Phil Grainger, Lead Product Manager

BMC Software

March, 2016

—

An A-Z of *System Performance* for DB2 for z/OS

The Challenge

- Simplistically, DB2 will be doing one (and only one) of the following at any one time:
  - Doing useful work
  - Waiting for something
    - I/O
    - CPU
    - Locked DB2 Object
    - Latch, drain, dataset operation etc.

So tuning is simple

- Maximise the time and effort involved in doing useful work
- Minimise the time spent waiting for something to happen

- Simple!

So tuning is simple

- Let’s look at an A-Z of areas to consider when determining where to look for potential improvements
- And where to look for possible problems

- I’ll finish off with some thoughts about a “performance database”

Access Path stability

- Introduced with PK 52523
  - AKA Plan Stability
    - But it's not for plans!

- Allows rebinding to take place whilst retaining old access paths for regression
- Removes the major fear of rebinding
  "If I rebind and get an access path I don't like, there is no way back to where I was"
- Rebind now has "fallback"

Access Path stability

- REBIND PACKAGE ….
  PLANMGMT(OFF, BASIC or EXTENDED)
  - Default defined in DSNZPARM
  - Note, not ALL package options can be changed during this type of rebind
  - But great for a REBIND to get a new access path
    - In case you might want to fall back
  - Also applies to trigger packages

Bufferpools - A Strategy

- Separating of production DB2 objects across bufferpools can offer several advantages:
  - Object/Bufferpool separation delivers isolation
  - DB2 performance can improve as objects are tuned based on processing needs
- Such as
  - BP0 - Catalog & Directory
  - BP1 - Indexes
  - BP5 - Tablespaces
  - BP7 - DSNDB07(Workfile Database)
- Expand by separating like-processed objects
  - Random ones
  - Sequential ones
  - Consider refining default bufferpool configurations

Bufferpools - Read Processing

- Hit Ratio/Hit Density for random access should be watched
- Watch Workload Changes
- Particularly:
  - DM Critical Threshold Reached
  - Prefetch Disabled - No Buffer

Bufferpools - Write Processing

- VDWT-Vertical Deferred Write Threshold(0 % - 90 %) is for a single dataset Sum of IN-USE and Updated pages
  - Default 10 %, consider Zero or One
  - Can also be a number of pages
    - Useful for very large bufferpools
- Pages Written divided by Asynchronous Writes
  - Max of 32 pages per Write
  - If Low, lower Write Thresholds
- Most frequent writes occur at DB2 checkpoints
- PGSTEAL Option - LRU or FIFO
  - FIFO can reduce Chain Latch Maintenance significantly
  - But can affect re-reading of pages

Clustering

- Clustering (and clusterratio) describes whether the data is held in the same sequence as an index
  - Usually the CLUSTERING index
- Ensures that any sequential access by this key will be efficient

Clustering

- BUT if the data is NEVER accessed by this clustering key
- Why bother keeping the data clustered?
  - And why reorg “because clusterratio is low”?

- DB2 10 provides RTS REORGCLUSTERSENS
  - How often a table is accessed IN THE CLUSTERING KEY SEQUENCE since a reorg
  - If low (or zero) – why bother!
  - Or did you choose the wrong clustering key completely?

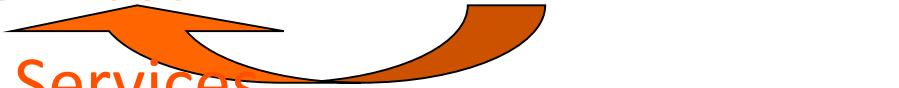
Compression

- Table space compression CAN improve performance
  - Data is compressed on DASD and in the bufferpools
  - Only expanded when it is passed back to the application
  - So more data is held in the same bufferpool space
- Index compression ONLY saves DASD space
  - A compressed 4K index page will be expanded to 8K, 16K or 32K in the bufferpool

- Also note that table space compression can be immediate in DB2 10
  - DB2 creates a compression dictionary dynamically

Dispatching Priority

- Nothing has Changed
  - IRLM
  - DB2 System Services
  - DB2 Database Services
  - DB2 DDF



Dynamic Statement Cache - Sizing

- DSNZPARM parameters now allow independent sizing of certain EDMPOOL items:
- EDMPOOL
  - Size of EDMPOOL itself
  - NOT including the following two areas
- EDMSTMTC
  - How space for dynamic statement caching
  - Default = via EDMPOOL definition
- EDMDBDC
  - How much space for DBD caching
  - Default = via EDMPOOL definition

EDM Pool

- Available Pages are allocated first for new pages
  - Inactive Pages are stolen when needed, If no room, Error returned
- FREE PAGES IN FREE CHAIN vs. PAGES IN EDM POOL (too Big, too Small)
  - Increased Response time & I/O for loading of SKCT, SKPT, and DBD
  - Preparation on Caching Dynamic SQL
- Hit Ratio for REQUESTS NOT in EDMPOOL
  - Cursor and Package Sections - 85 % to 95 %

Explain

- Don’t forget to always Explain everything
  - A new table, DSN_STATEMNT_TABLE, can also be populated when Explain is run
- There is also the DSN_STATEMENT_CACHE_TABLE
  - Which is populated by EXPLAIN STMTCACHE ALL

- In production, why would you ever bind EXPLAIN(NO)??
  - This is just asking for trouble!

Locking

- Row vs Page
- MAXROWS
- PAGESIZE
- LOCK SIZE
- CURRENTDATA YES/NO
- ISOLATION (UR)

Logging

- Active Log Dataset write time
  - DASD Speed - As low as 2 ms
  - As always, watch I/O Contention
- UNAVAILABLE OUTPUT LOG BUFFERS
  - Hopefully Zero
  - Size of OUTPUT LOG BUFFER
    - Benefit Recover and Restart
- COMMIT Logic can have impact on Performance(Rollback)
- Fast Log Apply for Recover and Restart
  - LOGAPSTG
  - Suggest minimum of 100M
- LOG NO is NOT for performance

Page Size

- 4K, 8K, 16K, 32K
  - More Choices
- When Row Length is greater than 4K
  - What is the best page size ?
  - Example: 5K row
  - If Processed Sequentially - 32K would be better - 6 objects in 32K
  - If Processed Randomly - 16K may be better
- Also, be careful of table-based partitioning
  - A bigger pagesize can mean more partitions
- And INLINE LOBs?

Page Size and Partitioning

DSSIZE	4K pages	8K pages	16K pages	32K pages
1-4 Gb	4,096	4,096	4,096	4,096
8Gb	2,048	4,096	4,096	4,096
16Gb	1,024	2,048	4,096	4,096
32Gb	512	1,024	2,048	4,096
64Gb	256	512	1,024	2,048

- The problem lies in the allowable size of a page number

Rebinding

- It is now possible to separate rebinding from getting a new access path
- Sometimes IBM recommend rebinding for “other reasons”
- `APREUSE` (`none/error/warn`) can be used to retain existing access paths on rebinds

SORT

- Sort record is the sort key + all selected columns
  - Limit sort records and size

SELECT C1, C2, C3, C4, C5
FROM T1 WHERE….
ORDER BY C2, C4

SORT RECORD - C2, C4 *PLUS* C1, C2, C3, C4, C5

- Watch these statistics
  - MERGE PASSES DEGRADED
    - Should be zero
  - WORKFILE REQUESTS REJECTED
    - Should be zero
  - WORKFILE PREFETCH NOT SCHEDULED
    - Should be zero
  - Prefetch Quantity less than 4 or 5

Sort Work Database
aka “Workfile Database” or “dsndb07”

- DSNDB07
  - or Named database for Data Sharing
- Suggested minimum of 5 or 1/5 Max # of Partitions
- Placement of Datasets
  - Across Channels and Control Units
- Own Bufferpool
  - VPSEQT Set to 95 to 98
  - Watch DWQT and VDWQT
- Big change from DB2 9
  - Favouring 32K ones not 4K!

Sort Work Database

- PK70060
  - Now that work files and DGTTs/static scrollable cursors share temp space
  - Do you allocate a secondary amount or not?
    - You might want DGTTs to extend but not sort work files
  - With this APAR:
    - Work files with SECQTY = 0 will be **preferred** for sort work data
    - Work files with SECQTY > 0 will be **preferred** for DGTTs and static scrollable cursors

Sort Work Database

- PM02528
  - Introduced zparm switch to CONTROL this behaviour
    - Rather than it being a preference
  - WFDBSEP:
    - YES – Separation IS maintained
      - If YES, the separation is a HARD separation
      - Failures may occur as overflow is not allowed
    - NO – Separation is NOT maintained
      - If NO, the separation is a SOFT separation
      - DB2 will try and keep usage apart BUT overflow IS allowed
  - See II14587 for recommendations

Sort Work Database

- MAXTEMPS
  - Lastly (?)
  - MAXTEMPS controls maximum amount of storage
    - Of ALL types
    - sort work files, GTTs, DGTTs, scrollable cursors
  - That any single agent can use

Utilities

Keep up with utility performance improvements/options

- RECOVER INDEX with IMAGE COPY
  - 5 - 10 times faster
  - COPY ENABLE on CREATE/ALTER INDEX
- REUSE on REORG and LOAD
  - Skips DELETE/DEFINE
- ONLINE REORG
  - Mapping Table Management Improvements
- COPY & RECOVER Lists
  - TABLESPACE & INDEX
  - From / To DASD
- REORG REBALANCE

Wait Times

- IN DB2 TIME COMPARED TO IN DB2 CPU TIME
  - How much difference ?
  - What are we waiting for ?
- Service Task Wait Time Broken Down to:
  - Open / Close
  - Dataset Create/Extend/Delete
  - SYSLGRNX Downlevel Detection
  - Phase 2 Commit/Abort
- ALWAYS have accounting class 3 turned on
  - Gives much more wait time information

Summary

- Remember Applications accessing DB2 will do one(and only one) of the following at one time:
  - Use CPU
  - Wait for something
    - I/O, CPU, Locked DB2 Object
- Watch SQL related areas
- Keep current on maintenance
- Manage and monitor changes
- Keep current on what YOUR version of DB2 can do!

THE PERFORMANCE DATABASE

Performance Database

- Often we hear talk of a “Performance Database”
  - But what is it?
- It’s a historical collection of performance data
- YOU choose what to store
- YOU choose how long to store it for
- YOU choose where to store it
- YOU choose what to do with it

Performance Database

- My first performance database was simply a flat file containing
  - CICS transaction ID
  - IN DB2 TIME
  - IN DB2 CPU TIME
  - TOTAL ELAPSED TIME
- For each DAY
- And I kept 3 months of data

Performance Database

- You could choose to keep much more data
- And for much longer

- A flat file is pretty simple, but also not very easy to search
- And useless for complex analysis

- Perhaps your performance database should BE a database

Performance Database

- Where you source the data from is also your choice
- Most DB2 monitors will externalise anything you ask for
- Or you could just use DB2s Accounting or Performance trace data
- Or perhaps you have a performance analysis tool for DB2 that has its own Performance Database
  - If so, the definitions should be well documented

Tracking, Truthfulness and Trending

- Tracking is performance is easy with a performance database
- It's IMPOSSIBLE without one
- Say a transaction is running slower than usual
  - Your tracking data may show it's NOT a DB2 problem (IN DB2 time is the same as usual)
  - Or it IS a DB2 problem, but IN DB2 CPU is the same as usual…..

Tracking, **Truthfulness** and Trending

- Has anyone ever reported something is “running slower than yesterday”?
- How do you know?
- Do people always tell the truth?
- Do you waste much time chasing problems that don’t exist
  - Or are not your responsibility?
- First stop – the Performance Database
- What do TODAYS figures look like compared with earlier?

Tracking, Truthfulness and Trending

- Now that you have performance data tracked over time
- It's easy to look for trends
  - Is performance staying the same?
  - Is it degrading?
    - How much by and how fast?
  - Are we running out of a critical resource?
- You can see (and fix) problems BEFORE they arise
  - And before the phone rings
- And you can even validate & quantify any tuning that you do
  - How much did things improve?
  - How much of a star are you?

Thank you.