#SHAREorg

SHARE
Technology · Connections · Results

DB2 10 for z/OS Buffer Management

Jeffrey Berger
IBM

August 7, 2012
Session 11297

SHARE in Anaheim 2012

#SHAREorg

SHARE
Technology · Connections · Results

Disclaimer

The information contained in this presentation is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided "as is" without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans strategy, which are subject to change by IBM without notice. IBM shall not be responsible for damages arising out of the use of, or otherwise related to, this presentation or any other documentation. Nothing contain in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM (or its suppliers or licensors), or altering the terms and conditions of any agreement or license governing the use of IBM products and/or software.

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.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks The following terms are trademarks or registered trademarks of other companies and have been used in at least one of the pages of the presentation:

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both: DB2, DB2 Connect, DB2 Extenders, Distributed Relational Database Architecture, DRDA, eServer, IBM, IMS, iSeries, MVS, z/OS, zSeries

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

References

- DB2 for z/OS Best Practices, webcast
  http://www.ibm.com/developerworks/data/bestpractices/db2zos/
  - Best Practices for DB2 for z/OS Local and Group Bufferpools, John Campbell
  - Best Practices for DB2 for z/OS Inline LOBs (Large Objects), Jeffrey Berger
- *DB2 Version 10.1 for z/OS Command Reference*, SC19-2972-04
- *DB2 Version 10.1 for z/OS Managing Performance,* SC19-2978-05
- *Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS V4R1 Report Reference,* SC18-9984-02
- *DB2 9 for z/OS Performance Topics*, SG24-7473
- *DB2 10 for z/OS Technical Overview*, SG24-7892
- *DB2 10 for z/OS Performance Topics*, SG24-7942

SHARE
Technology · Connections · Results

Agenda

- Buffer pool allocation
- Page size selection
- LOB table space considerations
- Random/sequential page classification and LRU processing
- Asynchronous prefetch I/O
- Deferred writes
- Critical thresholds
- Data sharing and LOG NO
- Long term page fixing and 1MB page frames
- In-memory buffer pools
- Free space considerations
- Indirect references

DB2 Buffer Pools

- 4K Page Size: BP0, BP1, ..., BP39
- 8K Page Size: BP8K0, BP8K1, ..., BP8K9
- 16K Page Size: BP16K0, BP16K1, ..., BP16K9
- 32K Page Size: BP32K, BP32K1, ..., BP32K9
- BP0, BP8K0, BP16K0 and BP32K used for DB2 catalog/directory
  - Usually small, but it's good to increase the size (especially BP0) while migrating the DB2 catalog to a new DB2 release

Buffer Pool Tuning

- 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 BP tuning parameters for customized tuning
    - ALTER BPOOL is synchronous and effect immediately, except for BP contraction because of wait for updated pages to be written out
  - Reduced BP latch contention
- Catalog/directory is in BP0, BP8K0, BP16K0 and BP32K0
- Minimum of 4 user BPs: user index (4K) and user data (4K) and work files (4K and 32K)
- Don’t fragment your buffer pool space too much

Buffer pool allocation

- BPs are created in DB2's DBM1 address space, above the 64-bit bar
  - Created at first data set Open
  - Buffer pool deleted when all referenced data sets are closed
  - DB2 10 allocates and extends piecemeal on demand
    - Avoids real storage penalty of over-sizing a buffer pool if PGFIX(YES) is used
- AUTOSIZE (YES)
  - WLM may direct DB2 to increase or decrease VPSIZE by up to 25% based on system needs
  - If DB2 is restarted, the 25% is relative to the size when DB2 is restarted

Buffer pool parameters

- How many buffer pools are needed and how to assign objects to buffer pools
- Buffer pool size - VPSIZE
- Virtual Pool Sequential Threshold – VPSEQT (default 80%)
- Horizontal Deferred Write Queue Threshold – DWQT (default 30%)
- Vertical Deferred Write Queue Threshold – VDWQT (default 5%)
- To page fix or not to page fix – PGFIX (YES or NO) (default NO)
- Page steal method – PGSTEAL (LRU, FIFO, or NONE) (default LRU)
- Auto size – AUTOSIZE (YES or NO) (default NO)

Page size selection

- 4K pages usually optimize the buffer hit ratio
- Special considerations
  - The page needs to be large enough to store the max row size
  - DB2 can store at most 255 rows on a page
- When rows are large, a large page helps minimize DASD space consumption
  - On average, each page wastes a half-row of space
    - E.g. If you average 10 rows per page, you waste 5% of the space
- Index considerations
  - A large page is necessary for index compression
  - A large page minimizes index splits
  - A large page reduces the number of index levels
- A large page (8K or 16K) provides better sequential performance
- With DB2 10, a large page size helps enable inline LOBs, which may help improve I/O and CPU performance significantly

LOB table space page size considerations

- A page in a LOB table space contains only one LOB (i.e. one row)
  - The effect of LOBs on a buffer pool is the same as an ordinarily table that contains one row per page
- The auxiliary index is where most LOB performance problems emanate
- A small page size always provides the most space efficiency, especially when LOBs are small.
- If a LOB fits in one page, then it only takes one I/O to read the LOB
- Otherwise it takes a minimum of two I/Os
  - The second I/O will read up to 128KB
- DB2 10: Special tuning considerations for inline LOBs
  - See “DB2 for z/OS Best Practices website”

SHARE
Technology · Connections · Results

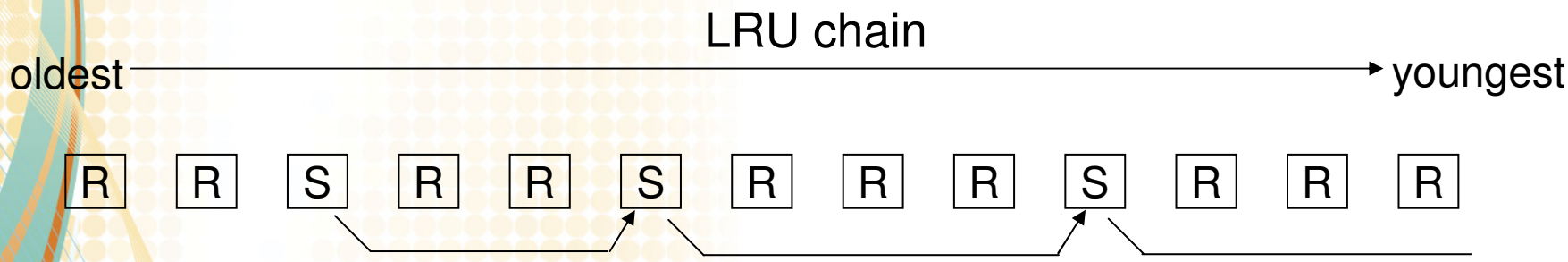
Three types of DB2 prefetch

- Sequential prefetch
  - Used for sequential table scans and by utilities
- Dynamic prefetch
  - Used for organized index scans and index-to-data access when the cluster ratio is high
- List prefetch
  - Used to scan disorganized data and, starting with DB2 10, disorganized indexes

Page classification and LRU processing…

- Pages in a BP are classified as either random or sequential in order to protect OLTP from the effects of queries
- Pages read from DASD
  - A page read in via synchronous I/O is classified as random
  - A page read in via any form of prefetch I/O is classified as sequential
- Pages that already exist in the BP from previous work
  - A random page is never re-classified as sequential
  - In DB2 V8, a sequential page was re-classified as random when subsequently touched by a random Getpage.
  - In DB2 9 or 10, pages are not reclassifed.
  - Sequoia is expected to behave like V8.

LRU management



- If the length of the SLRU chain is greater than VPSEQT, then DB2 always steals the oldest buffer from the SLRU chain. Otherwise it steals the oldest buffer on the LRU chain.
- When a buffer is referenced, it becomes the “youngest” or “most recently used” buffer on the chain

…Page Classification and LRU Processing

- DB2 has a mechanism to prevent sequentially accessed data from monopolizing the BP and pushing out useful random pages
  - Maintains two chains
    - LRU with all pages (random and sequential)
    - SLRU with only the sequential pages
  - Steals from the LRU chain until VPSEQT is reached, and then steals preferentially from the SLRU chain
- General recommendations
  - Set VPSEQT to 99% for the sort workfile BP, 90% for other workfile usage

Buffer Reclassification

- Buffer misses
  - A sequential buffer that is stolen for use by a random Getpage is supposed to be taken off the sequential LRU chain.
    - But this stopped happening in DB2 9
    - Impact: LRU is not really LRU anymore, meaning that a recently used buffer might get stolen more quickly than a less recently used buffer
    - Worst case: Mixture of sequential and random I/O with a moderately high VPSEQT such as 80%.

Watch for PM70270 to fix this problem

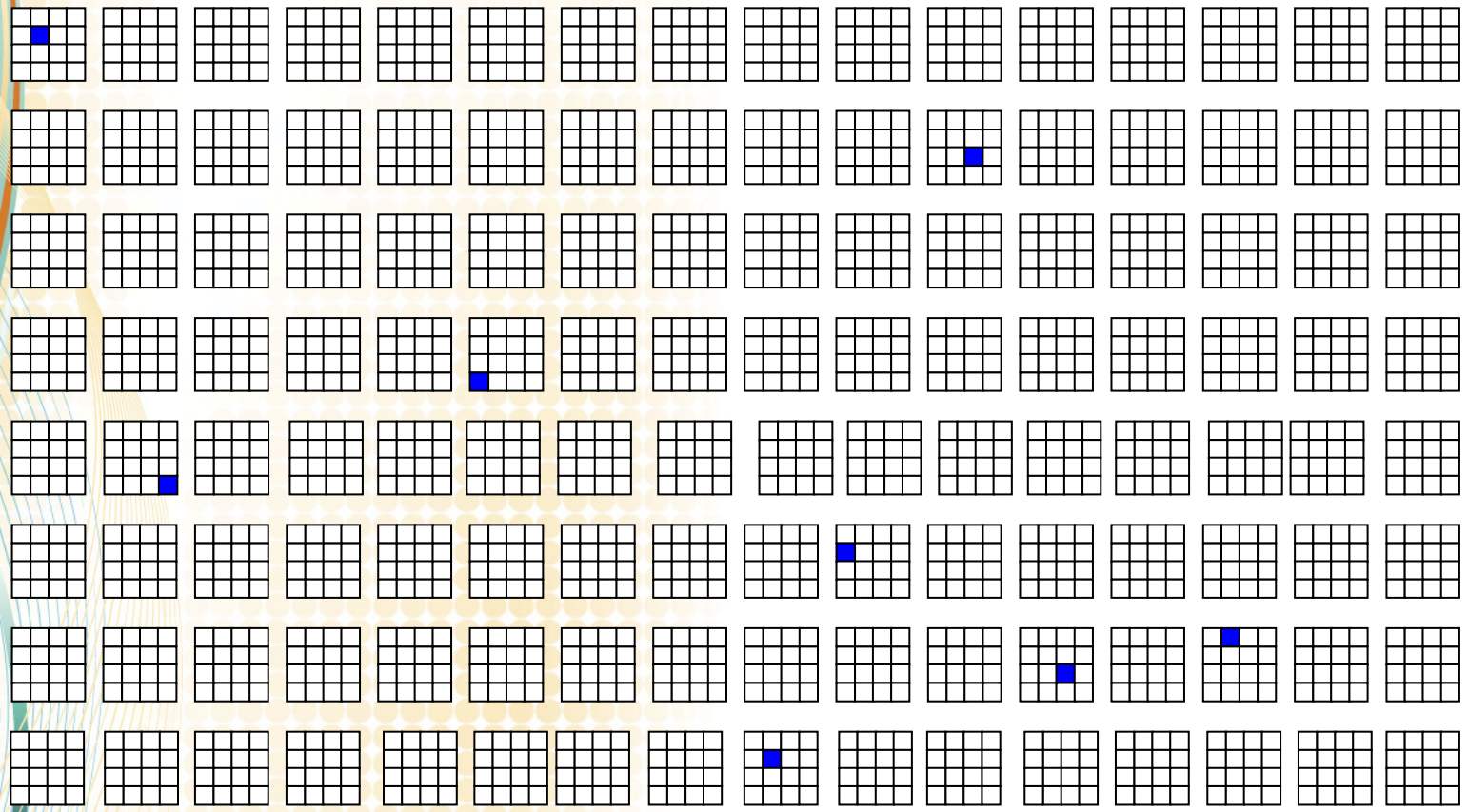
- Buffer hits
  - A sequential buffer that is touched by a random Getpage is supposed to be taken off the sequential LRU chain.
    - But this stopped happening in DB2 9
    - Impact: LRU is not really LRU anymore
    - Worst case is when OLTP and queries share the same objects and VPSEQT is low
    - Expect Sequoia to fix this problem

…Page Classification and LRU Processing

- Sequoia is expected to revamp the way pages are classified or reclassified
  - When dynamic prefetch is used, Sequoia will classify the "page-sequential" pages as sequential instead of random (unlike all prior DB2 versions)
    - Consequently dynamic prefetch cannot overwhelm the buffer pool
  - When list prefetch is used for disorganized index scans or RID list scans, the Getpages will be classified as sequential
    - Consequently these uses of list prefetch cannot overwhelm the buffer pool
  - Unlike DB2 9 or 10, a buffer may become reclassified from sequential to random

#SHAREorg

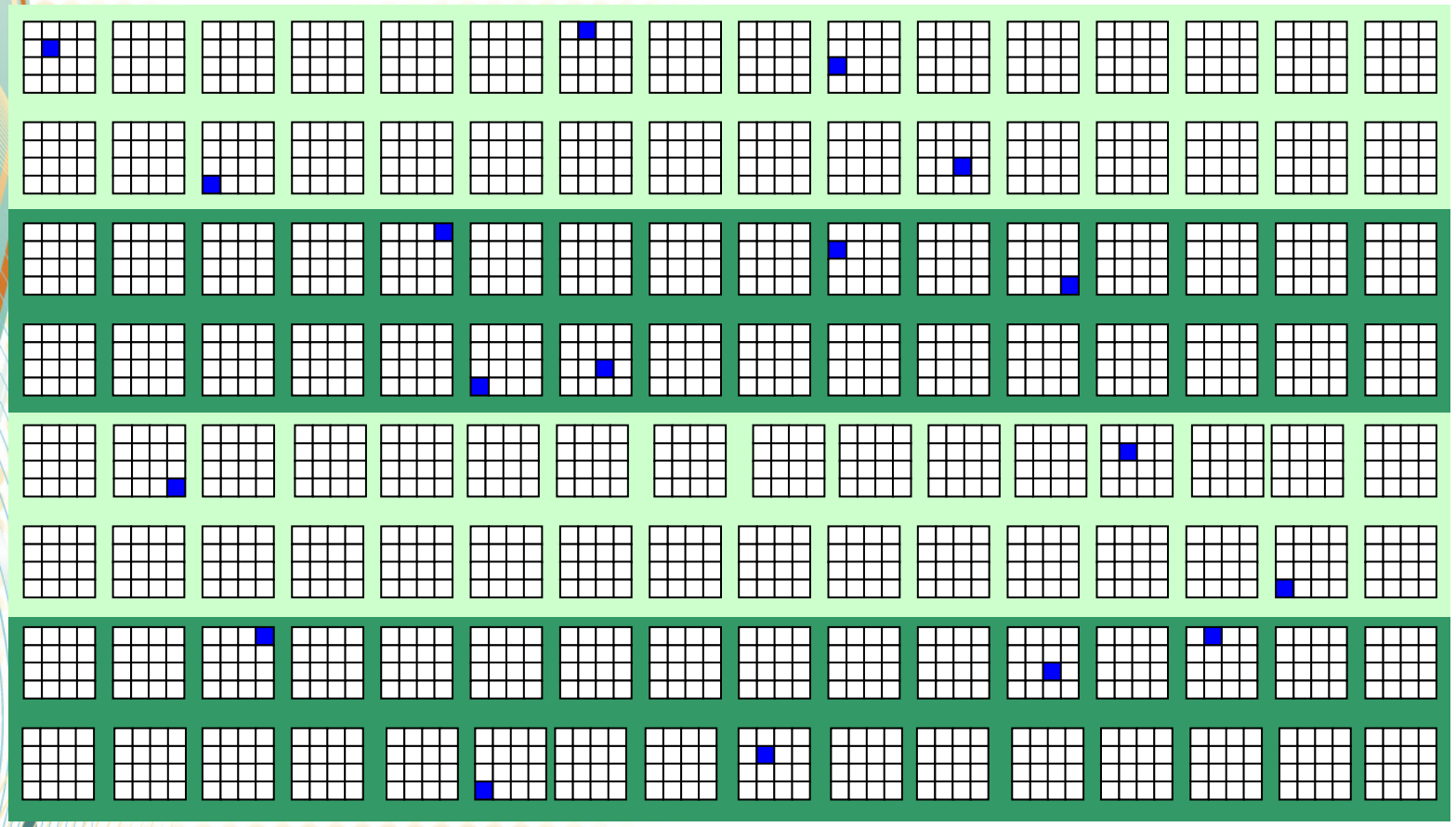
Sparse skip sequential with DB2 10



- DB2 10 does synchronous I/O for sparse pages

SHARE in Anaheim 2012

Dense skip sequential



- DB2 prefetches all of the pages, 32 pages per I/O
- When the Getpages “skip” pages, DB2 uses buffers and reads the skipped pages from DASD

DB2 Omegamon Performance Expert

$$\text{Buffer Hit Ratio} = \frac{\#\text{Getpages} - \#\text{Synch I/Os} - \#\text{Asynch Pages}}{\#\text{Getpages}}$$

Possible causes of a negative hit ratio

- If a page is prefetched that is “skipped”, the BP hit ratio may be negative, because #Asynch Pages may exceed #Getpages
  - This is not a problem, and increasing VPSIZE won’t change a thing

If you observe lots of page skipping, then possibly lower VPSEQT to limit the impact of dynamic prefetch on random I/Os

- Pages are being prefetched and then buffers stolen before the Getpages
  - This is a problem. Possible solutions are:
    - Increase VPSIZE or VPSEQT
    - Decrease the amount of sequential parallelism
    - Separate some objects into a different buffer pool

SHARE
Technology · Connections · Results

Sequential prefetch

- Starting with DB2 9, sequential prefetch is only used for table scans
  - Not good for skip sequential, or for a mixture of sequential and random access
- Uses prefetch trigger pages
  - For example, every 64th page triggers a prefetch for 64 pages
- One I/O at a time (except for the initial 2 I/Os)
- SQL prefetch quantity
  - If VPSIZE x VPSEQT >= 160MB, then sequential prefetch reads 256K per I/O. Else 128K.
  - Using 256K improves throughput by up to 20%
- Utility prefetch quantity
  - If VPSIZE >= 320MB, then utility sequential reads and writes transfer 512K. Else 256K.
  - With zHPF, using a larger I/O quantity improves 4K format writes by up to 50%.

Tuning for sequential I/O

- DB2 9 and above
  - If VPSEQT * VPSIZE >= 160MB, then for SQL, DB2 doubles the prefetch quantity
  - If VPSEQT * VPSIZE >= 320MB, then utilities double the prefetch quantity and format write quantity
- If you're channels are fast, consider setting VPSEQT low, but not so low as to reduce the prefetch quantity, or create a shortage of buffers for prefetch I/O

*Ensure VPSEQT * VPSIZE >= 320MB to get best sequential I/O performance.*

- It is best to keep "sequential tables" and "random tables" in different buffer pools, especially when dynamic prefetch is heavily used. (This will become less critical with Sequoia.)

Dynamic prefetch – sequential detection

- In DB2 9, some queries switched from Sequential Prefetch to Dynamic Prefetch
- Improved in DB2 10 – Row Level Sequential Detection
  - The first prefetch done by a Getpage after 5 out of 8 sequential *rows* are scanned. (DB2 9 only counted *pages*.)
- DB2 10 also introduced a progressive prefetch quantity.
  - The first prefetch reads 32K, then 64K, then 128K thereafter
- Prefetches sequential pages and does synchronous I/O for random pages
- Up to two I/Os at a time, 128K per I/O

List Prefetch

- Prefetch a specific list of pages
  - Scattered reads
  - Up to two I/Os at a time, 128K per I/O
- When does DB2 use list prefetch?
  - Queries
    - Starting with DB2 10, a scan of a disorganized index
    - Index-to-data access when the index cluster ratio is below 90%
    - Multiple index access pathIndex ANDing and ORing
    - LOBs
  - Utilities
    - Fast log apply
    - Incremental copy
    - New to DB2 10: Runstats table sampling

Optimizing list prefetch

Use IBM DS8700 or DS8800 storage wth zHPF, which uses List Prefetch Optimizer

- List Prefetch Optimizer works best with RAID 5 and storage pool striping

Upgrade to FICON Express 8S for best channel performance

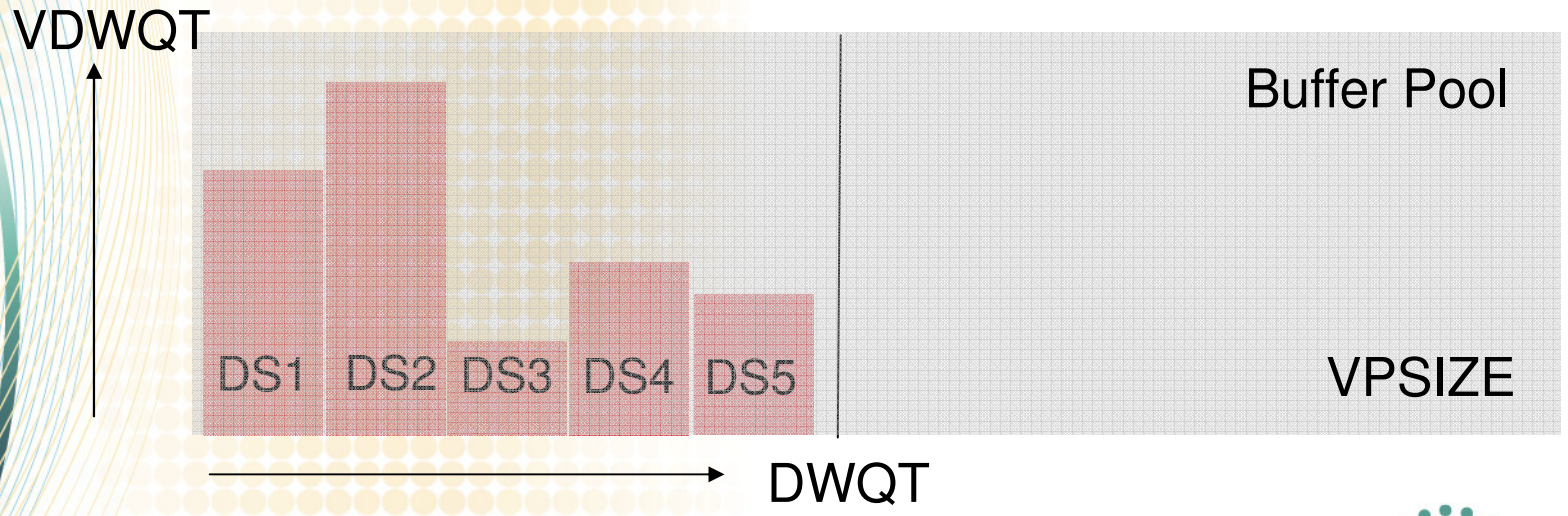
Tuning for sequential I/O versus random I/O

- Faster data paths (channels, switches, control units host adapters and device adapters) to optimize sequential I/O.
- If your data paths are fast, but your disks are slow, then lower VPSEQT to maximize your random buffer hit ratio.
  - However, be aware that lowering VPSEQT could also hurt your list prefetch buffer hit ratio (with DB2 9 or 10). So, if you don't have List Prefetch Optimizer, lowering VPSEQT could hurt you.

Anticipate that Sequoia will likely provide better performance when VPSEQT is small

Deferred Writes

- VDWQT (Vertical Deferred Write Queue Threshold) based on the data set level as a % of VPSIZE or number of buffers
  - DB2 schedules a write for up to 128 pages, sorts them in sequence, and writes them out in at least 4 I/Os. A page distance of 180 pages is applied to each I/O to avoid high page latch contention, since buffers are latched during I/O.
- DWQT (horizontal Deferred Write Queue) at the BP level



…Deferred Writes

- Setting VDWQT to 0 is good if the probability of re-writing pages is low
  - DB2 waits for up to 40 changed pages for 4K BP (24 for 8K, 16 for 16K, 12 for 32K) and writes out 32 pages for 4K BP (16 for 8K, 8 for 16K and 4 for 32K)
- Setting VDWQT and DWQT to 90 is good for objects that reside entirely in the buffer pool and are updated frequently
- In other cases, set VDWQT and DWQT low enough to achieve a "trickle" write effect in between successive system checkpoints
  - But… Setting VDWQT and DWQT too low may result in writing the same page out many times, short deferred write I/Os, and increased DBM1 SRB CPU resource consumption
    - If you want to set VDWQT in pages, do not specify anything below 128

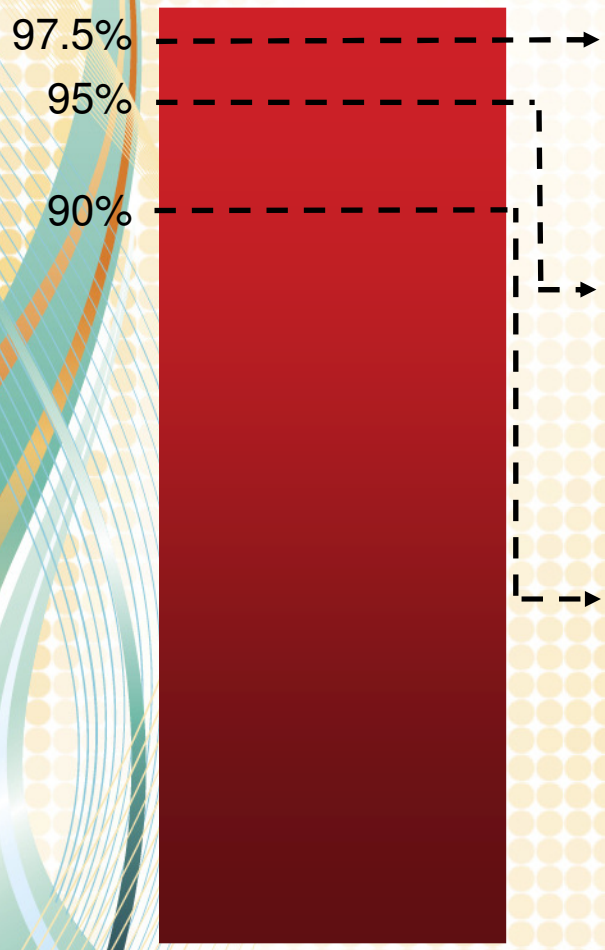
- Prefer hitting VDWQT instead of DWQT to increase the number of pages per I/O and reduce the number of I/Os
- Prefer hitting DWQT instead of VDWQT if you want to spread the I/Os over more data disks

Checkpoint interval

- Periodically, the checkpoint interval flushes out all of the buffer pools to DASD, causing a burst of I/Os
- DB2 restart has to read the logs since the start of the prior checkpoint interval and apply those log records
- Reducing the checkpoint interval will help avoid the burstiness of the writes, and reduce the restart time for DB2, but it may cause more write I/O

#SHAREorg

Critical Thresholds



Immediate Write Threshold

Checked at page update

>>After update, synchronous write

Data Management Threshold

Checked at page read or update

>>Getpage can be issued for each row sequentially scanned on the same page – potential large CPU time increase

Sequential Prefetch Threshold

Checked before and during prefetch

90% of buffers not available for steal, or running out of sequential buffers (VPSEQT with 80% default)

>>Disables Prefetch (PREF DISABLED – NOBUFFER)

Critical Thresholds

Minimize PREF DISABLED – NO BUFFER (*)

Keep DATA MANAGEMENT THRESHOLD to 0

- If non-zero
  - Increase BP size and/or
  - Reduce deferred write thresholds
  - Increase system checkpoint frequency (reduced CHKFREQ)
- (*) Can be much higher if prefetch intentionally disabled via VPSEQT=0
  - Interesting option for data in memory BP
    - Avoid the overhead of scheduling the prefetch engines when data is already in BP
  - No problem in that case
  - Consider using FIFO instead of LRU for PGSTEAL or, better yet with DB2 10, consider using PGSTEAL(NONE)

Data sharing and LOG NO

- If an object is defined as LOG NO, the Commits will synchronously force the pages to DASD or the CF
- If data sharing is used and there is inter-system R/W interest with GBPCACHE(YES), the deferred writes go to the CF, not DASD. Commits also write pages to the CF.
- Cast out writes from a GBP use write threshold similar to local buffer pool
  - Up to 64 pages per I/O, unlimited by the 180 page range, because there is no danger of page latch contention

SHARE
Technology · Connections · Results

Causes of poor write performance

- Problems with remote replication
  - E.g. Network contention
  - E.g. A poor control unit implementation can cause reads to queue behind the writes
- Poor local disk performance
  - E.g. RAID 5 rank is overloaded, 10K RPM HDDs
- Poor CF performance
  - E.g. CF links are overloaded
- Low write buffer residency time
  - Caused by VDWQT set too low?
  - Insufficient memory

Long-Term Page Fix for BPs with Frequent I/O

- DB2 BPs have always been strongly recommended to be backed up 100% by real storage to avoid paging

In a steady-state, PAGE-IN for READ/WRITE < 1% of pages read/written

- Given that there is no paging, you might as well page fix each buffer just once to avoid the repetitive CPU cost of page fix and free for each and every I/O
  - Option: ALTER BPOOL(name) PGFIX(YES|NO)
  - Requires the BP to go through reallocation before it becomes operative
    - Means a DB2 restart for BP0
  - Up to 8% reduction in overall IRWW transaction CPU time

1MB Page Frames

- Large (1 MB) page frames, requires z10 or z196, PGFIX(YES)
  - **Save 1-4% CPU**
- Specify LFAREA in IEASYSxx
  - z/OS sets aside this amount of real storage for 1MB frames
  - IPL required to modify LFAREA
- For PGFIX(YES) buffer pools
  - DB2 tries to allocate from 1MB frames.
  - If there is not enough 1MB frames, then DB2 will use 4K frames

#SHAREorg

PGSTEAL

- LRU – Least recently used
- FIFO – First in first out
  - Avoids CPU cost of maintaining LRU chain
  - DB2 9 automatically uses MRU for Copy utility anyway
- DB2 10 introduces PGSTEAL(NONE)
  - "In-memory" buffer pool
  - Similar to setting VPSEQT(0) in that it turns off prefetch
  - Also avoids LRU management cost, like PGSTEAL(FIFO)
    - But does not disable query parallelism.
  - DB2 will sequentially load data sets when they are opened
  - Enables the buffer pool to be loaded more quickly after a DB2 restart, without doing any random synchronous I/Os
  - If the buffer pool is too small, performance will be unpredictable
  - Consider setting VDWQT and DWQT very high to minimize write I/O

Indirect references

- If a variable length row needs to be updated and it can no longer fit on its original page, and indirect reference is created
  - The indexes continue to point to the original location, which then indirectly points to the new page
- Indirect references always require a synchronous read, even when doing a table scan
- If a row is updated again, and if the row shrinks, the indirect reference may go away, but otherwise the only way to clean them up is Reorg
- Use MAXROWS if you can to prevent inserts from consuming all space on each page
- Once created the only way to remove indirect references is REORG

Free space considerations

- PCTFREE is the percentage of space on each page that the DB2 utilities (LOAD and REORG) reserve
- Why reserve free space?
  - For table spaces, to preserve clustering.
  - In some cases, to avoid indirect references
    - You can set MAXROWS to avoid indirect references
  - For indexes, to avoid index splits, but usually that's not a good idea
- Why is reserving free space bad?
  - Use more DASD space
  - Worse buffer hit ratio
  - Elongated scans and utilities

#SHAREorg

SHARE
Technology · Connections · Results

Questions?

Email : bergerja@us.ibm.com