# What in the world is a page! An introduction to the life of a DB2 page.

**Quentin Presley**
*IBM*

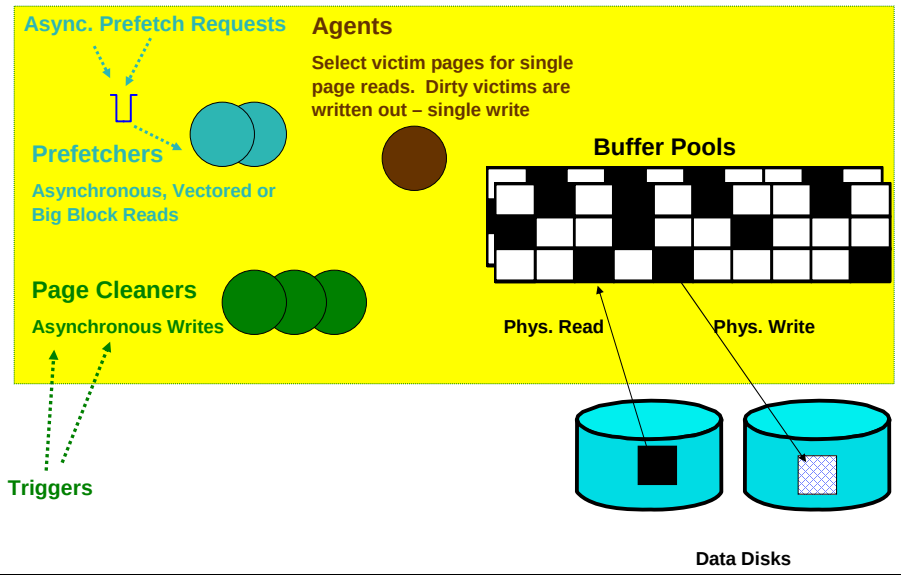
Session Code: D9
Wednesday, Oct 16, 2013 (08:30 AM - 09:30 AM)

INTERNATIONAL DB2 USERS GROUP
Independent • Not-for-Profit • User Run
Founded 1988

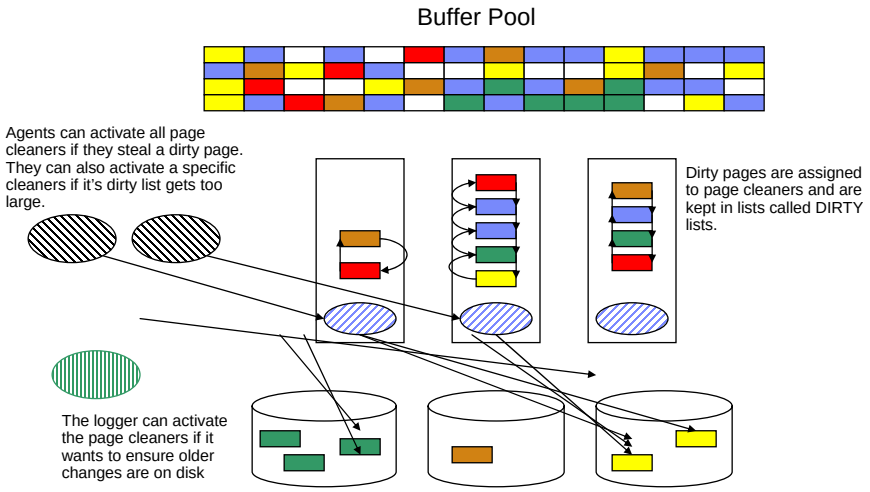
## Agenda

- Buffer and I/O Management
- Sizing Your Buffer Pools
- Page Cleaning
- Prefetching
- Block Based Buffer Pools

## How A Page Flows Through The Buffer Pool



## Page Cleaning



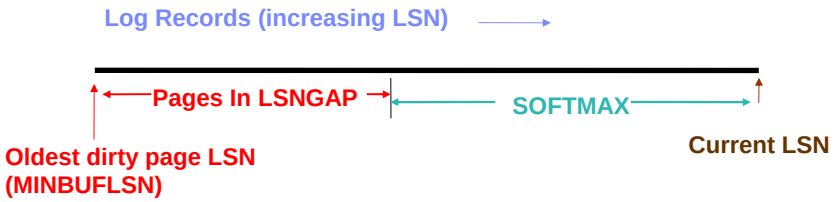

Dirty lists are doubly linked, and are ordered by oldest update to the page.

## Page Cleaning

- NUM_IOCLEANERS is used to configure the number of Page Cleaners:
  - Rule of thumb:
    - AIX, Windows, Solaris – 1 for every 25 to 30 spindles
      - Max out at number of processors
    - Other platforms - ~ 1 per spindle
- There are two types of dirty pages that are written by Page Cleaners:
  - 'Old' pages
  - 'Cold' pages
- DB2 Page Cleaners respond to 'triggers'
  - LSNGAP
  - Threshold
  - Dirty Steal

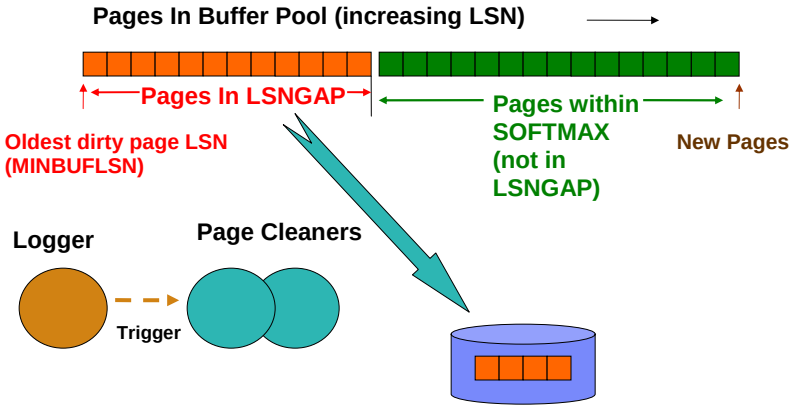
## Page Cleaning – LSNGAP Trigger

- **SOFTMAX defines which pages are 'old':**
  - High SOFTMAX = Less page cleaning and longer recovery time
  - Low SOFTMAX = More page cleaning and shorter recovery



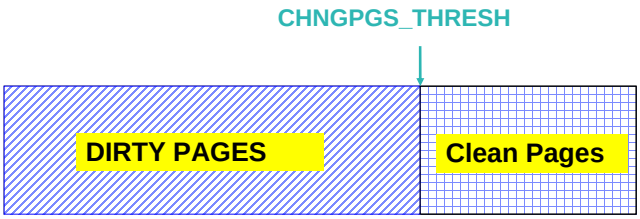
## Page Cleaning – LSNGAP Trigger

- **SOFTMAX defines which pages are 'old':**



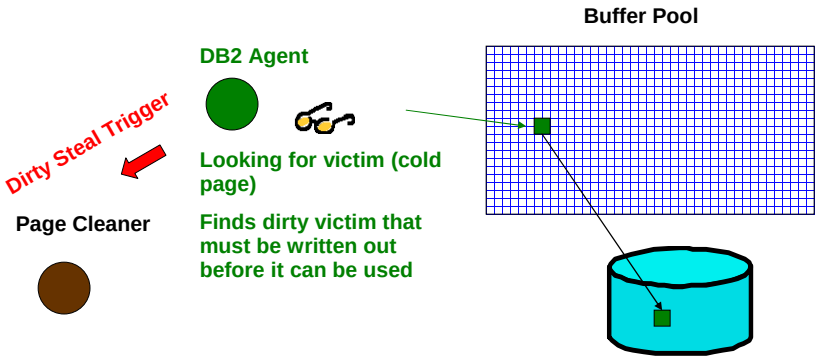
## Page Cleaning – Threshold Trigger

- Trigger is based off CHNGPGS_THRESH database configuration parameter.
  - CHNGPGS_THRESH value specifies what % of Buffer Pool should be dirty before Page Cleaning is triggered. Default is 60%.
  - Page Cleaners write out 'cold' pages when triggered for Threshold



## Page Cleaning – Dirty Steal Trigger

- Dirty Steal triggers are what you want to PREVENT by configuring NUM_IOCLEANERS and CHNGPGS_THRESH properly.



## Page Cleaning – Dirty Steal Trigger

- When servicing this trigger, Page Cleaners write out 'cold' pages
- Page "temperature" is determined when the page is released by a transaction using a weighting scheme

## Page Cleaning – Best Practices

- Start with NUM_IOCLEANERS set to AUTOMATIC, leave CHNGPGS_THRESH set to the default value and tune from there.
- Asynchronous I/O is best. Watch your synchronous I/O particularly writes as they can usually be prevented through better configuration of page cleaning (CHNGPGS_THRESH and NUM_IOCLEANERS)
  - pool_data_writes
  - pool_index_writes
  - pool_async_data_writes
  - pool_async_index_writes
- Monitor your triggers, particularly Dirty Steal, and consider making Page Cleaning more aggressive through better configuration of CHNGPGS_THRESH and / or NUM_IOCLEANERS
  - pool_drty_pg_steal_clns

## Alternate Page Cleaning (APC)

- db2set DB2_USE_ALTERNATE_PAGE_CLEANING = ON
- APC is a proven alternative on heavy pure OLTP workloads (e.g. TPC-C)
  - APC does not clean from block based buffer pools at all.
  - APC has been shown to work well in many mixed workloads.

Traditional Page Cleaning

- Threshold trigger
- Dirty steal trigger
- LSNGAP trigger

Alternate Page Cleaning

- No external triggers exist

## Alternate Page Cleaning (APC)

- CHNGPGS_THRESH configuration parameter is ignored
- SOFTMAX configuration parameter is still respected.

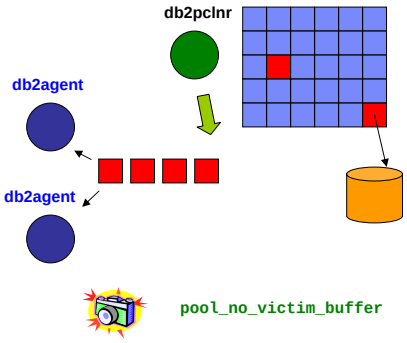
### How does it work ?

#### Victim selection

- The page cleaner pre-selects victim pages for agents.
- If this list of pages falls below 1% of the buffer pool, agents internally trigger cleaners. Cleaners populate, this list up to 2% of the buffer pool.

#### SOFTMAX (LSNGAP) cleaning

- Instead of waiting until a lsn gap interval occurs, page cleaners use a predictive algorithm to prevent a lsn gap from occurring.
- The result is a much more even write I/O throughput.

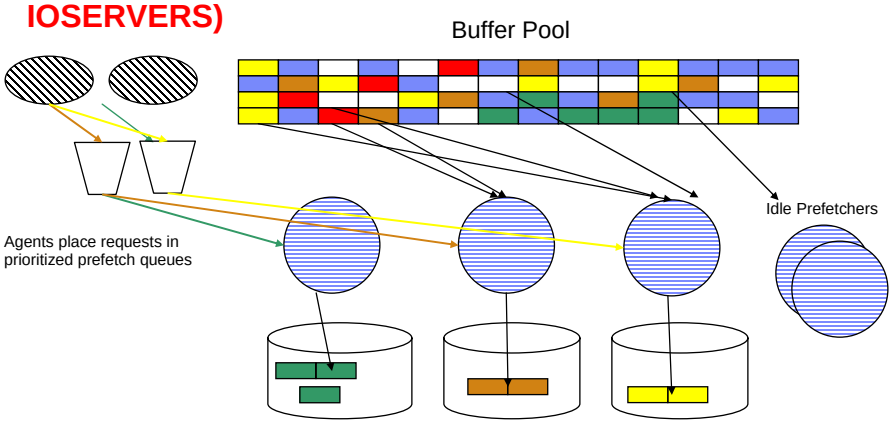


## Rules of thumb when considering APC

- Pure OLTP workloads can benefit from APC.
- For mixed and DSS workloads stick with traditional page cleaning.
- Page clean more aggressively with APC than with traditional page cleaning.
  - 32 I/Os outstanding with traditional
  - 256 I/Os outstanding with APC
- You may be able to get away with less page cleaners when using APC. Normally there is no benefit from reducing the number of page cleaners however.
- Don't consider APC if DB2 does not use platform asynchronous I/O.
  - consider it on AIX, Windows and Solaris
  - consider it on Linux if you have enabled AIO

## Prefetching

### I/O Prefetching done by DB2 by Prefetchers (aka. IOSERVERS)




Pages are read directly into buffer pool memory on most platforms. Due to limitations on some platforms I/O is done in big block reads to a private buffer within the prefetcher then copied into buffer pool memory

## Prefetching

- DB2 performs three kinds of Prefetching
  - RANGE
    - Sequential access either in the query plan or through sequential detection at run time.
  - LIST
    - Prefetches a list of pages that are not necessarily sequential (although they may be)
    - DB2 will convert the LIST request into one or more RANGE request if sequential ranges exist
  - LEAF
    - Used to Prefetch an Index leaf page and the Data pages pointed to by the leaf.
      - Leaf page is done as a single I/O.
      - Data pages on the leaf are submitted as a LIST request

## Prefetching – Best Practices

- Set NUM_IOSERVERS to the maximum amount of I/O parallelism you need
  - Start with a setting of AUTOMATIC which is generally good for most workloads. Tune from there if required.
  - For RANGE requests, each Prefetcher will read ONE Extent worth of page
  - To read N Extents in parallel, you need at least N Prefetchers
- Set PREFETCHSIZE appropriately
  - Start with a setting of AUTOMATIC and tune from there if required.
  - Good rule of thumb is Extent Size X Num. of Spindles or some multiple of this

## Prefetching – Best Practices

- The unread_prefetch_pages monitor element can indicate over configured prefetching.
  - A high value indicates that many pages that were prefetched on behalf of an application are being evicted out of the Buffer Pool BEFORE the application that needed them was able to use them.
  - Tune by either reducing the PREFETCHSIZE or check to ensure you don't have too many Prefetchers configured (NUM_IOSERVERS)

- The prefetch_wait_time monitor element advises of under configured Prefetching (applications waiting for prefetching to complete).
  - For high prefetch_wait_time, consider making Prefetching more aggressive through increased PREFETCHSIZE or check to ensure that you have enough Prefetchers configured (NUM_IOSERVERS)

## Prefetching

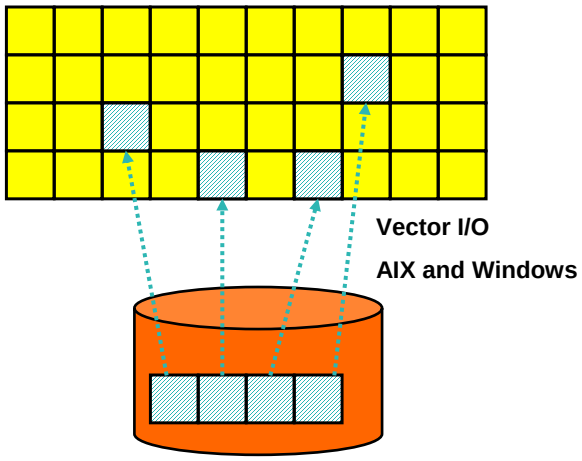
- **Example A:**
  - Two containers, each is a SCSI disk, Extent Size = 32
    - Good NUM_IOSERVERS = 2
    - Extent Size = 32
    - Good PREFETCHSIZE = 2 x 32 = 64
- **Example B:**
  - Three containers, each is a RAID 5 (6+1) device, Stripe Size = 64K, Page Size = 4k
    - Good NUM_IOSERVERS = 6 x 3 = 18
    - Good Extent Size = Stripe Size / Page Size = 64K/4K = 16
    - Good Prefetch Size = 3 x 6 x 16 = 288
    - DB2_PARALLEL_IO = ON

## How I/O Is Done During RANGE Prefetch

- A Prefetcher performing a RANGE prefetch will select the 'coldest' victim pages from the Buffer Pool, regardless of their location in the Buffer Pool (i.e. they may not be in sequential memory locations)
- 'Vector' or 'Scatter/Gather' I/O – an operating system primitive, supported well on some operating systems (AIX and Windows) allows Prefetchers to provide good I/O throughput regardless of the fact that sequential pages on disk are being read into a non sequential memory area

## Vector I/O

- Non Block Based Buffer Pool with Vector I/O

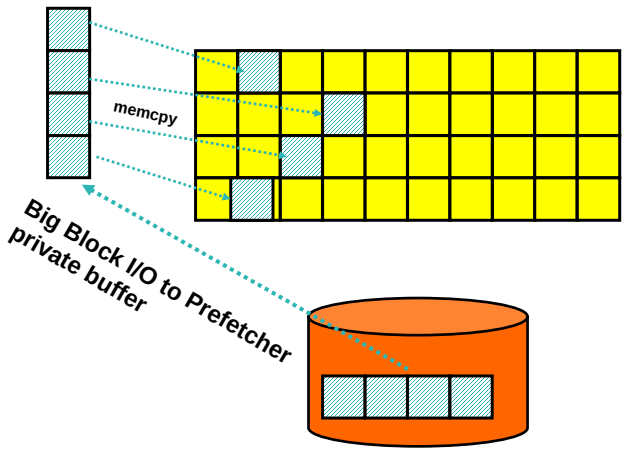


## There are some problems with Vector I/O

- Largest limitation of Vector I/O is that it is not supported efficiently everywhere (e.g. Solaris)
  - On these platforms DB2 does a large block read of the Extent into a private memory buffer and a memcpy into the Buffer Pool
  - This results in good I/O throughput (large block I/O) but unnecessary CPU cycles and CPU cache pollution
- On AIX there is a limit of 16 Pages that can be read on one Vector I/O call
  - Thus a Prefetcher reading an Extent of > 16 pages will have its Vector read request broken up – Inefficient.

## Inefficient Vector I/O - Solaris

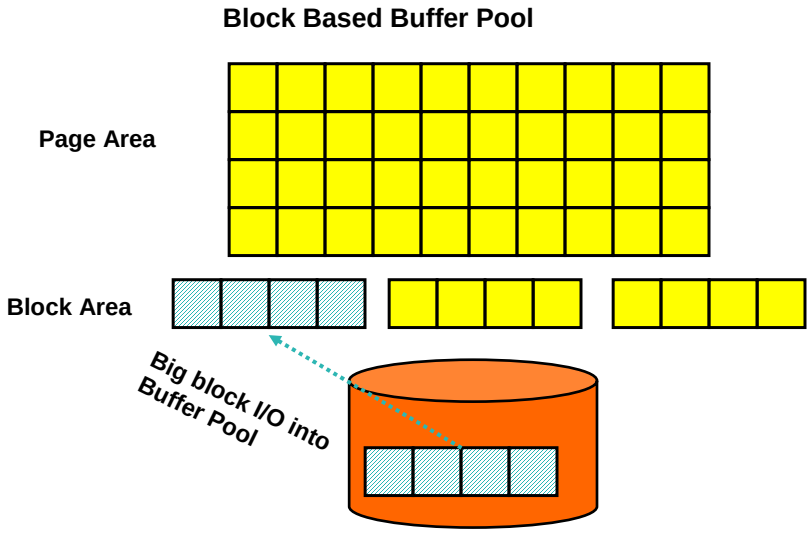
Non Block Based Buffer Pool with inefficient Vector I/O (Solaris)



## Block Based Buffer Pools

- A Block Based Buffer Pool has both a 'Page Area' and a 'Block Area' whereas traditional Buffer Pools only have a 'Page Area'
- The 'Block Area' is only visible to Prefetchers doing RANGE prefetch
- When looking for victim pages to prefetch into, Prefetchers are thus able to get a full Block of pages that are contiguous in memory
- Prefetchers can then do a big block read DIRECTLY into shared Buffer Pool memory

## Block Based Buffer Pools



## Block Based Buffer Pools

- Where can you expect the most benefit from Block Based Buffer Pools?
  - Platforms that do not support good Vector I/O (e.g. Solaris)
  - On AIX when Extent Size is much > 16 Page readv limit
- How big to make the Block Area?
  - Depends on the application. Typical sizes range from 10% to 50% of total Buffer Pool size.
- What are the disadvantages of Block Based Buffer Pools?
  - The Block Area is only visible to RANGE Prefetch. No RANGE Prefetch means it will reduce the effective size of your Buffer Pool.

## Buffer pool layout

- The buffer pool is made up of two blocks of memory
- Page memory
  - Where physical data is stored in memory
- Page descriptor memory
  - Where internal information about a particular page is stored
- Each page is linked 1 to 1 with a page descriptor

## Hate List

- At database activation time every page descriptor is added to a hate list (all pages at activation are hated)
  - List obviously includes unused pages
- Number of hate lists has an upper bound of 8
  - May be made smaller to balance out the size of each list
- Hate list size is a function of BP size and number of hate lists
  - Roughly (BPSize/numHateLists)

## Dirty List

- List of pointers to pages with updates that have not yet made it to disk
  - Ordered by LSN
  - Page Cleaners are assigned to work on a set of Dirty Lists across all existing buffer pools

## Clock Pointer

- Used to walk all page descriptors in a circular order
- Each visit starts where the previous visit ended
- Clock pointer only sees pages that are used. Unused pages would be on the hate lists

## Victim Selection

- Victim selection occurs when a requested page is not found in the buffer pool
- Start looking for victims on the hate lists
  - Each visit to the hate lists starts on the next hate list from where the previous visit left
  - If victim page is no in use, use it
  - If victim page is in use do further processing before choosing
    - Is it CLEAN/DIRTY

## Victim Selection (cont.)

- If a victim is not found on the hate list move to the clock pointer
  - Starts where previous clock scan stopped
  - For each page viewed
    - Check if Dirty
    - Check for Weight
  - After some number of pages viewed just choose one instead of continuing.

## Summary

- Scan this presentation for a list of configuration parameters highlighted and use the information to help you set them for best performance
- Use the database monitor elements to check the health of your system and adjust its configuration appropriately
- Start with AUTOMATIC settings and tune from there.

**Quentin Presley**

IBM

*qpresley@ca.ibm.com*

Session

What in the world is a page! An introduction to the life of a DB2 page.

INTERNATIONAL DB2 USERS GROUP
Independent • Not-for-Profit • User Run
Founded 1988