JC's Greatest Hits, War Stories and Best Practice 2018 Parts 1 and 2

John Campbell and Maureen (Mo) Townsend

IBM Db2 for z/OS Development

Session code: **A01**

Monday, 5th November at 10:40 - 11:40 and 12:00 – 13:00

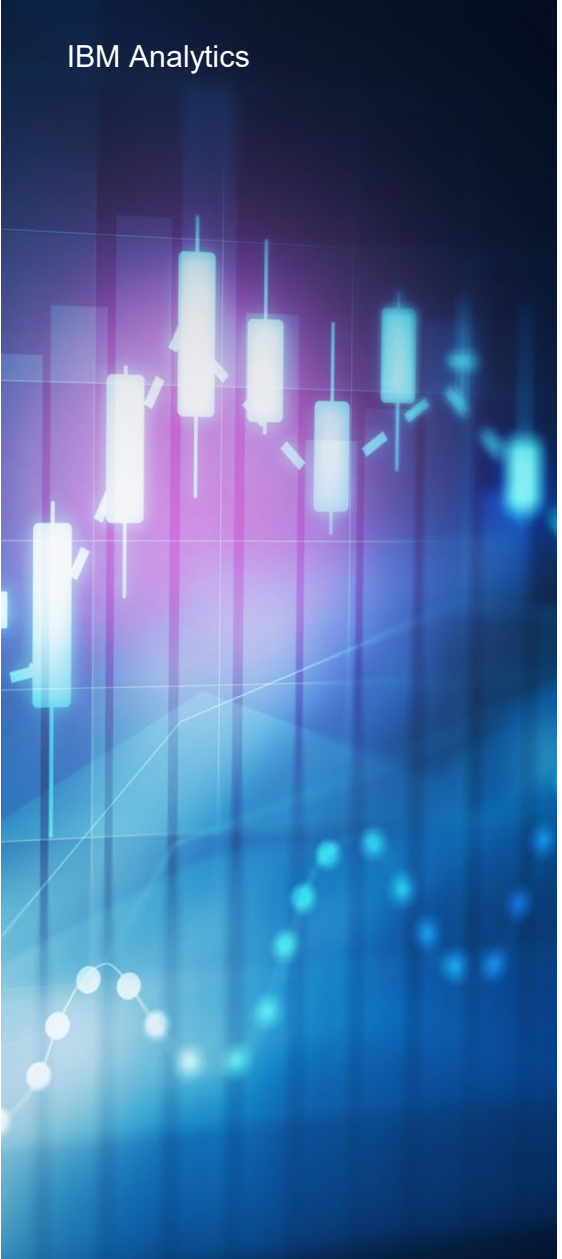
Platform: Db2 for z/OS

Objectives

- Meet new Db2 for z/OS Director of Development & learn high-level future strategy
- Learn recommended best practice
- Learn from the positive and negative experiences from other installations


Db2 for z/OS is investing for leadership in the cloud, mobile, and analytics era, while maintaining our mainframe heritage of 24x7 availability, security, scalability, and performance, and improving automation of platform-specific tasks.
IBM **Analytics**

Db2 for z/OS Strategy



Smarter analytics

- Hybrid Transactional/ Analytics Processing
- IDAA integration
- Expanded OLAP

Simplification

- DBaaS and DevOps
- Enhanced RESTful services
- SQL improvements
- Db2ZAI, embedding ML for self-driving Db2

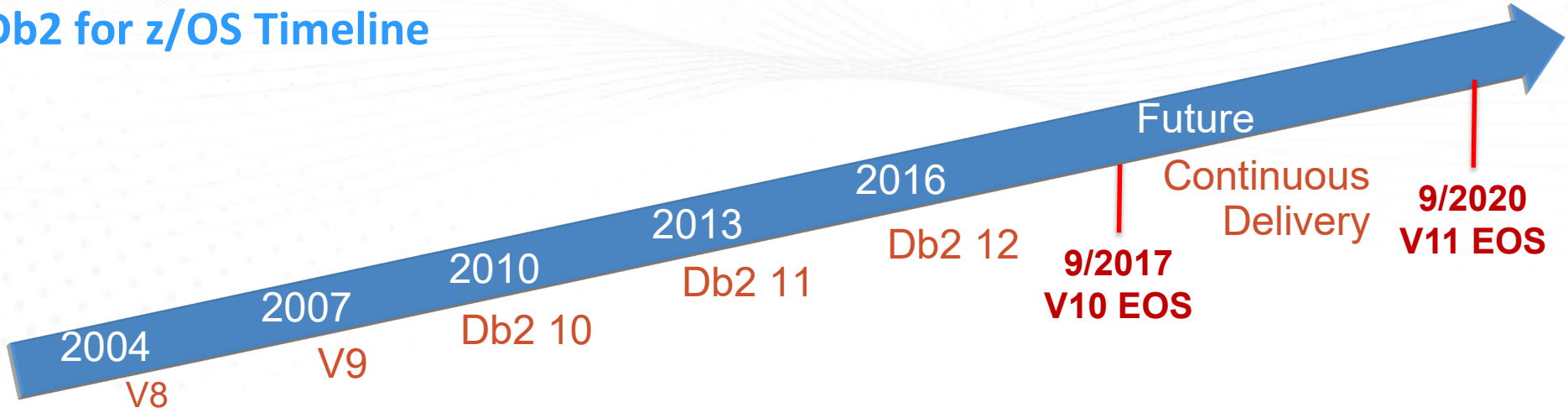
Always on

- More non-disruptive schema changes
- Enhanced online catalog migration

z Platform optimization

- Enhanced Compression
- Faster I/O
- In-memory improvements
- Further Data Sharing Optimization

Db2 for z/OS Timeline



- **No Single Version Charging – same price for any Db2 version**
- **Db2 12 GA October, 2016**
- **Db2 12 adoption rate comparable to previous releases**
- **Quality metrics, continuous improvement:**
- **V12 better than V11, which is better than V10**

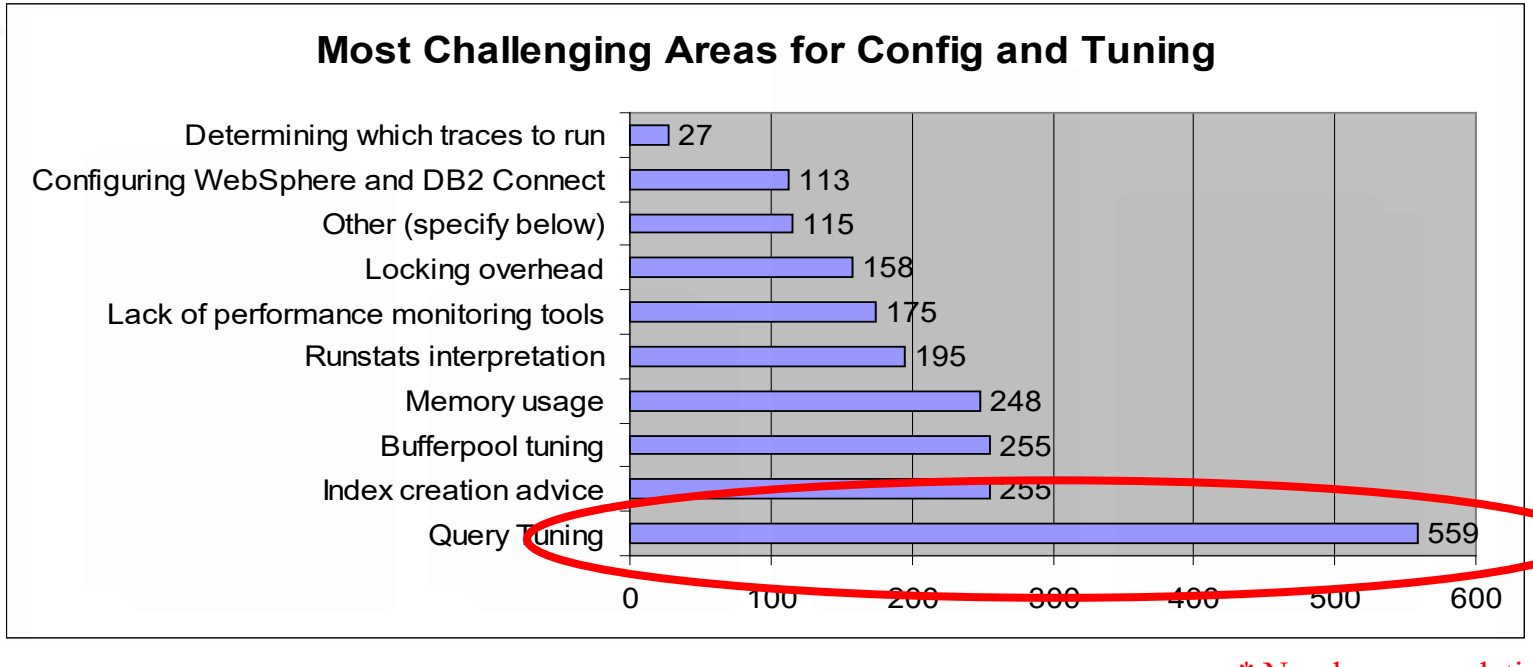
Db2 12 Function Levels – Our Future Delivery Model

- **Start slow, speed up as we go**
  - Quality, stability is priority #1
  - Function levels (FLs) are the mechanism to activate new features on V12
    - System level and application level controls
    - FL 500 is base V12 "NFM". FLs 501, 502, … beyond that
    - Goals: faster delivery, easier to consume for customers (compared to traditional release migration)
  - https://www.youtube.com/watch?v=ldHOnyJXT3M
  - REDP5469 Exploring IBM Db2 for z/OS Continuous Delivery
- **FL 501 – 1st post-GA delivery**
  - LISTAGG
- **FL 502 – April, 2018. APAR PI95511**
  - Transparent Dataset Encryption: Db2 DBA controls
  - Casting numeric to GRAPHIC/VARGRAPHIC
- **FL 503 – Sept., 2018. APAR PH00506**
  - Db2 AI for z/OS (Db2ZAI)
  - Migration support on DATA CHANGE OPERATION for temporal auditing
  - Enablement for replication of system-period temporal tables and generated expression columns

Machine Learning in Db2 – which high value problem to tackle first?

"25-30% of customer DBA time is spent fighting access path problems which cause performance degradation and service impact."

~John Campbell, IBM Db2 Distinguished Engineer



*

* Numbers are relative. Biggest pain point.

Introducing: IBM Db2 AI for z/OS

- Leverage machine learning to address complex challenges within the Db2 for z/OS engine
- Begin with optimizer access paths to deliver immediate value with no application changes
- Manage ML models throughout life cycle
  no need for data science skills to achieve benefits
- Explore future opportunities to improve performance & reduce CPU, free up DBA & system programmer talent to higher value tasks, and to mitigate risks, e.g., in applying maintenance
- Announce: September 11, 2018
- Available: September 21, 2018
- Watch our video at: https://www.youtube.com/watch?v=t5fTNxfehQA

JC's Agenda

- Running out of basic 6-byte log RBA addressing range
- Diagnosing resolving slow-downs and hangs
- Hung Db2 threads
- New ZSA keyword on HIPER and PE APARs
- Insert free space search algorithm
- Insert with APPEND
- Fast un-clustered insert (Db2 12)
- Hidden ROWID support to partition
- Cost management using CPU capping
- Fast REORG with SORTDATA NO RECLUSTER NO
- Requirement for increased RID Pool size (Db2 12)
- Setting initial STATISTICS PROFILE (Db2 12)
- System parameter REALSTORAGE_MANAGEMENT

JC's Agenda …

- Transaction level workload balancing for DRDA traffic
- Use of High-Performance DBATs
- Overuse of UTS PBG tablespaces and MAXPARTS
- Cleaning up after STATCLUS = STANDARD
- "MBA lock"
- IRLM query requests for package break-in
- Db2 maintenance issues with batch utilities
- Running CHECK utilities with SHRLEVEL CHANGE
- Local buffer pool allocation – virtual and real memory

Running out of basic 6-byte log RBA addressing range

- **Background**
  - Increasingly common for the basic 6-byte log RBA (256 TB) addressing range to be exhausted for Db2 subsystems
  - Tiny number of installations close to exhausting the basic 6-byte log LRSN for a data sharing group
  - BSDS must be converted to extended 10-byte RBA format before migrating to Db2 12

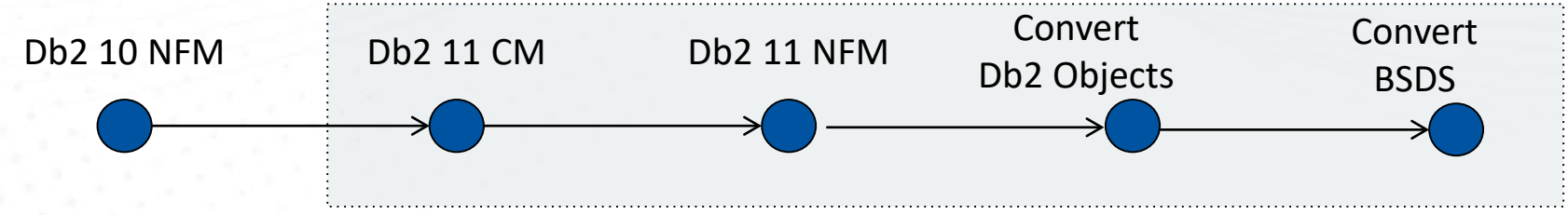
- **Problem areas**
  - After BSDS conversion to extended 10-byte log RBA, non-data sharing Db2 subsystems will accelerate with increased velocity towards end of basic 6-byte log RBA addressing range!
  - After converting the BSDS, Db2 stops generating the DSNJ032I warning messages, even if there is imminent danger of reaching the 6-byte RBA soft limit (non-data sharing) or 6-byte LRSN soft limit (data sharing) for table spaces or index spaces in 6-byte basic format
  - Many installations embarked on an aggressive but unnecessary “crash project” to reorganise Catalog/Directory and application database objects to extended 10-byte of RBA or LRSN format

Running out of basic 6-byte log RBA addressing range …

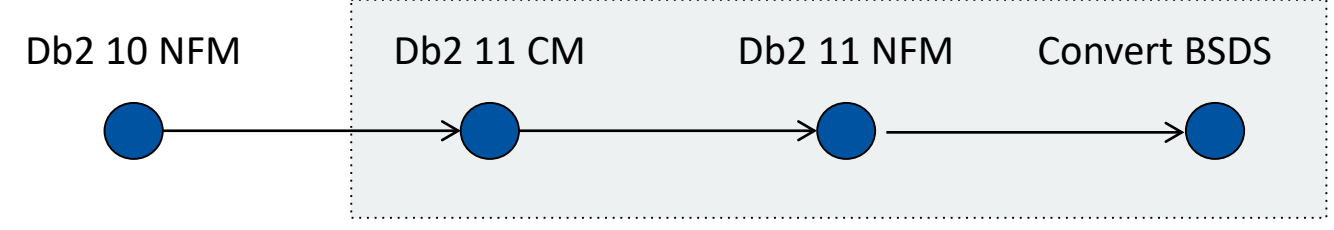
- **Recommendations**
  - Non-data sharing - getting to extended 10-byte log RBA format
    - Must convert the Catalog, Directory and application objects via REORG to extended 10-byte log RBA format
    - Must convert BSDS of the problem Db2 subsystem to extended 10-byte log RBA format
    - Leave converting the BSDS to the very end of the overall conversion process, otherwise will accelerate towards end of basic 6-byte log RBA addressing range with increased velocity
  - Data sharing - getting to extended 10-byte log RBA format for a specific Db2 member
    - Just convert BSDS of the problem Db2 member
    - No need to convert Catalog, Directory and application objects via REORG to extended 10-byte LRSN format
  - Data sharing – getting to the extended 10-byte LRSN format
    - Must convert the Catalog, Directory and application objects via REORG to extended 10-byte extended LRSN format
    - Must convert BSDS of each Db2 member to extended 10-byte extended log RBA/LRSN format
    - Convert the BSDS at the start of the overall conversion process to get incremental benefit from LRSN spin avoidance as Catalog, Directory and application objects are converted to extended 10-byte LRSN format

Reaching the limit conversion strategies

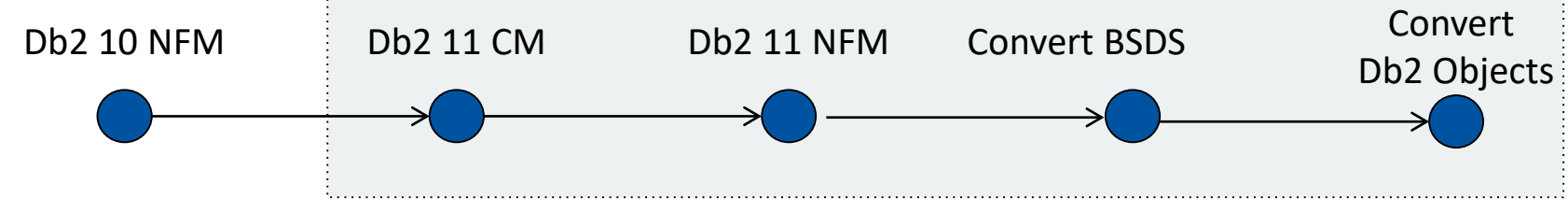
Non Data Sharing – End log RBA problem



Data Sharing – End log RBA problem only on one or more members



Data Sharing – End LRSN problem



Diagnosing resolving slow-downs and hangs

- **Very rare condition, with little or no customer operational experience**
- **Techniques**
  - Issue Db2 and IRLM commands for each Db2 member and if a Db2 member/IRLM does not respond, then should take that Db2 member out even if it means using the z/OS CANCEL command
    - Sample commands to be issued
      - -DIS THD(*) SERVICE(WAIT) to each Db2 member
      - MODIFY xxxxIRLM,STATUS,ALLD to each IRLM
      - -DIS THD(*) SERVICE(WAIT) SCOPE(GROUP) to each Db2 member
  - Manually trigger rebuild of the LOCK1 structure into alternate CF based on the PREFERENCE LIST in CFRM policy
    - Issue SETXCF START, REBUILD, STRNAME=DSNxxxx_LOCK1,LOCATION=OTHER
    - Structure rebuild may clear the condition
    - However if the structure rebuild fails, find the connector which did not respond and IPL the non-responsive z/OS LPAR

Hung Db2 threads

- **Things are hung, do not cancel … not yet**
- **Dump first, cancel changes the layout of the control blocks and may hide the problem**
- **Bring out the smallest size "hammer" first**
  - CANCEL the thread in Db2
  - FORCEPURGE CICS TRAN or CANCEL BATCH JOB
  - CANCEL Db2
    - CANCEL IRLM
    - FORCE MSTR
    - FORCE DBM1
  - IPL z/OS LPAR

New ZSA keyword on HIPER and PE APARs

- **Introduced to provide a "rating" to try and help customers determine when to apply a particular HIPER/PE PTF**
- **The ZSA keyword is under APAR Error Description, usually as an 'Additional Keyword'**
- **ZSA ratings, ZSA1 (low), ZSA2, ZSA3, ZSA4, ZSA45, ZSA5 (highest) described below**
  - System Outage: 4.5 and 5 is an example for adding the consideration for the number of customer who has hit the problem, it should be the same for other HIPER category
    4: system outage should automatically get a 4
    4.5: If there are 1-5 customer already hit the problem
    5: If there are more than 10 customer already hit the problem
  - Data Loss:
    4: non-recoverable or pervasive, common
    3: recoverable, incorrout output, but with few conditions
    2: recoverable, incorrout, but fairly rare to hit it
    1: super rare cases
  - Function Loss:
    4: pervasive causing application outages
    3: likely common
    2/1: rare
  - Performance:
    4: looping indefinitely
    3: degrades >=5%
    2: degrades <5%
    1: not noticeable
  - Pervasive:
    4: automatically
  - MSysplex:
    4: automatically

Insert free space search

- **Insert performance is a trade off across**
  - Minimising CPU resource consumption
  - Maximising throughput
  - Maintaining data row clustering
  - Reusing space from deleted rows and minimising space growth
- **Warning: insert space search algorithm is subject to change and it does**

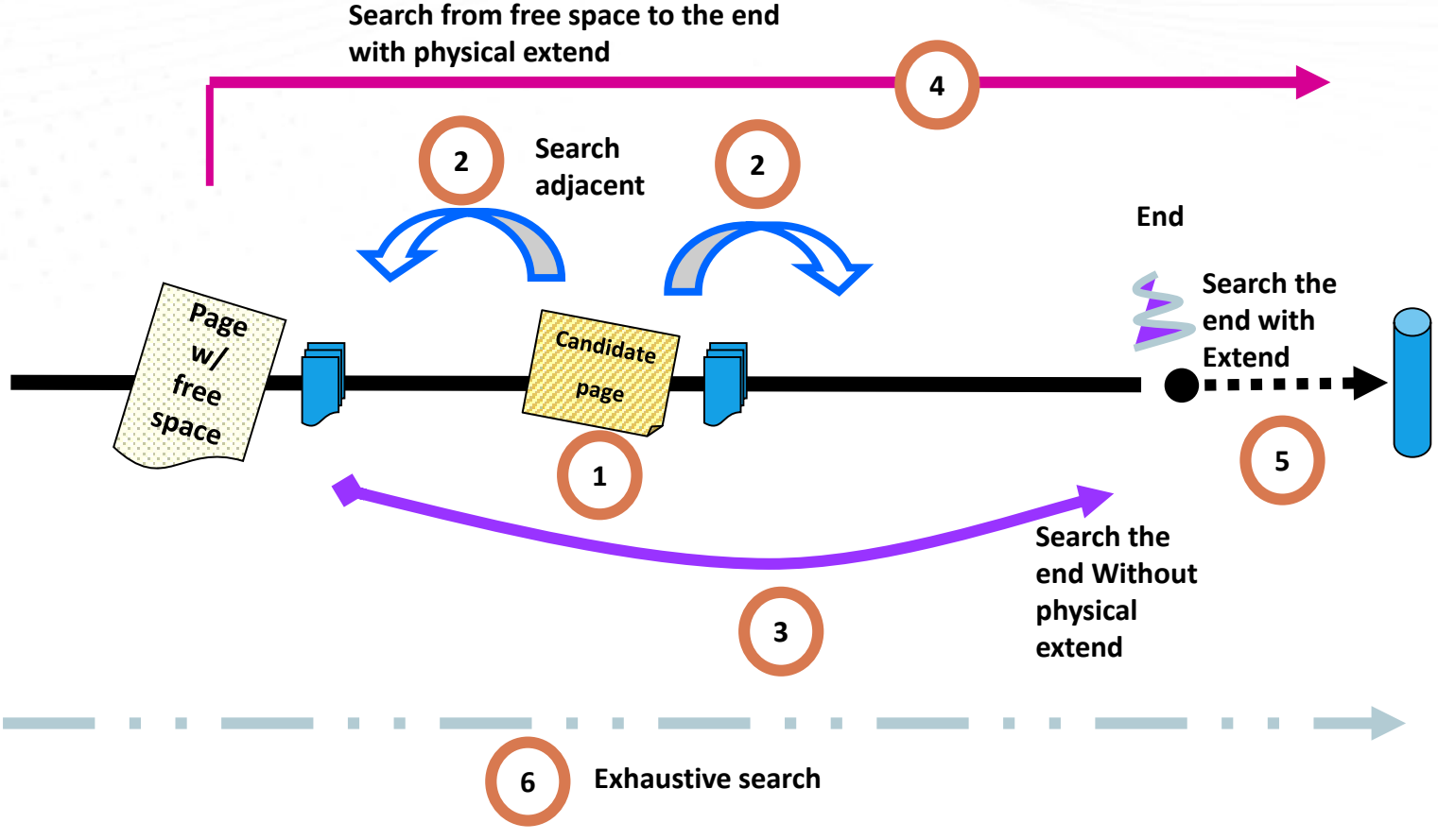
Insert free space search …

- **For UTS and classic partitioned table space:**
  1. Index Manager will identify the candidate page (next lowest key rid) based on the clustering index
     - If page is full or locked, skip to Step 2
  2. Search adjacent pages (+/-) within the segment containing the candidate page
     - For classic partitioned it is +/-16 pages
  3. Search the end of pageset/partition without extend
  4. Search the space map page that contains lowest segment that has free space to the last space map page up to 50 space map pages
     - This is called "smart space exhaustive search"
  5. Search the end of pageset/partition with extend until PRIQTY or SECQTY reached
  6. Perform "exhaustive space search" from front to back of pageset/partition when PRIQTY or SECQTY reached
     - Very expensive i.e., space map with lowest segment with free space all the way through
- **For classic segmented table space, steps are very similar except the sequence:**
  - 1->2->3->5->4->6

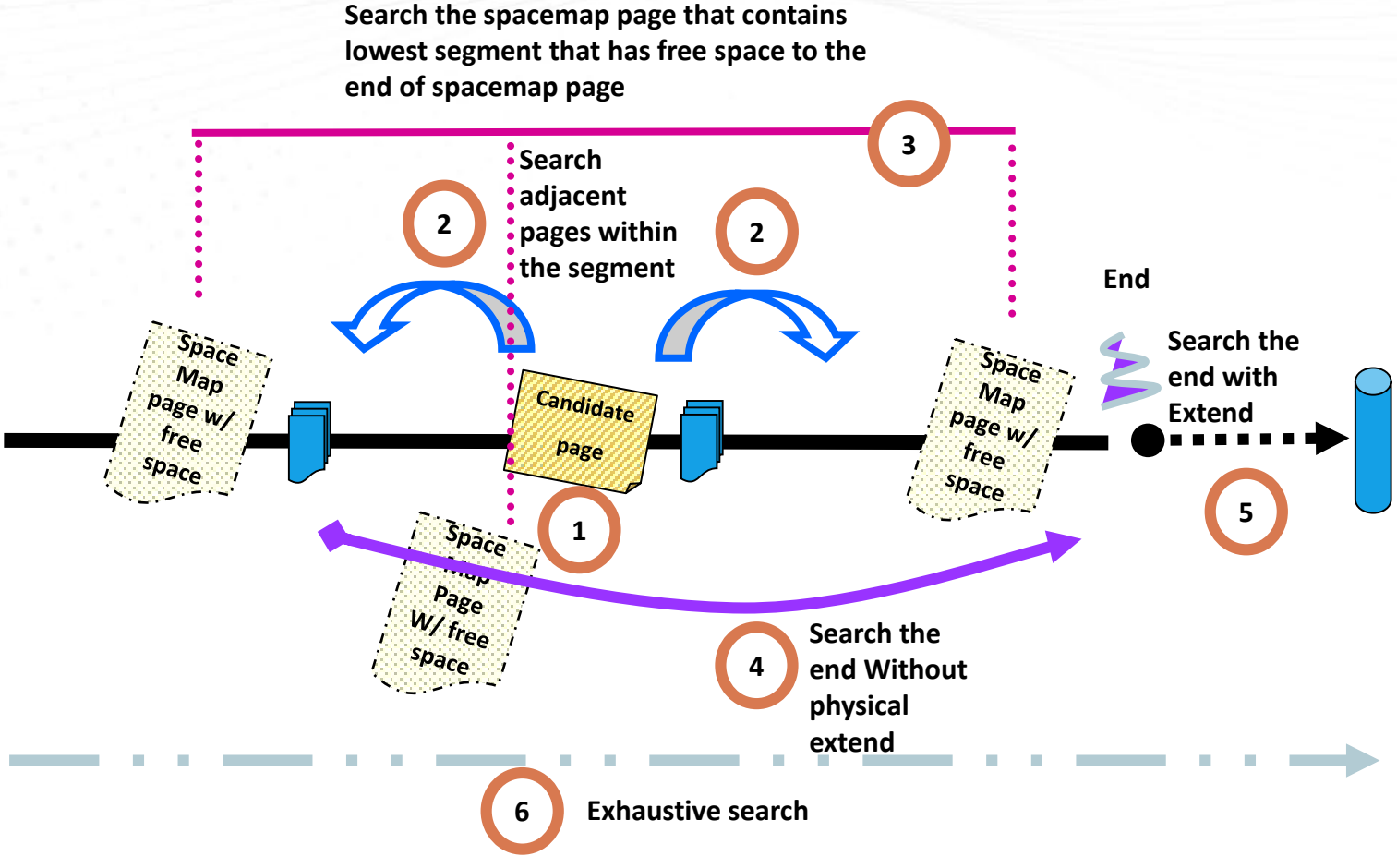
Insert free space search …

- **Each member maintains for each pageset/partition:**
  - First/lowest space map page with free space
  - Current space map page with free space
  - Last space map page
- **In general, each step has 3 pages for “false lead” and 100 for lock failure threshold**
  - In step 3 (search at end of pageset/partition without extend) there is a little more search than just 3 pages “false lead” to tolerate the hot spot case
- **For the search step where cannot extend at end of pageset/partition**
  - Search from the first space map page to the end
  - No false lead or lock threshold
  - Try every possible space before failure
- **Elsewhere for the “smart exhaustive search” before extend, it will follow the "lowest segment that has free space" going forward 50 space map pages then it will go to extend**
  - The 3 pages “false lead” and 100 page lock failures are applied to each space map page
- **If there are multiple inserts in the same UR, then 2nd insert will not search the previous repeated unsuccessful space map pages if the previous insert already gone through more than 25 pages**

Insert – free space search steps (UTS and classic partitioned tablespace)



Insert – free space search steps (Classic segmented table space)



INSERT with APPEND

- **With APPEND**
  - Always insert into the end
  - No attempt to use free space created by deleting rows
  - Space search will always start on last data page of the last space map page
  - Search -64 pages prior to the last page of the last space map page
- **Ideal for application journal and event logging tables with little or no update or delete**
- **APPEND can be beneficial as it avoids the "normal" space search**
- **With high concurrency may not see any improvement due to space map contention at the end of table**
  - So in data sharing environment with high concurrency, APPEND should be combined with MEMBER CLUSTER for tables which require faster insert and do not need to maintain the data row clustering
    - Similar to V8 MEMBER CLUSTER, PCTFREE 0, FREEPAGE 0
- **With LOCKSIZE PAGE|ANY, space growth can be excessive with partially filled data pages**
  - With LOCKSIZE ROW, the space growth should be slower
- **When using LOAD with DUMMY, do NOT use PREFORMAT option**
  - DASD space size will be reduced after LOAD

Fast un-clustered INSERT

- **Insert workloads are amongst the most prevalent and performance critical**
- **Performance bottleneck will vary across different insert workloads**
  - Index maintenance?
  - Log write I/O?
  - Data space search (space map and page contention, false leads)
  - Format write during dataset extend
  - PPRC disk mirroring
  - Network latency
  - etc
- **Common that Index insert time may dominate and mask any insert speed bottleneck on table space**

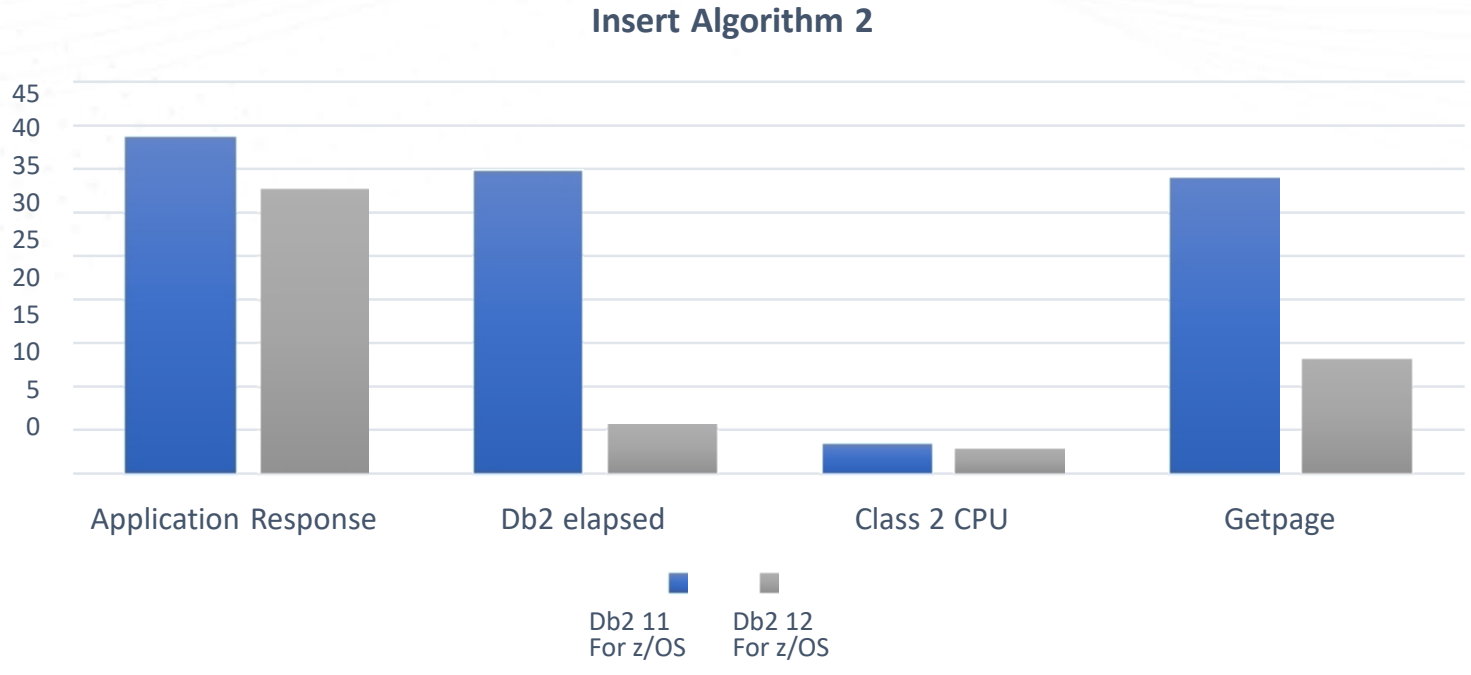
Fast un-clustered INSERT …

- **Officially referred to as "Insert Algorithm 2 (IAG2)"**
- **Sometimes referred to as "Smart Insert" or even "Fast Insert"**
- **Potentially delivers significant improvement for un-clustered inserts (e.g., journal table pattern) where both**
  - Heavy concurrent insert activity (many concurrent threads)
  - Space search and false leads on data is the constraint on overall insert throughput
- **Applies to any UTS table space defined with MEMBER CLUSTER**
  - Applies to both tables defined as APPEND YES or NO
- **Implemented advanced new insert algorithm to streamline space search and space utilisation**
  - Eliminates page contention and false leads
  - Currently default is to use the new fast insert algorithm for qualifying table spaces after new function activation (FL=V12R1M5nn)
    - DEFAULT_INSERT_ALGORITHM system parameter can change the default
    - INSERT ALGORITHM table space attribute can override system parameter

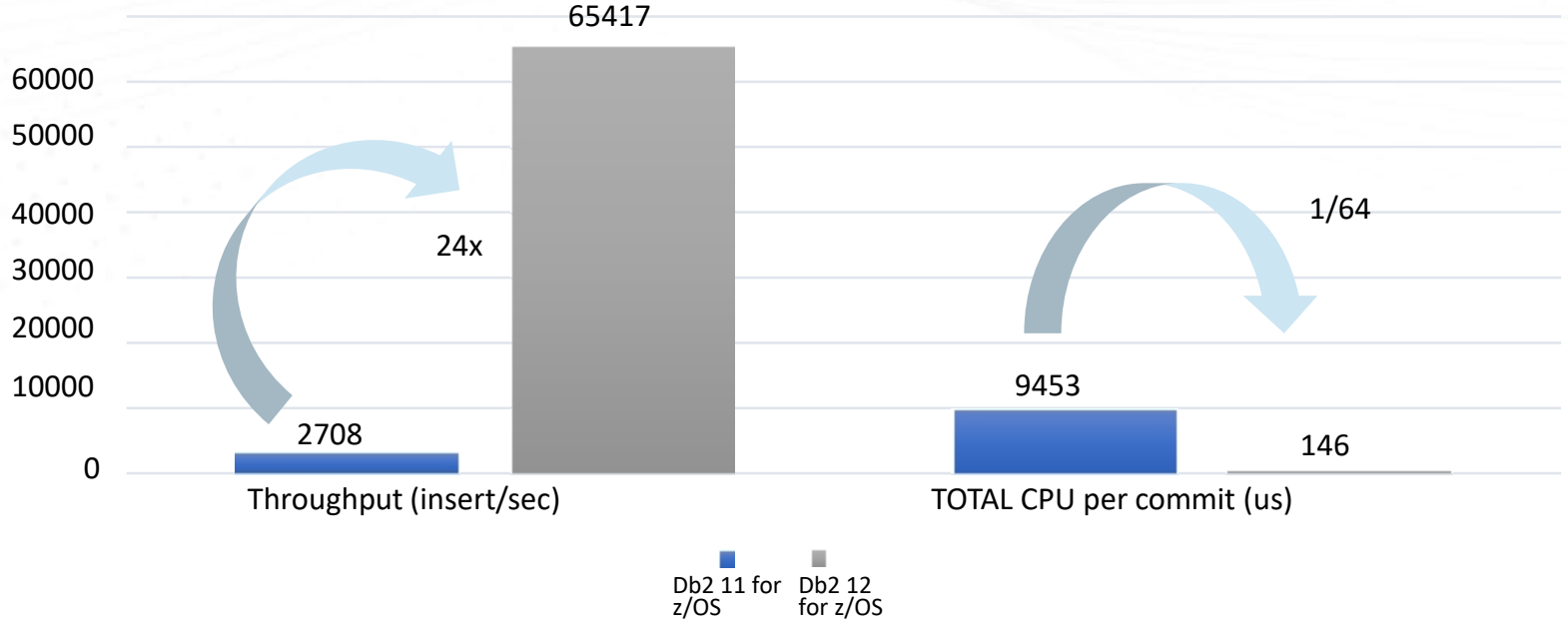
Fast un-clustered INSERT …

- **Your mileage will vary**
  - Many insert workloads will see no improvement and is to be expected
  - Some specific insert workloads may see significant improvement
- **Will shift the bottleneck to the next constraining factor**
- **LOAD SHRLEVEL CHANGE can also use Fast Un-clustered INSERT**
- **Fast Un-clustered INSERT will not be used when lock escalation occurs or use of SQL LOCK TABLE**
- **Available after new function activation (FL=V12R1M5nn)**
- **Must be very current on preventative service for robustness and serviceability – tagged with "IAG2"**
  - For example – see APAR PH02052 which implements automatic re-enablement with retry logic
- **Recommendations**
  - Change system wide default - set system parameter DEFAULT_INSERT_ALGORITHM = 1 (old basic insert algorithm)
  - One size probably does not fit all tablespaces
  - Not much difference/improvement for short running transactions
  - Use INSERT ALGORITHM 2 selectively at individual table space level to override system wide default

Fast un-clustered INSERT – Shifting The Bottleneck …



Fast un-clustered INSERT - Db2 11 for z/OS PMR Recreate …



UTS PBG with MEMBER CLUSTER, RLL, with 400 bytes per row, one index,
800 concurrent threads, 10 insert per commit

Hidden ROWID support to partition

- **ROWID can be used as a partitioning column**
- **Application impact if ROWID cannot be hidden**
  - APARs to support to define a hidden ROWID
    - PI76972, PI77310, PI77302 (Db2 12)
    - PI77718, PI77719, PI77360 (Db2 11)
- **Benefits**
  - Allows table to be partitioned where no natural partitioning key exists or candidate partitioning keys do not provide a good spread across partitions
  - Transparent to the application
  - Improved insert throughput
    - Less lock/latch contention on index and data

```
CREATE TABLE  PRDA.ZJSCNTP0
      (  CLIENT    VARGRAPHIC(3) NOT NULL,
         WI_ID     VARGRAPHIC(12) NOT NULL,
         LENGTH    SMALLINT,
         DATA      VARCHAR(1000),
         ROW_ID    ROWID NOT NULL
         IMPLICITLY  HIDDEN generated always
      )  PARTITION BY (ROW_ID)
          (PARTITION 1 ENDING AT (X'0FFF'),
           PARTITION 2 ENDING AT (X'1FFF'),
           PARTITION 3 ENDING AT (X'2FFF'),
           PARTITION 4 ENDING AT (X'3FFF'),
           :
           PARTITION 14 ENDING AT (X'DFFF'),
           PARTITION 15 ENDING AT (X'EFFF'),
           PARTITION 16 ENDING AT (MAXVALUE))
```

Performance cost management using CPU capping

- **There are different types of caps, based on different technologies and therefore different impact on workloads when capping kicks in**
- **Concerns**
  - WLM policy may not be defined in a way to defensively protect the service when CPU is constrained
  - Increased risk of experiencing erratic performance and sporadic system slowdowns and worse …
- **Recommended best practice for WLM setup with the right prioritisation - not related to CPU capping**
  - IRLM mapped to service class SYSSTC
  - Use Importance 1 and a very high velocity goal that is achievable for MSTR, DBM1, DIST, WLM-managed stored procedure system address spaces
  - Use Importance 2-5 for all business applications starting in I=2 for high priority IMS, CICS, DRDA workloads
  - Use CPUCRITICAL for Db2 system address spaces to prevent lower importance working climbing
  - Ensure WLM Blocked Workload enabled and set BLWLINTHD threshold time interval for which a blocked address space or enclave must wait before being considered for promotion to <= 5 seconds

Performance cost management using CPU capping …

- **CPU capping …**
  - There are no problems with caps provided the applications can tolerate the cap hitting and there is an escape mechanism when contention exists
  - If NOT there there can be sympathy sickness across a Db2 subsystem or across a whole data sharing group (sysplex wide) due to lock and/or latch contention
    - Multiple customer outages have been caused by capping
    - Cap hits and can freeze Db2 at the wrong time (latch)
      - Normal Db2 safeguards prove ineffective (Db2 targeted boost via WLM services, WLM Blocked Workload)
    - Caps easy to misconfigure scope (accidental use of a SYSPLEX wide cap instead of LPAR cap)
  - WLM Resource Group Capping is a cap due to which work can become non-dispatchable and therefore can have more severe impacts on workloads, and such as limiting the efficiency of the very valuable WLM Blocked Workload promotion

Performance cost management using CPU capping …

- **CPU capping …**
  - There are tools from various vendors which perform “soft” capping (i.e., cap the MSUs for a particular workload), some claiming intelligence and being automatic, others used by customers to micro manage with frequent intervention
  - Every installation is different and many factors influence the outcome, but there are some aggravating factors and/or side effects
    - Engine parking and unparking with HiperDispatch
    - Over use of I=1 work for application work requiring aggressive SLA (i.e., >= 70% capacity and little or no lower priority work to pre-empt
    - Specialised LPARs: online vs. batch LPARs i.e., only online interactive workload on an LPAR
    - Low and high priority work touching the same tables and the same pages
    - Frequent changes in PR/SM topology, especially when very small partitions are present

Fast REORG with SORTDATA NO RECLUSTER NO

- **Use cases: materialising pending alters, converting to extended 10-byte LRSN/LRBA, conversion to UTS**
- **Sample control statement**

```
REORG TABLESPACE SHRLEVEL CHANGE
SORTDATA NO RECLUSTER NO
STATISTICS TABLE(ALL) SAMPLE 1 REPORT YES UPDATE NONE
```

- **Considerations and risks when using RECLUSTER NO**
  1. Recalculation of CLUSTERRATIO
     - If no explicit STATISTICS option provided, REORG will collect implicit statistics
       - This will can probably lead to bad CLUSTERRATIO statistics
  2. Reset of RTS statistics
     - REORGUNCLUSTINS, REORGINSERTS and REORGUNCLUSTINS will be set to 0 even through no data rows have been reclustered
- **Recommendations**
  - Only use SORTDATA NO RECLUSTER NO for objects with very highly clustered data
    - Alternative is to update RTS columns after REORG and restore the before values which have been saved away
  - Use explicit STATISTICS with REPORT YES UPDATE NONE to avoid update to CLUSTERRATIO
    - SAMPLE 1 is to reduce CPU resource consumption

Fast REORG with SORTDATA NO RECLUSTER NO …

- **Have to use tablespace level REORG to go through the process of converting classic partitioned tablespaces to UTS PBR**
- **Db2 has supported data unload/reload partition parallelism on classic partitioned and UTS PBR tablespace since Db2 9 GA**
- **However, this parallelism not enabled when REORG converts a classic partitioned tablespace to UTS PBR tablespace**
  - Restriction has been removed with APAR PI72455 (Db2 12) and APAR PI71930 (Db2 11) which enables REORG to perform data partition parallelism in the conversion case
- **Can also experience very long elapsed times with most of the elapsed time spent rebuilding the clustered index because the keys were not sorted first**
  - If the data is not well clustered to start with, you end up with “death by random I/O”
  - At the same time, REORG already uses sort for the not-clustered index keys
  - With APAR PI90801, available for both Db2 11 and Db2 12, REORG will now sort the clustering index keys into key order for this scenario

Requirement for increased RID Pool size (Db2 12)

- **Increased space requirement for RID Pool as a result of RID value increase 5 -> 8-byte value**
  - Internally Db2 for z/OS uses a normalized 8-byte RID value to allow for future expansion
  - More RID blocks will be used for the same query because each RIDLIST holds fewer RIDs
  - RID Pool memory usage will be roughly 60% higher (for smaller lists it will be up to 2x higher)
  - May have to increase MAXRBLK (RID Pool size) by up to 60%
  - Data Manager logical limit (RIDMAP/RIDLIST) reduced from 26M (26,602,376) RIDs to 16M (16,625,976) RIDs
  - More RID Pool overflow to workfile is to be expected

Setting initial STATISTICS PROFILE (Db2 12)

- **Statistics profile is automatically created on first BIND/REBIND/PREPARE/EXPLAIN after entry to Db2 12 for z/OS**
- **Statistics profile created based on all the existing statistics in the Catalog**
  - Stale or inconsistent "specialised" statistics may exist in the Catalog that have not been collected for a long time
  - Statistics no longer collected because either too expensive to collect with RUNSTATS or were ineffective in solving access path problems, and not ever cleaned out
- **"Specialised" statistics will now be MERGED into the respective statistics profile and will then be collected again on every execution of RUNSTATS with USE PROFILE**
- **After the initial profile create, cannot tell from the subject statistics profile what statistics are the ones that were the older/inconsistent statistics**
- **Stale or inconsistent statistics may already be causing sub-optimal access path choices prior to Db2 12**

Setting initial STATISTICS PROFILE (Db2 12) …

- **So it is important to clean up any (SYSCOLDIST) statistics that you do not intend to regularly collect before first BIND/REBIND, PREPARE or EXPLAIN after entry to Db2 12 for z/OS**
  1. Simplest way to find these is to look for tables with rows having different STATSTIMEs in SYSCOLDIST
  2. Reset access path statistics back to default using RUNSTATS with RESET ACCESPATH option
     1. Sets relevant Catalog column values to -1, and clears out entries in SYSCOLDIST
  3. Run “regular” RUNSTATS after the RESET operation

System parameter REALSTORAGE_MANAGEMENT

- **The frequency of contraction mode is controlled by system parameter REALSTORAGE_MANAGEMENT**
- **Unless you have ample real memory headroom and can tolerate some memory growth, recommend running with REALSTORAGE_MANAGEMENT=AUTO (default)**
  - With RSM=AUTO, Db2 will regularly discard unused REAL memory frames
    - RSM=AUTO with no paging (AUTO-OFF) → "Soft Discard" at Thread Deallocation or after 120 commits
    - RSM=AUTO with paging (AUTO-ON) → "Soft Discard" at Thread Deallocation or after 30 commits – STACK also DISCARDED
  - "Soft Discard" = Db2 uses DISCARD with KEEPREAL(YES)
  - Pros of the "Soft Discard"
    - Reduced REAL memory use
    - Reduced exposure to SPIN lock contention
  - Cons:
    - CPU overhead in MSTR/DIST SRB time (which can be greatly minimised with good thread reuse)
      - Worse on IBM z13 and z14 hardware
    - 64-bit shared and common real memory counters are not accurate until paging occurs
      - The memory is only "virtually freed"
      - RSM flags the page as freed or unused, but the frame is still charged against Db2

System parameter REALSTORAGE_MANAGEMENT (RSM) …

- **REALSTORAGE_MANAGEMENT = OFF**
  - Do not enter 'contraction mode' unless the system parameter REALSTORAGE_MAX boundary is approaching OR z/OS has notified Db2 that there is a critical aux shortage
- **But … provided an installation is certain about the generous allocation of REAL memory and avoiding any paging at all times then setting to OFF can generate CPU savings where there is poor thread reuse and/or surge of DBAT termination**
  - CPU savings can be seen in Db2 MSTR and DIST system address spaces
- **Setting system REALSTORAGE_MANAGEMENT = OFF requires a long-term commitment to maintaining generous REAL memory "white space" at all times**

Transaction level workload balancing for DRDA traffic

- **Re-balancing mechanism to adjust the workload distribution across Db2 members to meet the goal**
- **Workload distribution can be not uniform, can by spiky, can see "sloshing" effect**
- **Drivers up to today's latest available level have the two main issues**
  1. Balancing tends to favor highest weighted member(s)
     - Less load: Driver starts traversing through server list and always selects the first Db2 member in the server list, because it is available to take a new transaction based on a non-zero weight
     - Heavy Load: Driver starts traversing through server list from first Db2 member and checks whether the Db2 member is available to take a new transaction based on the weight
  2. Failover to another Db2 member has caused client driver to excessively retry
- **Client driver team has proposed the following changes to address issues:**
  1. Driver knows what is the last Db2 member used, so it starts from next Db2 member
     - From first iteration through server list, driver will use every Db2 member once
     - Second iteration onwards the driver choses the Db2 member based on the weight and number of transactions already ran
  2. Failover to another Db2 member will only be attempted once and the DataSource address will be used

Transaction level workload balancing for DRDA traffic …

- **Recommendations (prioritised) to improve the workload distribution across both members**
  1. Validate whether or not the goal of DDF transactions are being met under normal operating conditions and adjust the WLM policy accordingly (adjust the goal so it can be met and/or break down the DDF transactions into multiple service classes)
  2. Upgrade to new Db2 Connect driver level 11.1 M4 FP4 (currently planned for 4Q2018) which will provide a new workload balancing distribution algorithm which is much better
  3. Only if necessary consider setting RTPIFACTOR in IEAOPTxx (e.g. 50%) to reduce the possibility of a sudden redirection of workload to the alternate Db2 member when PI >> 1
- **Supporting recommendations**
  - CONDBAT on each member should be set higher than the sum of all possible connections across all connection pools from all application servers plus 20% to avoid reduction in health of Db2 member
    - 80% of CONDBAT=Health/2, 90% of CONDBAT=Health/4

  MAXDBAT should be set to a value which permits normal workload levels, AND allow for peaks, AND possible Db2 member outage … but not so high as to allow a 'tsunami' of work into the system
  - MAXCONQN can be used as the vehicle to limit queuing → force early redirection of connections wanting to do work to the other Db2 member

Use of High-Performance DBATs

- **High-Performance DBATs have the potential to provide a significant opportunity for CPU reduction by**
  - Avoiding connection going inactive and switching back to active later
  - Avoid DBAT being pooled at transaction end i.e., going back into DDF thread pool for reuse by probably a different connection
  - Supporting true RELEASE(DEALLOCATE) execution for static SQL packages to avoid repeated package and statement block allocation/deallocation
- **Very powerful performance option, but can be dangerous … if not used intelligently**

Use of High-Performance DBATs …

- **Important operational considerations**
  - Do not use RELEASE(DEALLOCATE) on common widely shared across distributed workloads as it will considerably drive up the requirement for MAXDBAT
    - Risk of not having enough DBATs available for DRDA workloads that are not using High-Performance DBATs
    - Worst case running out of DBATs completely or escalating thread management costs
  - Check that all the ODBC/JDBC driver packages in collection NULLID are bound as RELEASE(COMMIT) – otherwise they must be rebound with RELEASE(COMMIT)
    - WARNING: Since 9.7 FP3a, default BIND option for Db2 client driver packages has been RELEASE(DEALLOCATE)
  - Be very careful about DDF workloads re-using common static SQL packages used by CICS and/or batch workloads bound with RELEASE(DEALLOCATE)
  - Do not over-inflate the application server connection pool definitions, otherwise it will considerably drive up the demand for High-Performance DBATs

Use of High-Performance DBATs …

- **Recommendations**
  - Consider rebinding with RELEASE(DEALLOCATE) the static SQL packages that are heavily-used and unique to specific high-volume DDF transactions
  - Enable High-Performance DBATs by using the -MODIFY DDF PKGREL(BNDOPT)
  - Be prepared to use -MODIFY DDF PKGREL(COMMIT)
    - Allow BIND, REBIND, DDL, online REORG materialising a pending ALTER to break in
    - Switch off High-Performance DBATs at first signs of DBAT congestion i.e. overuse of DBATs

Overuse of UTS PBG tablespace and MAXPARTS

- **Primary driver for the developing UTS PBG tablespace was the removal of the 64GB limit for classic segmented tablespace and avoid the disruptive migration to classic partitioned tablespace**
- **Some considerations**
  - All indexes are going to be NPIs
  - Limited partition independence for utilities (REORG, LOAD)
  - Partitioning not used for query parallelism
  - Degraded insert performance (free space search) as the number of partitions grow
  - If REORG a partition list/range, it may encounter undetected deadlock between applications and REORG during the SWITCH phase (i.e. drain and claim in different order)
  - REORG PART will fail for a full UTS PBG partition if FREEPAGE or PCTFREE are non-zero
  - Setting system parameter REORG_DROP_PBG_PARTS = ENABLE could lead to operational issues if the number of PARTs are pruned back
    - No point-in-time recovery prior to the REORG that prunes partitions
    - Cannot use DSN1COPY to move data between Db2 systems
- **Should not be using UTS PBG as the design default for all tables (with large number of partitions)**

Overuse of UTS PBG tablespace and MAXPARTS …

- **General recommendations for use of UTS PBG tablespace**
  - Only use UTS PBG tablespace as the alternative and replacement for classic segmented tablespace
  - A table greater than 60GB in size should be created as a UTS PBR tablespace
  - Good reasons to limit number of partitions - should have as few partitions as possible - ideally only 1
  - DSSIZE and SEGSIZE should be consistent with the target size of the object e.g.
    - Small size object: DSSIZE = 2GB and SEGSIZE = 4
    - Medium size object: DSSIZE = 4GB and SEGSIZE = 32
    - Large size object: DSSIZE = 64GB and SEGSIZE = 64
  - REORG at the table space level unless do not have sufficient DASD space for sort
  - Setting system parameter REORG_DROP_PBG_PARTS = DISABLE?
    - If required to prune back the number of partitions
      - Use online system parameter to temporarily enable for controlled use
    - Better still, in Db2 12, use the DROP_PART YES option of REORG

Cleaning up after STATCLUS = STANDARD

- **Problem**
  - In Db2 9, introduced a new Data Repeat Factor (DRF) statistic and implemented a new improved formula for calculating CLUSTERRATIO (CR)
  - These enhancement was covered by a new system parameter STATCLUS=STANDARD|ENHANCED
  - Some installations were of the opinion that STATCLUS=STANDARD was a good defensive choice
    - WRONG!
  - In Db2 11, Db2 dropped the system parameter STATCLUS and started enforcing STATCLUS=ENHANCED behaviour
    - Resulting in cases of poor access path choices after running tablespace partition level REORG and/or index REORG
  - Challenge is how to clean up?

Cleaning up after STATCLUS = STANDARD …

- **CLUSTERRATIO & DATAREPEATFACTORF are mostly influencing the non-clustering indexes**
  - As a “light weight” approach could just run index RUNSTATS on all indexes - without REORGs, and also without running tablespace RUNSTATS
    - However, not a good idea to collect STATISTICS on data that may not be well clustered/organised
    - CLUSTERRATIO degrades more quickly on a clustering index than other indexes due to unclustered inserts
  - As an alternative could run index REORGs only and update the CLUSTERRATIO = 100% for those indexes with CLUSTERING=‘Y’
    - Although that is statistics manipulation that I am not a huge fan of
  - Correct approach and safest choice is to run tablespace REORG with inline STATISTICS
    - But does have high cost/impact due to tablespace level REORG
    - Prioritise tables with more than 1 index

"MBA lock"

- **Read-LSN values are maintained at buffer pool level but only when LOBs and/or static sensitive scrollable cursors are used**
- **Each Db2 member maintains a single MBA structure containing read-LSN values for all buffer pools**
- **The MBA (BMC_MBAO) lock is used in data sharing to serialise SCA updates for read-LSN tracking**
  - Read-LSN value is tracked and maintained at the buffer pool level
  - But the buffer pool ID is not part of the resource name on the lock
- **The MBA lock is acquired by Db2 Buffer Manager to**
  - Register a read-LSN
  - Retrieve the read-LSN for the buffer pool and it's been marked as invalid - so it has to be recalculated from the information in the SCA
- **Common cause of 'L-Lock Other' contention**
- **This lock has a very large CTLTOV multiplier of 24 applied to lock timeout value**
  - Actual timeout may happen at (CTLTOV * timeout period) + 4 * DEADLOCK TIME value
- **Observed a number of situations where the IRLM Global Deadlock Detection Manager locks up completely for 30 seconds and there are long waiter chains for the MBA lock**
- **Resulting in excessive CPU resource consumption during IRLM deadlock detection, possibly resulting in IRLM messages DXR167E, DXR186I and DXR187I**

"MBA lock" …

- **Strong recommendation to apply PTFs for APARs PI78061 and PI89629 as a defensive measure**
  - Prior to APAR PI78061, Db2 member would also get the MBA lock during commit (declaim) when releasing the last read-LSN for the buffer pool
  - APAR PI78061 removed the locking at declaim, by leaving the read-LSN in place until the pseudo-checkpoint process cleans it up (within a few seconds)
    - This can also reduce the locking on the register side, particularly for the case where an agent is repeatedly doing register/commit/register/commit because at the second and subsequent registers the earlier one is probably still there, and lower than his new one
    - The burden of advancing the value is now on the system agent
    - PTF UI47007 superseded by PTF UI47278
  - APAR PI89629 provides further improvement by taking the MBA lock out of deadlock processing and changing the read-LSN invalidation notify message
    - Db2 member will no longer just mark the read-LSN as "invalid" and let all of the agents who want to retrieve it fight over the MBA lock to refresh the value from the SCA
    - Db2 member will now send the new value in the notify message, so the other Db2 members can update it immediately and it can be retrieved without having to get the MBA lock

IRLM query requests for package break-in

- An IRLM query request is used to see if there are X-package lock waiters on a package during commit as part of commit phase 1 broadcast
- The break-in at commit is charged to Acctg Class 2 CPU Time for both local allied and DBATs regardless under read-only or update transactions
- The IRLM query request is “fast path” and a single query request has little or no performance impact
- There is an IRLM query request per RELEASE(DEALLOCATE) package per commit (per lock token basis)
- If there are many RELEASE(DEALLOCATE) packages loaded by commit time in a long-running thread, then there will be an IRLM query request for each one of those packages at that time
- It is possible to accumulate many distinct RELEASE(DEALLOCATE) packages across 1000 successive commits on each CICS-Db2 thread i.e., REUSELIMIT is 1000 in CICS definitions, and many different transactions running against the same DB2ENTRY in CICS definitions, and multiple packages per commit
- A big multiplier exaggerates the CPU overhead of IRLM query requests
- Likely to be a problem when applications are using many fine-grained packages e.g., CA Gen (CoolGen)
- Solution: set system parameter PKGREL_COMMIT=NO and toggle on/off outside of peak period “4HRA”

Db2 maintenance issues with batch utilities

- **Requiring a fix on both Db2 and batch sides in order to take advantage of some improvement or new feature is BAU, is by design**
- **However, requiring a fix on both Db2 and batch sides in order to avoid causing a problem is just not acceptable to customers and is cause for a PTF to be marked in error (PE)**
  - Db2 should never ship a PTF that causes a problem when applied on one side and not the other, just like we would not tolerate it if Db2 caused a problem due to a PTF being applied to one member in data sharing group and not the other

Running CHECK utilities with SHRLEVEL CHANGE

- **Db2 CHECK utilities are critical diagnosis tools in the event of data corruption**
  - Identify objects that need repairing/recovering and assess the extent of the damage
- **Many installations not currently set up to run the CHECK utilities non-disruptively**
  - System parameter CHECK_FASTREPLICATION = PREFERRED
  - System parameter FLASHCOPY_PPRC = blank
- **Having system parameter CHECK_FASTREPLICATION = PREFERRED may become an availability exposure**
  - CHECK with SHRLEVEL CHANGE would be allowed to run, but would not be able to use FlashCopy to create the shadow objects → objects could be left in RO status for many minutes whilst they are being copied with standard I/O and this would likely be very disruptive causing an application outage
  - Will you take an application outage, or delay the discovery of the damage and the extent thereof?
- **Recommendations**
  - As an immediate defensive measure, set ZPARM CHECK_FASTREPLICATION = REQUIRED (default since Db2 10)
    - Protection against misuse of CHECK SHRLEVEL CHANGE
    - CHECK utility will fail if FlashCopy cannot be used

Running CHECK utilities with SHRLEVEL CHANGE …

- **Recommendations …**
  - Better still exploit data set FlashCopy to run the CHECK utilities non disruptively
    - Requires only a small investment in temporary disk storage to create shadow objects
    - Optional: system parameter UTIL_TEMP_STORCLAS can be used to specify a storage class for the shadow data sets (if left blank, the shadow data sets will be defined in the same storage class as the production pageset)
    - Some special considerations apply when using DASD-based replication
      - Shadow data sets must be allocated on a pool of volumes outside of async mirror (Global Mirror, z/OS Global Mirror, XRC)
      - With sync mirror (Metro Mirror, PPRC) and assuming Remote Pair Flashcopy
        - Set system parameter FLASHCOPY_PPRC = REQUIRED (default Db2 10)
        - Or shadow data sets must be allocated on a pool of volumes outside of the mirror

ZPARM UTIL_TEMP_STORCLAS = blank

```
COPY DATASET(FILTERDD(SYS00009)) DEBUG(FRMSG(DTL))-
REPLACEUNCONDITIONAL RENUNC( -
       (*.*.*.*.I0001.*,*.*.*.*.J0001.*), -
       (*.*.*.*.J0001.*,*.*.*.*.I0001.*)) -
FCNOCOPY FASTREPLICATION(REQUIRED ) -
FCTOPPRCPRIMARY(PRESMIRREQ ) -
SHARE CANCELERROR TOLERATE(ENQFAILURE)
```

ZPARM UTIL_TEMP_STORCLAS = SCDB2FC

```
COPY DATASET(FILTERDD(SYS00009)) DEBUG(FRMSG(DTL))-
REPLACEUNCONDITIONAL RENUNC( -
       (*.*.*.*.I0001.*,*.*.*.*.J0001.*), -
       (*.*.*.*.J0001.*,*.*.*.*.I0001.*)) -
STORCLAS(SCDB2FC ) BYPASSACS(**) -
FCNOCOPY FASTREPLICATION(REQUIRED ) -
FCTOPPRCPRIMARY(PRESMIRREQ ) -
SHARE CANCELERROR TOLERATE(ENQFAILURE)
```

Local buffer pool allocation – virtual and real memory

- **Difference in behaviour between Db2 10 vs. Db2 11/Db2 12 for two reasons**
  - Performance issue using 1MB increments in Db2 10
    - Taking quite huge amount of CPU time to allocate the buffers incrementally
      - DBM1 SRB time, zIIP eligible now but it was not in Db2 10
      - It costs large CPU immediately after Db2 restart until the buffer pool populates to VPSIZE
  - Functional issue related with serialisation during internal ALTER
  - In Db2 11, return to allocating the buffer pool up to VPSIZE straight away
    - Difference between small 4K size frames vs. large size 1M or 2G size frames
      - Large frames are allocated up front, page fixed, backed real memory
      - Small frames are allocated up front (virtual), but page fixed and backed by real memory only as used

Summary

- Learn recommended best practice
- Learn from the positive and negative experiences from other installations

John Campbell and Maureen (Mo) Townsend
IBM Db2 for z/OS Development
campbelj@uk.ibm.com and motown@us.ibm.com

Session code: A01

Please fill out your session evaluation before leaving!