Tridug

IBM

IBM DB2 10 for z/OS Performance Best Practices

Mark Rader mrader@us.ibm.com
IBM ATS

August 15, 2013

Please note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

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 is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including 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 results similar to those stated here.

IBM

Topics

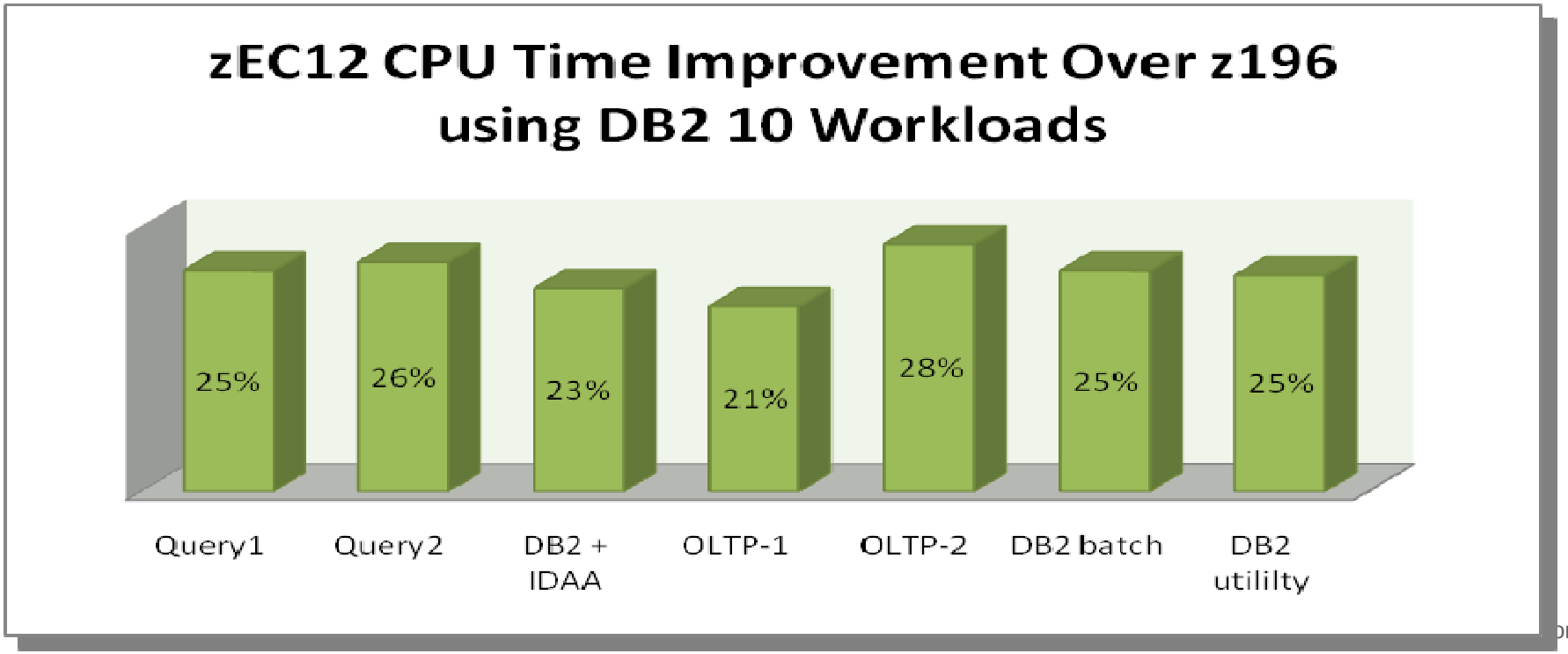
- DB2 and zEC12
- DB2 10 Performance Tips and Lessons Learned
- RELEASE (DEALLOCATE) and High Performance DBATs
- DB2 10 NFM
- UTS (PBG/PBR) Usage Tips

IBM

DB2 for z/OS and zEC12

zEnterprise EC12 and DB2 for z/OS

- **zEC12 Latest generation IBM zEnterprise System**
  - 'EC' = an enterprise class machine.
  - '12' = Twelfth generation of CMOS processors since 1994
- **Faster CPU compared to z196**
  - 5.5GHz, up to 101 processors, up to 3 TB real storage
  - Observed around 25% (20-28% range) of CPU reductions with various DB2 workloads



IBM

DB2 10 Performance Tips and Lessons Learned

IBM

DB2 10 Migration Performance – (1)

- **Agent CPU time reduction**
  - When equivalent or better access paths are taken
  - Good improvement from heavy SELECT from SYSDUMMYx users
    - Example : SELECT CURRENT TIMESTAMP FROM SYSIBM.SYSDUMMY1
    - DB2 10 can bypass table look up for simple SYSDUMMY1 queries
- **DB2 DBM1 SRB time reduction or cost reduction**
  - zIIP usage for prefetch and deferred write
  - Buffer pool scan avoidance
- **Concurrent insert throughput and CPU improvement**
  - Algorithm change
  - Log latch reduction
  - LRSN Spin avoidance
- **Good DBM1 virtual storage reduction**
- **Overall DB2 latch contention improvement**

DB2 10 Performance Experience – (2)

- No.1 reason of less improvement than expected or degradation
  - Vender products online monitoring
  - See II14701
- Fewer hits in package authorization cache (CACHEPAC)

```
PKG-AUTH SUCC-W/O CATALOG      1368.4K
PKG-AUTH SUCC-PUB-W/O CAT       124.9K
PKG-AUTH UNSUCC-CACHE          1011.5K
PKG CACHE OVERWRT - AUTH ID       0.00
PKG CACHE OVERWRT - ENTRY      1002.7K
```

    - Higher BP0 getpages for index for SYSPACKAUTH
  - CACHEPAC is applied to non RACF users in DB2 10 and needs more entries
    - Default CACHEPAC is changed from 100KB to 5MB, which solves most of customer issues; some needed to be 10MB

DB2 10 Migration Performance – (3)

- High rate of Package Table NOT FOUND in steady state
  - V8->DB2 10 migration using default EDM_SKELETON_POOL (10MB)

	QUANTITY	/SECOND	/THREAD	/COMMIT
PT REQUESTS	639224	10.7K	60.15	9.01
PT NOT FOUND	64538	1075.54	6.07	1.00

  - Increase EDM_SKELETON_POOL
- LAST USED PACKAGE in Real Time Stats
  - PM31614 less frequently checked to reduce the overhead
  - PM37672 to disable LASTUSED package checking
    - New ZParm DISABLE_EDMRTS YES
  - PM67558 high DB1M CPU time for accumulated package info that can't be externalized
- Index Probing
  - Real Time Stats lookup and index probing to choose better access path
    - Empty Runstats, literal predicates or REOPT is used
  - Unnecessary index probing can cause performance issues
  - PM54059, PM56542, PM60233 and PM60236

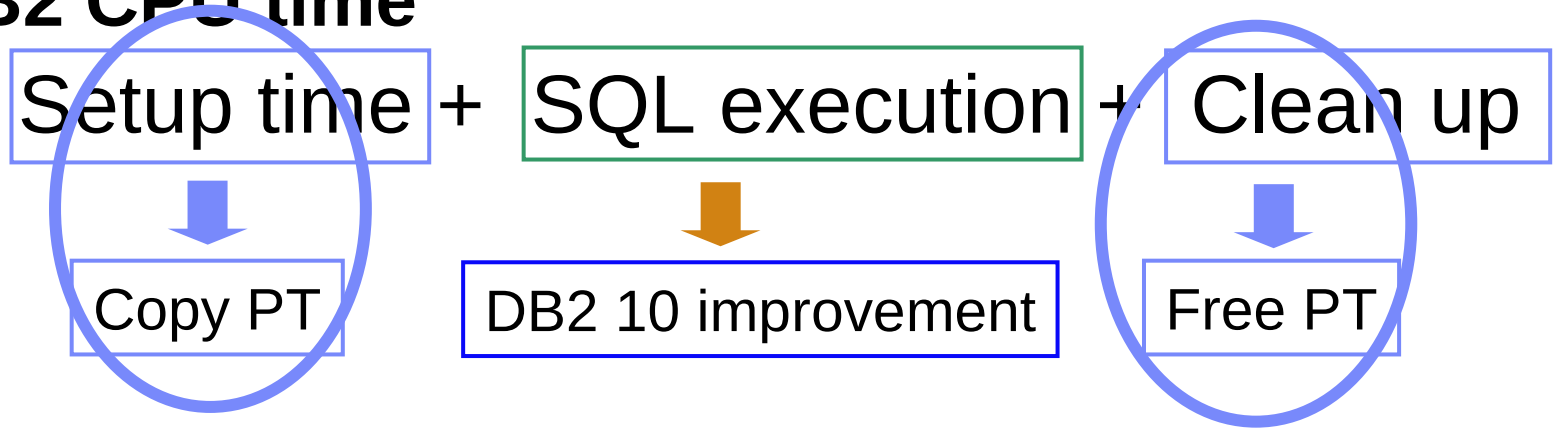
IBM

DB2 10 Migration Performance – (4)

- DB2 system CPU time
  - High DB2 MSTR SRB
    - PM65360 : Eliminate unnecessary z/OS discard requests
  - High DB2 MSTR TCB during idle time
    - Monitoring activity issuing z/OS COUNTPAGES serialization
    - PM49816 (DB2) and OA37821 (z/OS) to reduce CPU usage
- DB2 directory space increase (SPT01/DBD01) in NFM
  - Space increase observed in both base table space as well as LOB space
    - IN_LINE LOBs for SPT01 as ZPARM (PM27073/PM27811)
      - Default SPT01_INLINE_LENGTH (32K)
    - Externalized ZParm COMPRESS_SPT01 with default NO
  - PM64426 addressing LOB growth in DB2 directory
  - PM74659 addressing base table growth in DB2 directory

DB2 CPU Time and RELEASE(DEALLOC)

- **DB2 CPU time**



- **Majority of DB2 10 CPU improvement is in SQL execution**
- **REL(DEALLOCATE) or KEEPDYNAMIC could reduce setup and cleanup cost**
- **Choose the candidate programs understanding the impact and continue to monitor**

IBM

RELEASE DEALLOCATE

RELEASE - BIND and REBIND Option

- **Determines when to release resources that a program uses**
  - RELEASE (COMMIT) : Releases resources at commit
  - RELEASE (DEALLOCATE) : Releases resources when the program terminates (deallocation)
  - RELEASE (INHERITFROMPLAN) : Added by PM07087 only for package to inherit the value from plan
- **Default value**
  - BIND PLAN : COMMIT
  - BIND PACKAGE : value from plan
  - REBIND PLAN/PACKAGE : existing value
  - DB2Binder Utility for JDBC and SQLJ 9.7
    - COMMIT is default with DB2 9 and earlier release
    - DEALLOCATE is default with DB2 10
- **Catalog**
  - Column RELEASE of tables SYSPACKAGE and SYSPLAN

What Are Resources Kept With REL(DEALLOC) ?

- **Packages**
- **Statements**
- **Table space or partition level locks (parent locks)**
- **Information related with the objects accessed by SQL statements**
  - Lookaside buffer, dynamic prefetch tracking, etc.

IBM

Thread Reuse and REL(DEALLOC)

- **CICS or IMS thread reuse eliminates cost of thread allocation and deallocation**
  - Significant CPU saving with both REL(COMMIT) and REL(DEALLOCATE)
- **REL(DEALLOC) becomes effective with the reused thread by further reducing the cost**
  - Without thread reuse, REL(DEALLOCATE) still goes through package deallocation at thread termination

NOTES: CICS Protected Threads Overview

- **Only entry threads can be protected by specifying PROTECTNUM=n on the DB2ENTRY definition for an entry thread**
  - A protected thread is not terminated when a transaction ends, and the next transaction associated with the same DB2ENTRY reuses the thread.
  - If no eligible task to use thread then up to PROTECTNUM threads will be kept idle
- **Thread idle for up to two purge cycles**
- **Some confusion over thread reuse and protected threads**
  - Any thread, pool or entry, can be reused:
    - Protected threads can be reused within the purge cycle time.
    - Unprotected threads are reused by transactions queued, waiting to use it

High Performance DBATs

- What are High Performance DBATs ?
  - Support RELEASE(DEALLOCATE) bind option in DRDA
  - Avoid processing to go inactive and then back to active at every commit
    - Continue to cut accounting at commit
- How does it work ?
  - A DBAT stays active with connection until 200 commits are executed
    - Connection turns inactive after 200 times to free up DBAT
  - In-flight DBATs waiting for next message can be cancelled after the IDTHTOIN value has expired
  - DBATs in completed unit-of-work status become inactive after the POOLINAC value has expired

High Performance DBAT Capabilities

- **New -MODIFY DDF PKGREL(BNDOPT/COMMIT) command**
  - Effective only with ZPARM, CMTSTAT=INACTIVE
  - PKGREL(BNDOPT) honors package bind option
  - PKGREL(COMMIT) forces package bind option RELEASE(COMMIT)
    - Same as V9 inactive connection behavior and will be default processing until a –MODIFY DDF PKGREL command issued
- **Switching back to PKGREL(COMMIT) will occur gradually**
  - Any inflight high performance DBAT that commits will be terminated if RELEASE(DEALLOCATE) packages have been used
  - Any active DBAT marked as high performance (no active UOW pending) waiting for a new request from client will be terminated by DDF service task after 2 to 4 minutes.
  - Local connections need to be rebound to switch

DB2 10 : Memory Usage With REL(DEALLOCATE)

- **What to consider?**
  - Fat, Persistent Threads
- **Virtual and real storage usage**
  - Packages bound with DB2 9 or earlier as REL(DEALLOC) :
    - Package tables are stored below the 2GB bar
    - Impact on DBM1 virtual below the bar and real storage just like DB2 9
    - Reduction in LC24 contention
  - Packages bound with DB2 10 as REL(DEALLOC) :
    - Package tables are stored in thread storage above the 2GB bar
    - Some increase in DBM1 below the bar
    - Impact on DBM1 above the bar = real storage usage
  - Accumulated DB2 object information
    - Potential CPU cost for scanning the objects built up under the thread

CICS TS 4.2 Protected Threads Improvement

- **New REUSELIMIT(value) : Limit on the number of times that a thread can be reused**
  - A value of 0 : no limit, this was the situation before CICS TS 4.2.
  - Default of 1000 : provides sufficient protection against fat threads (over-allocating thread storage and/or EDM pool storage with RELEASE(DEALLOCATE) BIND option)
  - Use default and monitor DB2 storage usage and adjust the number if needed
- **PURGECYCE change to the DB2CONN definition**
  - This controls how long protected threads are allowed to stay dormant before either being reused or terminated. Again it contributes to the "fat thread" problem.
  - PURGECYCLE now allows a lower limit of 5 seconds
    - today it is 30 seconds which is also the default
    - If the lower limit is used then on average a protected thread will be purged after 7.5 seconds, as a protected thread has to be seen by two purge cycles before it is terminated.
    - The default remains at 30 seconds meaning on average a protected thread will be purged after 45 seconds.

Considerations : Concurrency

- **Locks for packages and parent objects are held for the life of threads**
  - BIND or REBIND operations against the packages :
    - Timeout because S-locks are held against the packages used in the persistent threads
  - DDL operations such as DROP, ALTER against the objects used any time in the thread life :
    - Timeout because intent parent locks are held in the persistent threads
  - With high performance DBATs, switching PKGREL to COMMIT solves the issue
    - In data sharing, switch has to be done in all members who are executing the package with REL(DEALLOC)
- **Applications using LOCK TABLE statements**
  - TABLE locks are held across commit
    - STOP DATABASE .. AT(COMMIT) can interrupt the persistent thread

Verification and Monitoring : Where To Look ?

1. **Verify RELEASE(DEALLOCATE) is working as expected**
2. **Identify the benefit**
3. **Monitor virtual and real storage**

IBM

(1) Is REL(DEALLOC) Working ?

- Examples of simple workload SQLJ IRWW with REL(COMMIT) vs REL(DEALLOC)

Statistics - EDM	COMMIT	DEALLOC
DBD Requests per commit	1	0.05
PT Requests per commit	6.3	0.2
Other Statistics	**COMMIT**	**DEALLOC**
Package Allocation Success	1	0.01
Package Authorization Success	1	0.01
Lock Requests	16	7.4

(2) Identify Benefit

- **General DB2 CPU time : Class 2 CPU time + non DDF address space time**
- **For distributed threads, alternatively statistics total CPU time**
  - Examples of simple workload SQLJ IRWW with REL(COMMIT) vs REL(DEALLOC) with 48 clients

Accounting (micro sec)	V9 COMMIT	V10 COMMIT	V10 DEALLOC
Class 1 CPU time	1233	1012	982
Class 2 CPU time	835	650	608
Statistics CPU (micro sec)			
DDF Address Space CPU per commit	1365	1131	1086
Non DDF Address Space CPU per commit	77	45	45
Total Address Space CPU per commit	1442	1177	1132

- **V9 commit vs. V10 commit = 18% DB2 CPU reduction**
- **V9 commit vs. V10 dealloc = 23% DB2 CPU reduction**

(3) Memory Usage

- **DBM1 storage statistics (IFCID 225, statistics class 1)**
  - Examples of 450 threads with REL(COMMIT) vs REL(DEALLOC)

DBM1 Below (MB)	COMMIT	DEALLOC
Total agent local storage (MB)	40	80
- Total system agent storage (MB)	10	10
- Total non-system agent storage (MB)	29	70
- Number of active user threads	447	453
- Per user thread (MB)	0.06	0.15
Real Storage		
64 bit shared memory pages	506819	581111
Shared memory roughly per thread (MB)	4.43	5.01

When To use ?

- **RELEASE(DEALLOCATE) is NOT meant for all packages**
- **Not recommended if**
  - Under real or virtual storage constraint
  - Concurrency with DDL, REBIND is important for local connection
  - Not for a thread which executes large variety of infrequently used packages and statements
  - Not for a thread which touches many DB2 objects
- **Effective when threads are reused and the programs are repeatedly executed across commits**
  - Higher CPU reduction rate when the packages frequently issue commits
  - Higher CPU time reduction with large packages but storage impact is higher, too
- **Continue to monitor the usage, benefit and storage impact**
- **Switching local connections is more involved than distributed (HPDB)**
- **DB2 11 introducing the possibility of automatic switching**

IBM

KEEPDYNAMIC

KEEPDYNAMIC - BIND and REBIND Option

- **Determines whether DB2 keeps dynamic SQL statements after commit**
  - KEEPDYNAMIC **NO**/YES
  - YES : DB2 keeps dynamic SQL statements after commit until,
    - Application process ends
    - A rollback occurs
    - An explicit PREPARE with same statement identifier
  - KEEPDYNAMIC YES does not apply for “EXECUTE IMMEDIATE”
- **Default is NO**
  - BIND PLAN NO
  - BIND PACKAGE NO
  - REBIND PLAN/PACKAGE Existing value
  - DB2Binder –keepdynamic for JDBC and SQLJ NO
  - DB2BaseDataSource.NOT_SET(0)
- **Catalog**
  - KEEPDYNAMIC in SYSPLAN and SYSPACKAGE

Performance of Prepare

- Full prepare vs. Short prepare
  - No cache hit as opposed to global cache hit
  - Immediate benefit by turning on CACHEDYN
  - Magnitude of hundreds times difference
    - Depends on complexity of the statement
    - One example ; 200 times less CPU time for short prepare vs. full prepare

- Short prepare vs. prepare avoidance via KEEPDYNAMIC
  - Global cache hit as opposed to local cache hit MAXKEEPD > 0
  - Application needs to avoid prepare
  - A few % to tens of % difference
    - Depends on the size of statement
  - With KEEPDYANMIC (YES), dynamic statements performs like static or even better in term of CPU consumption
    - Package and statements are kept in local thread like RELEASE(DEALLOCATE)
    - Already prepared
    - DBATs stay active

IBM

KEEDPDYNAMIC and Virtual/Real Storage

- **Statements are kept in user thread storage**
  - DB2 9 EDM and thread storage below the 2GB bar : virtual storage impact
  - DB2 10 above the 2GB bar : real storage impact

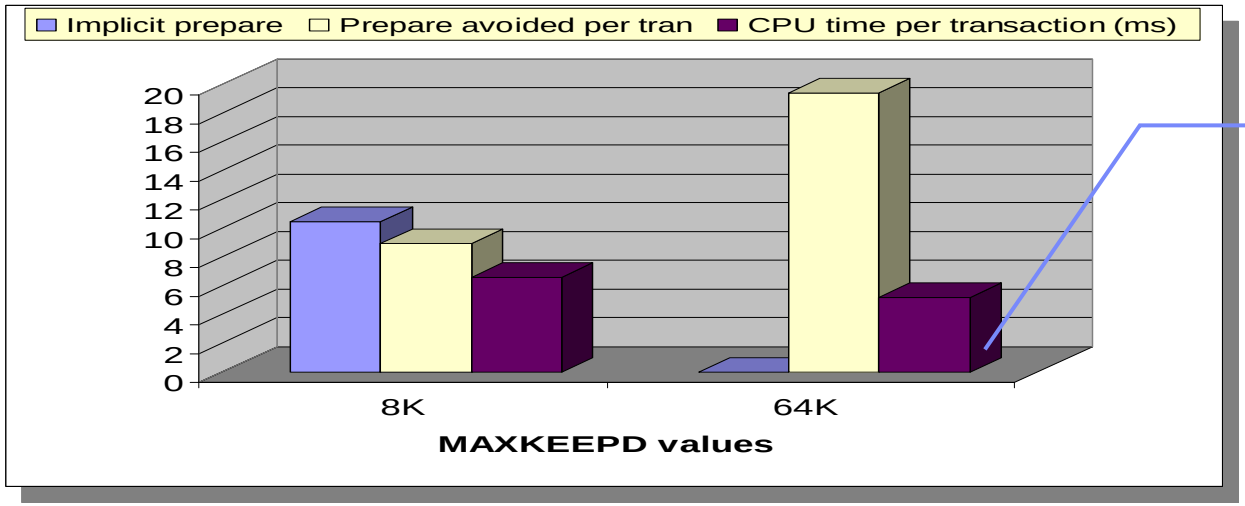
- **Controlling the storage usage**
  - Use parameter markers (or literal replacement) to reduce the number of statements
  - Cap with MAXKEEPD value
    - Limits number of cached statements in system level
    - Remove statements in local thread storage
      - New execution of the removed statements results in “IMPLICIT” prepare
    - MAXKEEPD =0 still benefit
      - Package allocation avoidance
      - Explicit prepare reduction
  - Storage contraction via CONTSTOR and MINSTOR to reduce unused storage footprint
  - CACHEDYN_FREELOCAL
    - Remove statements when threshold is reached
    - New execution of the removed statements results in “IMPLICIT” prepare

IBM

Monitoring – Effectiveness of KEEPDYNAMIC

Dynamic SQL STMT	COUNT	Comments
PREPARE REQUESTED	98786	Explicit prepare requested
FULL PREPARE	19088	Explicit and Implicit prepare, not found in global cache
SHORT PREPARE	79696	Found in global cache
GLOBAL CACHE HIT RATIO	80.68%	
IMPLICIT PREPARES	10510	KEEPDYNAMIC(YES) but statements not found in local cache
PREPARES AVOIDED	63834K	Statements found in local cache
CACHE LIMIT EXCEEDED	40596	Statements are invalidated due to MAXKEEPD or FREELOCAL
PREP STMT PURGED	0	Statements are purged from DDL or runstats
LOCAL CACHE HIT RATIO	99.98%	

DB2 10 and Larger MAXKEEPD



26% CPU reduction

MAXKEEPD	8K	64K
CPU time per transaction (ms)	**6.55**	**5.18**
#Concurrent thread	**844**	**844**
Prepare avoided per tran	**8.95**	**19.37**
Implicit prepare	**10.43**	**0**
DBM1 Below (MB)	**93.39**	**92.53**
Real storage usage (MB)	**5861**	**5874**

IBM

KeepDynamic DBAT Refresh

- **KEEPDYNAMIC (YES) causes DBATs stay active**
- **DB2 9 change (PK69339) to address long running DBATs with KEEPDYNAMIC(YES)**
- **Requires**
  - CMTSTAT = INACTIVE (Default)
  - Client IBM Data Server Driver/Client for JAVA
  - Sysplex Workload Balancing and /or Seamless Failover
    - DataSource "enableSysplexWLB" or "enableSeamlessFailover" set true
  - Data source KeepDynamic set
- **DDF will terminate the DBAT connection after**
  - Over one hour after it has been used OR
  - Over 20 minutes remained idle
- **SAP servers no longer need be manually stopped to relieve possible virtual storage constraint**

IBM

DB2 10 New Function Performance

INCLUDE Indexes (NFM)

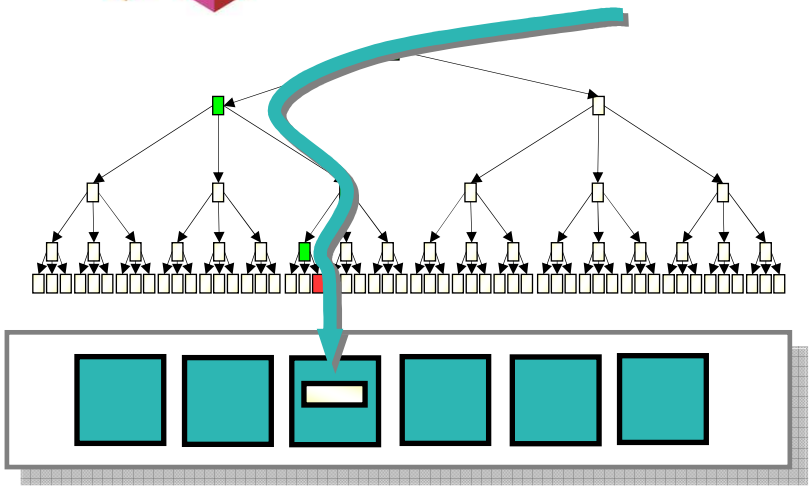
TWO indexes
CREATE UNIQUE INDEX **ix1** ON t1(C1,C2)
CREATE INDEX **ix2** ON t1(C1,C2,C3,C4)

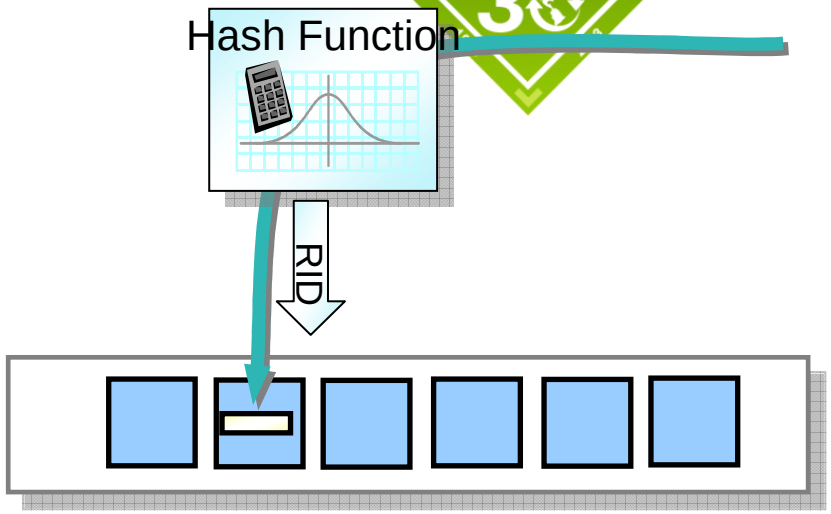
One INCLUDE index
CREATE UNIQUE INDEX **ix3** ON t1(C1,C2) INCLUDE (C3,C4)

- CPU saving in Insert
  - 2 Index vs 1 index with INCLUDE columns shows 30% cpu reduction in insert
- **More index only access**
- **DASD space saving**
- **More Stable Access Path Selection (ix1 or ix2 ???)**

- **Remember….**
  - **Include index will be larger than original unique index**
  - If the majority of usage is index ix1, you may see getpage, I/O impact

Index to Data Access vs. Hash Access (NFM)





Hash Access can provide CPU reduction

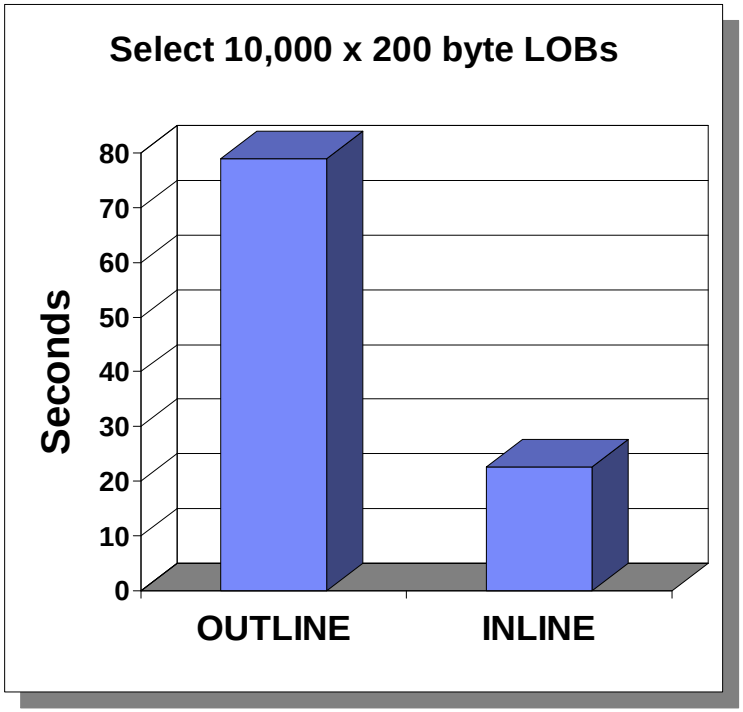
- DB2 locates a row without having to use an index
  - 5 to 30% CPU reduction observed
    - Better improvement with large tables with small rows
  - IRWW workload shows average 8% CPU reduction with subset of tables as Hash access

Challenge to find the right objects

- **Not ideal for sequential fetch nor insert**
- Sync I/O increase if accessed in clustering order
  - Impact on LOAD utility using input data with clustering order

Inline LOBs for Small LOBs (NFM)

Elapsed time in random select



- 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
    - Index Expression can be used
  - **Reduce DASD space**
    - No more one LOB per page
    - Inline portion can be compressed

Not ideal IF,

- Most of LOBs become “split LOB”
- Majority of SQLs do not touch the LOB columns anyway

Impact against base table access with Inline

- Buffer pool size needs adjustment with inline

IBM

UTS Performance

IBM

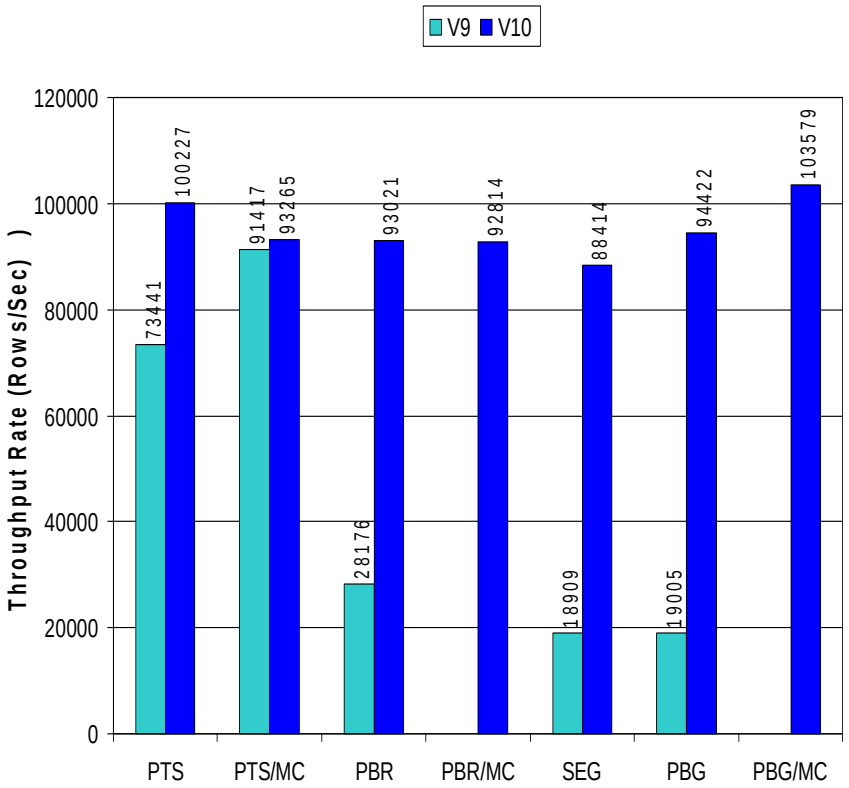
Some new features are UTS only …

- **DB2 9**
  - Clone tables
- **DB2 10**
  - Hash access
  - Currently Committed bind option / prepare attribute.
  - Inline LOB support

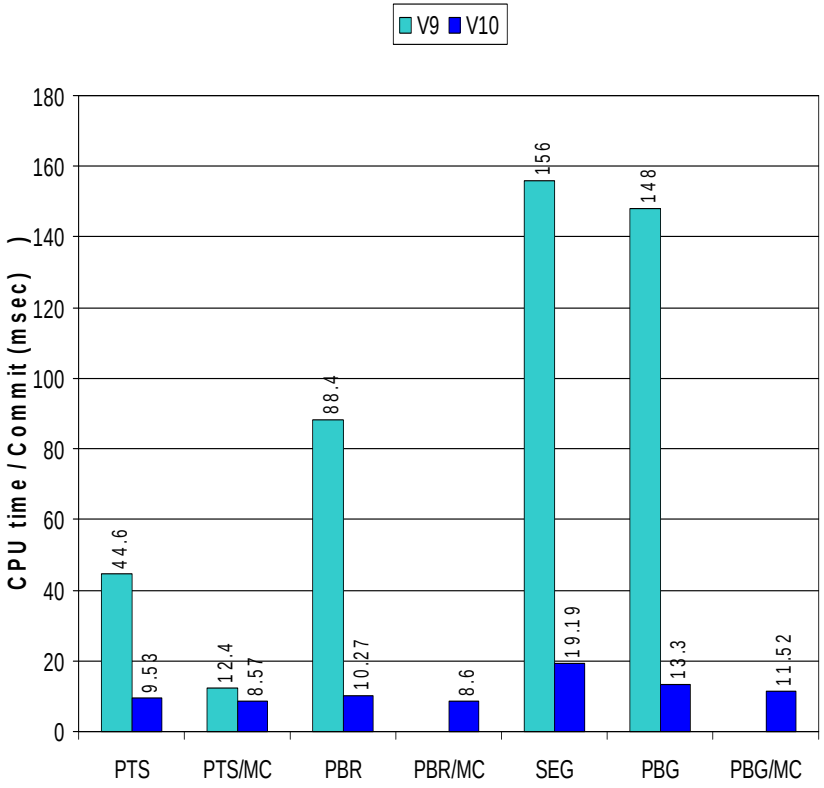
One Example of Insert Performance

Insert Rate and CPU Time Comparisons DB210 vs. 9
Sequential Inserts – Page Level Locking

Insert Rate Comparisons



CPU Time Comparisons



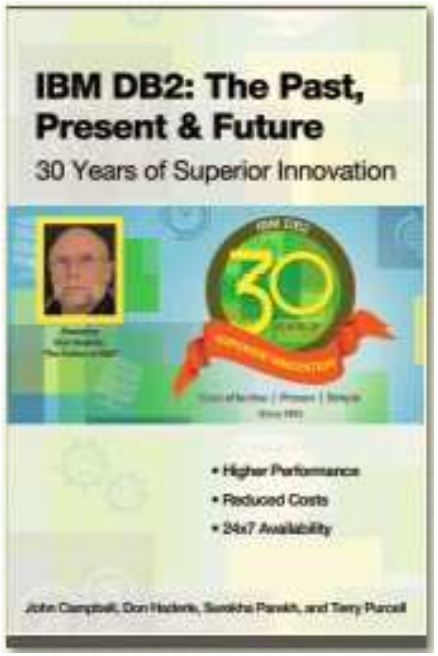
- Multi-Row Inserts (100)
- Page Level Locking
- 240 concurrent threads
- Commit every 3 inserts with no delay

UTS Usage Guidance

- **UTS Insert Performance in DB2 10**
  - Significant improvement from DB2 9 in concurrent insert
    - Insert algorithm change
    - Member cluster support (NFM)
  - Significant variation depending on concurrency, insert pattern, row size and number of indexes. In our workload,
    - PBG is generally better performer than classic Segmented TS
    - Page level locking : As good as classic table spaces
    - Row level locking : Seq insert into non-MC UTS/SEG is soft spot
  - Recommend specific test using own workloads
- **Partition By Growth Table Space**
  - Be aware of cost of high MAXPARTITIONS
    - NUMPARTS vs. MAXPARTITIONS
    - Recommend to specify realistic number of partitions, not 4096
    - PM57001 - Allows ALTER TABLESPACE MAXPARTITIONS to lower numbers (DB2 9 and 10)

Top DB2 for z/OS Communities

- **World of DB2 for z/OS** http://db2forzos.ning.com/
- **DB2 10 LinkedIn** http://linkd.in/IBMDB210
- **DB2 for z/OS What's On LinkedIn** http://linkd.in/kd05LH
- **DB2 for z/OS YouTube** http://www.youtube.com/user/IBMDB2forzOS
- **WW IDUG LinkedIn Group** http://linkd.in/IDUGLinkedIn
- **IDUG.ORG** http://www.idug.org
- **DB2 for z/OS Exchange Forum** http://ibm.co/DB2zHotline



Acknowledgements and Disclaimers:

Availability. References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.