Buffer Pool Tuning

John Campbell
Email: john.Campbell@triton.co.uk

Agenda

- Local Buffer Pool
  - Introductory comments
  - Common problems
  - Use multiple buffer pools
  - Key metrics for read efficiency
  - Page classification and LRU processing
  - Buffer pool simulation
  - Use of long-term page fixed buffers
  - Use of large size real memory frames
  - Contiguous buffer pool

Agenda …

- Group Buffer Pool
  - Common problems
  - Best practice for CFRM Policy settings
  - Key metrics and messages
  - Rough calculation of required number of directory entries and changed pages

Local Buffer Pool

Introductory Comments

- Establish clear tuning objects
  - CPU efficiency?
  - Improved response time and throughput?
  - Or both?
- Estimate potential for savings
  - Cannot make savings on read I/O which have been previously eliminated
  - Law of Diminishing Returns applies when moving out beyond “knee” onto flat part of buffer pool reference curve
    - Plot of Hit Ratio vs. Buffer Pool size (VPSIZE)
- Establish available additional real memory budget
  - Must have sufficient real memory available on the LPAR to fully back total buffer pool requirement
  - Must also have sufficient available (spare) memory to absorb Db2 dump without causing paging

Introductory Comments …

- Tune VPSEQT to stop sequentially accessed pages from monopolizing buffer pool
- Collect and analyze buffer pool performance data for complete 24-hour operational processing cycle
  - Include peak processing periods
  - Month or period end processing
- Especially when little or no available real memory available consider
  - Redistributing memory across buffer pools
  - Reassigning heavily accessed object to different buffer pool
- Ambition on buffer pool tuning should be proportional to the amount of effort you can afford to put in

Common Problems

- Misclassification of objects (random vs. sequential) assigned to the wrong buffer pool
- Too many buffer pools causing fragmentation of memory use
- Increasing VPSIZE and not reducing VPSEQT, VDWT, DWTH to
  - Maintain the same degree of “trickle” write in between successive system checkpoints
  - Limit sequential pages to the same degree
- Over aggressive tuning down the value of VPSEQT to drive preferential stealing of sequential pages
- Over commitment of real memory causing paging at the wrong time
- Heavily optimized tuning based on limited set of buffer pool performance data
- “Brute force” method of throwing increased real memory at buffer pool size generating limited benefit
- Not appreciating impact of shrinking buffer pool when WLM-managed buffer pool turned on
  - AUTOSIZE(YES) option on –ALTER BPOOL

Use multiple buffer pools

- Multiple buffer pools recommended
  - -DISPLAY BPOOL for online monitoring
  - Data set statistics via –DISPLAY BPOOL LSTATS (IFCID 199)
  - Useful for access path monitoring
- Dynamic tuning
  - Full exploitation of buffer pool tuning parameters for customized tuning
  - -ALTER BPOOL is synchronous and effective immediately, except for buffer pool contraction when must wait for updated pages to be written out
  - Reduced buffer pool latch contention
- Catalog/directory is in BP0, BP8K0, BP16K0 and BP32K0
- Minimum of 4 user buffer pools: user index (4K), user data (4K) and work files (4K and 32K)
- Do not fragment your buffer pool memory too much

Key metrics for read efficiency

- Hit Ratios (HR) = percentage of times the page was found in the buffer pool
  - System HR = (Total getpages – Total pages read) / Total getpages * 100
    - Total getpages = random getpages + sequential getpages
    - Total pages read = synchronous reads for random getpages + synchronous reads for sequential getpages + pages read via sequential prefetch + pages read via list prefetch + pages read via dynamic prefetch
    - Can be negative because of heavy use of dynamic prefetch
  - Application HR = (Total getpages – Synchronous reads) / Total getpages * 100
    - Total getpages = random getpages + sequential getpages
    - Synchronous reads = synchronous reads for random getpages + synchronous reads for sequential getpages
    - An overall hit ratio over 90% is a good goal, but will likely vary across intervals and is very workload dependent

Key metrics for read efficiency …

- Residency Time (RT) = average time that a page is resident in the buffer pool
  - System RT (seconds) = VPSIZE / Total pages read per second
    - Total pages read = synchronous reads for random getpages + synchronous reads for sequential getpages + pages read via sequential prefetch + pages read via list prefetch + pages read via dynamic prefetch.
    - Achieving a system residency time of over 45-60 seconds is a good target
  - Random page residency time (secs) = max(System residency time, (buffer pool size * (1-VPSEQT/100) / sync pages read per second)
  - Sequential page residency time (secs) = min(System residency time, (buffer pool size * (VPSEQT/100) / async pages read per second

Page classification and LRU processing

- Pages in a buffer pool are classified as either random or sequential in order to protect transaction performance from the effects of queries and batch
- Pages read from DASD
  - Page read in via synchronous I/O is classified as random
  - Page read in via any prefetch I/O is generally classified as sequential
- Pages that already exist in the buffer pool from previous work
  - Sequential page is re-classified as random when subsequently touched by a random getpage
  - Random page is never re-classified as sequential

Page classification and LRU processing …

- LRU management
  - LRU chain (oldest to youngest) contains all pages both random and sequential pages
  - SLRU chain (oldest to youngest) only contains sequential pages
  - When a buffer is referenced, it becomes the “youngest” or “most recently used” buffer on the respective chain(s)
- Db2 has a mechanism to prevent sequentially accessed data from monopolizing the buffer pool space and pushing out useful random pages
  - Db2 steals from the LRU chain until VPSEQT is reached, and then steals preferentially from the SLRU chain
  - VPSEQT option on ALTER BPOOL is your weapon of choice stop sequential pages dominating and pushing out useful random pages

Buffer pool simulation

- Db2 has an accurate buffer pool simulation capability built into the Db2 engine
  - No need for additional tools
  - No need for expensive performance traces to be collected
- Can simulate larger buffer pool size
- Cannot simulate
  - Reducing buffer pool size
  - Transferring buffers from one buffer pool to another
  - Moving objects to a different buffer pool

Buffer pool simulation …

- Db2 only uses additional memory for control blocks to track pages in the simulated buffer pool extension
  - No additional memory is allocated for the actual buffers supporting the extension (SPSIZE)
- Db2 resource requirements
  - Memory: 2% of SPSIZE*4K
  - CPU: approximately 1% for each simulated buffer pool

Buffer pool simulation …

- Triggered by new SPSIZE > 0 option on –ALTER BPOOL
  - SPSIZE is the increase in the buffer pool size value that complements VPSIZE
    - Total buffer pool size simulated is VPSIZE + SPSIZE
  - Simulation should be used for long time duration
  - Remember to turn the simulation off later when you have finished!
- Provides metrics on avoidable read IO
  - Statistics Trace Class 1 (IFCID 2)
  - -DISPLAY BPOOL DETAIL (see message DSNB432I)

Long term page fix

- Triggered by PGFIX(YES) on –ALTER BPOOL and requires re-allocation of buffer pool
- Opportunity for CPU saving
- Amount of CPU saving related to IO intensity
- Predicated by strong advice (as always) to fully back the buffer pool requirement with real memory

Large frame size

- Opportunity for CPU saving
  - Consolidating 4K size frames into 1M or 2G size large frames
  - Potential for improved hit ratio in Translation Lookaside Buffer (TLB) improving CPU efficiency
  - CPU saving related to getpage intensity
- Predicated on buffer pool using long term page fix and buffer pool reallocation

Considerations using Contiguous Buffer Pool

- At very large buffer pool size, scaling issues start to emerge because of running long (PMB) control block chains
  - Maximum size: few 100s of GB
- For super sized buffer pools, consider using Contiguous Buffer Pool (PGSTEAL NONE)
  - Above recommendation conditional on buffer pool adequately super sized to hold all the pages for all the objects (data in memory)
  - Eliminates LRU and Hash chain management

Considerations using Contiguous Buffer Pool …

- PGSTEAL(NONE) local buffer pool = Contiguous Buffer Pool under Db2 12
- When the object is physically opened
  - All the pages for the object are brought into the local buffer pool using sequential prefetch
  - After the open, prefetch is not much of a factor for PGSTEAL(NONE) objects
  - Except when there is GBP-dependency change
    - Pages in the local buffer pool are invalidated
    - APAR PI59168 introduced a change to invoke sequential prefetch for a PGSTEAL(NONE) object when a GBP-dependency change occurs
      - Db2 checks for VPSEQT=100 before invoking sequential prefetch after the GBP-dependency change
    - All users should set VPSEQT=100 to drive sequential prefetch to repopulate the local buffer pool with valid pages after a GBP- dependency change
- WLM AUTOSIZE is not supported
  - WLM AUTOSIZE feature has no concept of the contiguous buffer pool model
  - Contiguous Buffer Pool not registered to WLM

Considerations using Contiguous Buffer Pool …

- Increasing VPSIZE will not stop the use of the overflow area
  - Buffers for the object are allocated at physical open
  - If the object is extended with more pages
    - Db2 will attempt to allocate more contiguous buffers for the object, but not if the object is already using the overflow
  - Once an object starts using the overflow, the pages in the overflow buffers are scattered
    - Db2 has no support to quiesce and convert the pages in the overflow to contiguous buffers
  - Operational bypass
    - Physical close of the object will free up all the contiguous buffers used for the object
      - Next physical open can use the increased VPSIZE
    - Recycle Db2 member or subsystem
    - Use -STOP DB SPACENAM followed by –START DB SPACENAM, OR -ALTER BPOOL VPSIZE(0) followed by -ALTER BPOOL VPSIZE(n)

Considerations using Contiguous Buffer Pool …

- Incompatible change for PGSTEAL(NONE) when migrating from Db2 11 to Db2 12
  - For Db2 11, Db2 can use 2G size frames for PGSTEAL(NONE) buffer pools
  - For Db2 12 after APAR PH22469
    - Db2 will go ahead to honour the request to use 2G size frames
    - But Db2 will switch to using PGSTEAL(LRU) instead of using PGSTEAL(NONE)

Group Buffer Pool

Common problems

- No clear ownership or responsibility for ongoing tuning across z/OS and Db2 Teams
- Relying exclusively on XES Auto Alter but insufficient resources committed
  - SIZE (maximum size) limit has been reached on group buffer pool
  - RATIO on some group buffer pools has been adjusted to 1:1
- Shortage of directory entries resulting in directory entry reclaims, associated page invalidation in respective local buffer pool and incurring wasteful refresh of pages from DASD
- Low average page residency time resulting in poor XI read ratio and page in local buffer pool having to be refreshed from DASD

Common problems …

- Not adjusting group buffer pool (INITSIZE, SIZE, RATIO) when increasing VPSIZE of associated local buffer pool
- No recognition of the impact on group buffer pool as a result of implementing WLM-managed local buffer pool
  - AUTOSIZE(YES) option on –ALTER BPOOL

Best practice for CFRM Policy settings

- Use XES Auto Alter (autonomic)
  - Tries to avoid Structure Full and Directory Entry Reclaim conditions
  - Conservative – gradual change as opposed to quick radical change
- CFRM Policy
  - ALLOWAUTOALT(YES)
  - Set MINSIZE to INITSIZE
  - Set FULLTHRESHOLD = 80-90%
  - Set SIZE to 1.3-2x INITSIZE
- Perform periodic review and update to CFRM policy based on actual allocation and ratio
  - Especially applies when the allocated size reaches SIZE

Key metrics and messages

- Design for and monitor for 0 (zero)
  - Cross invalidations due to directory entry reclaims
    - See DSNB788I from -DISPLAY GBPOOL(*) GDETAIL(*) TYPE(GCONN)
  - Writes failed due to lack of storage
    - See DSNB762I from -DISPLAY GBPOOL(*) GDETAIL(*) TYPE(GCONN)
- Design for miss ratio of less than 10% for synchronous read due XI
  - Sync.Read(XI) miss ratio = SYN.READ(XI)-NO DATA RETURN / TOTAL SYN.READ(XI)
    - TOTAL SYN.READ(XI) = SYN.READ(XI)-DATA RETURNED + SYN.READ(XI)-NO DATA RETURN
  - See Db2 Statistics Trace Class 1

Key metrics and messages …

- Track for space shortage via automation and instigate action to investigate
  - DSNB319A (75% of available storage used)
  - DSNB325A (Critical, 90% of available storage used)
  - DSNB327I (adequate available storage)

Rough calculation of required no. of directory entries & changed pages

- Basics
  - Data page size is either 4K, 8K, 16K or 32K
  - Directory entry size is dependent on CFLEVEL
    - Assume 432 bytes for 4K page size
    - Assume 530 bytes for 32K page size
  - Determine peak no. of changed pages written per second from Db2 Statistics Trace Class 1
    - CHANGED PGS SYNC.WRTN
    - plus CHANGED PGS ASYNC.WRTN
    - plus 20% “padding” factor for unforeseen peaks
  - Establish target for average changed page residency time (30 – 180 seconds)

Rough calculation of required no. of directory entries & changed pages …

1. Calculate total number of changed pages required (A) by multiplying number of changed pages per second by target time for average changed page residency
   - Calculate total memory requirement (B) for changed pages by multiplying (A) by data page size
2. Calculate total number of directory entries required (C)
   - (VPSIZE * No. of Db2 members) + total changed pages written per second across all Db2 members
   - Calculate total memory requirement (D) for directory entries: (C) * directory entry size
3. Calculate total memory requirement for GBP: (B) + (D)
4. Calculate RATIO for GBP: (C) / (A)

Questions?