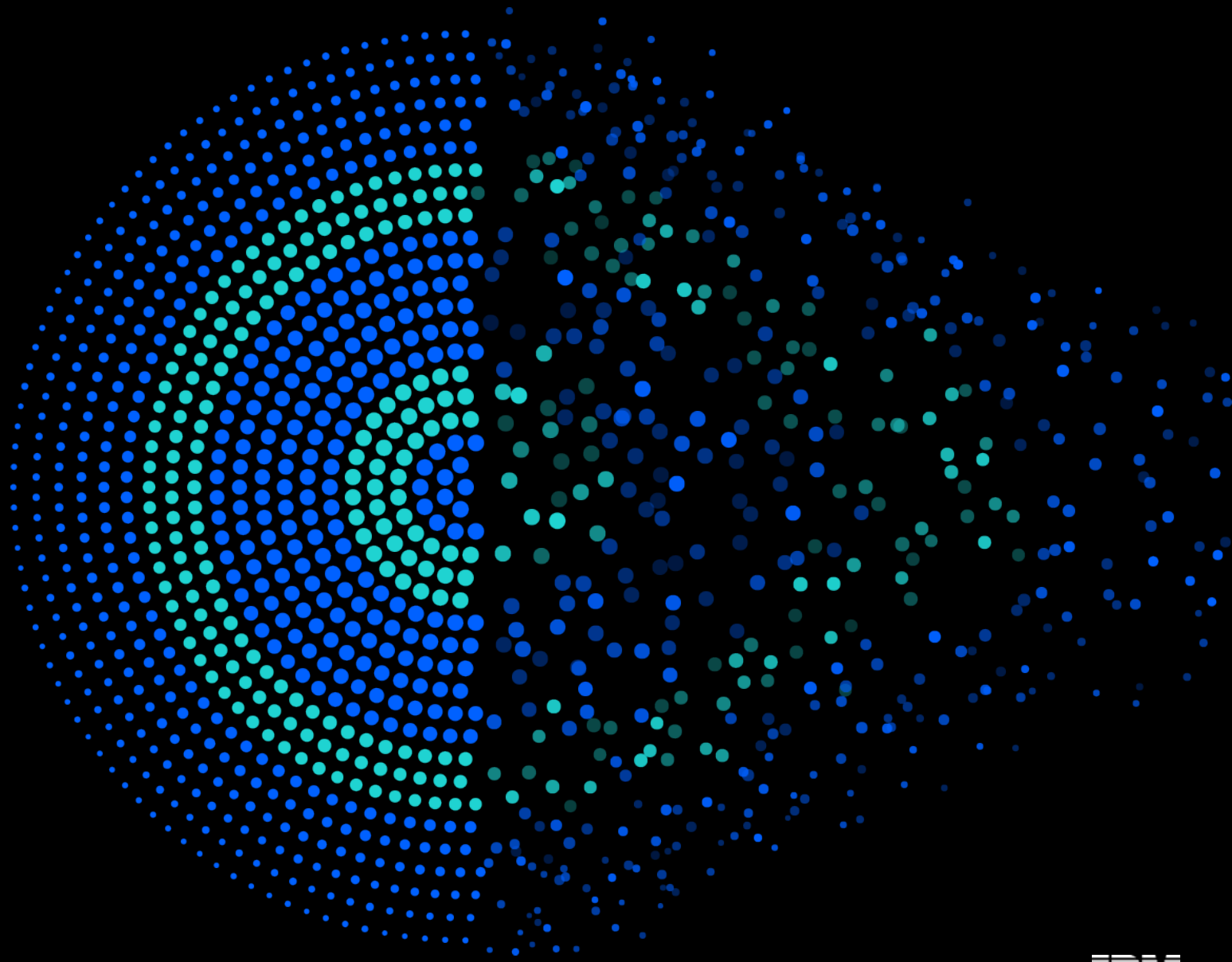
Data and AI

Db2 for z/OS:
Hot Topics and Best Practices with John Campbell – Part 2



John Campbell
Db2 for z/OS Development

IBM

Objectives

- Learn recommended best practice
- Learn from the good, bad and ugly experiences from other installations
- Provide answers to important, frequent or difficult questions

Agenda

- COPY SHRLEVEL CHANGE FLASHCOPY CONSISTENT
- Considerations when migrating from segmented to multi-part PBG UTS
- Continuous Delivery
- Software Maintenance
- Point-In-Time (PIT) Recovery in Db2 12 with SCOPE UPDATED
- Point-In-Time (PIT) Recovery before Function Level (FL) activation and/or CATMAINT
- Fast Traversal Blocks (FTBs)
- Dynamic prefetch avoidance

Agenda …

- Local Buffer Pools and Real Memory allocation
- A single local buffer pool can be spread across different size real memory frames
- XXXL Size Local Buffer Pool
- Restrictions with Buffer Pool Simulation
- Using DSNJU999 service aid
- Using Large Frame Size (LFAREA) for Local Buffer Pools
- Growth in CPU time/transaction as CPU busy increases
- Tutorial on HiperDispatch and Efficiency differences between VH, VM, VLs
- REBIND very old plans and packages to avoid Autobind
- REBIND Phase-in and thread reuse with RELEASE(DEALLOCATE)
- Data Sharing - GBP Asynchronous cross-invalidation (XI)
- Data sharing - GBP deallocation
- Data Sharing – Peer recovery
- Database Housekeeping Rules
- Turning off CICS-Db2 thread protection

Local Buffer Pool Topics

- Local Buffer Pools and Real Memory allocation
  - Db2 10 allocates as needed for both virtual and real memory
  - Db2 11 allocates virtual storage immediately according to VPSIZE, but allocates real memory as needed when virtual is referenced
    - Exception – Buffer pools with PGFIX(YES) and FRAMESIZE is 1M or 2G
  - It has always been strongly recommended that Db2 buffer pools should be backed up 100% by real memory to avoid paging to AUX (or Flash Express)
- A single buffer pool can be spread across different real memory frame sizes
  - Especially if there is an ALTER BPOOL on VPSIZE
  - If Db2 needs to drop down to using 4K size frames, Db2 will not try to deallocate the 1M frames already allocated, just to make the frame size consistent
- XXXL size Local Buffer Pool
  - Maximum size: few 100s of GB
  - At very large size, scaling issues start to emerge because of running long (PMB) control block chains
  - For super size local buffer pools, use Contiguous Buffer Pool (PGSTEAL NONE)

Local Buffer Pool Topics …

- Restrictions with Buffer Pool Simulation
  - The maximum value for SPSIZE is documented under the -ALTER BUFFERPOOL command:
    - For 4K page size buffer pool, maximum SPSIZE is 250000000 - VPSIZE
  - At ALTER BPOOL time, Db2 performs some additional checks:
    - When SPSIZE is being altered, Db2 checks whether the alter would cause the total size of all buffers and simulated buffers to exceed a virtual storage limit of 1TB
      - This is checking against the defined sizes of *ALL* buffer pools and simulated buffer pools
      - If exceeded, Db2 will reject the ALTER of SPSIZE and will issue the message DSNB508I
    - If the simulated pool is being allocated or expanded, Db2 will check if the “soft limit” on buffer pool size, which is 2x real memory size on LPAR, will be exceeded
      - If so, Db2 will reject the ALTER of SPSIZE
      - Db2 is checking against the memory used for the allocated buffer pools and the control block memory used for the simulated buffer pools
      - Currently, 14K simulated buffers uses up 1MB of control block memory
      - If total buffer pool memory is greater than 2x real memory on LPAR, Db2 will issue message DSNB612I

Local Buffer Pool Topics …

- DSNJU999 service aid
  - Use to change size of local buffer pool and/or change pagefix attribute before starting Db2

```
//JOBLIB DD  DSN=USER.TESTLIB,DISP=SHR
//       DD  DSN=DB2A.SDSNLOAD,DISP=SHR
//SETBP1   EXEC PGM=DSNJU999,PARM='BP01 000007D0 N'
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSUT1   DD DISP=SHR,DSN=DSNC000.DB2A.BSDS01
```

  - This will change the VPSIZE of BP1 to 2000 and change pagefix to NO

Local Buffer Pool Topics …

- DSNJU999 service aid …
  - For DSNJU999 to work, there is a requirement that some load modules be authorized

```
//DSN7AUTH EXEC PGM=IEWL,REGION=32M,
//      PARM='LIST,XREF,NCAL,RENT,AMODE=31,RMODE=ANY,COMPAT=LKED'
//SYSPRINT DD SYSOUT=*
//SYSLIB   DD DSN=USER.TESTLIB,DISP=SHR
//        DD DSN=DB2A.SDSNLOAD,DISP=SHR
//SYSLMOD  DD DSN=USER.TESTLIB,DISP=SHR
//SYSUT1   DD UNIT=SYSVIO,SPACE=(CYL,(1,1))
//SYSLIN   DD *
 INCLUDE SYSLIB(DSN7SCAD)
 ENTRY DSN7SCAD
 MODE AMODE(24),RMODE(24)
 SETCODE AC(1)
 NAME DSN7SCAD(R)
 INCLUDE SYSLIB(DSN7GBPD)
 ENTRY DSN7GBPD
 MODE AMODE(24),RMODE(24)
 SETCODE AC(1)
 NAME DSN7GBPD(R)
 INCLUDE SYSLIB(DSNJU999)
 ENTRY DSNJU999
 MODE AMODE(24),RMODE(24)
 SETCODE AC(1)
 NAME DSNJU999(R)
//
```

Local Buffer Pool Topics …

- Using Large Frame Size (LFAREA)
  - Problem
    - Combination of
      - Misunderstanding and/or mismanagement of the LFAREA
      - Improper sizing
    - Do not implicitly rely on the breaking down of unused LFAREA memory to satisfy demand for 4K size frames
      - Unused LFAREA cannot be used to satisfy demand for preferred 4K size frame requests as used by Db2

Customer Example

total **GB**	%	256.00
z/OS resrv	1.6%	4.00
QUAD	12.5%	32.00
PLAREA	12.5%	32.00
LFAREA	**35.0%**	**89.60**
4k	Remainder	102.40
	Db2 used	101.32
	leftover	1.08

Local Buffer Pool Topics …

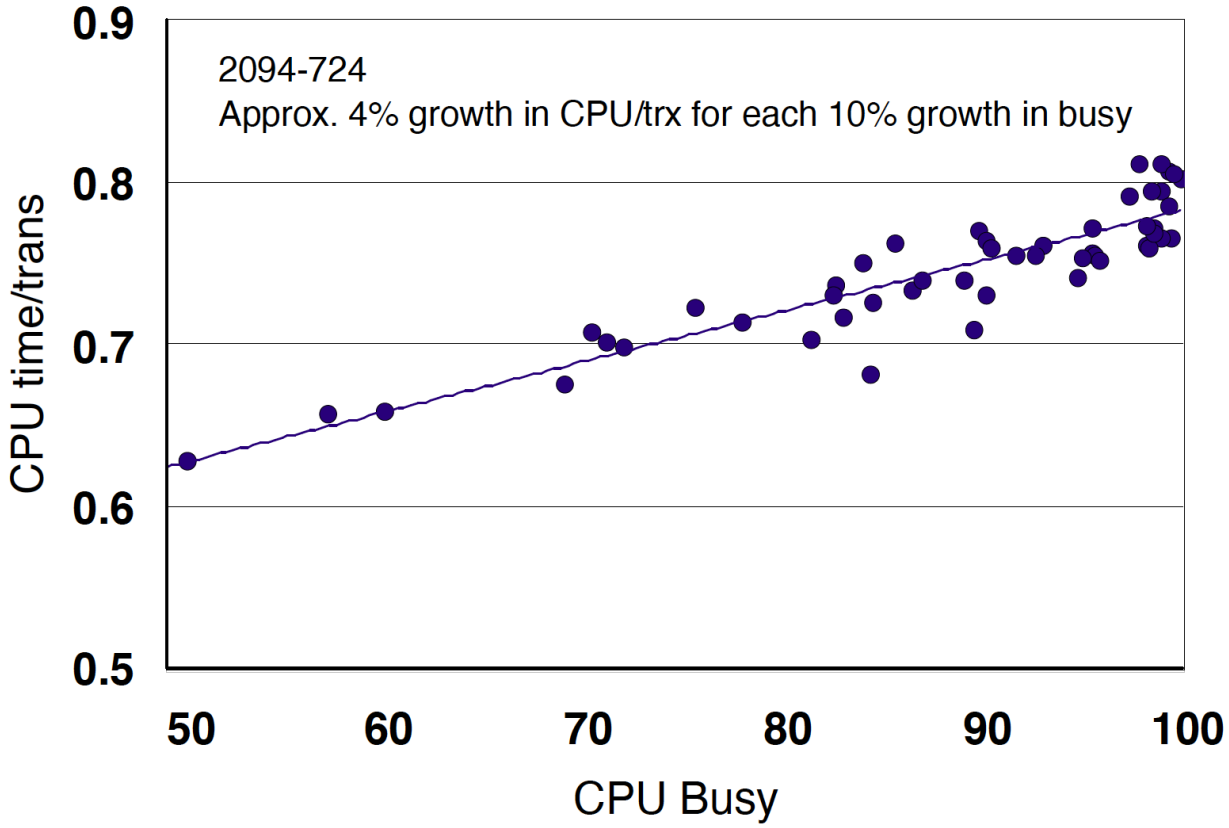
- Using Large Frame Size (LFAREA) …
  - Recommendations
    - #1 priority is to first ensure there are enough 4K size frames, both preferred and non-preferred
      - Avoid any paging and to avoid the breaking down of unused PLAREA, QUAD, or LFAREA
    - When planning the real memory budget must account for the following z/OS areas which are a percentage of the LPAR definition: z/OS reserved, QUAD, PLAREA
    - LFAREA setting should be set to minimum (what you need, what you can afford)
      - Do not over allocate
      - Size based on total buffer pool requirement for 1MB frames
      - Set as an absolute number and not as a percentage
      - Should be viewed as a high-water mark

Growth in CPU time/transaction as CPU busy increases

- Source of variation
  - CPU utilisation generally reflects the amount of work flowing through a fixed hardware and software configuration
    - The higher the workload rate, the higher the utilisation
  - As more work flows through a fixed configuration, the efficiency of the hardware and software is reduced
    - Smaller share of hardware resources (caches, buses) available to each work unit
    - Software manages more work units - longer queues, more contention
    - CPU time per transaction or job will grow
  - Magnitude of the effect is related to
    - Workload characteristics
      - Higher RNI workloads (as measured at higher utilisations) see higher impact
    - Size of the processor
      - Smallest N-way (say 1-4 way) are somewhat less sensitive

Growth in CPU time/transaction as CPU busy increases …

- OLTP Client Workload Example



Growth in CPU time/transaction as CPU busy increases …

- ROT: Approx. 4% growth in CPU/trx for each 10% growth in CPU busy
- Two implications for capacity planning
  1. May have less headroom on the CEC than you think
  2. When moving a workload, it may not fit in the new container
- Example
  - Assume a workload is running at 50% busy on a 2000 MIPS box
    - Without factoring in utilisation effect, it will be called a 1000 MIPS workload
    - In fact, it may be an 1200 MIPS workload when running at the efficiency of a 90% busy CEC
  - Caution:
    - There is NOT room to double this workload on the current CEC
    - If moved to a new CEC or LPAR, it will likely need a 1200 MIPS container (not 1000 MIPS) to fit
- Estimating the impact - conservative approach
  - For a change in utilisation of 10%, plan for the capacity effect to be
    - 3% for LOW RNI workloads, 4% for AVERAGE RNI workloads, 5% for HIGH RNI workloads

Tutorial on HiperDispatch and Efficiency differences between VH, VM, VLs

- HiperDispatch (HD) plays a vital role on the latest CEC models where cache performance has such a big impact
  - PR/SM and z/OS Dispatcher interfaces are establishing an affinity between
    - Units of work and logical CPs
    - Logical CPs and physical CPs
- Impact is the increased likelihood of a unit of work being re-dispatched to the same logical CP and executing on the same or nearby physical CP
  - Optimises the effectiveness of processor cache at every level, by reducing the frequency of processor cache misses
  - By reducing the distance (into the Nest) required to fetch data
- With HD active, based on LPAR weights and number of physical CPs, PR/SM will assign logical CP as
  - Vertical High (VH): 1:1 relationship with physical CP
  - Vertical Medium (VM): 50% share of physical CP
  - Vertical Low (VL): Low share of physical CP, subject to being “parked” when not in use

Tutorial on HiperDispatch and Efficiency differences between VH, VM, VLs …

- Customer example related to zIIP pool of processors
  - Customer added 2 additional zIIP physical processor to the CEC, but did not see much better zIIP offload
  - # of logical zIIP engines and relative LPAR weights
    - Did not align with documented best practices for HD
    - Did not match with the actual demand from the respective LPARs
    - No changes were made
  - Consequences
    - Unnecessary redirect of zIIP eligible workload to the GCPs
    - GCP resource was already constrained during peak periods
    - Loss of TCO benefit and introducing elapsed time latency for application processes
  - Recommendations for this customer situation
    - Align # of logical zIIP engines and relative LPAR weight with the actual demand for zIIP capacity demand from LPAR
      - Need more dedicated zIIP capacity for the LPAR e.g.
        - Increase weight of LPAR to ensure that at least 3 VH zIIP engines are assigned
      - Should reduce zIIP redirect to GCP and increase efficiency of zIIP dispatching

Configuring zIIP processors for Db2

- What is zIIP help function ? (or “needs help”)
  - IIPHONORPRIORITY parameter in the z/OS IEAOPTxx member of SYS1.PARMLIB determines whether zIIP-eligible work is routed from zIIP engines to GCP processors when the zIIP engines do not have enough capacity
  - When IIPHONORPRIORITY is set to NO, Db2 does not allow system agents to become eligible for zIIP processing
  - General recommendations:
    - Generously configure zIIP resources and sustain that position going forward
    - IIPHONORPRIORITY should be set to YES
- However, there are a few limitations associated with tasks running on zIIPs
  - WLM Blocked Workload support which can promote CPU starved address spaces, is not enabled for zIIP processors
  - zIIP “needs help” function even when using IIPHONORPRIORITY = YES may get delayed in a zIIP-constrained environment
  - Work classified as Discretionary to WLM does not benefit from the zIIP “needs help” function
- Further Recommendations
  - Even with the zIIP “needs help” function, it is vitally important to have enough zIIP capacity, as the zIIP engine itself needs to be able to signal for help from other zIIPs or general purpose processors
  - All LPARs running a “production” Db2 subsystem should be assigned at least 1 VH zIIP engine
  - Use Importance of 5 instead Discretionary

Configuring zIIP processors for Db2 …

- zIIP capacity requirement in Db2 system agents
  - Some zIIP-eligible workloads, such as CPU parallel queries, utility operations or low priority DDF transactions may be able to tolerate increases in latency due to dispatching delays
  - System agents executed under MSTR and DBM1 address spaces cannot tolerate significant CPU delay
    - These system agents provide the subsystem level services such as prefetch and deferred write I/Os, castout, and log write I/Os which tend to perform very small unit of works but can create bursts of zIIP eligible-work
  - Therefore, it is necessary to ensure there is always zIIP capacity available to the LPAR where Db2 runs to maximise zIIP offload, and provide the capacity needed to support the latency requirements
- Consider the following actions to protect latency-critical Db2 subsystems and workloads
  - Add more zIIP weights on the LPARs where critical Db2 for z/OS workloads run
    - To ensure a physical zIIP engine is available to be dispatched on-demand for latency critical Db2 task means managing the weight and logical engine assignments so the LPAR has at least one Vertical High logical zIIP engine assigned
  - Utilise z/OS Simultaneous Multi Threading in the z/OS LPAR on z13 or later processors
    - MT_ZIIP_MODE parameter in IEAOPTxx in SYS1.PARMLIB enables Simultaneous Multi Threading for zIIP engines and can result in overall zIIP capacity improvement

Configuring zIIP processors for Db2 …

- Monitoring zIIP usage
  - Use Db2 Accounting and Statistics traces
  - Utilise the following z/OS instrumentation to monitor zIIP usage
    - SMF Record Types 70 and 72: CPU Activity and Workload Activity Reports
      - Breaking Db2 system address spaces (DBM1, MSTR, DIST and IRLM) out into separate WLM Reporting Classes allows them to be monitored independently for zIIPs or other resource constraints
      - Key metric = APPL% IIPCP
        - Percentage of the processor time used by zIIP eligible transactions running on general purpose processors
        - Execution delays due to zIIP
  - Also see
    - SMF Record Type 113: hardware capacity and statistics report
    - SMF Record Type 98: High-frequency throughput statistics (HFTS)
  - Very important to review WLM policies and PR/SM LPAR weights periodically to ensure that reasonable service goals are assigned and are achieved

Rebind very old plans and packages to avoid Autobind

- Problem
  - Pre-Db2 10 plans and packages using 31-bit run-time structures are no longer supported under Db2 12
  - If you do not explicitly REBIND these very old plans and packages before leaving Db2 11, an autobind will occur the first time the associated application runs under Db2 12
  - If you allow autobind to occur under Db2 12 then run the risk of facing
    - Difficult to resolve performance regression issues
    - Disrupting the Db2 12 migration window
    - Potential authorization issues that may exist that cause autobind to fail
    - Application service outages

REBIND very old plans and packages to avoid Autobind …

- Recommendations
  - Best practice when migrating to Db2 12 is to eliminate the risk associated with very old plans and packages
  - Use DSNTIJPM pre-migration job under Db2 11 to identify the affected plans and packages
  - Explicitly REBIND these plans and packages under Db2 11 well ahead of migrating to Db2 12
    - Allow adequate time to address any performance regression issues
    - Avoid risk of disrupting the Db2 12 migration window
    - Resolve any potential authorization issues that may exist
  - REBIND the packages with APREUSE(WARN) and PLANMGMT(EXTENDED)
    - APREUSE(WARN)
      - Try and reuse the existing access paths
    - PLANMGMT(EXTENDED)
      - Save current run-time structures and access paths away in the PREVIOUS (and ORIGINAL if empty) package copy slots
      - If there is performance regression and still running under Db2 11, can use REBIND with SWITCH(PREVIOUS) to restore the old run-time structures and access paths
  - REBIND the plans during a very quiet period
- Note: APREUSE(WARN) with Autobind will be delivered with APAR PI15896

REBIND Phase-in and thread reuse with RELEASE(DEALLOCATE)

- Function level 505 under Db2 12 introduces support for REBIND Phase-in
  - Allows rebind of a package concurrently with execution of the package
    - REBIND PACKAGE creates a new copy of the package
    - When REBIND PACKAGE operation completes
      - New threads created can use the new package copy immediately
      - Existing threads continue to use the copy that was in use prior to the REBIND (the phased-out copy) without disruption
    - It does still require setting the zparm PKGREL_COMMIT to YES
  - Most importantly provides for recovery from performance regression without incurring an application outage
    - Switching back to previous run time structures and access paths gradually (switch to the previous access path is phased-in)
  - REBIND Phase-in will still work with thread reuse with RELEASE(DEALLOCATE)
    - Thread loads a package bound with RELEASE(DEALLOCATE) for execution
    - Thread does not free/release the package on COMMIT or ROLLBACK
    - Thread will still use the phased-out copy on the next successive units of work reusing the thread
    - For CICS protected threads, consider adjusting the REUSELIMIT = 1000 (default) on the DB2CONN entry if you rebind RELEASE(DEALLOCATE) packages and want threads to use the new package copy sooner

Data Sharing - GBP Asynchronous cross-invalidation (XI)

- When a changed page is written to the GBP, XI(s) are sent synchronously to invalidate now down-level “stale” version of page in the buffer pools of other Db2 member(s)
- Goal: Improve elapsed time performance by sending “cross invalidation” asynchronously when writing updated pages, including during commit
  - With async XI, Db2 must drive a sync-up call before releasing the locks that protect the written pages
  - For commit duration L-locks, sync-up happens prior to end-commit
  - For P-locks, happens at commit (usual case) or as part P-lock negotiation exit processing
- Time to invalidate a buffer in the other member(s) is a function of the distance between coupling facility and the other member(s)
  - Save CPU (zIIP eligible) during GBP writes
  - But the sync-up calls generate additional GBP requests which consumes extra CPU
- Internal IBM measurements show that the benefit is visible already at zero distance (~7%) and increases for long distances, e.g. 10 km up to ~21%
  - These numbers apply to Accounting Class 2 elapsed times

Data Sharing - GBP Asynchronous cross-invalidation (XI) ...

- Prerequisites:
  - APAR OA54688 for z/OS V2.2 and V2.3
  - CFLEVEL 23
  - Db2 12 with APARs PH05193 and PH10577 (enabler)
  - It is function level independent
- Considerations
  - Does not apply to 32K page size
  - >=8-page threshold is be applied to the total number of pages to write for an object in one invocation
- “Secret” commands for serviceability and to evaluate performance
  - Disable via -ALTER GBPOOL(GBPx) SERVICE(1) SERVDATA(CFLEVEL23OFF)
  - Re-enable via -ALTER GBPOOL(GBPx) SERVICE(1) SERVDATA(CFLEVEL23ON)

Data sharing - GBP deallocation

- Question: Why does GBP remain allocated when there are no GBP-dependent objects
- Answer:
  - GBP connect occurs when an object becomes GBP-dependent
  - GBP disconnect will not happen even if there are no objects that are GBP dependent
    - GBP is actually disconnected from when the associated local buffer pool (LBP) is deallocated
    - LBP is deallocated when the LBP use count goes to zero
    - LBP use count is typically decremented when the table space or index space is logically closed
      - Logical close can happen much later than the physical close when the dataset is closed
    - Logical close decrements the PB (control block) use count, but Db2 tries to keep the PB around for reuse
    - If Db2 deletes the PB, then Db2 will decrement the LBP use count, which could cause the LBP to be deallocated
- Note
  - Physical open and close is the MMSRV CONNECT and DISCONNECT to open/close the data set
  - Logical open finds or creates the PB control block and it will increment it's use count
  - LBP for the pageset will be allocated if not already allocated and the buffer pool use count will be incremented

Data Sharing – Peer Recovery

- Most customers already have automation in place through ARM and/or System Automation to perform
  - Restart a failed Db2 member in place on the “home” LPAR
  - Cross-system restart of a failed Db2 member when loss of LPAR and/or CEC

- Db2 now provides an alternative method for cross-system restart of failed Db2 member when the “home” LPAR or CEC fails
  - Alternate peer Db2 members do NOT automatically recover retained locks for a failed Db2 member
  - Rather a peer Db2 member will automatically restart the failed Db2 member on an alternate LPAR
  - Which LPAR is a “race” condition depending on which peer Db2 member successfully acquires the global lock first
  - Example
    - Member DB2A runs on LPAR PRDA
    - Member DB2B runs on LPAR PRDB
    - LPAR PRDB fails!
    - Member DB2A will trigger the restart of member DB2B with LIGHT(YES) on LPAR PRDA

Data Sharing – Peer Recovery …

- Peer Recovery is certainly not a panacea and it has some limitations to consider
  - It will always use LIGHT(YES)
    - So if in a cross-system restart scenario, and you do not restart the CICS/IMS regions on the alternate LPARs, and there are some in-doubt URs
      - Restarted Db2 member will not come down
      - Manual intervention is required to manually stop the Db2 member before it can be restarted on it's "home" LPAR
    - If you try to automate that scenario, you will end up with much more complex logic than simply using ARM or System Automation to code a simple restart statement with LIGHT(NOINDOUBTS) for cross-system restart
  - No differentiation between Db2 failure and LPAR/CEC failure
    - If Db2 member has crashed but the LPAR is still there, why would you use LIGHT(YES), possibly on another LPAR, instead of using a normal restart in place on the "home" LPAR?
- Recommendation
  - Continue to use ARM and/ or System Automation which provide a more powerful and robust solution for cross-system restart of a failed Db2 member when loss of LPAR and/or CEC

Database Housekeeping Rules

- Problem areas
  - Tablespace level REORG
    - Potential for generating too many REORGs for “unclustered inserts”
    - Not generating enough REORGs for “relocated rows”
  - Not performing enough index REORGs
    - Ignoring pseudo deleted index entries and pseudo empty index pages
    - Index candidates which have a high percentage of leaf page splits are not being reorganized
  - No investigation about which rules are triggering
    - Most candidates for REORG
    - Frequently occurring candidates for REORG

Database Housekeeping Rules …

- List of candidates for REORG should be built dynamically based exclusively on RTS
  - Most critical rules to trigger REORG (*)
    - For indexes:
      - **Pseudo-deleted entries: REORGPSEUDODELETES/TOTALENTRIES > 5%**
      - Index splits: (REORGLEAFFAR×100)/NACTIVE > 10%
      - **Pseudo-empty pages: NPAGES > 5 AND NPAGES/NLEAF > 10%**
    - For non-LOB table spaces or partitions:
      - Unclustered inserts: REORGUNCLUSTINS /TOTALROWS > 10% AND REORGCLUSTERSENS > 0
        - Instead of a more “generic” rule on REORGINSERTS
      - **Relocated rows: (REORGNEARINDREF+REORGFARINDREF)/TOTALROWS > 5%**
      - Deleted rows: REORGDELETES/TOTALROWS > 25% (to control space growth)
    - For LOB table spaces:
      - REORGDISORGLOB/TOTALROWS > 50%
  - Also include objects in advisory-REORG pending or advisory-REBUILD pending
    - (*) Be careful about rules based on REORGINSERTS or REORGINSERTS/UPDATES/DELETES as they have the potential of generating many unnecessary REORGs

Database Housekeeping Rules …

- List of candidates for REORG should be built dynamically based exclusively on RTS …
  - Most important rules are related to
    - **Relocated rows**
    - **Unclustered inserts**
    - **Pseudo deleted index entries and pseudo empty index leaf pages**
  - Going forward should periodically benchmark your rules against the rules in current version of DSNACCOX
    - Note carefully: the rules on pseudo-deleted entries and pseudo-empty pages are not included in DSNACCOX
  - Regular REORG INDEX is very important for performance, including exploitation of FTBs
    - Important: Never collect access path STATISTICS as part of REORG INDEX
  - Keep information about trigger conditions for trend analysis and possible tuning for individual objects that are frequently REORGed
    - Use as input to evaluate alternative tuning options to reduce REORG frequency
      - Increase PCTFREE to reduce index page splits?
      - Define some PCTFREE FOR UPDATE?
      - Increase SEGSIZE to provide better chance of maintaining clustering?
      - Consider larger index page size to reduce index page splits?

Turning off CICS-Db2 thread protection

- Protected threads (PROTECTNUM) can only be set and altered on DB2ENTRY definitions so can use either of the following techniques to change the number
  1. CEMT SET DBENTRY(name) PROTECTNUM(Value=0) - To set the number of protected threads to zero
     - Syntax: https://www.ibm.com/support/knowledgecenter/SSGMCP_5.4.0/reference/transactions/dfha7n5.html
     - Need to know the names of all the DB2ENTRY definitions using the plan/packages that touch the tables and also know the correct number to use when setting the protected threads back again
  2. EXEC CICS SET DB2ENTRY(name) PROTECTNUM(value=0) - Issued from user written CICS program (need to allow SPI commands to be issued)
     - Syntax: https://www.ibm.com/support/knowledgecenter/SSGMCP_5.4.0/reference/commands-spi/dfha8_setdb2entry.html
     - As it is programmable, do a EXEC CICS INQUIRE DB2ENTRY START and then browse through all the entries using INQUIRE NEXT performing a SET action against each - need to save away the original PROTRCTNUM value in order to set it back
  - You can then use the same commands to set the PROTECTNUM to non-zero again to reinstate the number of protected threads

Summary

- Learn recommended best practice
- Learn from the good, bad and ugly experiences from other installations
- Provide answers to important, frequent or difficult questions

Top DB2z Social Media Channels

#DB2z

- Join the World of DB2 www.worldofdb2.com
- Follow @IBMDB2 on Twitter https://twitter.com/IBMDB2
- Join DB2z LinkedIn Group
- https://www.youtube.com/user/IBMDB2forzOS
- DB2z on Facebook
  - https://www.facebook.com/IBMDB2forzOS/

Now … Live Q&A with John Campbell