IDUG

Akiko Hoshikawa, IBM

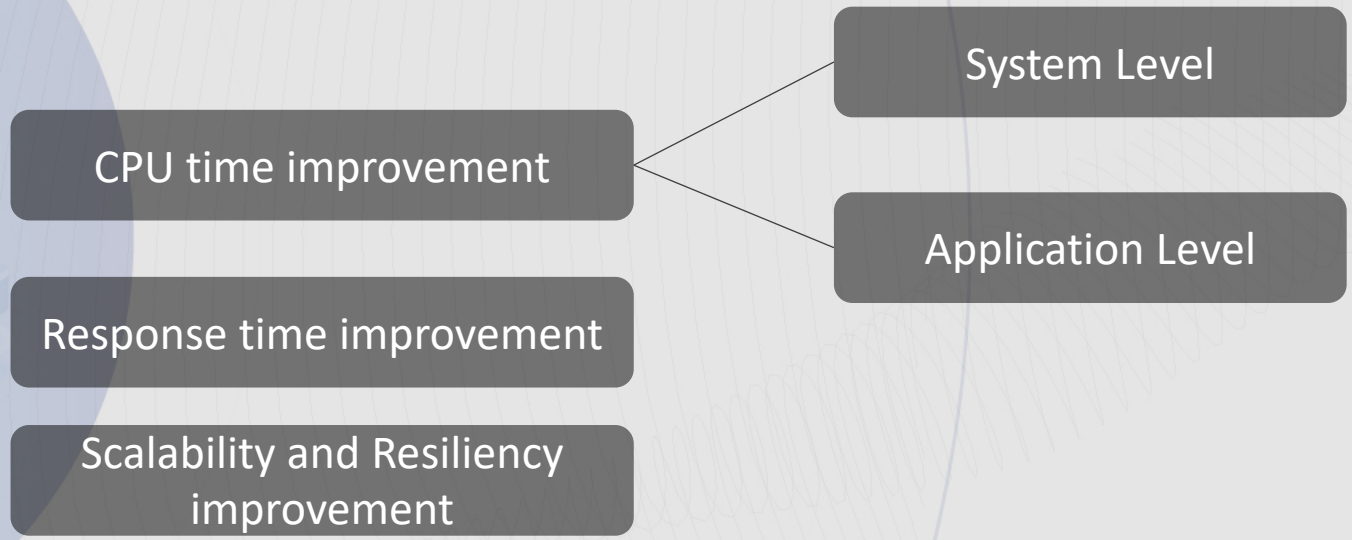
Db2 for z/OS Performance Improvement Opportunities

Db2 for z/OS

Disclaimer

- IBM’s statements regarding its plans, directions, and intent are subject to change or withdrawal without notice and 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.

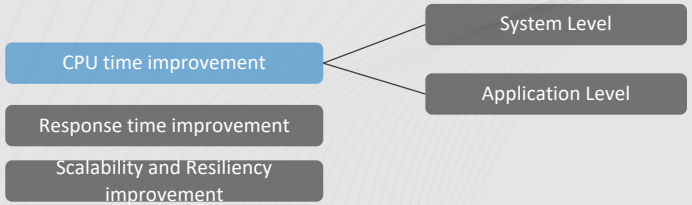
Improvement Opportunity



- Performance improvement outside of ”out of box” improvement.
- Out of box improvement = internal optimization that you get automatically by migrating to new releases or updating the maintenance or simply REBIND

- Sharing as a Good materials , something old that you have heard but forgotten, something new that you might have not yet heard.
- Will not contain in 1 hour slot
- Let me know the feedback if you like the format, and I should continue to maintain this..
- Out of the box improvement
  = enabled as default such as
  - Partial decompression
  - Xproc
  - Async XI
  - Access path improvement

CPU Improvement Opportunity



- General CP usage reduction
  - IDAA
    - Offload eligible queries VS trade off on data maintenance
      - IDAA modeling feature to understand the saving
    - Integrated synchronization reduced the cost of incremental updates
  - zIIP eligible work
    - Java application
    - CPU parallelism
      - Be aware of aggressive parallelism degree
      - Cost of enabling parallelism
  - Db2ZAI
    - SQL optimization provides access path and runtime improvement for eligible queries
      - Trade off between improvement vs. data collection
      - Likely eligible queries can be found though IBM provided catalog/plan table queries

- Selective Db2 features
  - System level
    - Expanded LRSN – write intensive app
    - Statement stability
    - zSort acceleration
    - Sort and sparce index memory
    - Fast Traverse Blocks
    - Buffer pool options
  - Application level
    - Basic application cost Reduce the number of columns
    - Release Deallocate, High performance DBATs, CICS thread reuse, IMS connection pooling
    - Index options
    - Overflow avoidance
    - DGTT

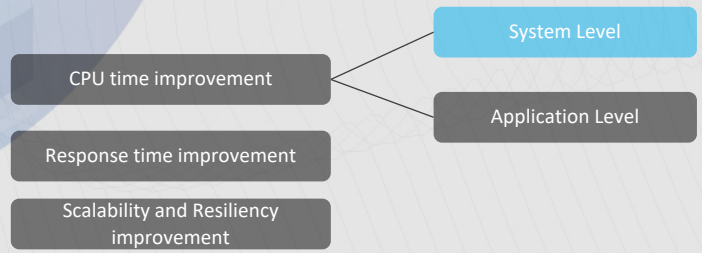
Reference : Db2 for z/OS V11 & 12 Language / API CPU Cost Comparison Using IBM Relational Warehouse Workload (IRWW)

Language/API	Base CPU/Tran Cost	Billable CPU/Tran Cost after zIIP redirect
COBOL Stored Proc	**1X (BASE)**	0.74x
C Stored Proc	1.02x	0.75x
SQLJ Stored Proc	1.71x	1.16x
JDBC Stored Proc	2.19x	1.54x
Native SQL Stored Proc	1.07x	**0.47x**

- IRWW : very simple online transaction workload
- Measured with z14 processors in non-data sharing environment
- Stored Procedures are called from IBM Type 4 JCC driver Client – clients on zLinux
- zIIP benefit is applicable only for remote Store Procedures called via TCP/IP DRDA

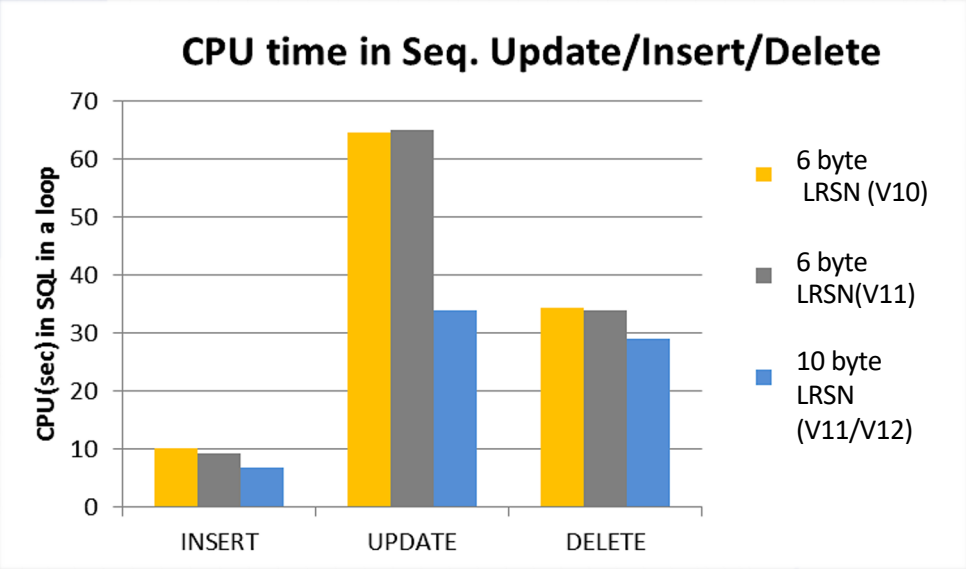
CPU Improvement Opportunity

System CPU time



CPU - System level : Extended log RBA/LRSN

- What it is : Extended LRSN : 10-byte log sequence number
  - BSDS, Db2 catalog and user objects
  - V12 requires BSDS to be 10-byte format
- Why : CPU reduction by eliminating LRSN spin
  - Update intensive objects in data sharing environment should be converted to 10-byte LRSN to avoid LRSN spins completely



Notes:

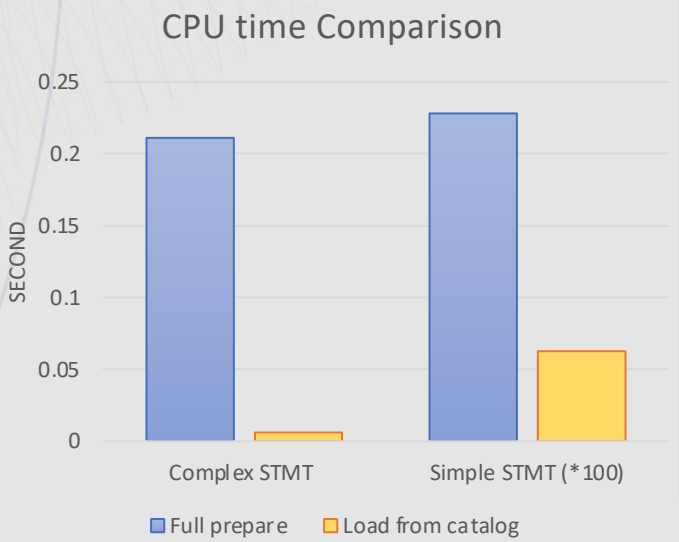
LRSN spin can be observed in the service fields in statistics

- QJSTSPNN
- QJSTSPNI

- Introduced in V11 6-byte to 10-byte format for both RBA and LRSN
  - BSDS, Db2 catalog and user objects
  - Workfiles are always 10 byte format
- Once the RBA/LRSN reached the soft limit, results in failed updates.
- V12 requires BSDS to be converted to 10-byte extended format at the migration
- 10-byte LRSN in data sharing can avoid LRSN spin completely to generate unique LRSN
  - No need to spin for RBA
  - 16 microseconds to 1 picosecond)

CPU - System level : Dynamic Statement Stability

- What it is
  - Capture a prepared dynamic statement in Db2 catalog to be reused once cached statements are stolen from statement cache
  - Introduced at V12 function level 500
- Why
  - Stabilize the access path of critical dynamic applications
  - Loading from catalog can be cheaper than full-prepare
- Considerations
  - Maintenance of SYSDYNQRY catalog table



- Introduced in V12 function level 500
- CACHEDYN_STABLIZATION
  - BOTH (default)
- START DYNQUERYCAPTURE command to define the scop
- DISPLAY DYNQUERYCAPTURE command to monitor
- FREE STABLIZED DYNAMIC QUERY command to free the captured statements

CPU saving from more memory - z15 Z Sort Acceleration

- What it is :
  - z15 Sort accelerator uses SORT LIST (SORTL) instruction to perform sort with multiple input list, exploited by
    - DFSORT & Db2 REORG invoking DFSORT
    - Db2 RDS sort
- Why :
  - REORG TABLESPACE utility UP to 12% CPU saving and elapsed time saving was observed
    - Db2 APAR PH28183 & UTILS_USE_ZSORT zparm YES
    - Requires more member during SORT phase (up to 2x observed)
  - Db2 RDS sort exploitation of SORTL
    - APAR PH31684
    - Enabled automatically for ORDER BY and GROUP BY when the condition met
      - When z15 is detected and row size is small enough – seen average 8% (0 to 30% range) CPU saving for sort intensive queries
      - Db2 will use up to sort pool size (SRTPOOL zparm). Larger SRTPOOL, better CPU saving

- IBM z15 includes a new coprocessor called the Integrated Accelerator for Z Sort, which is driven by the new SORT LISTS (SORTL) instruction.
- z/OS DFSORT and Db2 REORG utility and Db2 RDS sort take advantage of Z Sort.
- Available now on z15 with z/OS DFSORT APAR PH03207, PTFs UI90067, and UI90068 for z/OS V2R3 and V2R4.
- Db2 REORG APAR PH28183
- How to enable
  - OPTION ZSORT in DFSORT control statement or ICEMAC/PRM
  - Db2 zparm UTILS_USE_ZSORT = YES (NO as default)
  - Consideration – requires more memory during sort phases
- Db2 RDS APAR PH31684
  - Enabled automatically when conditions are met
  - Recorded in IFCID 96 and statistics

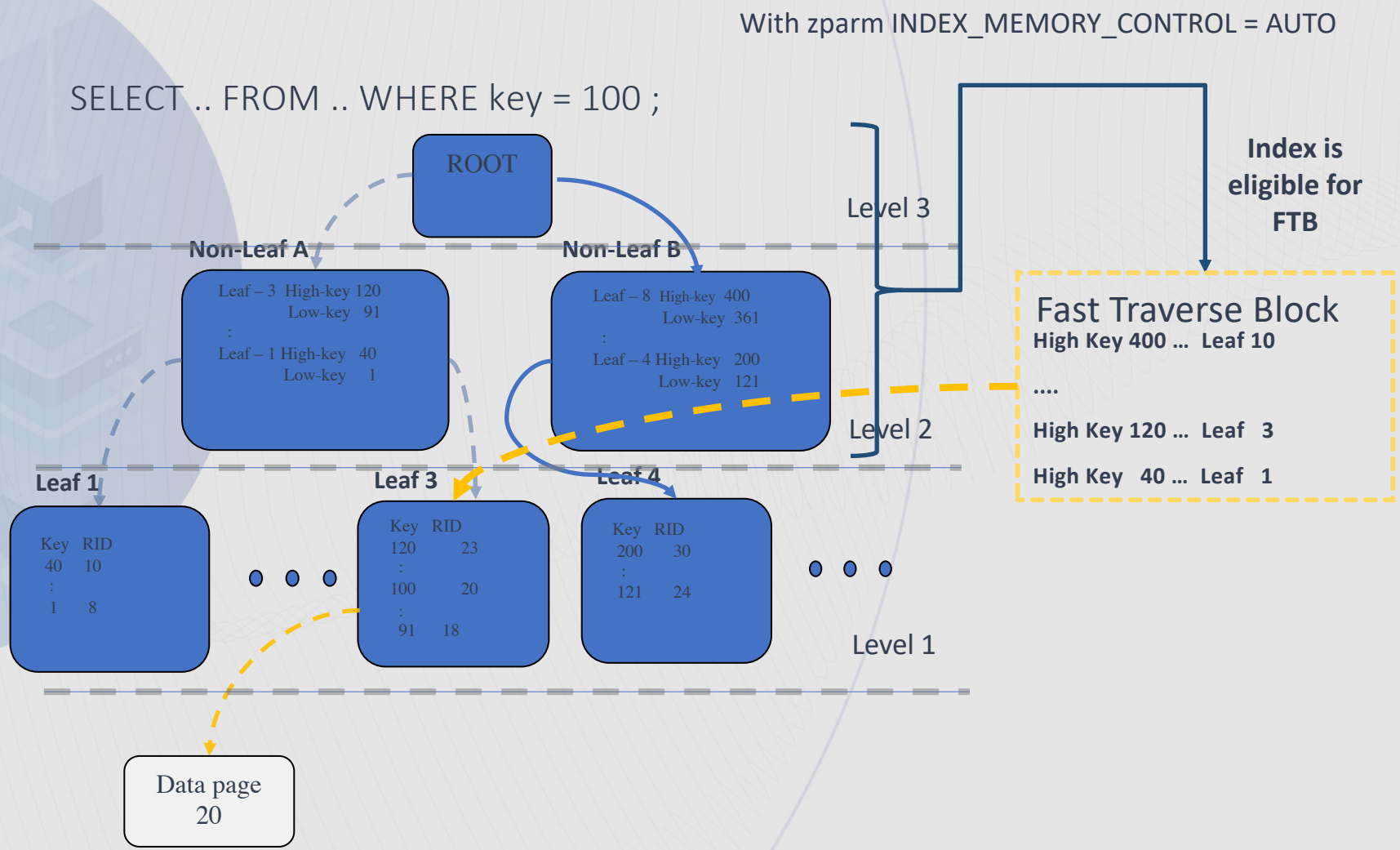
CPU saving from more memory - Sort and sparse index

- What it is : In-memory sort
  - zparm MAXSORT_IN_MEMORY
  - Max value for in-memory sort operation per thread
    - Operate RDS SORT in memory instead of materializing in workfile
    - Very hard to estimate – start with 2-8MB to observe the improvement (CL2 CPU time improvement with ORDER BY/GROUP BY queries)
- What it is : sparse index data cache
  - zparm MXDTCACH
  - Max value for data caching for sparse index operation
    - Showing noticeable CPU reduction (10-20% range) in eligible queries with larger spare index cache value
    - Watch statistics QXSISTOR THE NUMBER OF TIMES THAT SPARSE INDEX WAS DISABLED BECAUSE OF INSUFFICIENT STORAGE and increase the default as needed

- MAXSORT_IN_MEMORY
  - 1M to SRTPOOL size, default is 1MB
  - Max allocation of storage for a query with sort to use in-memory processing
  - Monitor the usage in the statistics
  - Field Name: QISTSISC
  - CUR STOR (SORT) IN-MEM (KB)
  - The total space used for currently active in-memory work files created bythe SORT component.
- MXDTCACHE
  - 0-512 default 20MB
  - Max amount of memory in MB to be allocated for data caching for each threads to access sparse index during join.
  - Check if there are queries with PRIMARY_ACCESSTYPE = “T” or JOIN_TYPE=“P”
  - IFCID 27 for detail recording
- Monitor increase of memory usage using statistics IFCID 225 64bit shared storage QW0225ShrStg_Real
-

CPU saving from more memory - FTB (Fast Traverse Blocks)

How it works?



- FTB is outside of buffer pools.
- FTB structure contains root and non-leaf information and memory efficiency compared to the pages in the buffer pools.
- Once there are enough traversal, Db2 creates Fast Traverse Block
- The query predicate evaluation can access FTB directly with a smaller number of getpages.
- In this particular case, the getpage from index will be reduced from 3 to 1.
- In addition to random select, FTB can be effective to random index access for insert, update or delete with small numbers of structure modification

CPU saving from more memory -FTB (2)

Where to Look & Expectation

- Benefit shows as
  - Accounting Class 2 CPU time & GETPAGE reduction
- Cost shows in Statistics
  - DBM1 SRB (zIIP) time, possibly IRLM SRB time and NOTIFY

Accounting CL2 CPU time

Accounting IX BP GETPAGE count

Statistics CPU DBM1 zIIP time

Statistics NOTIFY& IRLM CPU time

- Improvement from FTB should be visible in the accounting as class 2 CPU time (= Db2 CPU time)
- If FTB is effective, numbers of getpages from index buffer pools should be reduced as well
- Once FTB is enabled, the daemon tasks monitors candidates and create/drop FTBs. This is accounted in DBM1 address space CPU time and 100% eligible for zIIPs.
- In data sharing environment, creation and index structure updates triggers NOTIFY SEND/RCV increase and possibly corresponding IRLM CPU time

Reference : Latest Important Maintenance for FTB

Db2 for z/OS News from the Lab : search with #Db2Znews

Take a new look at fast index traversal (FTBs) in Db2 12

https://community.ibm.com/community/user/hybriddatamanagement/blogs/paul-mcwilliams1/2020/10/08/new-look-ftb-db2-12

APAR / PTF	Correction for...
PH19484 / UI67068	Handling variable-length index keys with include columns
PH21916 / UI68631, PH29336/ UI71351 PH36531/ (OPEN)	FTB p-lock handling at various index operations in data sharing environments
PH25801 / UI70116	Timing window between mass-delete and FTB creation
PH26109 / UI70271	Error handling of SYSIBM.SYSINDEXCONTROL entries
PH26845 / UI70523	Avoid a loop in insert/delete against the index with FTB
PH28182 / UI71784	Improved index look-aside with FTB when the index is updated sequentially for SQL insert or delete operations
PH35596 / UI74814	Handling a timing issue during creation of FTB in data sharing environments

- List of the most important FTB related APARs to apply, at the time of writing. It is not a comprehensive list of the APARs and PTFs with FTB, however, if you apply them, all other relevant APARS and PTFs will be pulled in.
- Highly recommend to use the up-to date maintenance to utilize FTB feature.
- FTB FIXCAT DB2FTB/K

CPU saving from more memory – Buffer Pool Parameters

- What it is : Buffer pool options
  - Buffer pool simulation SPSIZE to determine reasonable VPSIZE
    - Reduce numbers of I/O and increase residency time. sync I/Os and associated CPU cost - use buffer pool simulation or adjust VPSEQT
  - PGFIX(YES/NO)
    - Page fix buffer pools – CPU saving during I/O & CF operations
    - Use for buffer pool with high I/O intensity or with high rate of GBP operations
  - FRAMESIZE (4K, 1M,2G)
    - Large frames (1MB/2GB) – CPU saving during accessing the pages in buffer pools
    - Use for buffer pool with high get page intensity
  - PGSTEAL(NONE) – Further CPU saving during accessing the pages at in-memory buffer pools
    - Use for the object with high getpage intensity with steady size
    - Object getpage intensity = GETPAGE per object (RTS) / NACTIVE
    - An 8% class 2 CPU time reduction and a 7% reduction in elapsed time was observed for the benchmark run that used PGSTEAL(NONE) compared to PGSTEAL(LRU).

- ALTER BUFFERPOOL command
- SPSIZE and SPSEQT
  - Simulated additional buffer pool size
- PGFIX (YES/NO)
  - NO is default
  - Page fix during I/O operations
- FRAMESIZE (4K, 1M, 2G)
  - 4K is default
  - 1M/2G requires z/OS LFAREA setup
  - 1M/2G requires PGFIX = YES
  - 2G cannot be used for PGSTEAL(NONE)
- PGSTEAL(LRU, FIFO, NONE)
  - LRU as default

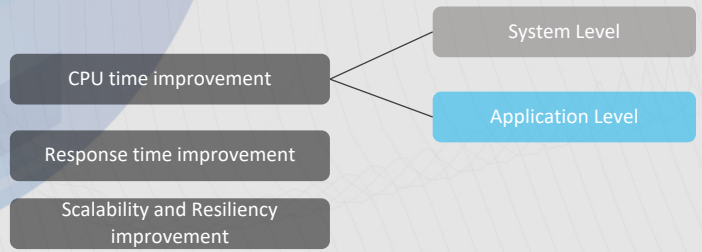
Reference : PGFIX (YES) and 1MB Page Frames

	SIZE	Get Page	Sync Read	Pre-Fetch	Write	Hit Ratio	I/O Intensity	GP Intensity	PG Fix	1MB
BP0	3K	138	0	0	0.06	100%	0	5		
BP1	524.3K	1496.3K	0.03	0	589	100%	0	285	Y	Y
BP2	2097K	160.4K	404	0	402	100%	0	8		
BP3	524.3K	93.6K	2101	35300	197	98%	7	18	Y	Y
BP4	2097K	40.9K	9873	2530	433	76%	1	2	Y	

- PGFIX (YES): recommended for high I/O intensity buffer pools – BP3 and BP4
- 1MB or 2GB page : recommended for high getpage intensity buffer pools, such as BP1 & BP3
- Recommendation : use PGFIX(YES) and 1M or 2G frames for BP1, BP3 and use PFGIX(YES) and any size of frames for BP4
- Run buffer pool simulation for BP4 to see SYNC read can be reduced

CPU Improvement Opportunity

Application level



Reference – Basic cost of SQL

- Factors influences cost of a SQL statement
  - Number of columns and types of columns
    - TIMESTAMP > CHAR
  - Number of indexes updated for insert/delete
  - Number of rows qualifying
    - Number of rows in table * product of FF of all predicates,where FF = Filter Factor
  - Number of predicates evaluated
  - Number of pages scanned
    - Effectiveness of index and data look aside
  - Number of rows scanned by Data Manager
    - Number of rows in table * FF of chosen access path
  - Number of rows returned

Reference – General Application Best Practice

General Application Programming

- Locking and concurrency - Be sensitive of isolation and holding the resource
  - Transaction isolation should be CS unless necessary to use RR/or RS.
  - JDBC uses the default as cursor with hold, which would hold the resource and threads across the commit boundary. Use cursor without hold unless necessary
  - Close the cursor, statements or connections as needed to release resources, locks and memory associated with.
  - Do not use AUTOCOMMIT (ON), which is default of Java, but use frequently enough commit to release the locks.
- Use multi-row insert or update as needed to improve insert performance.
- Use parameter markers to match statement cache or literal replacement
- Retrieve only the needed columns - don't use SELECT * unless necessary
- Don't use the mismatch data type between Java and Db2

General Application Monitoring

- Set ClientInfo to record the granular accounting info to identify the transaction
- Monitor long running UR and readers via Db2 messages:
  - DSNR035I that is controlled by URCHKT
  - DSNB260I through LRDRTHLD
  - DSNJ031I URLGWTH.
- Monitor timeout and deadlock messages OR use IFCID 196 (TIMEOUT record) and IFCID 172 (DEADLOCK record) in the statistics to review contention for resources, packages, statements, holders, and types of locks.

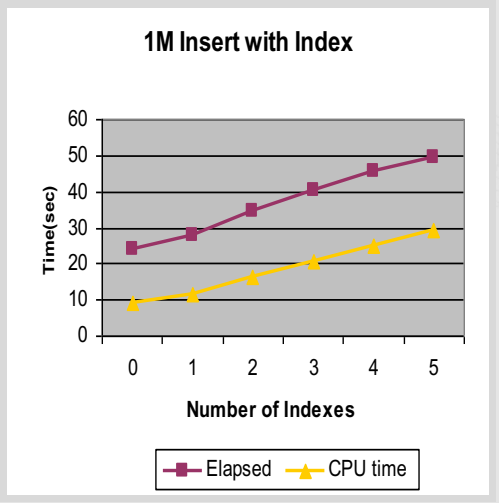
CPU - Application Interface : Thread Reuse & Release Deallocate

- What it is : RELEASE (DEALLOCATE) BIND/REBIND option
  - Keep resources across commit
  - Effective combined with CICS or IMS thread reuse or batch applications
  - DDF High Perf DBATs are introduced in Db2 10
- Why?
  - Combined with thread reuse, eliminate the copy processing for package, parent locks, statement execution information
  - Range of a few % to 15% CPU saving for frequently executed transactions
- Consideration
  - Challenge in REORG, REBIND, DDL
    - Break-in through PKGREL_COMMIT, then REBIND phase - in

- REL(DEALLOCATE) keeps resources across commit. they are,
  - Table space or partition locks
  - Packages
  - Statements for static
  - Lookaside, prefetch tracing
- V11 introduced
  - Break-in
  - internal optimization to avoid degradation due to accumulated information
- V12 Function level 505 introduced
  - REBIND phase-in
- Details are Deep Dive into RELEASE(DEALLOCATE) and KEEPDYNAIC(YES) in 2012 NA IDUG by Akiko Hoshikawa

CPU saving – Index Option

- What it is : INCLUDE non-key columns in a unique index
  - Include additional columns that are not part of a unique constraint
- Why
  - Reduce the number of indexes to improve INSERT
  - Achieve index-only access



- What it is : EXCLUDE NULL KEYS
  - Suppress including NULL rows from indexes
- Why
  - Reduce INSERT, Utility and index scan performance depending on the ratio of NULL value
  - A case study with 50% NULL value -> 50% REORG index, 5-20% insert CPU time, 5-10% select using the index

- INCLUDE index and EXCLUDE NULL KEYS is introduced in V11 New Function Mod
- INCLUDE columns in a unique index can be used to enable index-only access.
- Default is INCLUDE NULL KEYS and an index entry will be created when every key column contains the NULL value
- LASTUSED column in SYSIBM.SYSINDEXSPACESTATS to identify the indexes which are used for “index access”, not for insert
- Db2 11 also added pseudo index clean up as a background task to improve overall index performance
  - Enabled as default

CPU saving – Index On Expression

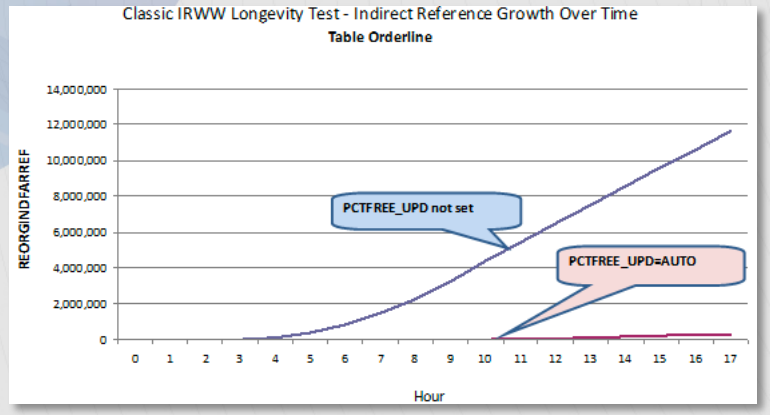
- What it is : Stores the results of the expression in the index
- Why
  - Orders of magnitude improvement if a predicate using such an index
  - Extra cost in index maintenance

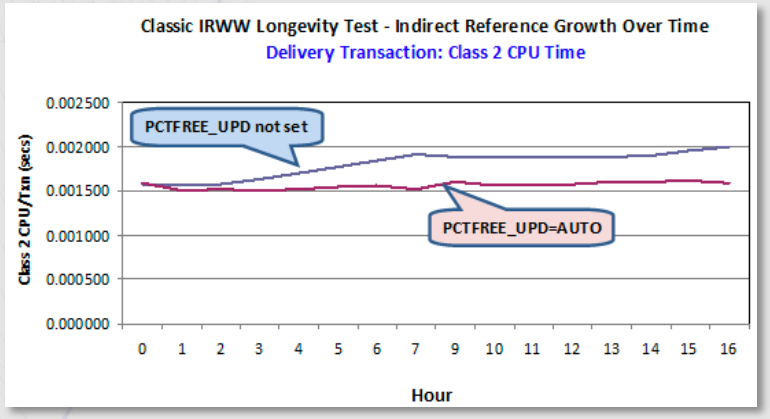
```
Create INDEX
        CREATE INDEX UPPER_NAME
        ON emp
        (UPPER (lastname, 'EN_US'),
        UPPER (firstname, 'EN_US'))

Query
        SELECT id  FROM emp WHERE
        UPPER (lastname, 'EN_US') = 'SMITH'
        AND
        UPPER (firstname, 'EN_US') = 'JOHN'
```

- Introduced in Db2 9
- Significant CPU saving for the matching application
- Extra cost in index maintenance and not suitable with heavily updated indexes
  - Load, Insert, Update on key value, Rebuild Index, Check Index, and Reorg tablespace
  - Not Reorg index as expressions are evaluated in Insert or index rebuild
  - Not eligible for zIIP offload

CPU saving – Reduction of Overflow Records

- What it is :
  Reduce overflow record creation through update statement by saving enough space during the insert, LOAD or REORG.
  - PCTFREE x% FOR UPDATE y% per object level or PCTFREE_UPD zparm
    - X% free space per data page reserved by LOAD / REORG
    - Y%. Free space per data page reserved by INSERT, LOAD
    - FOR UPDATE -1 : Db2 to determine the value using internal RTS UPDATE history
  - PCTFREE_UPD xx/AUTO is system default
- Why?
  - Overflow record adds CPU time to access – extra locking and getpages. Reduce the need of running REORG

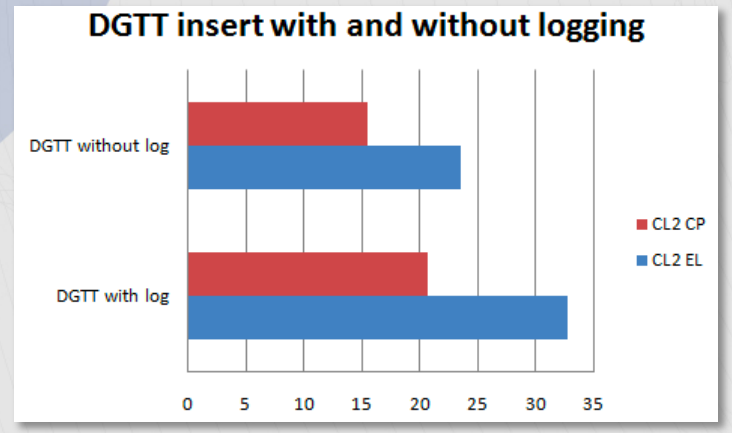


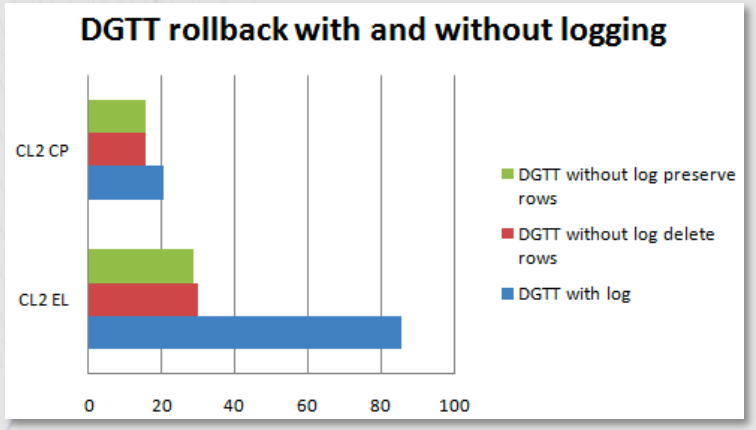


- Overflow record (indirect reference) is created during UPDATE against variable length rows or compressed rows
- **Impact caused by indirect references**
  - Additional getpages, potentially additional I/Os to the overflow pages
  - Lower clustering
  - REORG TS is necessary to remove indirect references
- PCTFREE FOR UPDATE is introduced in V11 NFM
- Zparm PCTFREE_UPD

CPU saving – DGTT NOT LOGGED

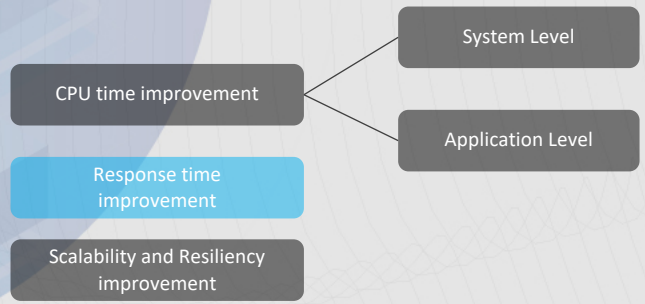
- What it is : NOT LOGGED option – default is LOGGED
  - DECLARE GLOBAL TEMPORARY TABLE dgtt NOT LOGGED
    - ON ROLLBACK DELETE ROWS
    - ON ROLLBACK PRESERVE ROWS
- Why
  - Less CPU from log processing in DGTT updates
    - >20% CPU/elapsed time improvement seen in DGTT INSERT, UPDATE or DELETE
  - Faster rollback/error processing
    - 60% elapsed time reduction rolling back 5 million insert (and CPU time from system address space time)





- DGTT NOT LOGGED option was introduced in V11 New Function Mode
- Requires DECLARE statement updates
- NOT LOGGED ON ROLLBACK DELETE ROWS:
  - This option specifies that you do not want logging to occur for this table, and during ROLLBACK or ROLLBACK TO SAVEPOINT, all rows in the DGTT are deleted if anychange was made since the last COMMIT.
- NOT LOGGED ON ROLLBACK PRESERVE ROWS
  - This option specifies that you do not want logging to occur for this table, and during ROLLBACK or ROLLBACK TO SAVEPOINT, the rows in the DGTT are preserved as they

Elapsed Time Improvement Opportunity

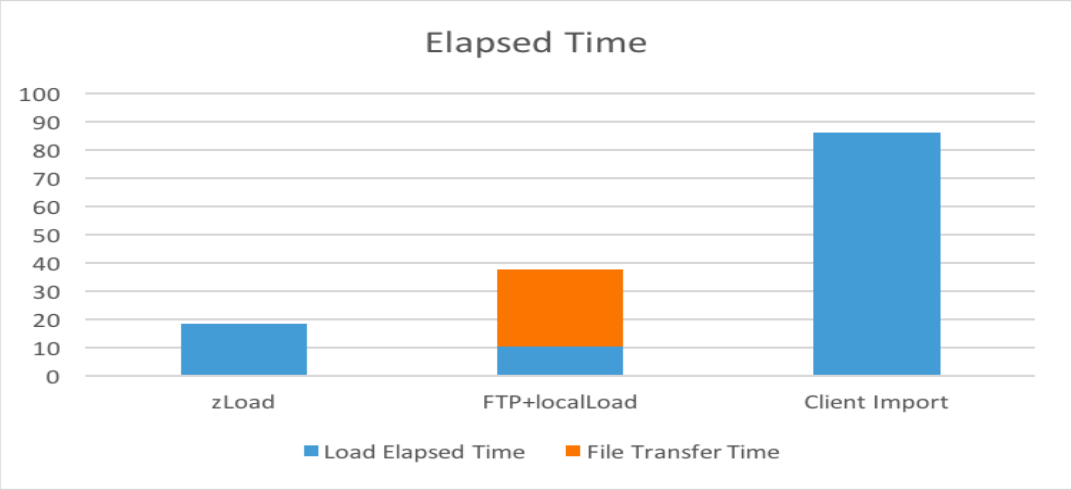


Elapsed time Improvement Opportunity

- Integration with System Z
  - zEDC and z15 on-chip compression
  - Huffman compression
  - zHylerwrite
  - zHyperLink
  - SMC-D
  - Async Cross invalidation (enabled as default)
- Db2 features
  - zLOAD
  - Inline LOB
  - IAG2

Response time (Elapsed time) – zLOAD

- What it is : DRDA Fast Load (zLOAD)
  - Execute Db2 LOAD utility statement from a remote client program
- Why
  - Quick and easy loading of data from files at remote locations

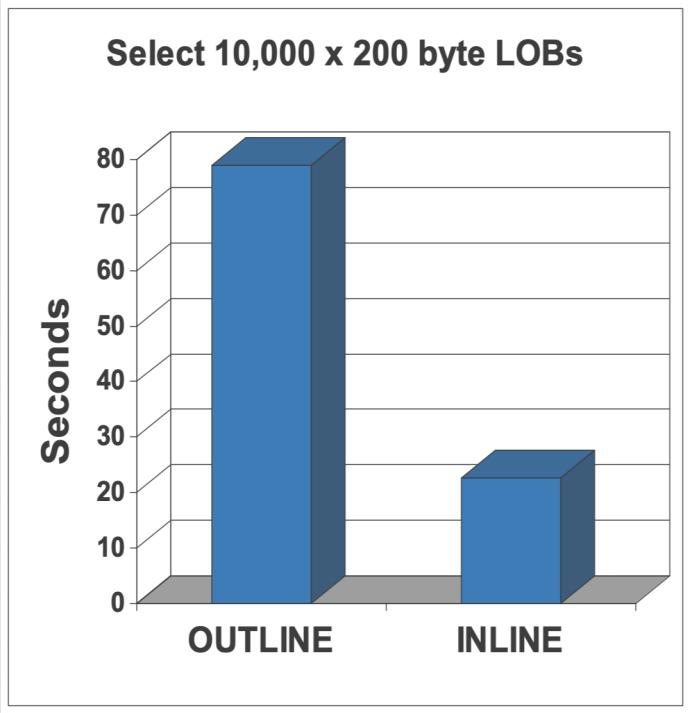


- zLOAD is supported
  - CLI, JDBC, CLP
  - Requires Data Server Client V11.1 fix pack 1 or above
- Detail description in Db2 12
- https://www.ibm.com/docs/en/db2-for-zos/12?topic=tables-loading-data-drda-fast-load-zload

Response time (Elapsed time) – Inline LOB for mostly <32K LOBs

- What it is
  - Store all or a part of small LOB into a base table
- Why
  - Elapsed & CPU time reduction for mostly inline LOB (mostly <32K length, but a few LOBs are big)
  - Not suitable for average length >32K

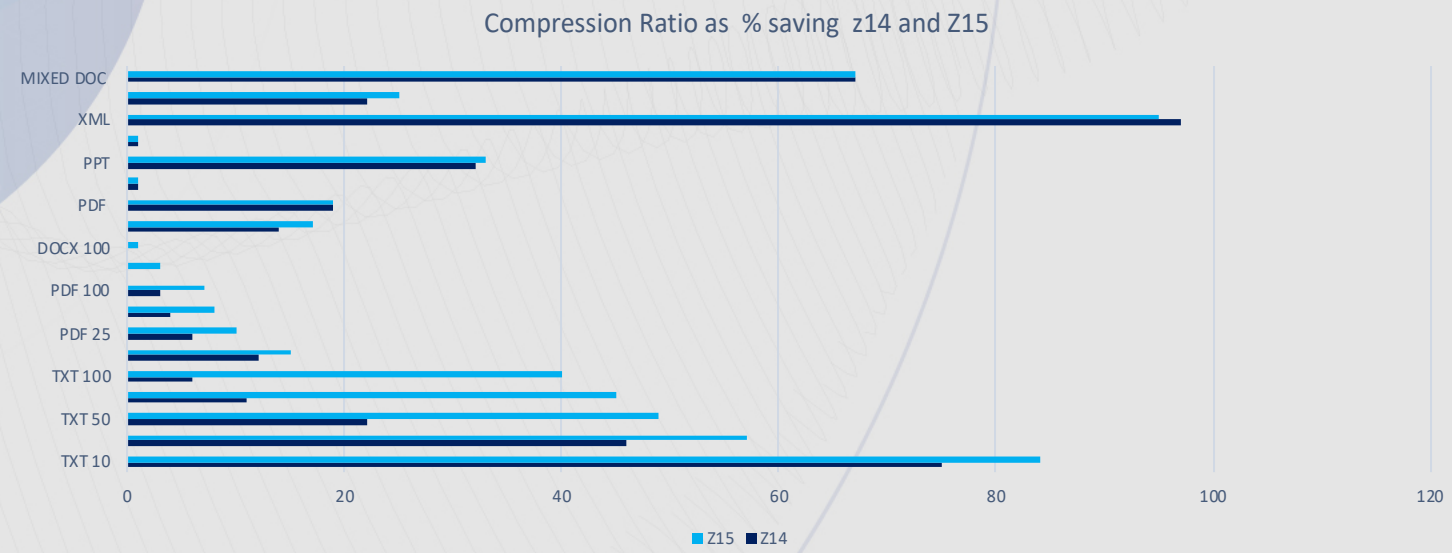
Elapsed time in random select



- CREATE or ATLTER 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

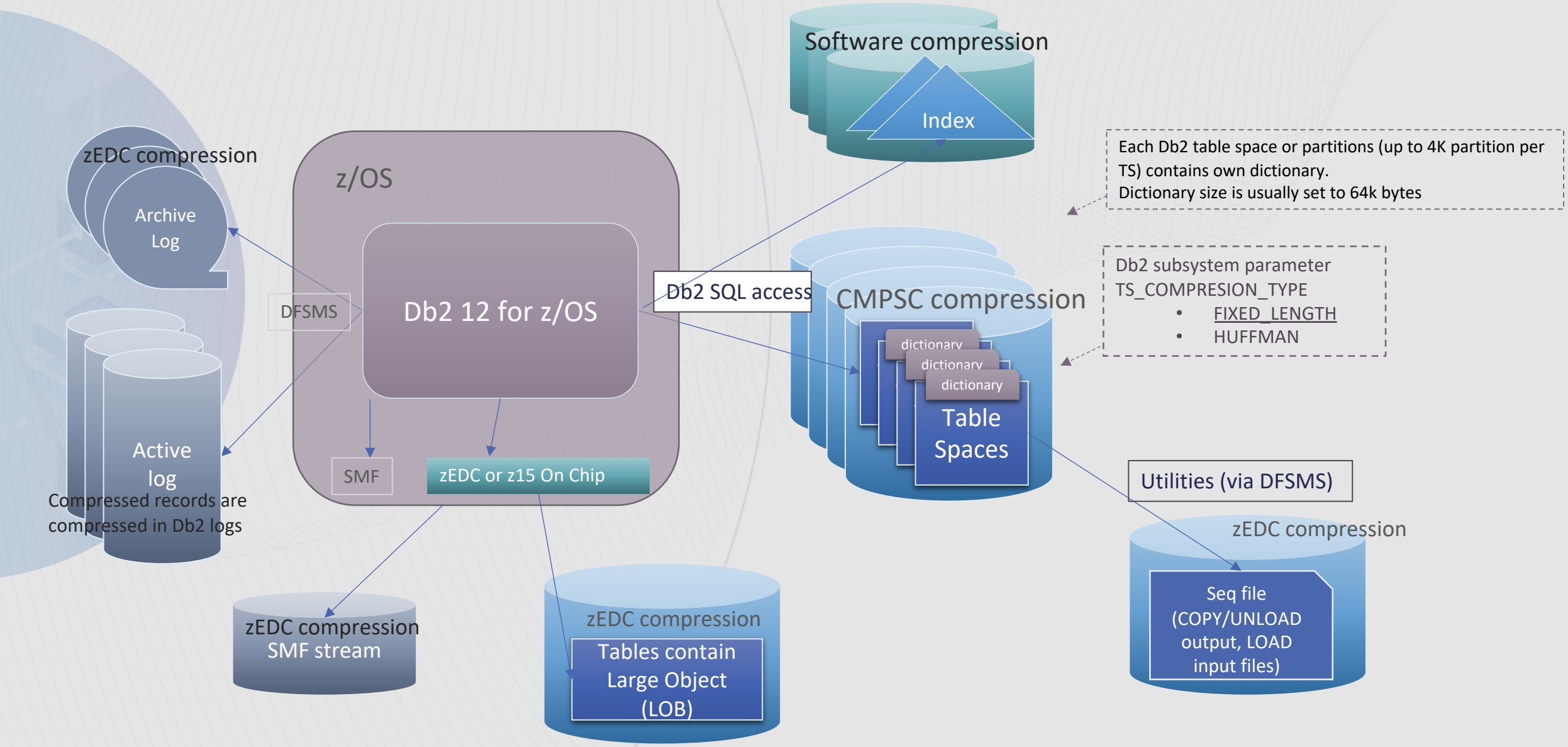
Response time (Elapsed time) – z15 On chip compression

- What it is : zEnterprise Data Compression (zEDC) or z15 on chip compression
  - LOB compression in V12 FL500
  - Utility sequential files - output of COPY, UNLOAD, input of LOAD, input of RECOVERY
    - Customer has seen 60-70% disk space saving
  - Db2 Archive log
  - Db2 trace in SMF (SMF streaming required)
- Why ?
  - Disk space saving
  - Elapsed reduction for accessing large objects which are compressed well
- z14 vs. z15 LOB compression
  - Better Compression Rate with z15 z
  - The set of various LOB shows average 32% better compression rate with z15 compared to z14, results in better elapsed and CPU time processing LOB



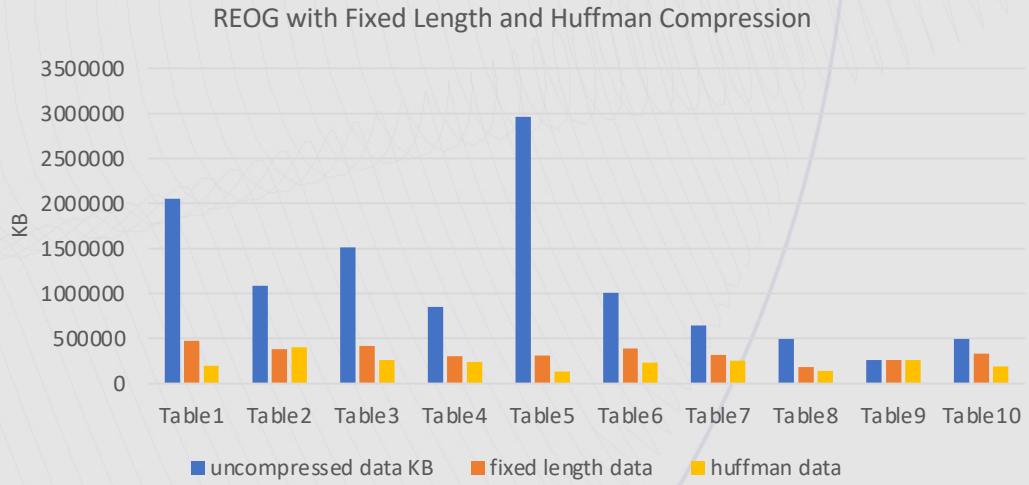
- zEDC supported in zEC12 to z14
- Db2 user table LOB compression requires,
  - Db2 12 FL500 and above
  - zEC12 GA2 and above with zEDC Express feature
  - UTS with COMPRESS YES attribute
  - Total length of LOB must be larger than defined page size
- To compress Db2 directory LOB requires additionally,
  - DSN6SPRM COMPRESS_DIRLOB YES (NO is default)
    - Using a customer's data, we saw up to 80 % page saved
- DSN1COMP to estimate the LOB compression space saving
- The examples file types typically compressed well
  - DOCX, XML, some PDF
- The examples of file types typically do not compress well
  - JPG, PNG, PPT, XLSX

Reference : Db2 and Data compression (1)



Elapsed time Improvement Opportunity – Huffman Encoding

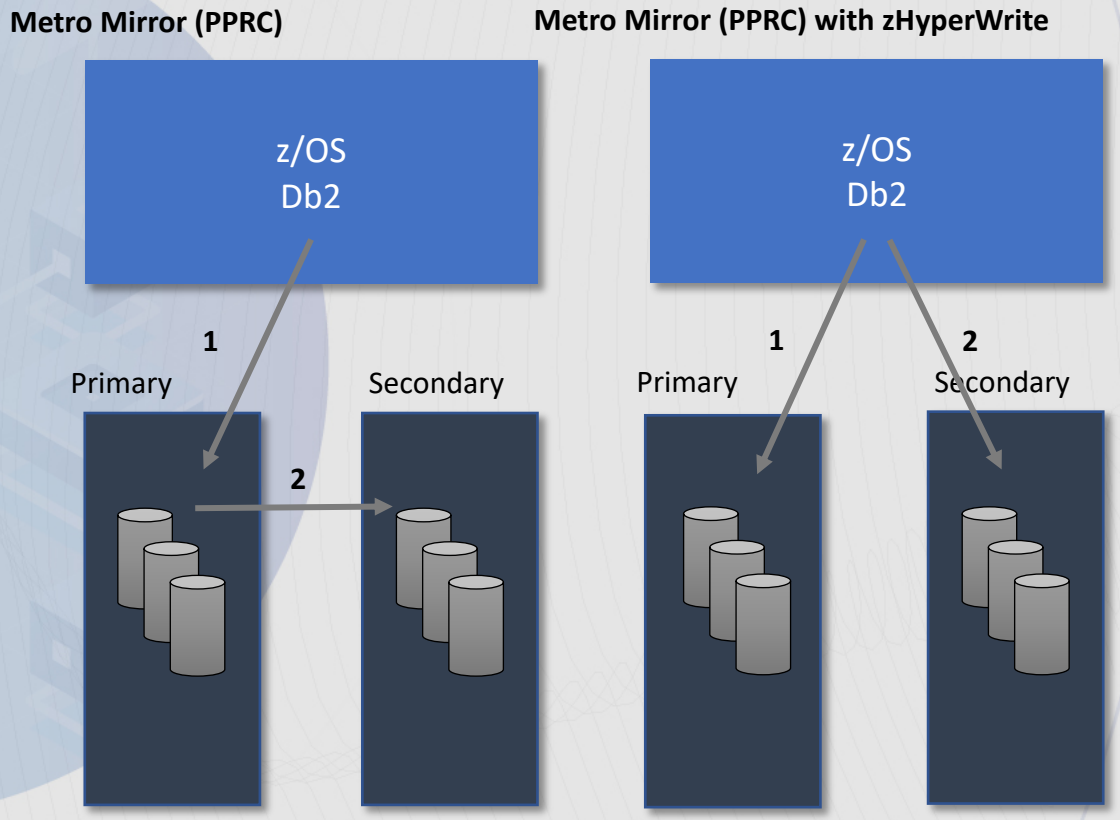
- What it is : TS compression algorithm introduced in Db2 12 and z14 above
  - Zparm TS_COMPRESION_TYPE
  - Observed 10-30% disk space saving with Huffman encoding
- Why
  - Possible elapsed time saving for sequentially accessed objects
  - Less need of disk and buffer pools
- Factors influencing the performance
  - Compression ratio, data access pattern, ratio of SELECT and predicate list vs. total numbers of columns, impact to APS and processor types



- Huffman compression encoding is introduced in Db2 12 function level 504 as system level option
- TS_COMPRESION_TYPE
  - FIXED_LENGTH
  - HUFFMAN
- Db2 V12 FL509 introduced object level Huffman compression option for easy transition
- Catalog update to indicate which encoding TS is using
- PH34808 : DSN1COMP enhancement to compare the estimate space saving converting from FIXED_LENGTH to HUFFMAN
- No support for partial decompression and possible CPU degradation depending on the queries

Reference : ZHYPERWRITE and ZHYPERLINK

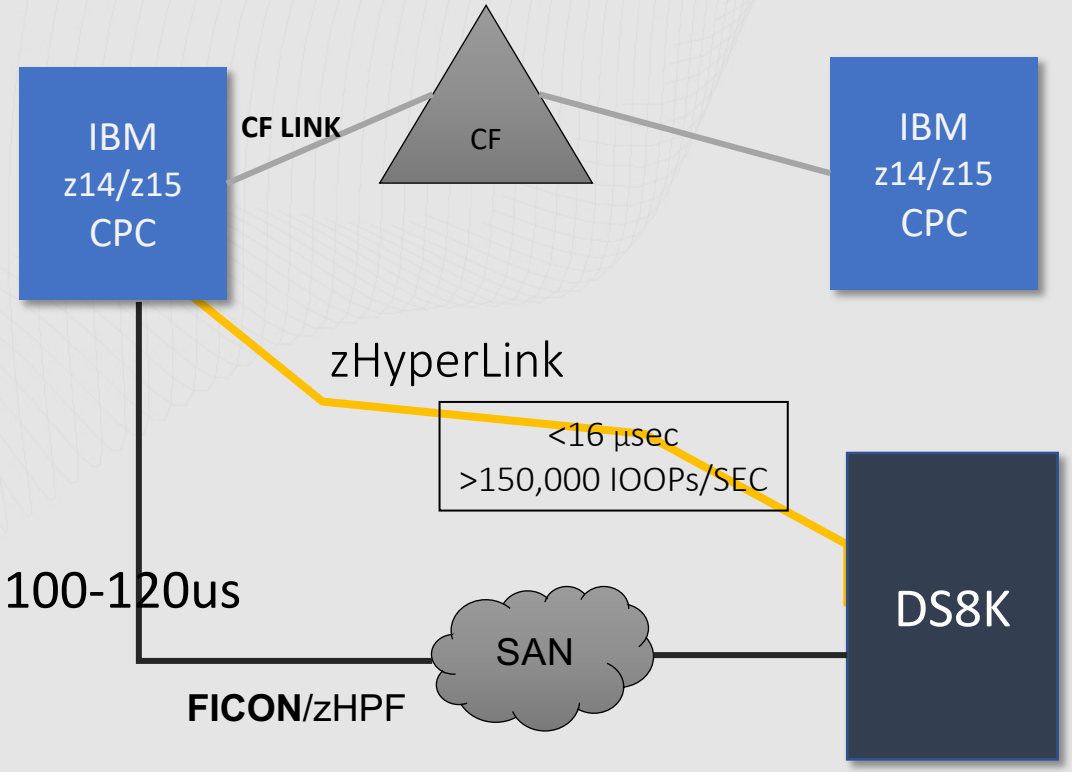
ZHYPERWRITE



Active log under PPRC

REMOTE_COPY_SW_ACCEL = ENABLE/DISABLE

ZHYPERLINK



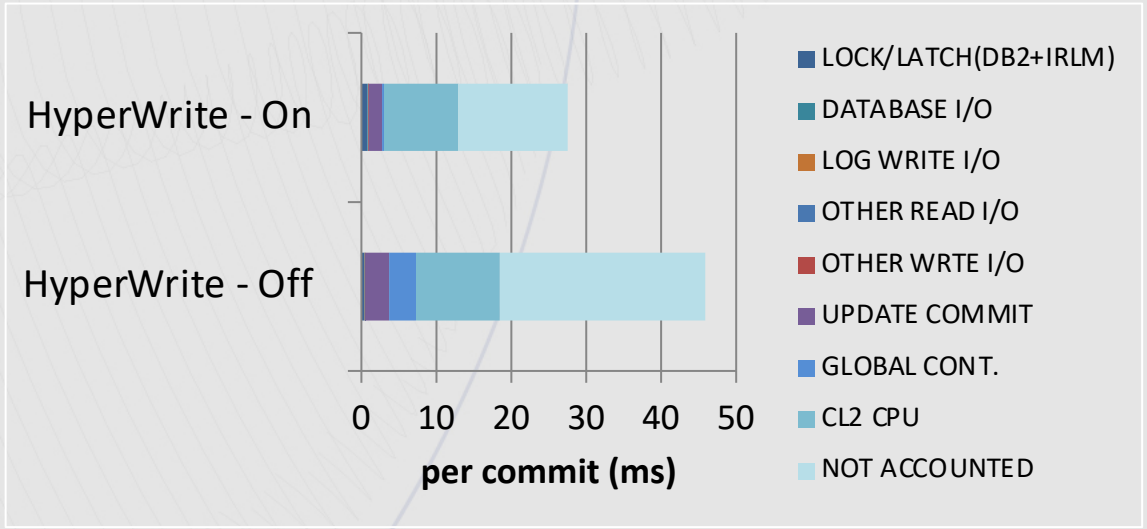
Sync I/O : DB read and Active log write

ZHYPERLINK= ENABLE/ACTIVELOG/DATABASE/DISABLE

Elapsed time Improvement Opportunity – zHyperWrite

- What it is : zHyperWrite
  - Utilize software control remote copy process for active log output in peer to peer remote copy (PPRC) environment
  - Zparm REMOTE_COPY_SW_ACCEL
    - DISABLE/ENABLE (default is DISABLE)
- Why
  - Elapsed time reduction for update intensive workload

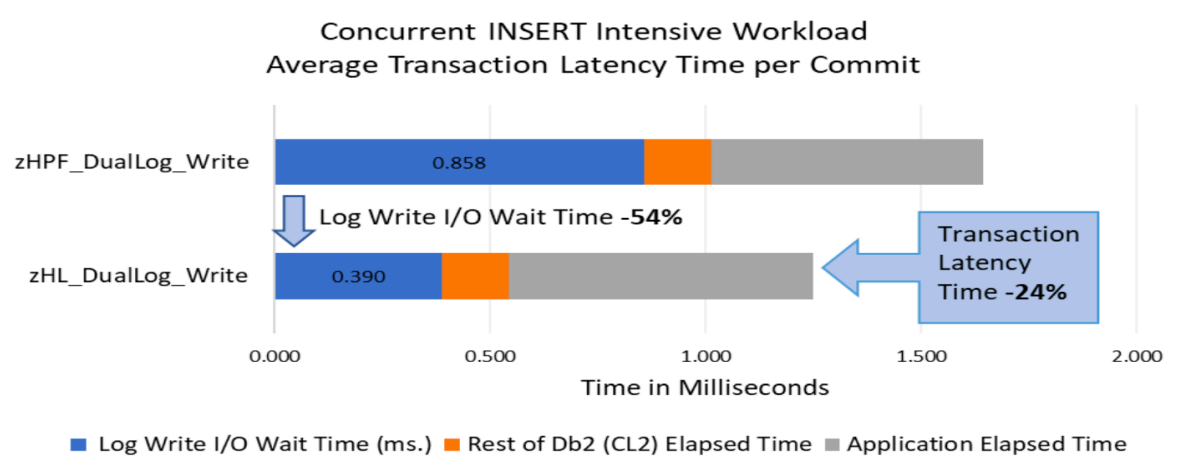
- Customer example: running multiple updates jobs with 20 updates per commit
- 43% Reduction in Update Commit time and 40% DB2 elapsed time reduction



- Applicable for PPRC environment to speed up active log write
- Parallel log write in primary and secondary to reduce the response time
- Customer's experiment showing 20-40% elapsed time saving for update intensive batch applications

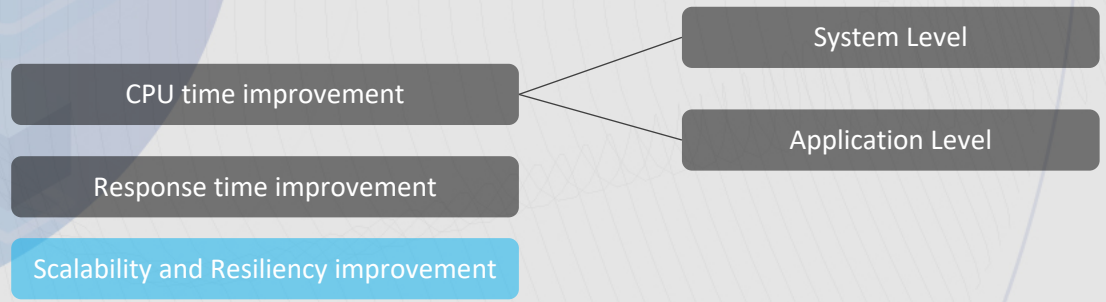
Elapsed time Improvement Opportunity – zHyperLink

- What it is : zHyperLink
  - Reduce I/O latency for sync I/O for 4K pages cached in DS8K
    - Data base sync read I/Os
    - Active log write I/Os
  - Zparm ZHYPERLINK
- Why
  - Significantly cut down the I/O response time in Db2 transaction
    - Data base read intensive workload (for read)
    - Update intensive workload (for write)
  - Latest updates
    - ZHYPERLINK write is now supported for global mirror configuration with z/OS 2.3
    - DFSMS and Db2 support the improvement of dual log write
- Trade off
  - CPU time increase



- zHyperLink is a new short distance mainframe link technology, connecting directly between the CPU and the I/O device.
- ZHYPERLINK
  - DISABLE (default)
  - ENABLE (read&write)
  - DATABASE (read only)
  - ACTIVELOG (write only)
- See the IDUG article for details
  - https://www.idug.org/browse/blogs/blogviewer?blogkey=1701882d-c5d4-4819-9242-7f21f9a8d10a
- zHyerlink parallel log write support
  - DFSMS OA57833, OA58134,OA59581
  - Db2 12 PH29407
- FIXCAT HYPERL/K

Resiliency, Scalability Updates



Resiliency / Scalability Improvement Opportunity - DSMAX

- Concurrently open data set
  - APAR PH27493/PH33238 : pro-actively close datasets that were only opened for utility processing and prioritize closing utility-only datasets when DSMAX is hit. Benefit is to keep the number of concurrently open data set lower
- Future release of z/OS
  - Consider to deliver the enhancement to move portion of below the bar storage to above the bar
  - Db2 to plan to exploit the feature

- Db2 APAR PH09189
  - Reduce risk of hitting DSMAX
    - Increase the amount of buffer prior to hitting the DSMAX limit to drive dynamic close of datasets earlier, allowing more time for asynchronous close processing to occur
  - Prevent application failure when hitting DSMAX
    - Drive synchronous closes of datasets prior to retrying the open request
- Db2 APAR PH27493 (now APAR PH33238)
  - Proactively close datasets that were opened for utility processing
  - Prioritize closing utility-only datasets when DSMAX is hit

Resiliency / Scalability Updates

- Db2 tends to challenge z/OS limitation, especially with DRDA clients
  - Recent challenges exposed via DDF workloads
    - PH34378 to reduce WML request for High performance DBATS
    - PH35068   to reduce WML storage requests
    - PH34200 to improve thread deallocation after hitting profile exception
    - PH36114 (open) to reduce the impact from deallocation afer hitting POOLINAC
      - This can be mitigated by REALSTORAGE_MANAGEMENT OFF
  - RACF related
    - PH30164 to reduce Db2 latch contention class 10 (authorization latch)
- Recommendation
  - Install the maintenance & Monitor using SMF 98 (High frequency throughput statistics)
  - Utilize Db2 granular statistics interval by STATIME_MAIN PH18658

- Db2 tends to challenge z/OS limitation, especially with DRDA clients
- DDF CPU spikes due to z/OS SRM spin lock contention – PH34378 is available to reduce WML request for High performance DBATS
- DDF CPU spikes due to z/OS VSMFIX spin lock contentions – PH35068 is available to reduce WML storage requests
- DDF CPU spikes due to z/OS RSM spin lock contentions – PH34200 for profile, PH36114 (open) for pool inac behaviod change
  - This can be mitigated by Realstorage_management OFF
- PH30164 improves Db2 to release class 10 latch while invoking RACF during thread termination
  - FYI: OA60285 to correct GRS orphaning cells which delayed ENQ/DEQ processing
- RMF III CF related monitor triggered CPU spikes due to IXLSHELL spin lock contention
  - Resolved by z/OS OA58772 RSU2006

Thank you for joining the session!

akiko@us.ibm.com

Speaker: Akiko Hoshikawa
Company: IBM
Email Address: akiko@us.ibm.com

Session code:

Please fill out your session evaluation before leaving!!!!