

Understanding Buffer Pool Performance and Tuning in DB2 UDB v8.2

Jeff Brokaw
Product Manager
Quest Central® for DB2

November 2, 2005

Agenda

- Buffer pool overview
- Buffer pool management
- Collecting performance metrics
- Evaluating performance metrics
 - Physical vs. logical IO
 - Asynchronous vs. synchronous IO
 - Page cleaner tuning
- Q&A

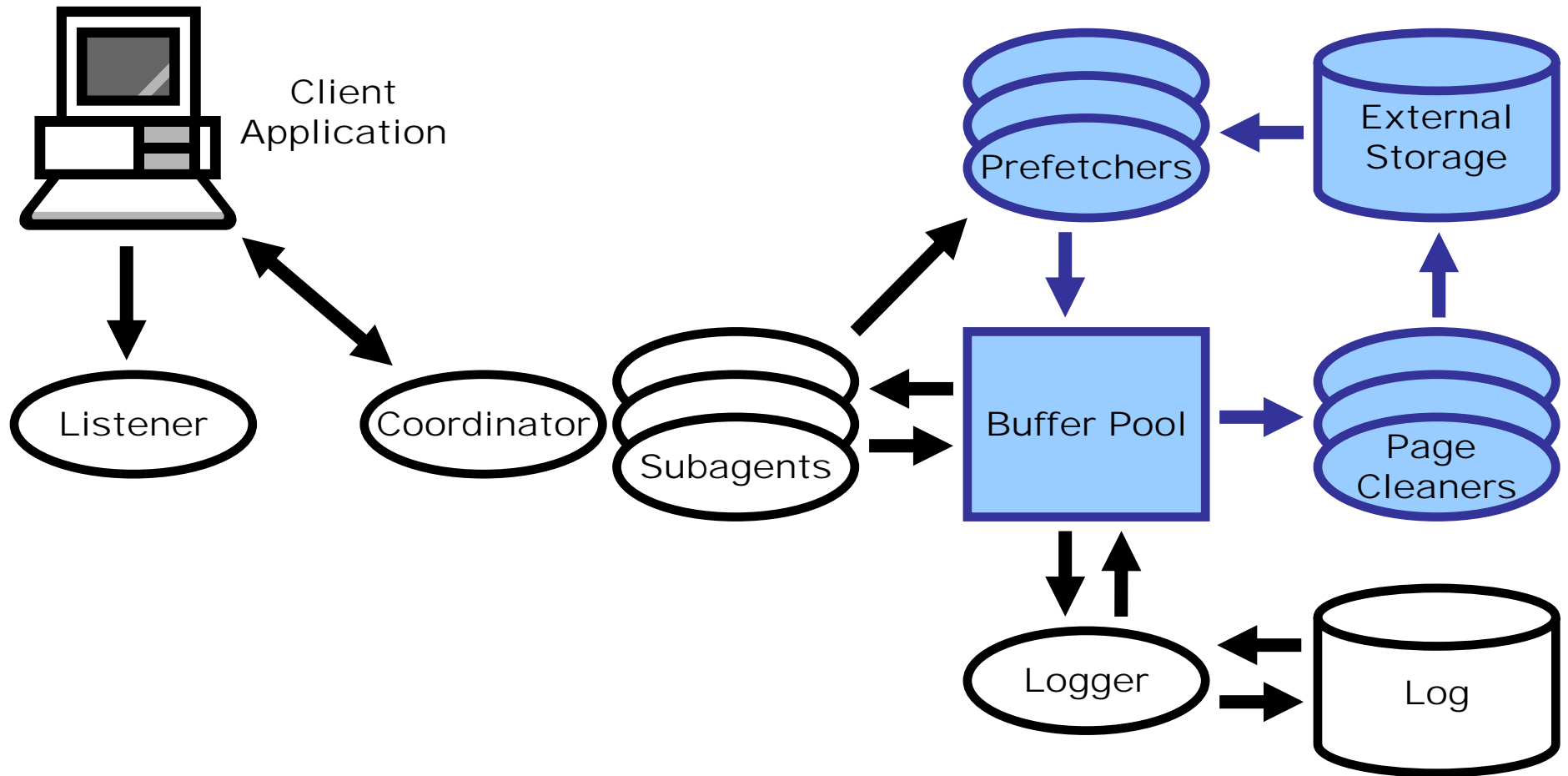
Introduction

Q: With so many tuning areas to focus on why should we pay special attention to the buffer pool?

A: DB2 uses the buffer pool to offset the performance disparity between CPU and disk, so they are vital for optimizing transaction throughput. From the 8.2 Admin Guide:
Performance (emphasis mine):

“Because most data manipulation takes place in buffer pools, configuring buffer pools is the single most important tuning area.”

You Are Here – DB2 Process Model



Buffer Pool Overview

Basics and Terminology

- Area of memory into which database pages are read, modified, and held during processing
 - A *hit* occurs when required page is found in buffer pool
 - A *miss* occurs when required page not found in buffer pool
- Pages in buffer pool exist in one of three states:
 1. In use
 2. Dirty
 3. Clean

Buffer Pool Overview

Basics and Terminology (cont'd)

- Pages read into buffer pool from disk by:
 - Agents, using synchronous I/O
 - Prefetchers, using asynchronous I/O
- Pages written to disk from buffer pool by:
 - Agents, using synchronous I/O
 - Page cleaners, using asynchronous I/O

Buffer Pool Overview

Tuning Goals

- Recovery vs. Performance
 - Recovery wants everything on disk
 - Database manager wants everything in memory
 - Which goal is more important – largely depends on data
 - Which combination works best for both goals
- High hit ratio means logical reads usually successful, avoiding disk access
- Normally want to maximize asynchronous I/O to minimize waits

Buffer Pool Overview

Prefetchers and Page Cleaners

- DB2 uses prefetch and page cleaner processes to reduce the time that agents have to wait for IO operations to complete
- New in 8.1.4: Proactive Page Cleaning, AKA Alternate Page Cleaning
 - `DB2_USE_ALTERNATE_PAGE_CLEANING=ON`

Buffer Pool Overview

Prefetch

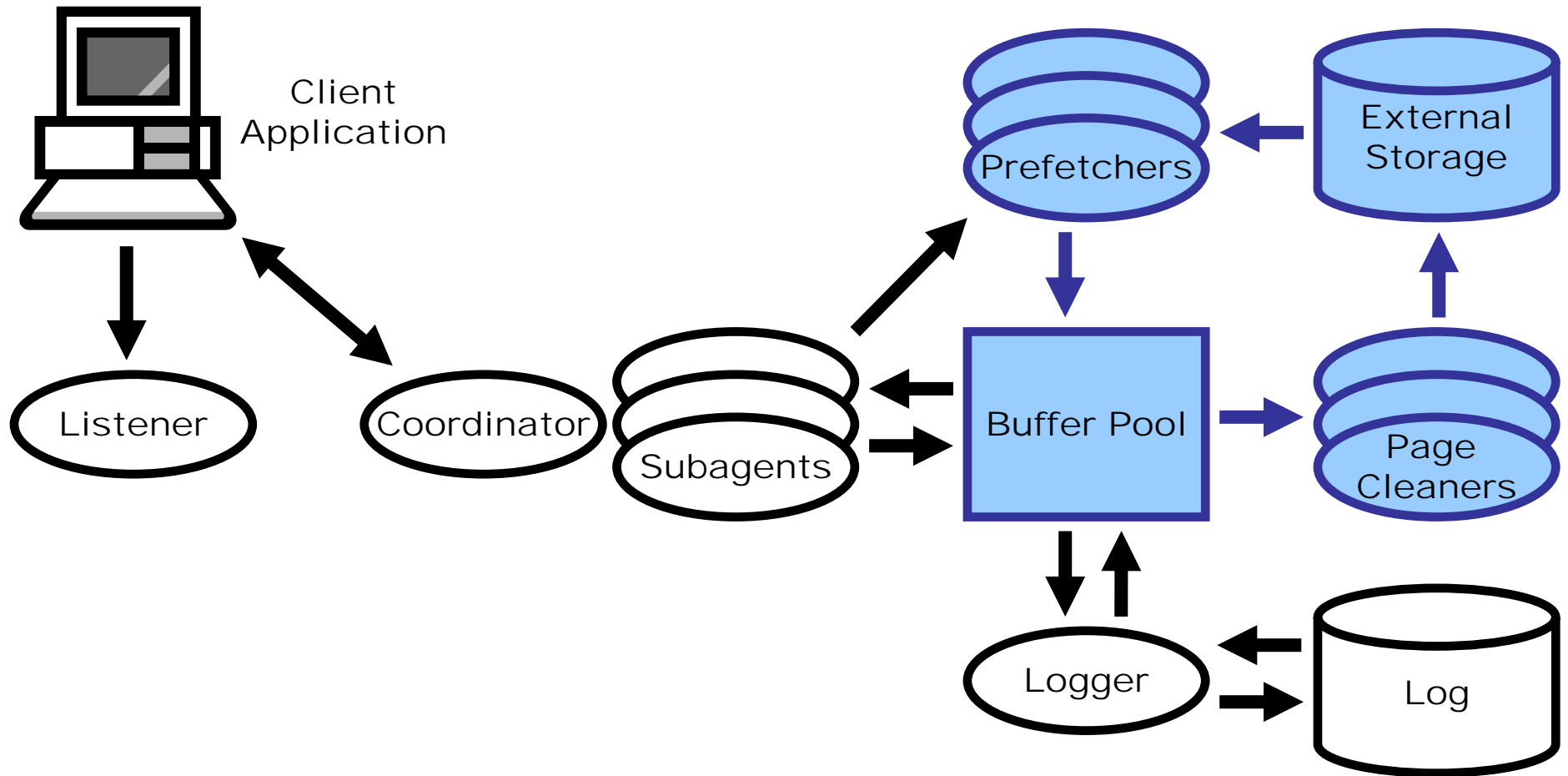
- Prefetchers try to ensure that agents doing tablespace scans never wait for disk I/O
- Agents send asynchronous read-ahead request to common prefetch queue
- Prefetchers service the request by bringing the requested pages into the buffer pool
- Should result in logical read, drive hit rate up
- Two types:
 - List prefetch, efficiently reads a set of non-consecutive data pages
 - Sequential prefetch, reads consecutive pages

Buffer Pool Overview

Page Cleaners

- Two main goals
 - Ensure that agents reading a page into the buffer pool never need to first flush a dirty page to disk to free up a slot
 - Speed recovery time by reducing # log files needed
- Configuration parameters
 - num_iocleaners – number of page cleaners for a database
 - chngpgs_thresh – percentage of changed pages at which the page cleaners will be started, if not currently active
- Alternate page cleaning (8.1.4 and later)
 - List of “good victim pages” (dirty pages just written out) kept, prevents searching in many cases
 - LSN gaps anticipated, prevents I/O spikes
 - chngpgs_thresh is ignored

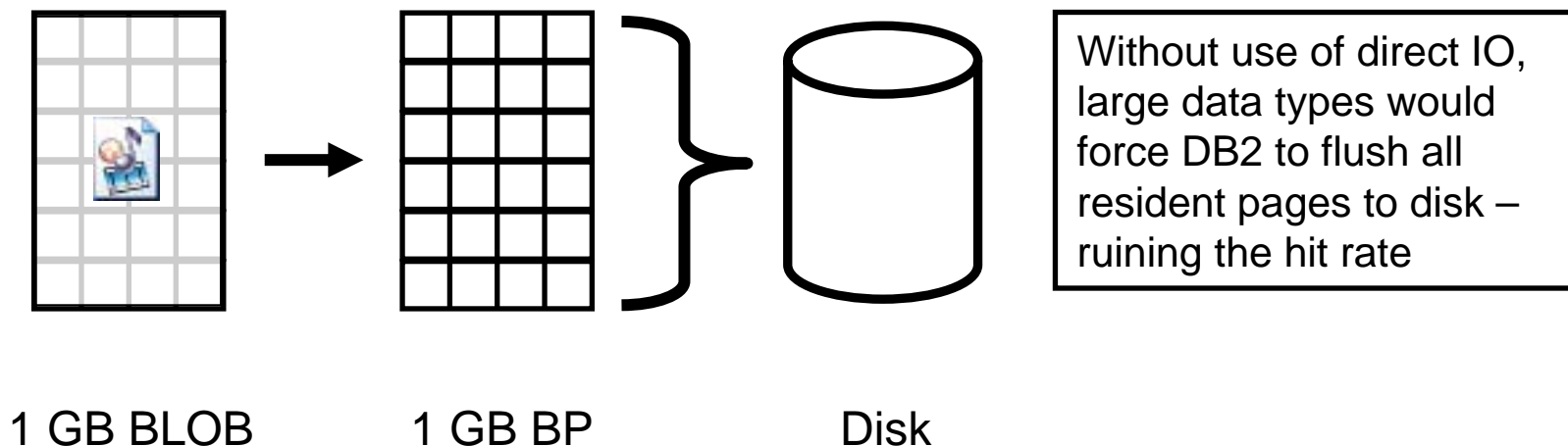
You Are Here – DB2 Process Model



Buffer Pool Management

Buffer pools not used with all data types

- LONG VARCHAR, BLOB and CLOB retrieved directly from disk using non-buffered (direct) IO
- Keeping large object pages out of buffer pools helps DB2 maintain effective caching for regular data types



Buffer Pool Management

Creating/Altering Buffer Pools

- IBMDEFAULTBP automatically created with each database
- Additional buffer pools added with the CREATE BUFFERPOOL statement
- SYSCTRL or SYSADM privileges are required to work with buffer pools

```
CREATE BUFFERPOOL bufferpool_name SIZE number of pages
[IMMEDIATE | DEFERRED]
[PAGESIZE integer [K]]
[[NOT] EXTENDED STORAGE
| NUMBLOCKPAGES number of pages [BLOCKSIZE number of pages]]
[ALL DBPARTITIONNUMS
| DATABASE PARTITION GROUP db_partition_group_name [,...]]
[EXCEPT ON DBPARTITIONNUM[S] (db_partition_number1 [TO db_partition_number2]
SIZE number of pages [,...])]
```

Buffer Pool Management

Memory Usage

- Version 8 makes DB2 easier to manage by using database global memory (DATABASE_MEMORY) instead of database heap (DBHEAP)
- Prevents resizing of DBHEAP with bufferpool resizing

```
CREATE BUFFERPOOL bufferpool_name SIZE number of pages
[IMMEDIATE | DEFERRED]
[PAGESIZE integer [K]]
[[NOT] EXTENDED STORAGE
| NUMBLOCKPAGES number of pages [BLOCKSIZE number of pages]]
[ALL DBPARTITIONNUMS
| DATABASE PARTITION GROUP db_partition_group_name [,...]]
[EXCEPT ON DBPARTITIONNUM[S] (db_partition_number1 [TO db_partition_number2]
SIZE number of pages [,...])]
```

Buffer Pool Management

Extended Storage

- 32-bit DB2 can't support creating buffer pools larger than 4G due to addressable storage limits
- Extended storage allows DB2 to flush dirty pages to memory rather than disk, so future access to the pages is much faster
- Requires NUM_ESTORE_SEGS and ESTORE_SEG_SIZE configuration parameters

```

CREATE BUFFERPOOL bufferpool_name SIZE number of pages
[IMMEDIATE | DEFERRED]
[PAGESIZE integer [K]]
[[NOT] EXTENDED STORAGE
| NUMBLOCKPAGES number of pages [BLOCKSIZE number of pages]]
[ALL DBPARTITIONNUMS
| DATABASE PARTITION GROUP db_partition_group_name [,...]]
[EXCEPT ON DBPARTITIONNUM[S] (db_partition_number1 [TO db_partition_number2]
SIZE number of pages [,...])]
    
```

Buffer Pool Management

Block-Based Buffer Pools

- Contiguous blocks of pages to be moved into contiguous portions of memory
- Block-based buffer pools will be wasted if your application doesn't perform sequential prefetch

```

CREATE BUFFERPOOL bufferpool_name SIZE number of pages
[IMMEDIATE | DEFERRED]
[PAGESIZE integer [K]]
[[NOT] EXTENDED STORAGE
| NUMBLOCKPAGES number of pages [BLOCKSIZE number of pages]]
[ALL DBPARTITIONNUMS
| DATABASE PARTITION GROUP db_partition_group_name [,...]]
[EXCEPT ON DBPARTITIONNUM[S] (db_partition_number1 [TO db_partition_number2]
SIZE number of pages [,...])]
    
```

Buffer Pool Management

Hidden Buffer Pools

- For improved availability only
- DB2 creates small (16 page) buffer pools of each page size at database activation (4, 8, 16, 32k)
- Hidden from the user, and used only when ordinary buffer pools cannot be allocated
- Performance will be noticeably impacted, and DB2 will write a warning message to the administration notification log

Buffer Pool Management

Related Objects

- CREATE TABLESPACE specifies which buffer pool to use
- Page sizes of buffer pool and tablespace must match – valid values are 4, 8, 16, and 32 kilobytes
- CREATE BUFFERPOOL statement can specify partition group, to control partitions in which a given buffer pool is created

Collecting Performance Metrics

Tuning – An Iterative Process

- Collect performance data
 - Snapshot monitoring
 - Event monitoring
- Review the results
 - Understand SQL workload
 - Evaluate performance (calculations)
- Take action
 - Resize buffer pools
 - Define more page cleaners
 - Break out tables with “like” workloads
- Start the process again

Collecting Performance Metrics

Snapshot monitor	Event Monitor
“Point in time” representation of data	“Event” based representation of data
Less overhead (~5%)	More overhead (~10-20%)
Need to reissue over time	“Create” it once and it runs until it’s told to stop
Monitor Switches need to be turned on to collect data	“Create” takes care of everything
“Real time” balance of application and database performance	“Historical” main focus on Application statistics

Collecting Performance Metrics

DB2 Snapshot Switches

- Seven Switches
 - Buffer Pool (dft_mon_bufpool)
 - Lock (dft_mon_lock)
 - Sort (dft_mon_sort)
 - Statement (dft_mon_stmt)
 - Unit of Work (dft_mon_uow)
 - Table (dft_mon_table)
 - Timestamp (dft_mon_timestamp)

Collecting Performance Metrics

DB2 Snapshot Switches (cont'd)

- Turning switches on at the instance – take care here

```

UPDATE DATABASE MANAGER CONFIGURATION USING
    dft_mon_bufpool ON
    dft_mon_lock ON
    dft_mon_sort ON
    dft_mon_stmt ON
    dft_mon_table ON
    dft_mon_uow ON
IMMEDIATE
    
```

- Turning switches on for session

```

UPDATE MONITOR SWITCHES USING bufferpool ON
    
```

Collecting Performance Metrics

- Issuing a SNAPSHOT request
 - Instance level (requires “ATTACH”)
 - For all active databases
 - | `GET SNAPSHOT FOR ALL BUFFERPOOLS`
 - For a specific database
 - | `GET SNAPSHOT FOR BUFFERPOOLS ON database_name`
 - For a specific database partition
 - | `GET SNAPSHOT FOR BUFFERPOOLS ON database_name`
 `AT DBPARTITIONNUM db_partition_number`
- BUFFERPOOL and DATABASE snapshot calls are essential for tuning buffer pools

Collecting Performance Metrics

- BUFFERPOOL Snapshot

```

Bufferpool name           = BASEBP1
Database name             = BASEBALL
Database path             =
C:\DB2\NODE0000\SQL00005\
Input database alias      = BASEBALL
Buffer pool data logical reads = 300
Buffer pool data physical reads = 3490
Buffer pool data writes   = 3050
Buffer pool index logical reads = 67
Buffer pool index physical reads = 34
.
.
.
Node number               = 0
Tablespaces using bufferpool = 4
Alter bufferpool information:
  Pages left to remove    = 0
  Current size             = 1000
  Post-alter size         = 1000
    
```

Collecting Performance Metrics

- DATABASE Snapshot

```

Database name                = AUTO_PRD
Database path                 = C:\DB2\NODE0000\SQL00009\
.
.
.
Buffer pool data logical reads    = 165079
Buffer pool data physical reads   = 36264
Asynchronous pool data page reads = 10546
Buffer pool data writes          = 3991
Asynchronous pool data page writes = 3873
Buffer pool index logical reads   = 4014
Buffer pool index physical reads  = 558
Asynchronous pool index page reads = 332
Buffer pool index writes         = 26
Asynchronous pool index page writes = 26
Total buffer pool read time (ms)  = 8329
Total buffer pool write time (ms) = 121852
Total elapsed asynchronous read time = 5353
Total elapsed asynchronous write time = 121292
Asynchronous read requests       = 1681
LSN Gap cleaner triggers         = 24
Dirty page steal cleaner triggers = 63
Dirty page threshold cleaner triggers = 33
Time waited for prefetch (ms)    = 626
    
```

Evaluating Performance Metrics

Know your workload

- Is the SQL workload primarily:
 - Update/Insert/Delete (OLTP)
 - Select (ad hoc, data warehouse)
- Event Monitoring can help if unsure

Evaluating Performance Metrics

Logical IO vs. physical IO – minimize number of times DB2 must access physical disks

Key I/O Metrics:

- Overall hit rate
- Data hit rate
- Index hit rate
- Physical IO read rate

Evaluating Performance Metrics

Overall Hit Rate

- Includes all data and index reads into buffer pool

$$BufferpoolOHR = \left[\frac{(bp.data.logical.reads + bp.index.logical.reads)}{(bp.index.logical.reads + bp.index.physical.reads + bp.data.logical.reads + bp.data.physical.reads)} \right] * 100$$

- Low values = more physical IO
- High values = more logical IO
- 85 – 90% is a good starting point

Evaluating Performance Metrics

Data Hit Rate

- Includes all data reads into buffer pool

$$\text{Bufferpool DHR} = \left(\frac{bp.data.logical.reads}{bp.data.logical.reads + bp.data.physical.reads} \right) * 100$$

- Low values = more physical IO
- High values = more logical IO
- 85 – 90% is a good starting point

Evaluating Performance Metrics

Index Hit Rate

- Includes all index reads by buffer pool

$$\text{Bufferpool IHR} = \left(\frac{bp.index.logical.reads}{bp.index.logical.reads + bp.index.physical.reads} \right) * 100$$

- Low values = more physical IO
- High values = more logical IO
- 85 – 90% is a good starting point

Evaluating Performance Metrics

Physical IO Read Rate

- Total number of physical reads by buffer pool over time

$$PhysicalIOReads = \left(\frac{bufferpool.data.logical.reads + bufferpool.data.physical.reads}{total.collection.time} \right)$$

- Use:
 - | `GET SNAPSHOT FOR TABLESPACES ON database_name`
- Could indicate the need for...
 - Additional containers
 - Buffer pool breakout

Evaluating Performance Metrics

Prefetch – Asynchronous vs. Synchronous I/O

- Waiting is always bad
- Make DB2 read ahead for you as much as possible, given application mix

Approach:

- Evaluate prefetch ratio
- Know the workload
- Evaluate prefetcher configuration
- Use block I/O

Evaluating Performance Metrics

Prefetch Ratio

$$PrefetchRatio = \left(\frac{async.pool.data.reads + async.pool.index.reads}{bp.logical.reads + bp.index.logical.reads} \right) * 100$$

- Lower values = more synchronous
- Higher values = more asynchronous
- Are there enough defined?

Evaluating Performance Metrics

Block-based Buffer Pools

- If using sequential prefetch, enable buffer pools to utilize block IO

```
ALTER BUFFERPOOL basebp2 NUMBLOCKPAGES 30;
```

```
ALTER BUFFERPOOL basebp2 BLOCKSIZE 10;
```

- NUMBLOCKPAGES should be a multiple of block size

Evaluating Performance Metrics

Evaluating Page Cleaner Performance:

- LSN Gap Triggers
- Dirty Page Threshold Triggers
- Dirty Page Steal Triggers

Approach:

- Evaluate workload and recovery goals
- Evaluate each metric against all others
- Modify parameters influencing page cleaner activity
- Enough page cleaners defined?
- More automated in V8.1.4 and above

Evaluating Performance Metrics

Log Cleans Ratio

$$LCR = \left(\frac{lsn.gap.cleaner.triggers}{\left(dirt.page.steal.cleaner.triggers + lsn.gap.cleaner.triggers \right) + dirty.page.threshold.cleaners.triggers} \right) * 100$$

- Has an affect on recovery times
- Too small or too large could be detrimental to buffer pool performance
- Affected by SOFTMAX DB CONFIG parameter
- 0 if ALTERNATE_PAGE_CLEANING=ON

Evaluating Performance Metrics

Threshold Cleans Ratio

$$TCR = \left(\frac{\text{dirty.page.threshold.cleaner.triggers}}{\left(\text{dirt.page.steal.cleaner.triggers} + \text{lsn.gap.cleaner.triggers} \right) + \text{dirty.page.threshold.cleaner.triggers}} \right) * 100$$

- CHNGPGS_THRESH DB CONFIG parameter
- Finding a “happy medium” is key
- 0 if ALTERNATE_PAGE_CLEANING=ON

Evaluating Performance Metrics

Victim Cleans Ratio

$$VCR = \left(\frac{\text{dirty.page.steal.cleaner.triggers}}{\left(\text{dirt.page.steal.cleaner.triggers} + \text{lsn.gap.cleaner.triggers} \right) + \text{dirty.page.threshold.cleaner.triggers}} \right) * 100$$

- Above 40%, typically means buffer pool needs to be larger
- Could also mean SOFTMAX parameter is too high (not enough page turnover)
- 0 if ALTERNATE_PAGE_CLEANING=ON

Conclusion

- **Focus on three main areas**
 1. BP hit ratios (logical vs. physical)
 2. Prefetcher tuning (asynchronous vs. synchronous)
 3. Page cleaner tuning (if not using newer alternate cleaning)
- **Collect performance data over time**
 - Sampling methodology – snapshots every 5-15 minutes over the course of the day, or during a batch run
- **Review the results**
 - Decipher data and plug into equations
 - Know the workload
 - Chart performance over time
- **Take action**
 - Various solutions depending on findings

Quest Software: You Can Expect More



<http://www.quest.com/db2/>