

# Key Metrics for DB2 for z/OS Subsystem and Application Performance Monitoring (Part 2)

Central Ohio DB2 Users Group

June 25, 2013



# Agenda

## ■ Part 1

- DB2 monitor-generated reports versus online displays
- Application performance: DB2 monitor accounting reports (and displays)

## ■ Part 2

- Subsystem performance: DB2 monitor statistics reports (and displays)
- The best bits in DB2 and CICS DISPLAY command output
- Important DB2-related stuff in z/OS monitor reports and displays

# Subsystem performance: DB2 monitor statistics reports (and displays)

# How are your buffer pools doing?

## ▪ Key metric: total read I/Os per second

–Referring to sample report output below, total read I/O rate for buffer pool BP2 is 3059.56 per second (sum of synchronous reads plus all prefetch reads)

- If more than 1000 I/Os per second, enlarge the pool, if possible (more on this to come)
- If between 100 and 1000 I/Os per second, consider enlarging the pool if LPAR memory resource is adequate

BP2 READ OPERATIONS	/SECOND
-----	-----
SYNCHRONOUS READS	2417.42
SEQUENTIAL PREFETCH READS	199.12
LIST PREFETCH READS	83.58
DYNAMIC PREFETCH READS	359.44

Add these up

## Can a buffer pool be made larger?

- If buffer pool configuration is too large relative to real storage, performance of non-DB2 work can be negatively impacted
- Check z/OS LPAR's demand paging rate (z/OS monitor)
  - If demand paging rate is in low single digits per second (or less) during busy processing periods, the z/OS LPAR's memory resource is not stressed – should be OK to enlarge buffer pool
    - That said, I generally do not like to see the size of a DB2 subsystem's buffer pool configuration exceed half of the size of LPAR memory
  - If demand paging rate is in the high single digits per second (or more), I'd be reluctant to increase the size of the buffer pool configuration
    - But you could increase size of BPx by decreasing size of BPy by same amount
- In DB2 data sharing group, increase in size of local buffer pool could require enlargement of corresponding group buffer pool

## Buffer pool monitoring: data manager threshold

- Data manager threshold (DMTH) is reached when 95% of the buffers in a pool are unavailable (meaning, either in-use or holding changed pages that have not yet been externalized)
  - When DMTH is hit, DB2 does a GETPAGE for every row retrieved from a page in the pool – result can be big CPU utilization spike
- Usually, DMTH hit because buffer pool is too small
  - Another cause I've seen: horizontal, vertical deferred write thresholds are set too high for a pool dedicated to work file table spaces
    - Setting these thresholds higher for work file buffer pools than for other buffer pools is OK (work file pending writes don't affect DB2 restart time), and that can save some cycles, but don't overdo it

BP2 WRITE OPERATIONS	QUANTITY
-----	-----
DM THRESHOLD	0.00

*Good!*

## More on buffer pools dedicated to work files

- Referring to sample report output below, you want zeros in all of these fields – if some are  $> 0$ , buffer pool probably too small
- Check statistics for the pool used for the 4K work file table spaces and the one used for the 32K work file table spaces
- Starting with Version 9, DB2 uses 32K work file space a LOT more than before
  - Probably want at least as much 32K work file space as 4K work file space

BP7 SORT/MERGE	QUANTITY
-----	-----
MERGE PASS DEGRADED-LOW BUF	0.00
WORKFILE REQ.REJCTD-LOW BUF	0.00
WORKFILE NOT CREATED-NO BUF	0.00
WORKFILE PRF NOT SCHEDULED	0.00

## Is your EDM pool large enough (DB2 9)?

- Broken into sections – I've shown "RDS pool below" (others are RDS pool above, skeleton pool, DBD pool, statement pool)
- Want to see zeros for "fails due to pool full" for each section
- I like to see the number of free pages for a pool be at least 10% of the pages in the pool
- Note: `RELEASE(DEALLOCATE)` + thread reuse can reduce the number of free pages in the RDS pools – watch that!

EDM POOL	QUANTITY
-----	-----
PAGES IN RDS POOL (BELOW)	38400.00
HELD BY CT	58.21
HELD BY PT	2455.31
FREE PAGES	35886.48
FAILS DUE TO POOL FULL	0.00



## DB2 10: changes for package storage

- With DB2 10:
  - Thread-related package virtual storage above 2 GB “bar” *for packages bound in DB2 10 system* (some above bar with DB2 9, none with DB2 V8)
  - This virtual storage comes out an agent local pool, not the EDM pool
- Performance implications:
  - Elimination of latching related to use of EDM pool storage for copies of packages used by threads
  - Lots more virtual storage “head room” for using RELEASE(DEALLOCATE) to improve CPU efficiency
- Monitoring: no longer limited by EDM pool size (see previous slide), but as you use more virtual storage for packages (more concurrent threads and/or more RELEASE(DEALLOCATE)), watch system’s demand paging rate

## How many DB2 checkpoints?

- I generally like to see a DB2 checkpoint frequency of one every 5-10 minutes – some folks who want shorter DB2 restart times aim for a checkpoint every 2-5 minutes
  - You're balancing restart time versus overhead of checkpointing
  - The snippet below is from a Statistics Long report that spanned a 2-hour period, so 8 checkpoints works out to one every 15 minutes – that's a little less frequent than I'd like to see
  - Via ZPARMs, you can set checkpoint frequency in terms of minutes between checkpoints or log records written between checkpoints (or, starting with DB2 10, both – whichever limit is reached first)

SUBSYSTEM SERVICES	QUANTITY
-----	-----
SYSTEM EVENT CHECKPOINT	8.00

## What about pseudo-close activity?

- When a data set open for read/write goes for a pseudo-close interval without being updated, it's switched to read-only state (back to read/write at next data-changing SQL statement)
  - Pseudo-close interval determined via two ZPARMS: PCLOSEN (specifies a number of checkpoints) and PCLOSET (specifies a number of minutes) – interval is based on which limit is reached first
  - As with checkpointing, a balancing act: faster data recovery (with more pseudo-close activity) versus overhead of pseudo-close
  - My opinion: pseudo-close frequency of 20-40 per minute is reasonable (value in report snippet below equates to 81 per minute – on high side)

OPEN/CLOSE ACTIVITY	/SECOND
-----	-----
DSETS CONVERTED R/W -> R/O	1.35

## RID list processing

- If DB2 V8 or DB2 9 runs short on storage in processing a RID list, it will revert to table space scan for query being executed
  - Not good – if it was going to be a TS scan, you'd probably prefer for DB2 to do that from the get-go, vs. starting first down the RID list path
  - In the report snippet below, field **A** indicates that virtual storage was exhausted, and field **B** indicates a too-small RID pool (might want to enlarge if either value is non-zero)

RID LIST PROCESSING		QUANTITY
-----		-----
TERMINATED-NO STORAGE	<b>A</b>	0.00
TERMINATED-EXCEED PROC.LIM.	<b>B</b>	0.00

- DB2 10: much larger RID pool default size (400 MB) – if that's not enough, DB2 will process RID list using work file space

# DBATs

- Snippet shows that during report period (two hours), there were almost 3 million times when a DBAT was needed to service a DRDA transaction, and DB2 had to create new DBAT 256 times
  - That's a very good use of pooled threads – helps CPU efficiency
  - If you saw more create DBAT activity and less pool DBAT reuse, might want to increase value of POOLINAC in ZPARM (number of seconds that a DBAT can be idle in the pool before being terminated)

GLOBAL DDF ACTIVITY	QUANTITY
-----	-----
DBATS CREATED	256.00
POOL DBATS REUSED	2919.8K

## DB2 10 high performance DBATs

- The report fields shown below indicate the use of high performance DBATs in the DB2 system
- Recall that a “regular” DBAT becomes a high performance DBAT when it is used to execute a package bound with `RELEASE(DEALLOCATE)`
- Note: high performance DBATs deplete supply of pooled DBATs
  - If you’re going to use `RELEASE(DEALLOCATE)` for some DRDA-invoked packages, you might want to up the value of `MAXDBAT` in `ZPARM`

GLOBAL DDF ACTIVITY	QUANTITY
-----	-----
CUR ACTIVE DBATS-BND DEALLC	0.00
HWM ACTIVE DBATS-BND DEALLC	0.00

## DB2 address space CPU utilization

- Note (and this snippet is from a report covering a 2-hour period):
  - IRLM, DB2 system services address spaces use very little CPU time
  - Database services address space CPU time mostly related to database writes, prefetch reads (DB2 10: async I/O is zIIP-eligible)
  - DDF address space uses very little CPU time with respect to “system” tasks (TCB and non-preemptable SRB time)
  - CPU time associated with DDF preemptable SRBs is basically the cost of SQL execution for statements coming through DDF – just as SQL statement CPU time is charged to (for example) CICS regions

CPU TIMES	TCB TIME	PREEMPT SRB	NONPREEMPT SRB	PREEMPT IIP SRB
-----	-----	-----	-----	-----
SYSTEM SVCS	12.646	0.000	2:03.883	N/A
DB SVCS	5:33.773	0.004	50:05.190	0.394
IRLM	0.003	0.000	21.245	N/A
DDF	1:02.659	3:13:21.699	2:10.283	2:55:43.179

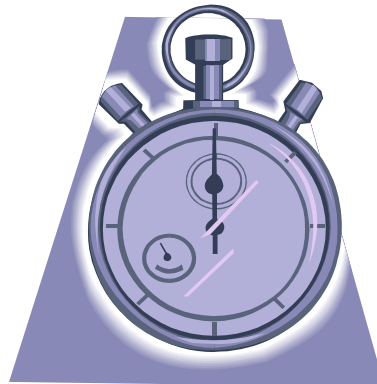
Preemptable SRB time consumed on general-purpose engines, consumed on zIIPs

# The best bits in DB2 and CICS DISPLAY command output



# My favorite: -DISPLAY BUFFERPOOL

- My preferred form of the command:
  - DISPLAY BUFFERPOOL(ACTIVE) DETAIL
- My preferred way of using the command:
  - Issue the command during a busy time on the system
  - Wait an hour, and issue the command again
  - Take the output of the second issuance of the command, and divide activity numbers by 3600 to get per-second figures



# -DISPLAY BUFFERPOOL output (abridged)

```

DSNB401I  -DB1P BUFFERPOOL NAME BP3, BUFFERPOOL ID 3, USE COUNT 121
DSNB402I  -DB1P BUFFER POOL SIZE = 40000 BUFFERS
           IN-USE/UPDATED      =      26    BUFFERS ACTIVE      =      40000
DSNB406I  -DB1P PGFIX ATTRIBUTE -
           CURRENT = NO (YES good for high-I/O pools - CPU savings)
           PENDING = NO
           PAGE STEALING METHOD = LRU (DB2 9: FIFO good for "pinned-object" pools;
DSNB404I  -DB1P THRESHOLDS - (DB2 10: NONE good for "pinned-object pools)
           VP SEQUENTIAL      = 80
           DEFERRED WRITE     = 50    VERTICAL DEFERRED WRT   = 10, 0
DSNB409I  -DB1P INCREMENTAL STATISTICS SINCE 10:00:08 JUN 28, 2010
DSNB411I  -DB1P RANDOM GETPAGE      = 26054523 SYNC READ I/O (R) =
623335 (A)
           SEQ.    GETPAGE      = 1253010 SYNC READ I/O (S) = 37095 (B)
(0 is good!) → DMTH HIT          =          0 PAGE-INS REQUIRED =          0
DSNB412I  -DB1P SEQUENTIAL PREFETCH - (memory probably not under pressure)
           REQUESTS      =      28880    PREFETCH I/O      =      9229 (C)
DSNB413I  -DB1P LIST PREFETCH -
           REQUESTS      =          0    PREFETCH I/O      =          0 (D)
DSNB414I  -DB1P DYNAMIC PREFETCH -
           REQUESTS      =     158785    PREFETCH I/O      =     21123 (E)

```

Read I/Os per second = (A + B + C + D + E) / seconds in interval

# -DISPLAY BUFFERPOOL output (continued)

If > 0, pool may be undersized

```

DSNB415I  -DB1P PREFETCH DISABLED -
          NO BUFFER                =          0  NO READ ENGINE      =          0
DSNB420I  -DB1P SYS PAGE UPDATES  = 2770777  SYS PAGES WRITTEN =
436472    ASYNC WRITE I/O          = 17836    SYNC WRITE I/O          = 15
          PAGE-INS REQUIRED          =          0  ← (memory probably not under pressure)
DSNB421I  -DB1P DWT HIT            =          0  VERTICAL DWT HIT      =
3373

```

If these thresholds not hit, or hit only a few times in an hour, consider lowering DWQT and/or VDWQT. If hit one or more times per second, consider increasing DWQT and/or VDWQT (zero values here are OK - due to very low rate of buffer updates).

# -DISPLAY DDF DETAIL output (abridged)

```

DSNL080I  @ DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I  STATUS=STARTD
DSNL082I  LOCATION                LUNAME                GENERICLU
DSNL083I  ABCDE123                USATLGA.ABCDE123     -NONE
DSNL084I  TCPSPORT=446            SECPORT=0            RESPORT=5001        IPNAME=-NONE
DSNL085I  IPADDR=: :1.22.33.444
DSNL086I  SQL          DOMAIN=prodmvs.demo.las.com
DSNL090I  DT=A          CONDBAT= 200  MDBAT= 100
DSNL092I  ADBAT= 1      QUEDBAT= 0    INADBAT= 0  CONQUED= 0
DSNL093I  DSCDBAT= 0    INACONN= 0
DSNL099I  DSNLTDDF DISPLAY DDF REPORT COMPLETE
  
```

With inactive DDF threads,  
number of client connections  
to DB2 can exceed number of  
active DBATs

Cumulative number of times (since  
last DB2 restart) that work had to  
wait for a DBAT to become available  
- if > 0, consider increasing  
MAXDBAT in ZPARM

Current number of queued  
connection requests

# DSNC DISPLAY STATISTICS (CICS command)

DFHDB2014 07/09/98 14:35:45 IYK4Z2G1 Statistics report follows for RCTJT accessing DB2 DB3A

DB2ENTRY	PLAN	CALLS	AUTHS	W/P	HIGH	ABORTS	-----COMMITTS-----	
							1-PHASE	2-PHASE
*COMMAND		1	1	1	1	0	0	0
*POOL	*****	4	1	0	1	0	2	0
XC05	TESTP05	22	1	11	2	0	7	5
XP05	*****	5	2	0	1	0	1	1

DFHDB2020 01/17/98 15:45:27 IYKA4Z2G1 The display command is complete.

Asterisks mean that dynamic plan allocation is used (I'm not big on that)

Indicates, for DB2ENTRY resources, the number of times transactions overflowed to the pool (assuming THREADWAIT(PPOOL) specified); for pool threads, indicates the number of times that transactions were queued up waiting for a thread  
 • If > 0, may need to increase number of pool threads - and make sure that TCBLIMIT is sufficiently large in DB2CONN resource definition

# Important DB2-related stuff in z/OS monitor reports and displays

# RMF CPU activity (abridged)

```

1                C P U   A C T I V I T Y

z/OS V1R10                SYSTEM ID P01                START 05/24/2010-09.00.00
                        RPT VERSION V1R10 RMF          END   05/24/2010-11.00.00

-CPU   xxxx   MODEL   yyy   H/W MODEL   zzz
0---CPU---   ----- TIME % -----
  NUM   TYPE   LPAR BUSY   MVS BUSY
  0     CP     89.96     98.65
  1     CP     89.94     98.59
  2     CP     89.93     98.50
  3     CP     89.89     98.41
  4     CP     89.85     98.29
  5     CP     89.80     98.19
  6     CP     89.66     97.98
TOTAL/AVG   89.86     98.37
0 7     IIP     45.39     45.45
  8     IIP     47.19     47.25
TOTAL/AVG   46.29     46.35
    
```

- Higher zIIP utilization is good, because zIIP MIPS cost less than general-purpose MIPS
- Client-server (DDF) work can drive zIIPs to high utilization during the day, but that might go way down at night
- Some organizations looking to boost nighttime zIIP utilization are binding some batch packages with DEGREE(ANY) - query parallelization is zIIP-eligible (can limit degree of parallelization via PARAMDEG in ZPARM)



# RMF Summary Report (abridged)

```

1           R M F   S U M M A R Y   R E P O R T

z/OS  V1R10           SYSTEM ID P01           START 02/15/2010-08.30.00
           RPT VERSION V1R10 RMF           END   02/15/2010-11.00.00

0
NUMBER OF INTERVALS 5           TOTAL LENGTH OF INTERVALS 02.29.57
-DATE    TIME          INT    CPU    DASD  DASD  SWAP  DEMAND
MM/DD   HH.MM.SS    MM.SS   BUSY  RESP  RATE  RATE  PAGING
002/15  08.30.00   29.59   78.3   1.2 13191  0.00   2.65
 02/15  09.00.00   30.00   90.5   1.2 14783  0.00   3.89
 02/15  09.30.00   29.59   87.0   1.2 13327  0.00   1.42
 02/15  10.00.00   30.00   86.0   1.2 12542  0.00   1.46
002/15  10.30.00   29.59   88.3   1.2 13029  0.00   4.03
-TOTAL/AVERAGE           86.0   1.2 13375  0.00   2.69
  
```

- As noted previously, demand paging rate should be in the low single digits per second (or less)
- 4-6 per second is "yellow light" territory
- High single digits per second (or more): LPAR memory resource is under more pressure than you'd like - add memory or reduce memory consumption



Robert Catterall  
[rfcatter@us.ibm.com](mailto:rfcatter@us.ibm.com)