Information Management

IBM

December 7, 2011

Baltimore/Washington
DB2 Users Group

The MOST VALUABLE Information I get from DB2 Monitor Accounting and Statistics Reports

Robert Catterall
IBM
rfcatter@us.ibm.com

Agenda

- Using your DB2 for z/OS monitor to generate accounting and statistics detail reports
- Accounting: what is the nature of your DB2 for z/OS workload?
- Accounting: how CPU-efficient are your DB2 applications?
- Statistics: how are your DB2 buffer pools doing?
- Statistics: is your DB2 subsystem operating efficiently?

Using your DB2 for z/OS monitor to generate accounting and statistics detail reports

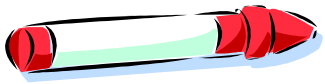
DB2 for z/OS monitors – typical usage

- In my experience, virtually every organization that uses DB2 for z/OS has a DB2 monitoring tool
  - These tools are available from a several vendors, including IBM
- Also based on my experience, it seems that many organizations use their DB2 monitors exclusively in online mode
  - That is to say, organizations use their monitors only to check out DB2 application and subsystem performance metrics in real time – i.e., to see “what’s going on right now”
  - Online monitoring has its place (indeed, it’s vitally important when you’re troubleshooting a performance problem), but you’re missing out on a lot of value if that’s the ONLY way you use your monitor

The value of monitor-generated reports

- For my money, NOTHING beats monitor reports for in-depth analysis of application and subsystem performance
  - Accounting reports: application-level view
  - Statistics reports: subsystem-level view

And you can mark them up with a red pen!



- Also great for trend analysis, and for determining what changed from a time before a performance problem arose to a time after the problem arose
  - When I worked in the IT department of a mainframe DB2-using organization, we had a great DB2 monitor reporting set-up:
    - Every day, DB2 accounting and statistics detail reports were “printed” to data sets in GDGs – browse-able via TSO/ISPF
    - Kept a rolling 60 days of reports – fantastic performance research and analysis resource

Good news for online monitor users

- Virtually every major DB2 monitoring tool has a batch reporting feature
  - Usually documented in a manual with a title akin to “Batch Reporting Users Guide”
    - Among other things, shows DB2 trace classes needed for reports
- Generating reports is a matter of executing a batch job that includes a DD statement pointing to a data set containing DB2 trace records (usually directed to SMF)
  - Batch job has a control statement in SYSIN, which indicates:
    - “From” and “to” dates/times
    - Report type (e.g., ACCOUNTING LONG or STATISTICS DETAIL)
    - Filtering criteria (e.g., include or exclude a DB2 plan name)
    - Report data organization options (e.g., group by connection type)

My favorite DB2 monitor reports

- Accounting detail (aka “accounting long”), with:
  - “From” and “to” times encompassing either a busy 1- or 2-hour time period, or a 24-hour time period
  - Data grouped by (for some monitors, it’s “ordered by”) connection type
    - Gives you a detailed report for each DB2 connection type: CICS, IMS, DRDA, TSO, call attach, utility, etc.
    - If you need more detail, can get data at correlation-name level (e.g., CICS transaction ID or batch job name), primary auth ID level, etc.
- Statistics detail (aka “statistics long”), with:
  - “From” and “to” times as indicated for accounting reports, above
- Loads of other reports available, but these are most useful – and they’re pretty inexpensive (they format records generated by low-overhead DB2 trace classes)

Accounting: what is the nature of your DB2 for z/OS workload?

What's the biggest component of your DB2 workload?

- A pretty simple question, but – believe it or not – one that plenty of DB2 people can't readily answer
- By "biggest," I mean biggest in terms of aggregate class 2 CPU time
  - So called because information comes from DB2 accounting trace class 2
  - Also known as "in-DB2" CPU time
  - Indicates the CPU cost of SQL statement execution
- By "component," I'm referring to connection type (e.g., CICS, batch, DRDA, etc.)

Answering “biggest component” question

- Use accounting detail report, with data grouped by connection type
- For each connection type, perform a simple calculation (referring to sample report output on following slide):

  (average class 2 CPU time) X (number of occurrences)

  - “Number of occurrences” = number of trace records
    - Usually one per transaction for online, one per job for batch
  - Reports generated by different monitors can look a little different, but you should be able to find the fields shown in the sample report
  - Note also that I’m leaving out some report lines and columns because putting all on slide would require a too-small font size
  - Data in sample report happens to be for a 2-hour time period

Sample report output

```
CONNTYPE: DRDA

AVERAGE          DB2  (CL.2)      HIGHLIGHTS
----------       ----------       ------------------------
CP CPU TIME        0.003614       #OCCURRENCES   :   3087344

SE CPU TIME        0.003348
```

Don't forget this! (SE = "specialty engine," which usually means zIIP)

(avg CL 2 CPU) X (# of occurrences) = 0.006962 X 3,087,344

= 21,494 seconds

In a DB2 data sharing environment, do this for each member of the group to get TOTAL DRDA SQL cost, TOTAL CICS-DB2 SQL cost, etc.

Comments on DB2 workload components

- Quite often, DRDA-related activity is the fastest-growing component of an organization's DB2 for z/OS workload
- At some sites, DRDA-related activity is the largest component of the DB2 for z/OS workload – bigger than CICS-DB2, bigger than batch-DB2
  - Again, "largest" refers to total class 2 CPU time
- I have found that people – even mainframe DB2 people – are often unaware of this
  - Not uncommon for senior IT managers to think of the mainframe as just the server where the "legacy" applications run
  - In fact, the mainframe DB2 platform is evolving to become a "super-sized" (and super-available and super-secure) data server for multi-tier apps

More on DB2 for z/OS and client-server

- A DBA at one site reported that a Java developer told him that he "didn't want to have anything to do with a mainframe"
  - At that same site, developers are writing Java programs with JDBC calls that access a DB2 for z/OS database – it looks to them like any other relational database
- At another site, an ADABAS application is being migrated to DB2 for z/OS, and the front-end code will be written in Perl
- The point: DB2 for z/OS is a great data server for multi-tier, client-server applications
  - This is happening at many mainframe DB2 sites, and data on the size and growth of that workload can help in highlighting that work – and changing people's perceptions of the mainframe platform

DB2 people: you have an opportunity to lead here, not just accommodate

Another important workload characteristic

- Is the DB2 workload CPU-constrained?
- A good place to check: “not accounted for” time in the DB2 monitor Accounting Detail report
  - What it is: in-DB2 (i.e., class 2) elapsed time that is not CPU time, not suspension time (the latter being class 3, or “waiting for” time)
  - Basically DB2 saying, “this was time, related to SQL statement execution, that I can’t account for”
  - In my experience, usually associated with DB2 wait-for-dispatch time
    - In other words, DB2 (vs. application) tasks are not being readily dispatched
  - DB2 address spaces usually have a high priority in the system, so if not-accounted-for time is relatively high for a transactional workload, it could be that you’ve hit a processing capacity wall

Sample report output (1)

```
CONNTYPE: CICS

CLASS 2 TIME DISTRIBUTION
-----------------------------------------
CPU     |================> 30%
SECPU   |
NOTACC  |==> 5%
SUSP    |================================> 65%
```

- I get concerned if not-accounted-for time is greater than 10% for a high-priority transactional workload such as CICS-DB2 (or, often, DRDA)
  - Not so concerned if this time exceeds 10% for batch DB2 workload – that's not uncommon

Sample report output (2)

```
CONNTYPE: CICS

AVERAGE          DB2  (CL.2)
-------------    ----------
ELAPSED TIME       0.085225 (A)

CP CPU TIME        0.025313 (B)

SE CPU TIME        0.000000 (C)

SUSPEND TIME       0.055708 (D)

NOT ACCOUNT.       0.004204
```

- If your monitor report does not have the "bar chart" elapsed time breakdown shown on the preceding slide, it will likely have a "not accounted for" field in the "class 2" time column (in red at left)
- If "not accounted for" time is not provided, calculate it yourself:

A - (B + C + D)

What if not-accounted-for time is high?

- One solution: add processing capacity (could just be an LPAR configuration change)
- If that's not feasible…
  - May see what you can do to reduce CPU consumption of the DB2 workload (more on that to come in this presentation)
  - Ensure that dispatching priorities are optimized for throughput in a CPU-constrained environment
    - IRLM should be in the SYSSTC service class (very high priority)
    - DB2 MSTR, DBM1, DIST, and stored procedure address spaces should be assigned to a high-importance service class (my opinion: somewhat higher priority than CICS AORs)
      - Also, may need to go with PRIORITY(LOW) for CICS-DB2 transaction TCBs (this is relative to priority of CICS AOR main task – default is HIGH)
    - Classify DRDA transactions so they won't run as "discretionary" work

How is your DB2 I/O performance?

Sample report output

```
CONNTYPE: DB2CALL

                                        (A)            (B)
CLASS 3 SUSPENSIONS    AVERAGE TIME    AV.EVENT
-------------------    ------------    --------
SYNCHRON. I/O               6.520800     6133.32
```

Average service time for synchronous I/Os = A / B

- These times are getting to be astoundingly low (in this case, 1.06 ms)
  - Has much to do with advances in I/O-related hardware and software: faster channels, parallel access volumes (greatly reducing UCB-level queuing), huge amounts of disk controller cache (and sophisticated management of same)
- A time greater than 5 ms represents opportunity for improvement
- A time greater than 10 ms could indicate a performance problem

Accounting: how CPU-efficient are your DB2 applications?

What are you looking to reduce?

Sample report output

```
AVERAGE          DB2  (CL.2)
-----------      ----------
CP CPU TIME       28.311773 (A)

SE CPU TIME        0.000000 (B)
```

- Usually, you're aiming to reduce A, above
  - Note that, sometimes, reducing A can be accomplished by increasing B (more on this to come)

Average CPU time – average per what?

Sample report output

```
AVERAGE          DB2  (CL.2)
-----------      ----------
CP CPU TIME       28.311773

SE CPU TIME        0.000000
```

- That depends on the granularity of the information in the report, which you specify in generating the report

- Could be average:
  - Per transaction or job for a connection type (e.g., all DRDA, or all call attach)
  - Per transaction for a CICS AOR
  - For a given batch job or CICS transaction (correlation name)
  - Per transaction or job for a given DB2 authorization ID
- Larger-scale granularity can be appropriate focus when planning a change of the "rising tide lifts all boats" variety (e.g., a page-fixed buffer pool)

Information at the program (package) level

Sample report output

```
M123456B                        TIMES
------------    -------------
CP CPU TIME     13:35.566002

SE CPU TIME           0.000000
```

- Very useful if a batch job or transaction involves execution of multiple programs
- Requires data from DB2 accounting trace classes 7 and 8

- If looking at program-level data, where to start?
  - Your monitor may show in the Accounting Detail report the top programs by elapsed time (class 7)

```
PROGRAM NAME    CLASS 7 CONSUMERS
D789123Y        |=> 3%
M123092G        |=======> 15%
I273459Z        |> 1%
```

Boosting efficiency: thread reuse

This happens to be for a CICS-DB2 workload

Thread reused, auth ID changed

Thread not reused

Thread reused, no auth ID change

NORMAL TERM.	AVERAGE
NEW USER	0.79
DEALLOCATION	0.01
RESIGNON	0.20

- Report fragment above shows a thread reuse rate of 99% - very good
- To increase CICS-DB2 thread reuse, define protected entry threads for high-use transactions (PROTECTNUM in DB2ENTRY RDO resource)
  - Non-protected thread likely to be deallocated after transaction completes
  - Protected thread will stick around for 45 seconds (default) after transaction completes – can be reused by another transaction associated with same DB2ENTRY if plan name doesn't change

Maximizing benefit of thread reuse

- Bind packages associated with reused threads with RELEASE(DEALLOCATE)
  - What that means: table space locks, EDM pool control blocks retained until thread deallocation, vs. being released at commit (i.e., end of tran)
  - Benefit: if package is executed repeatedly via the same thread, these resources won't have to be reacquired, and that improves CPU efficiency
- Impact: potential to reduce CPU consumption by several percentage points for affected transactions
- Considerations? Not many…
  - Table space locks are rarely exclusive
  - If using DB2 V8 or DB2 9, keep an eye on EDM pool space
    - RELEASE(DEALLOCATE) will increase amount of non-stealable space

DB2 10: a new thread reuse option

- Talking about high performance DBATs
  - Instantiated when DBAT used to execute a package bound with RELEASE(DEALLOCATE)
    - Prior releases of DB2 treated packages bound with RELEASE(DEALLOCATE) as though bound with RELEASE(COMMIT) when executed via DBAT
  - High performance DBAT not pooled – remains dedicated to connection through which it was instantiated
    - Terminated after 200 units of work to free up resources
  - Used to greatest advantage with simple, high-volume DRDA transactions (may want to bind IBM Data Server Driver packages with RELEASE(DEALLOCATE))
  - Monitoring: DB2 monitor Statistics Detail report (stay tuned)

Key determinant of CPU cost: GETPAGES

TOTAL BPOOL ACTIVITY	AVERAGE
GETPAGES	359.66

- For my money, the most important factor in a transaction's or job's CPU cost
  - Reducing average GETPAGEs per execution is highly likely to reduce CPU time per execution
- GETPAGE reduction is usually a matter of changing the access path used to execute a query
  - Might involve adding indexes or modifying existing indexes
  - Might involve rewriting the query to get a better-performing access path

Dynamic SQL statements: cache hits

DYNAMIC SQL STMT	AVERAGE
NOT FOUND IN CACHE	0.26
FOUND IN CACHE	1.05

- Tends to be particularly important for DRDA transactions, as these often involve execution of dynamic SQL statements
  - Recall that when programs issue JDBC or ODBC calls, these are executed as dynamic SQL statements on the DB2 for z/OS server
  - CPU cost of full PREPARE of a statement can be several times the cost of statement execution
- One way to boost statement cache hits: enlarge the dynamic statement cache (it's been above the 2 GB "bar" since DB2 V8)'
- Also helpful: parameterization of dynamic SQL statements (not always possible)

DB2 10 and dynamic statement caching

DYNAMIC SQL STMT	AVERAGE
CSWL - MATCHES FOUND	0.24

- New CONCENTRATE STATEMENTS WITH LITERALS attribute of PREPARE statement (can also be enabled on DB2 client side via specification of a keyword in the data source or connection property)
  - If match for dynamic statement with literals is not found in the cache, literals replaced with & and cache is searched to find a match for the new statement (if not found, new statements is prepared and placed in the cache)
- CPU savings less than that achieved with traditional dynamic statement caching and parameterized dynamic statements, but quite a bit less expensive than full PREPAREs of statements

Offloading work to zIIP engines

AVERAGE	DB2 (CL.2)	
CP CPU TIME	28.311773	A
SE CPU TIME	0.000000	B

← Aim: reduce A by increasing B

- Options:
  - If it's a DRDA workload, and you're using traditional DB2 stored procedures, switch to native SQL procedures (available with DB2 9 in NFM)
  - If it's a batch workload, consider binding some packages with DEGREE(ANY) to enable query parallelization
  - Migrate to DB2 10 (if not there already) – prefetch processing is zIIP-eligible, and so is XML schema validation processing

Statistics: how are your DB2 buffer pools doing?

Key metric: read I/Os per second

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

- Total read I/O rate for this pool is 3059.56 per second
- If the figure is greater than 1000 per second, enlarge the pool, if possible (more on this to come)
- If the figure is greater than 100 per second and less than 1000 per second, consider enlarging the pool if LPAR memory resource is adequate

Can a pool be made larger?

- Check the z/OS LPAR's demand paging rate (available from a z/OS monitor)
  - If demand paging rate is less than 1 per second during busy processing periods, z/OS LPAR's memory resource is definitely not stressed – should be OK to enlarge buffer pool
    - That said, I generally don't 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 close to 10 per second, I'd be reluctant to increase the size of the buffer pool configuration
    - But you could increase the size of BPx by decreasing the size of BPy by the same amount
- If running DB2 in data sharing mode, increase in size of "local" buffer pool could necessitate enlargement of group buffer pool

Watch out for the data manager threshold!

BP2 WRITE OPERATIONS	QUANTITY
DM THRESHOLD	0.00

- Data manager threshold (aka 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, and that can cause a major CPU utilization spike
- Usually, DMTH hit because a buffer pool is too small (make it bigger!)
  - Another cause I've seen: deferred write thresholds set too high for a pool dedicated to work file table spaces
  - Yes, these thresholds can be higher for that pool than for others (because work file table spaces aren't recovered on DB2 restart), but don't overdo it

Speaking of the work file buffer pools…

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

- You want to see zeros in all of these fields – if some have non-zero values, buffer pool is probably too small
- Be sure to check the pool used for the 4K work file table spaces and the one used for the 32K work file table spaces
- If currently running with DB2 V8 for z/OS, keep in mind that DB2 9 and DB2 10 will likely need a LOT more 32K work file space
  - Probably want to start out with at least as much 32K as 4K work file space

Statistics: is your DB2 subsystem operating efficiently?

Is your EDM pool large enough?

EDM POOL	QUANTITY
PAGES IN SKEL POOL (ABOVE)	3941.00
HELD BY SKCT	46.15
HELD BY SKPT	3725.56
FREE PAGES	169.29
FAILS DUE TO SKEL POOL FULL	0.00

- Broken into sections – I've shown the skeleton pool section (others are RDS pool below, RDS pool above, DBD pool, and 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 (the 169 free pages indicated above looks small to me)
- Note: RELEASE(DEALLOCATE) + thread reuse can reduce the number of free pages in the RDS pools – watch that!

How many DB2 checkpoints?

SUBSYSTEM SERVICES	QUANTITY
SYSTEM EVENT CHECKPOINT	8.00

- 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 above is from a Statistics Detail 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 between-checkpoints limit is reached first)

What about pseudo-close activity?

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

- When a data set that is open for read/write access goes for a pseudo-close interval without being updated, it is switched to a read-only state (switched back to read/write at next data-changing SQL statement)
  - Pseudo-close interval determined via two ZPARMS: PCLOSEN (specifies a number of minutes) and PCLOSET (specified a number of checkpoints) – interval is based on which of these two limits is reached first
  - Pseudo-close is analogous to checkpointing, in that you are balancing faster restart time (quicker pseudo-closes) versus overhead of pseudo-close
  - I think that a pseudo-close frequency of 20-40 per minute is reasonable (value in report snippet above equates to 81 per minute – on the high side)

RID list processing

```
RID LIST PROCESSING                    QUANTITY
---------------------------            --------
TERMINATED-NO STORAGE            (A)       0.00
TERMINATED-EXCEED PROC.LIM.      (B)       0.00
```

- If DB2 V8 or DB2 9 runs short on storage in processing a RID list, it will revert to a table space scan for the query being executed
  - That's a bummer – if it was going to be a TS scan, you'd probably prefer for DB2 to do that from the get-go, versus starting first down the RID list path
  - In the report snippet above, the A field indicates that DBM1 storage was exhausted (seems unlikely in a 64-bit world), and the B field indicates a too-small RID pool (might want to make that bigger if B value is non-zero)
- DB2 10: MUCH larger RID pool default size (400 MB), and if virtual storage is insufficient, DB2 will process RID list using work file space

DBATs

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

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

DB2 10 high performance DBATs

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

- These fields 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)
- Because high performance DBATs deplete the supply of pooled DBATs, if you're going to use RELEASE(DEALLOCATE) for some DRDA-invoked packages then you might want to up the value of MAXDBAT in ZPARM

And finally, the DB2 address spaces

CPU TIMES	TCB TIME	PREEMPT SRB	NONPREEMPT SRB
SYSTEM SVCS ADDRESS SPACE	12.646021	0.000000	2:03.883502
DB SVCS ADDRESS SPACE	5:33.773976	0.004446	50:05.190835
IRLM	0.003833	0.000000	21.245941
DDF ADDRESS SPACE	1:02.659197	3:13:21.699476	2:10.283841

- Note (and this snippet is from a report covering a 2-hour period):
  - IRLM and DB2 system services address spaces use VERY little CPU time
  - Database services address space uses a fair amount of CPU time – primarily related to database writes and prefetch reads
  - DDF address space uses very little CPU time with respect to "system" tasks (TCB and non-preembtible SRB time)
  - The large amount of CPU time associated with DDF preemptible SRBs is basically the cost of SQL statement execution charged to DBATs – just as SQL statement CPU time is charged to CICS-DB2 subtask TCBs

Robert Catterall
rfcatter@us.ibm.com