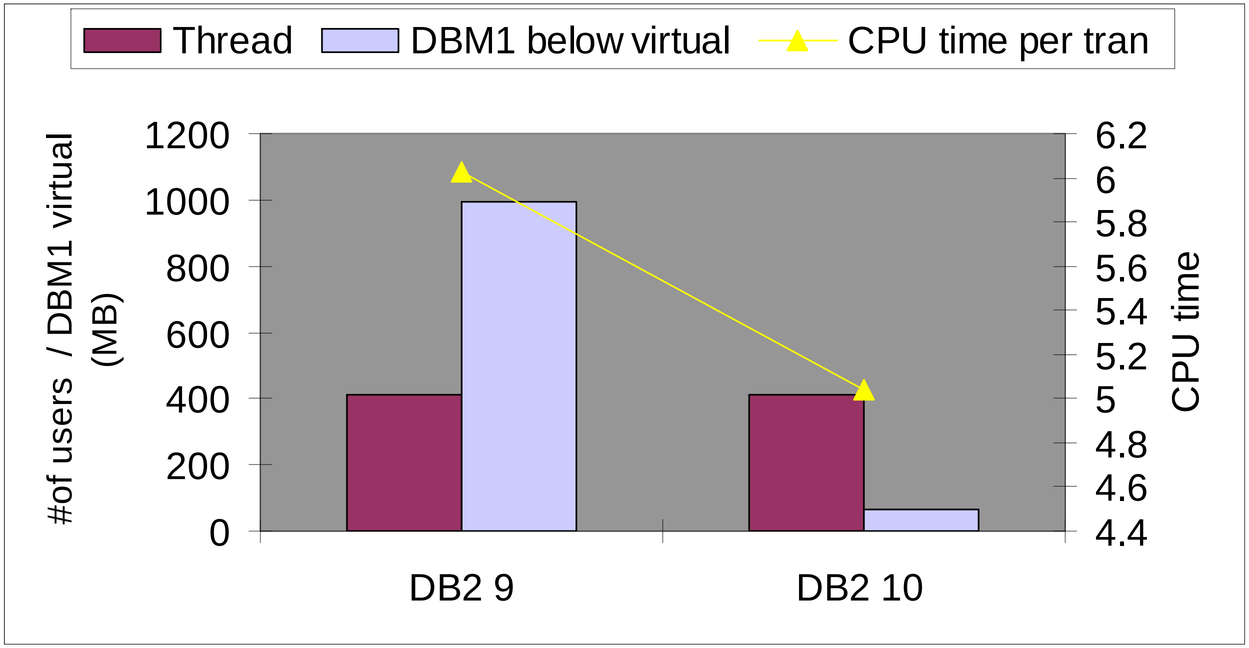
```
GLOBAL DDF ACTIVITY            QUANTITY
---------------------------   ---------
CUR ACTIVE DBATS-BND DEALLC       5.39
HWM ACTIVE DBATS-BND DEALLC      10.00
```

High Performance DBATs and CPU Reduction



- **Workload: distributed IRWW workloads with different interfaces show 4-7% further CPU reduction using High Performance DBATs**
- **37% CPU reduction with simple transactions with AutoCommit ON**

Virtual Storage Reduction from SAP Workload



- **Same 412 concurrent threads**
- **Virtual storage below the bar 997 MB with DB2 9 -> 63 MB in DB2 10**
- **No significant increase in real storage**
- **16% CPU time reduction**

DBM1 VSCR Monitoring

- **More focus on**
  - **Real storage usage**
  - **Common storage (ECSA and ESQA) usage**
- **New statistics in IFCID 225 reports**
  - **DBM1 address space : virtual below and above, real, #of user threads**
  - DIST address space : virtual below and above, real
  - Common and storage usage

```
DBM1 AND MVS STORAGE BELOW 2 GB                     QUANTITY
-------------------------------------------   ------------------
TOTAL NUMBER OF ACTIVE USER THREADS                  2694.28
  NUMBER OF ALLIED THREADS                            386.00
  NUMBER OF ACTIVE DBATS                             2275.06
  NUMBER OF POOLED DBATS                               33.21

REAL AND AUXILIARY STORAGE FOR DBM1                 QUANTITY
 -------------------------------------------   ------------------
 REAL STORAGE IN USE                     (MB)         5396.07
   31 BIT IN USE                         (MB)          289.45
   64 BIT IN USE                         (MB)         5106.62
 HWM 64 BIT REAL STORAGE IN USE          (MB)         5106.64
```

Common Storage and Real Storage Usage

- **DB2 usage for ESQA**
  - SRB process
    - z/OS APAR OA33106 Reduction of suspend SRB process
- **DB2 usage for ECSA**
  - Common control blocks and distributed threads
    - Many are moved to HCSA or private storage above
  - Continue to monitor using IFCID225
    - QW0225GC + FC + VC
- **DB2 usage for Real Storage**
    - Private and shared storage
      - Buffer pool is 64 bit private storage
      - Majority of thread / stack storage is in 64 bit shared storage
    - Reporting will be added in IFCID 225

Performance Scalability - DB2 Latches (CM)

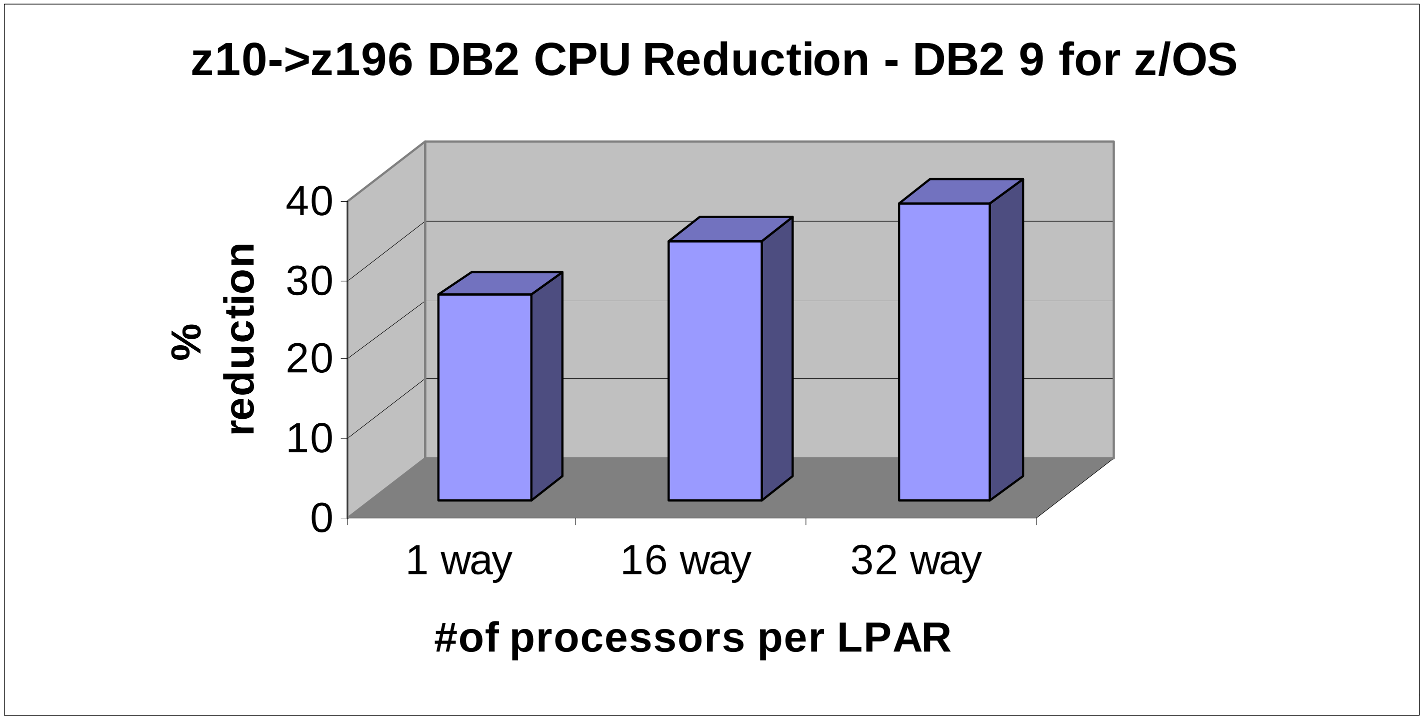
- **Faster process on latch suspend/resume**
- **Most of known DB2 latches are addressed in DB2 10**
  - LC12 : Global Transaction ID serialization
  - LC14 : Buffer Manager serialization
  - LC19 : Log write in both data sharing and non data sharing
  - LC24 : EDM thread storage serialization (Latch 24)
  - LC24 : Buffer Manager serialization (Latch 56)
  - LC25 : EDM hash serialization
  - LC27 : WLM serialization latch for stored proc/UDF
  - LC32 : Storage Manager serialization
- **Internal contention relief**
  - IRLM : IRLM hash contention
  - CML : z/OS Cross Memory Local suspend lock
  - UTSERIAL : Utility serialization lock for SYSLGRNG (*NFM)
  - Concurrent RE/BIND and most of DDL (*NFM)

Exploitation of system z Hardware

- **z10 and zEnterprise 196 prefetch instruction**
- **Large page frame size (1MB page frame) for buffer pools**
  - Why Large pages?
    - Significant reduction of hit miss in TLB (translation lookaside buffer)
  - z10 and z196
  - IEASYSXX LFAREA=(xx%| xxM | xxG| xxT) and RE-IPL
    - Backed by 256 contiguous 4K real frames and not page-able.
  - 1MB page is fixed, 64bit private pool
  - Buffer pools with long term page fix (PGFIX=YES)
    - If 1MB page frames are available, DB2 will request 1MB first
  - Long Term Page Fix from DB2 V8 to reduce CPU cost for I/O operations even without 1MB
  - 1MB page frames to reduce CPU cost during get pages and release pages
    - Buffer pools with large variations of getpage activities
  - Observed 1-4% CPU reduction in workload level by using 1MB page frames

DB2 and zEnterprise 196

- **Available in Sept 2010**
  - Larger processor cache (1.5MB L2 per core, 24MB L3 per chip, 129MB L4)
  - DB2 9 OLTP, Insert, Utility and query workloads observing 20% to 40% DB2 CPU reduction compared to z10 processors. Typical range is 25 to 35 %.
  - Higher DB2 CPU reduction can be achieved as #of processors per LPAR increases → Best fit with DB2 10 scalability
  - More than 20% improvement with DB2 10 compared to DB2 9 on z196 64 way



Buffer Pool Related Enhancements

- **Large real and DB2 managed in memory buffer pool**
  - z196 supports up to 3TB memory
  - 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
- **Buffer pools allocation as needed**
  - No more penalty for BP over-sizing
    - In DB2 9, an entire buffer pool is allocated when first used
    - If a defined size is bigger than actual used size and using PGFIX=YES
- **Table space buffer pools are no longer allocated when index-only access**
- **Avoid exhaustive local BP scan**
  - During p-lock negotiations in registering page or validity checking
  - Possible DB2 9 bypass
    - Increase PCLOSET/PCLOSEN
    - More and smaller buffer pools

Migration Story..

- DB2 9 NFM REBIND with PLANMGMT

- Migrate to DB2 10 CM without REBIND

- Enable 1MB page usage for key buffer pools
  - 1-3% CPU reduction

- Rebind step by step under DB2 10 CM
  - Virtual storage reduction
  - CPU reduction from avoiding “conversion” of packages,   SPROC re-enablement

- Rebind selective applications with Release Deallocate
  - Key online transactions frequently executed, DDF applications

- Enable DB2 10 NFM

- Enable DB2 10 performance feature

Migration Performance - Early Experience

- **Skip Migration Performance**
- **Catalog tables are replaced with UTS and LOB in NFM**
  - Performance of ENFM process
- **Concurrent REBIND/BIND/DDL in NFM**
  - CPU/Elapsed increase in single bind, utilize parallel bind jobs
    - Plan management is turned on as default
    - Inline LOB is planned for SPT01
  - Limitation on concurrent DROP database
- **Workload with very short running transactions**
  - May not see improvement
  - Good candidates for Release Deallocate or High performance DBATs
- **REBIND**
  - Virtual storage usage
  - Re-enable SPROC (Fast column processing)
    - IFCID 224 records the package with migrated SPROC
  - Packages with CPU Parallelism IFCID 360
    - IFCID360 records the migrated packages with CPU parallelism which causes incremental bind

Utility Performance

- **Significant improvement in DB2 9 and service stream**
- **Equivalent performance from DB2 9 in most of utilities**
- **Elimination of Utility serialization (UTSERIAL) in NFM**
- **Dataset Level FlashCopy support**
- **RUNSTATS**
  - Collect KEYCARD information as default
    - RUNSTATS / INLINE STATISTICS correlation-stats
    - Key cardinality statistics will always be collected with INDEX keyword
  - zIIP eligible for portion of RUNSTATS
    - Not for INLINE statistics
  - AUTO Sampling
    - Better sampling using page sampling based on Real Time Stats
    - Significant CPU & ET savings
    - TABLESAMPLE SYSTEM AUTO

Migration Performance – zIIP usage

- **More zIIP support for TCO improvement**
  - Portion of RUNSTATS utility (Class 1 CPU)
    - Redirection rate varies depends on the RUNSTATS option
  - Parsing process of XML Schema validation (Class 2 CPU)
    - 100% of new validation parser is eligible
      - Can be zIIP, zAAP, or zAAP on zIIP
      - Retro fit into DB2 9 via PK90032 (preconditioning), PK90040 (enabling)
  - Portion of DBM1 processes
    - Prefetch I/Os (DBM1 SRB)
    - Deferred write I/Os (DBM1 SRB)

Beta Customers' Feedback – Workload level

Workload	Results
CICS online transactions	Approx. 7% CPU reduction in DB2 10 CM after REBIND, 4% additional reduction when 50MB of 1MB page frames are used for selective buffer pools
CICS online transactions	Approx 12% CPU reduction
CICS online transactions	Approx 5% CPU reduction from DB2 8
CICS online transactions	No CPU reduction - Candidate of release deallocate usage
Distributed Concurrent Insert	50% DB2 elapsed time reduction, 15% chargeable CPU reduction **after enabling high perf DBAT**
Data sharing heavy concurrent insert	38% CPU reduction
Queries	Average CPU reduction 28% from V8 to DB2 10 NFM
Batch	Overall 20-25% CPU reduction after rebind packages

1. System Level Performance and Scalability

2. Application Level Performance
   - Migration performance without changing DDLs, applications
   - Improvement with DDL or application changes
     - **SQL Procedure Improvement**
     - **Insert, Update and Delete Performance**
     - **Fetch, SELECT performance**
     - **DDF , T2 access and dynamic SQL**
     - **Hash Access**
     - **LOB and XML**

3. Monitoring Support
4. Summary

SQL Procedure Performance (CM)

DB 9

Introduced native SQL Procedure

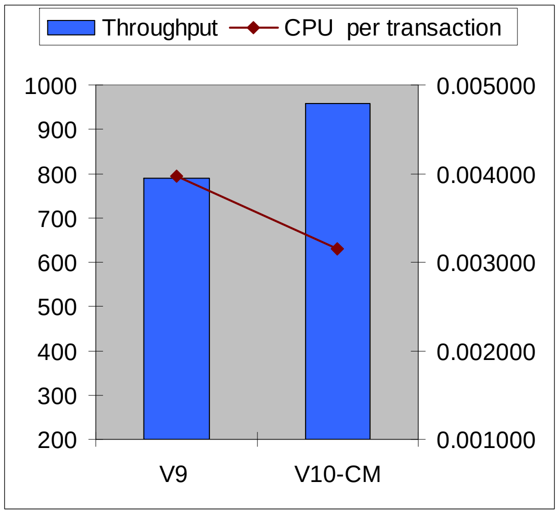
- Improvement by executing procedures in DBM1 instead of WLM address space
- Produce less SQL statements

DB2 10

- Performance optimization for common path
- Specific CPU reduction in commonly used application logic
- SELECT values FROM SYSIBM.SYSDUMMY1
- Chained SET statement support (NFM)

Note: Improvement for SELECT values FROM SYSIBM.SYSDUMMY1
- Applies any applications
- CPU reduction by not calling DM modules

Comparing Native SQLPL Workload DB2 9 vs. DB2 10 CM



- **OLTP using SQLPL**
  - 20% CPU reduction with DB2 10 CM
  - 89% DBM1 Below the Bar usage reduction
  - 5% resp time improvement due to latch contention relief
    - Faster latch resolution
    - LC27 reduction

1. System Level Performance and Scalability

2. Application Level Performance
   - Migration performance without changing DDLs, applications
   - Improvement with DDL or application changes
     - SQL Procedure Improvement
     - Insert, Update and Delete Performance
     - Fetch, SELECT performance
     - DDF , T2 access and dynamic SQL
     - Hash Access
     - LOB and XML

3. Monitoring Support
4. Summary

Insert Performance Improvement

DB2 9

- Large index pages
- Asymmetric index split
- More index look aside
- Data sharing
  - Log latch contention
  - LRSN spin loop reduction
  - Remove log force write at new page (Seg,UTS) via PK83735/PK94122
- Support APPEND option
- RTS LASTUSED support

DB2 10 CM

- Space search improvement
- Index I/O parallelism for non-Seg TS
- Log latch contention reduction and faster log I/O during commit
- Additional index look aside
- Log Buffer to pagefix

DB2 10 NFM

- INCLUDE index
- Support Member Cluster in UTS
- Further LRSN spin avoidance for tables.

Improvement Applies to all TS types

1. **Candidate selection in Sequential Insert for all TS types**
   - Optimize when index manager picks the candidate RID during sequential insert
     - Result: Higher chance to find the space and avoiding a space search
   - Significant improvement in high concurrent insert for all types of Table Spaces with sequential or skip sequential input

2. **Log latch reduction in both data sharing and non data sharing**
   - DB2 9 Log latch reduction in data sharing
   - DB2 10 reduction in both non data sharing and data sharing
   - More LRSN spin avoidance in NFM
     - Avoid LRSN spins when same data/index pages are updates
     - Typical case : MRI

Improvement Applies to all TS types.. Continued

3. Parallel log I/Os for dual logging

– Shorter LOG write I/O wait as I/O requests are done parallel for LOGCOPY1 and LOGCOPY2

4. Long Term Page Fix for output log buffers

– Reduce MSTR SRB time during log I/Os

5. Referential integrity check performance

– Avoid RI check for each insert of a child under the same parent

– Sequential detection and index look aside for RI

Focus on UTS

- **Wider adaptation of Universal Table Spaces**
  - Catalog Tables
  - New Functions : Hash Access, Inline LOB, Access Currently Committed
  - Alter Table Space types

1. **Segmented and UTS space search and CPU reduction**

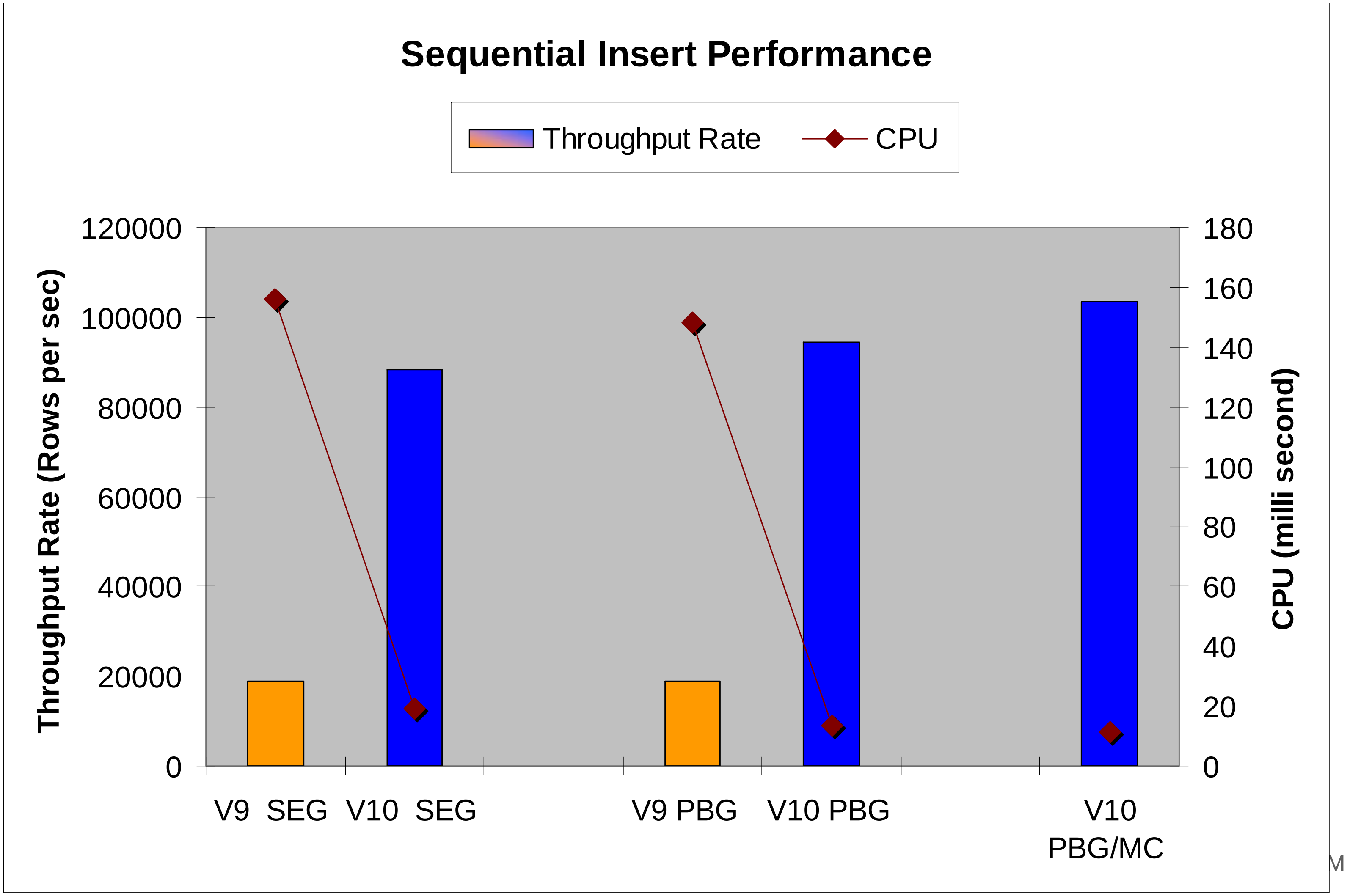
2. **Segmented and UTS space map page latch reduction**
   - Reduce the time held for space map latch

3. **Mass Delete Lock avoidance**

4. **UTS with MEMBER CLUSTER option (NFM)**
   - MEMBER CLUSTER does not maintain the clustering -> quicker space search in insert and remove the hot space map/data page in concurrent insert in data sharing
   - Same consequence during query due to loss of clustering

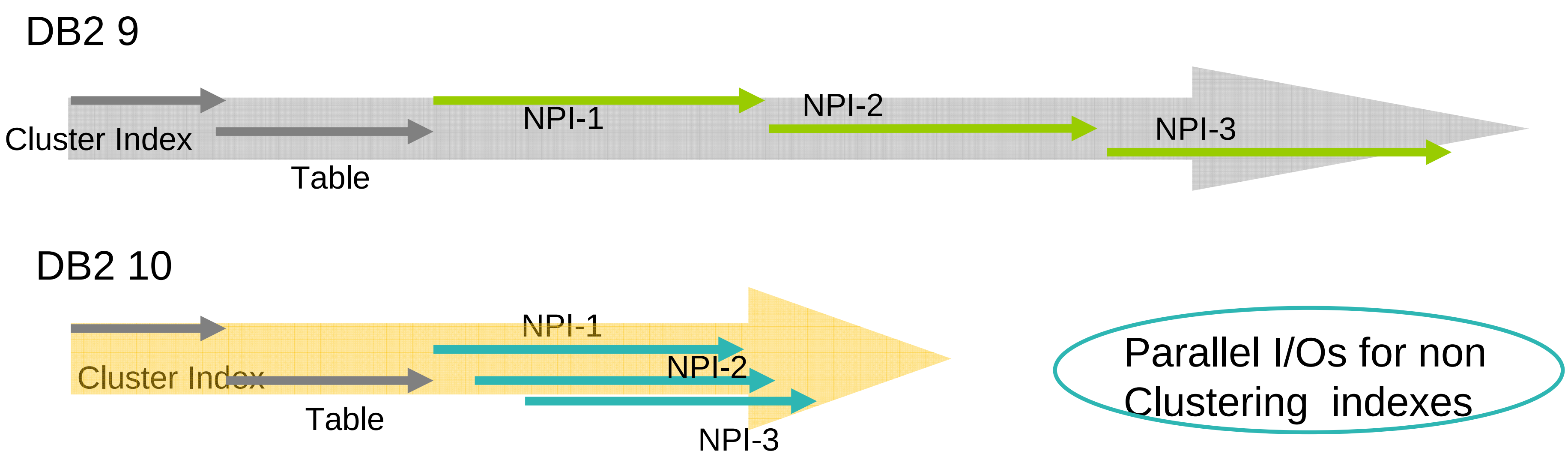
Insert Performance Improvement

- **Sequential key insert into 3 tables from JDBC 240 clients in two way data sharing members. Using Muti Row Insert (batch size 100). Each member resides on LPARs with z10 8CPs.**



I/O Parallelism for Index Maintenance (CM)

- **zPARM INDEX_IO_PARALLELISM (default ON)**
  – Parallel read I/Os for additional indexes by using prefetch
  – Enabled only when there are index I/Os (buffer pool miss)
  – Applicable with all TS type except segmented TS
  – Enabled at 3rd (or 2nd if MC/APPEND) index update
- **Elapsed time reduction**
- **Class 2 CPU time reduction with additional prefetch cost (DBM1 SRB)**



INCLUDE Indexes (NFM)

- **DB2 9 definition**

  CREATE UNIQUE INDEX i1 ON t1(C1,C2)

  CREATE INDEX i2 ON t1(C1,C2,C3,C4)

- **Possible DB2 10 definition**

  CREATE UNIQUE INDEX i1 ON t1(C1,C2) INCLUDE (C3,C4)

  or

  ALTER INDEX i1 ADD INCLUDE (C3,C4)

  and DROP INDEX i2

- **Note the index becomes REBUILD PENDING status after ALTER**

1. System Level Performance and Scalability

2. Application Level Performance
   - Migration performance without changing DDLs, applications
   - Improvement with DDL or application changes
     - SQL Procedure Improvement
     - Insert, Update and Delete Performance
     - Fetch, SELECT performance
     - DDF , T2 access and dynamic SQL
     - Hash Access
     - LOB and XML

3. Monitoring Support
4. Summary

Select/Fetch Performance Improvement

DB2 9

- Sort performance improvement , In memory workfile/Sparse index
- Index on Expression
- Many access path related improvements
  - Plan Stability for static SQL statements
  - Histgram statistics, and more..

DB2 10

- CPU reduction on index predicate evaluation
- RID Pool enhancement by using workfile
- Better performance using a disorganized index
- Row Level Sequential Detection
- Group by using Hash, More in memory workfile usage
- Dynamic statement cache support for literal constants
- Many access path related enhancements
  - Parallelism improvement
  - IN list access improvement
  - Auto stats…and more

CPU Reduction in Stage 1 Predicate Evaluation (CM)

```
SELECT name FROM employee
  WHERE  id > :H
AND status IN (‘a’,’b’,’c’,’d’) ;
```

- **Queries with stage 1 predicates with range, IN list, like**
  - No indication at Access Path Selection
  - Utilize dynamic optimization to evaluate the rows at runtime
  - Applicable in any workloads but higher improvement with queries where many rows are evaluated with multiple predicates
  - Performance improvement
    - Average improvement 20% from generic 150 queries
    - Individual queries shows between 1 and 70% improvement

CPU Reduction in Stage 2 Predicate Evaluation (CM)

SELECT name FROM employee WHERE (salary > 6700)

AND upper(code) ='iod' ← **stage2 predicate**

- **DB2 9 Stage2 Predicate => Evaluate in RDS**
- **DB2 10 Stage2 Predicate can be evaluated at Data Manager**
  - Indicated during access path selection in PUSHDOWN column in DSN_FILTER_TABLE
  - Performance Improvement
    - Measured 10- 30% improvement from queries with stage2 predicates pushdown to DM

CPU Reduction in Sort Process (CM)

- **Simple ORDER BY sort**
  - DB2 9   : In memory sort is used if the results fit in 32KB
  - DB2 10 : In memory sort can be used if the results fit in 1MB
- **Internal SQL sort :**
  - Examples: Sort Merge Join, Non-correlated sub query using IN list
  - DB2 9 : workfile was created
  - DB2 10 : In memory workfile can be used
- **GROUP BY sort**
  - DB2 9 :   Tournament sort
  - DB2 10: Hash sort for efficient GROUP BY operation

CPU Parallelism Enhancement

- **More Parallelism**
  - Parallelism with Multi Row Fetch with read only cursor
  - Parallelism with reverse index scan
  - Parallelism with workfile created by view, outer join, Tablefunction
- **Effective Parallelism**
  - Dynamic Record Range Partitioning
    - DB2 9 Key range partitioning based on LOW2KEY/HIGH2KEY
    - DB2 10 materialize the intermediate result in a sequence of join process, and divide into ranges with equal number of records
- **Balanced Parallel Tasks**
  - Straw Model
    - Smaller workgroup continues to work as it finishes

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
- **Early performance results**
  – 2 to 6 times faster with simple select SQL statements with small key size using list prefetch compared to Sync I/Os
  – 2 times faster using list prefetch for online REORG an disorganized NPI

Row Level Sequential Detection (CM)

- Problem : Dynamic prefetch sequential works poorly when the number of rows per page is large

1000, 1000, 1000, 1000, 2000, 1000, 1000, 1000, 4000, 1000, 1000,
1000, 1001, 1001, 1001, 1001, 3000, 1001, 1001, 1001, 2000, 1001

- Solution: Row Level Sequential Detection (RLSD)
  - Count rows, not pages to track the sequential detection

1000, 1000, 1000, 1000, 2000, 1000, 1000, 1000, 4000, 1000, 1000,
1000, 1001, 1001, 1001, 1001, 3000, 1001, 1001, 1001, 2000, 1001

- 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).
  - Progressive prefetch applies to indexes.

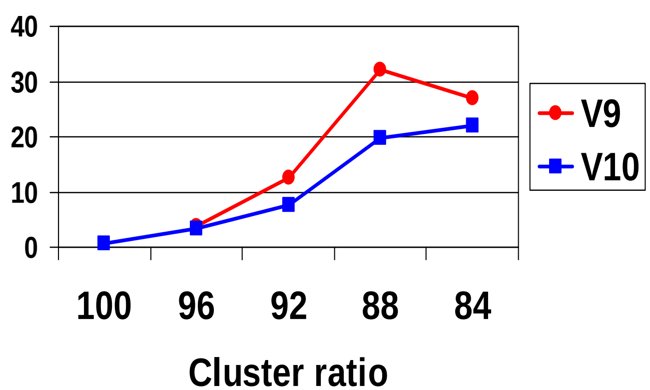
Index—>Data Range Scan

Row size = 80 bytes, page size = 4K (40 rows per page)

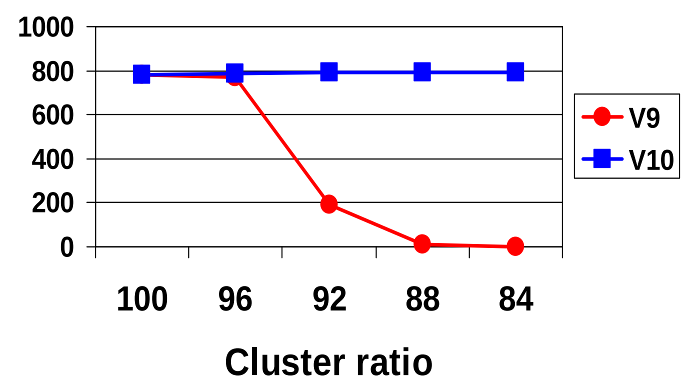
Test case	Cluster ratio	Cardinality	NPAGES
1	100%	20,000,000	50000
2	96%	20,400,000	52000
3	92%	20,800,000	54000
4	88%	21,200,000	56000
5	84%	22,400,000	58000

Read 10% of the rows in key sequential order

Query Time (seconds)



Dynamic Prefetch I/Os



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

1. System Level Performance and Scalability

2. Application Level Performance
   - Migration performance without changing DDLs, applications
   - Improvement with DDL or application changes
     – SQL Procedure Improvement
     – Insert, Update and Delete Performance
     – Fetch, SELECT performance
     – DDF , T2 access and dynamic SQL
     – Hash Access
     – LOB and XML
3. Monitoring Support
4. Summary

DDF Enhancements

- **High Performance DBATs**
- **DDF internal optimization**
  – Optimizes communication between DDF and DBM1
  – Bundle OPEN/FETCH../CLOSE as one network traffic
    • Just like SELECT statement
- **Optimized Special Register and Inactive thread processing**
  – Improves performance of distributed application that set special register frequently
- **REOPT(ONCE) CPU reduction**
  – Reduced CPU cost of processing REOPT(ONCE) in distributed access
- **UNICODEMGR support**
  – Use Unicode for DRDA metadata to avoid EBCDIC->Unicode conversion

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
  - More effective usage of zAAP for JVM
- **Expect significant performance improvement for applications with**
  - Queries that return more than 1 row
  - Queries that return LOBs

Dynamic Statements with Literals

▪ Dynamic Statement : Literal Replacement

– New prepare attribute CONCENTRATE STATEMENTS WITH LITERALS (CSWL)

– Treat like parameter markers

  - JCC 9.7 FP3 new connection property enableLiteralReplacement

– Significant improvement by avoiding full prepare

  - Up to 70% CPU reduction is observed compared to the prepare statement with literals without CSWL
  - Parameter marker is more efficient and still recommended for dynamic statement caching
  - Full prepare with literals may provide a better access path than CSWL with no REOPT

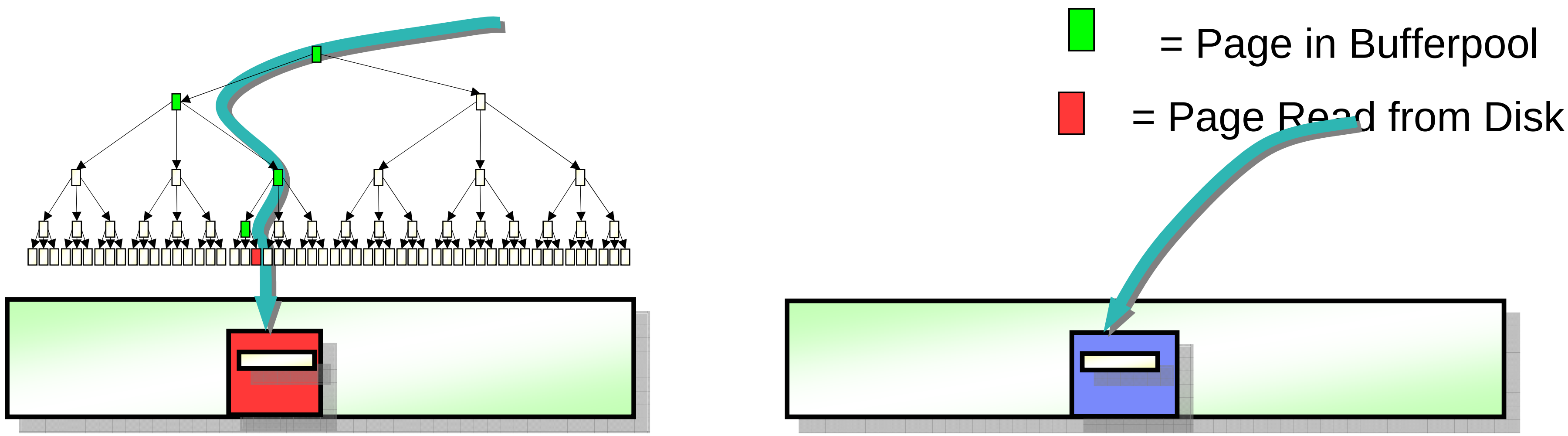
EXPLAIN Support for JDBC and CLI Clients

- **Provide EXPLAIN support to an application programmer**
  - DB2 9 supports EXPLAIN STMTCACHE for DBA
  - DB2 10 supports CURRENT EXPLAIN MODE special register
    - CURRENT EXPLAIN MODE=NO/YES/EXPLAIN
- **JCC and CLI support JCC 9.7 FP3a**
  - JCC
    - JCC datasource with connection property currentExplainMode='YES' | 'NO' | 'EXPLAIN'
  - CLI
    - Set DB2Explain=2 in db2cli.ini
- **Results are stored in DSN_STATEMENT_CACHE_TABLE, PLAN_TABLE**
- **Performance impact**
  - Meant for application development/tuning

1. System Level Performance and Scalability

2. Application Level Performance
   - Migration performance without changing DDLs, applications
   - Improvement with DDL or application changes
   - SQL Procedure Improvement
   - Insert, Update and Delete Performance
   - Fetch, SELECT performance
   - DDF , T2 access and dynamic SQL
   - Hash Access
   - LOB and XML
3. Monitoring Support
4. Summary

Index to Data Access Path vs. Hash Access

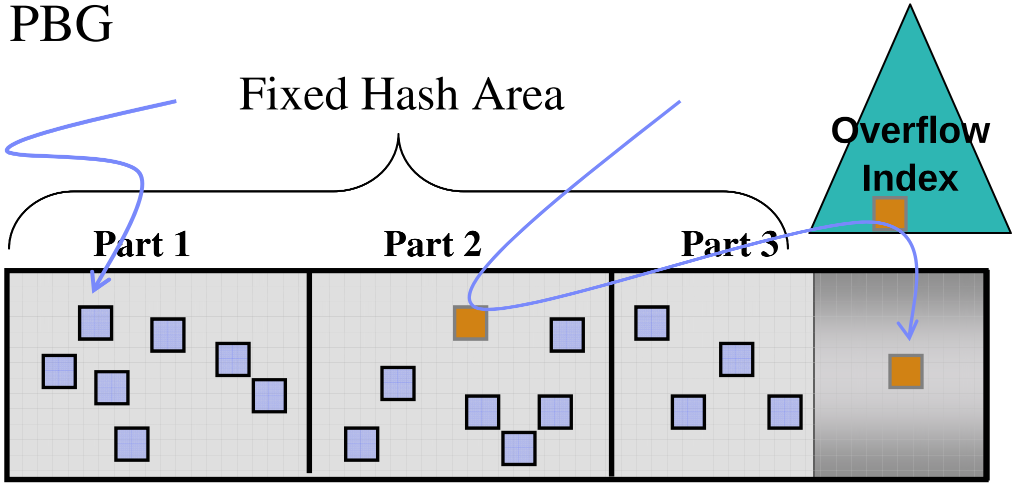


- Index->Data access
  - Traverse down Index Tree
  - For a 5 Level Index
    - 6 GETP/RELPs,
    - 2 I/O's
  - 5 index page searches

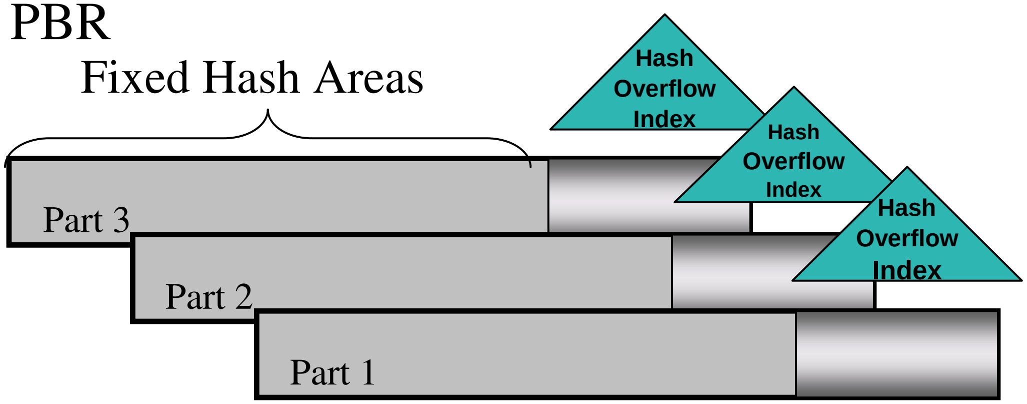
- Hash Access
  - Locate a row without having to use an index
  - Single GETP/RELP in most cases
  - 1 Synch I/Os in common case
  - Greatly reduced Search CPU expense

Hash Access and Hash Space

PBG



PBR



- Optimal to get from fixed area
  - 1 getpage, 1 I/O
- Overflow
  - 3 getpages, 2-3 I/Os
- Use REORG with **AUTOESTSPACE YES** unless you know better
- Real Time Statistics (RTS)
  - #of overflow **TOTALENTRIES**
  - **TOTALENTRIES / TOTALROWS < 10%**
- FREEPAGE is not valid for HASH space but PCTFREE is honored

Hash Access Summary

- **Performance benefit :**
  - Up to 30% DB2 CPU reduction with random access
    - 13 % better than Index-> Data access with NLEVEL 3
    - Higher improvement with large table with small rows
    - Savings in index maintenance once you remove the clustering index
  - Possible reduction in Hotspots
    - Rows are randomly distributed

- **Performance concern :**
  - Not for sequential fetch nor insert
    - Significant Sync I/O increase if accessed in clustering order
    - No Member Cluster support
    - Careful research is necessary on picking the candidate
      - Statement level of monitoring for GetPage and I/Os
  - Significant impact on LOAD utility using input data with clustering order
    - Relief is coming soon - by block fetch in hash order
  - Possible INCREASE in I/O or BP space in some cases
    - In case of small ‘active’ working set
    - In case of many “row not found”

1. System Level Performance and Scalability

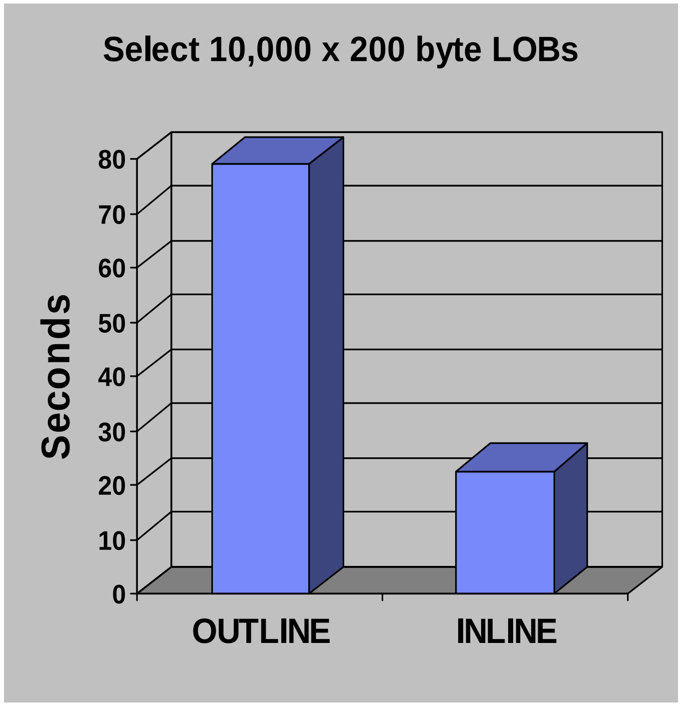
2. Application Level Performance
   - Migration performance without changing DDLs, applications
   - Improvement with DDL or application changes
   - SQL Procedure Improvement
   - Insert, Update and Delete Performance
   - Fetch, SELECT performance
   - DDF , T2 access and dynamic SQL
   - Hash Access
   - LOB and XML
3. Monitoring Support
4. Summary

Inline LOBs for Small LOBs (NFM)

- **CREATE or ALTER TABLE INLINE LENGTH on UTS**
  – INLINE to base table up to 32K bytes
- **Almost Completely Inline LOBs**
  – Save CPU and I/O
    - Less objects, less getpages, less I/Os for both LOB table space and LOB auxiliary index
    - Dynamic prefetch can be used
  – Reduce DASD space
    - No more one LOB per page
    - Inline portion can be compressed
  – 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

Inline LOBs

Elapsed time in random select



Very small LOBs select, insert shows Up to 70% elapsed time reduction with INLINE LOBs

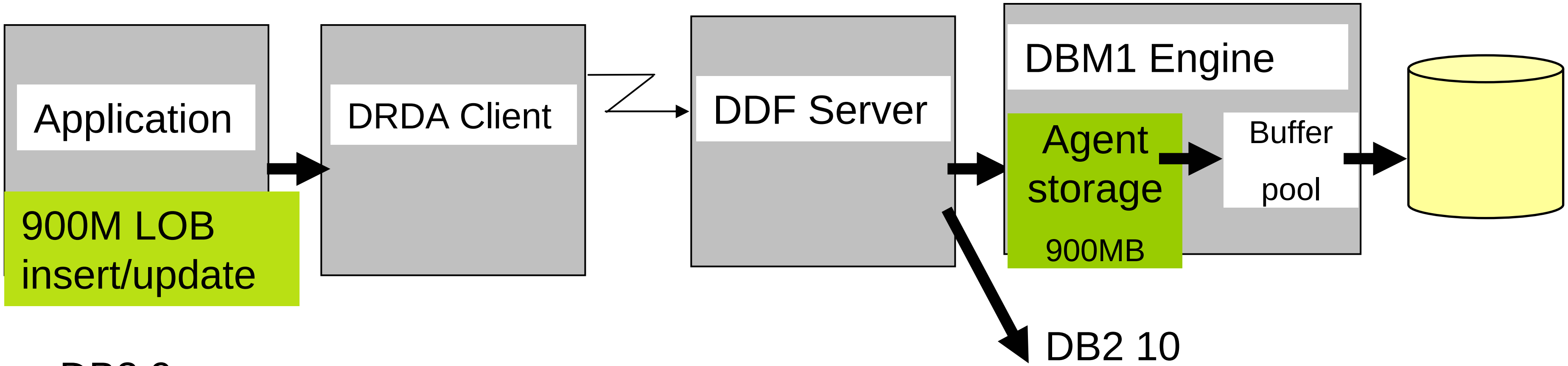
- Inline is good, if
  - Most of LOBs are small and only a few large ones
  - Compress well
- Inline is not good, if
  - Most of LOBs become “split LOB” unless indexing is important for inlined portion
  - Majority of SQLs do not touch the LOB columns
- Base table becomes larger with Inline
  - Buffer hit ratio for base table may decrease
  - Image copy of base table becomes larger

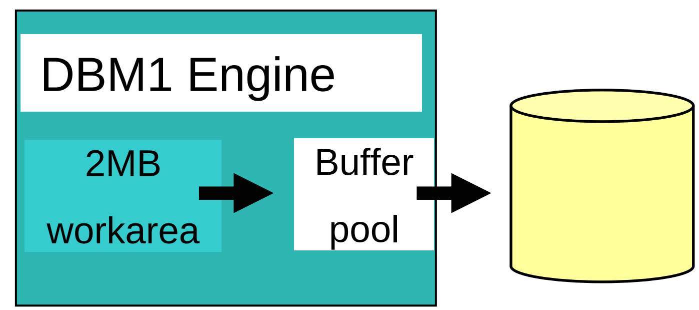
Other LOB Performance Enhancement

- **LOB and XML materialization avoidance**
  - Streaming instead of materialize entire large LOB or XML during insert
  - Significant saving in virtual storage usage
  - Elapsed time and CPU reduction for larger than 2MB LOB insert
  - For XML, stream size is 32K
    - Minor CPU reduction in XML

- **Support RECFM=VBS (Variable Blocked Spanned)**
  - Enables sequential SYSREC data sets to be used for LOB sizes greater than 32K
    - Supports all LOB sizes
  - Orders of magnitude faster than FRV for LOAD and UNLOAD

DB2 10: Eliminate Materialization of LOBs in DBM1





- DB2 9
  - Complete LOB is materialized in agent storage memory
- DB2 10
  - DB2 materializes up to 2M before inserting into the database
  - Storage will be reused for subsequent chunks until the whole LOB is processed.
  - Preformatting and update writes are overlapped with network transfer
- Considerably less virtual storage usage and less CPU time in DB2 10

XML performance improvement

- **Significant Performance improvement in DB2 9 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

  XML index matching with date/timestamp

  Schema Validation in engine
    - No more UDF call for validation
    - Utilizing XML System Service Parser for both validation and non-validation
      – 100% zIIP / zAAP eligible for parser cost

  XML Update
    - No more full document replace
    - Performance improvement in large document update

- **Significant functionality improvements in**
  – Automatic validation by using type modifier
  – XML versioning support
  – Check XML utility

Beta Customers' Feedback – Line Item Focused

Workload	Results
Multi row insert	33% CPU reduction from V9, 4x improvement from V8 due to LRSN spin reduction
Query with 10 stage 1 predicates	5 index matching, 1 index screening, range and IN predicates 60% CPU reduction with same access path
Parallel Index Update	30-40% Elapsed time improvement with class 2 CPU time reduction
Inline LOB	SELECT LOB shows 80% CPU reduction
Include Index	17% CPU reduction in insert after using INCLUDE INDEX
Hash Access	20-30% CPU reduction in random access No improvement or some degradation in CICS workload 16% CPU reduction comparing Hash Access and Index-data access. 5% CPU reduction comparing Hash against Index only access 20x elapsed time increase in sequential access

1. System Level Performance and Scalability
   - Migration performance without changing DDLs, applications

2. Application Level Performance
   - Migration performance without changing DDLs, applications
   - Improvement with DDL or application changes

3. Monitoring Support

4. Summary

SMF compression and ROLLUP

- **Compression for DB2 trace data in SMF with a new zparm SMFCOMP in DB2 10 CM**
  – Overhead is minimum (up to 1% measured w/acct cl 7,8,10)
  – 70-80% SMF data set saving from preliminary measurements
  – Trace formatter needs to be modified to call z/OS compression to decompress the data (Default is OFF)
- **Better ROLLUP accounting in DB2 10 CM**
  – Package accounting (IFCID 239) is rolled up
  – More useful ROLLUPs with DB2 10

Other DB2 10 Monitoring Enhancements or Changes

- **New Monitor class 29 for statement detail level monitoring NFM**
  - IFCID 316/318 for dynamic, IFCID 400/401 for static
    - Statement Level of elapsed, CPU, suspension time, getpages, etc.
- **Record Index split with new IFCID 359**
- **Improved thread and storage monitoring for DDF**
- **Accounting :   zIIP SECP values**
  - Possible redirection value is no longer supported, always zero
  - SE CPU (actual offloaded CPU time) continues to be available
- **Statistics trace interval**
  - Always 1 minute interval in V10 for basic statistics
  - 105-106, 199, 365 continue to honor STATIME values
- **Monitor the last used date in SYSPACKAGE catalog table**

1. System Level Performance and Scalability
   - Migration performance without changing DDLs, applications

2. Application Level Performance
   - Migration performance without changing DDLs, applications
   - Improvement with DDL or application changes

3. Monitoring Support

4. Summary

Examples of Workloads DB2 10 Tends to Show Improvement

- Online transactions with SQL intensive, large getpages per transaction
- Tuned to conserve virtual storage usage
  - Unable to use CICS protected thread and RELEASE(DEALLOC)
  - limiting MAXKEEPD with KEEPDYNAMIC YES
- SELECT or FETCH evaluating >10 rows thru index predicates
- The workload suffers from high latch contention or concurrency
  - High insert rate with LC19
  - Frequent package allocation in EDM pool LC24
  - Queries with CPU parallelism
  - Concurrent utility execution in data sharing
- Insert workload
  - Using concurrent threads
  - Insert with many indexes
- SELECT or FETCH from disorganized indexes/data

Continues.

- DRDA applications
- SQL PL applications
- JCC type 2 (e.g WAS on z/OS) or ODBC on z/OS, executing queries returning mostly more than 1 row
- Dynamic applications using literals as opposed to parameter markers
- Queries with stage2 predicates
- CPU parallel queries with unbalanced parallel tasks
- Insert or update tables with indexes on NULL columns
- Using small LOBs
- LOB insert
- XML application replacing part of large document (> 16K)
- And more…

(Note: this is not exhaustive lists)

DB2 10 System Performance Expectation

- **Migration System Performance**
  - DBM1 virtual storage constraint relief after REBIND
  - CPU reduction in larger scope compared to last releases
    - Without application/DDL changes but REBIND, BIND option change may be required
  - Higher improvement for
    - Workloads with scalability issues (DBM1 storage, DB2 latches)
    - Workloads which can take advantage of RELEASE DEALLOC
    - Using z10, z196 1MB page support
- **Monitoring enhancements**
  - Higher scalability

DB2 10 Application Performance Expectation

- **Application migration performance**
  - Significant CPU and Elapsed time saving in applications with heavy concurrent insert
  - CPU reduction with queries with many stage 1 predicates
  - SQLPL workload specific CPU reduction
  - DRDA applications with many open/fetch../close

- **Take an advantage of New Function**
  - Include index for index only and reduce index maintenance
  - Literal Replacement for dynamic SQL with literals
  - Inline for small LOB
  - Hash access for random access

Thank You!


IBM
Reliable
Global
Announcing DB2 10 for z/OS
Savings...right out of the box
Market Leader
Secure
Trustworthy
Innovative