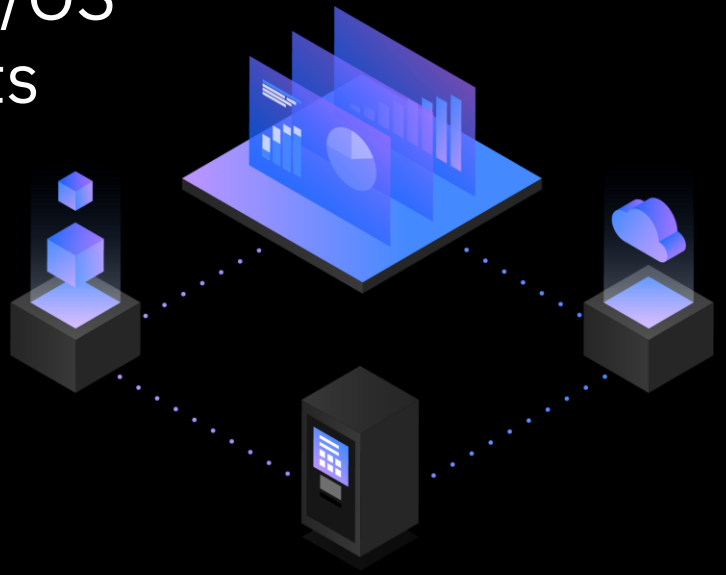
Getting the Most out of Db2 for z/OS Statistics and Accounting Reports



New England Db2 User's Group

December 2, 2021

Mark Rader
Db2 for z/OS Specialist
IBM Z Washington Systems Center

IBM

Agenda

- Introduction – what reports are we talking about?
- Statistics report – the subsystem view of Db2 for z/OS performance
- Accounting report – the application view of Db2 for z/OS performance

Introduction – what reports are we talking about?

How you get the reports

- They are generated by your Db2 for z/OS monitor
  - Most every Db2 performance monitor product, from any vendor, can generate statistics and accounting reports in batch mode, in addition to providing online monitoring capabilities
  - Job generation involves submitting a batch job that invokes Db2 monitor's reporting component and specifies type of report (e.g., statistics), reporting interval (i.e., "from" and "to" times), etc.
  - Typically, a DD statement in the job's JCL points to a data set with Db2 SMF trace records that provide input to the monitor's report generator
- If you haven't used your Db2 monitor to generate reports in batch mode, you can contact the vendor for assistance if needed

Detail and duration

- **Detail** – the reports of interest are ones that have a lot of detailed information (Db2 monitors can also generate reports with summary information)
  - In IBM OMEGAMON for Db2 for z/OS terminology, the reports you want are called the statistics long report and the accounting long report
    - Other Db2 monitors: similar reports might be called statistics (or accounting) detail report, or detailed summary of statistics (or accounting) information
- **Duration** – often useful to generate a report that captures one hour of activity during a time when the Db2 system is relatively busy
  - Shorter duration could be a little misleading due to short-duration peaks or drops in activity…unless you are specifically looking for a spike
  - Longer duration might overly smooth out activity peaks and valleys

A little more before looking at report info…

- For this session, terminology used and report samples shown will be related to IBM OMEGAMON Performance Expert for Db2 on z/OS
- As noted, other monitors can produce similar reports
  - What any monitor is actually doing to generate a statistics or accounting report is formatting and aggregating information in Db2 trace records
- Recommendations and guidelines are for production Db2 systems – performance tuning typically much less important for dev/test systems

Statistics report – the subsystem view of Db2 for z/OS performance

Buffer pools: get a handle on the most important metric

- Talking about the total read I/O rate (determine this for each buffer pool)
  - That's all synchronous + all prefetch reads for a pool, per second
    - There are 3 categories of prefetch read: sequential, list and dynamic

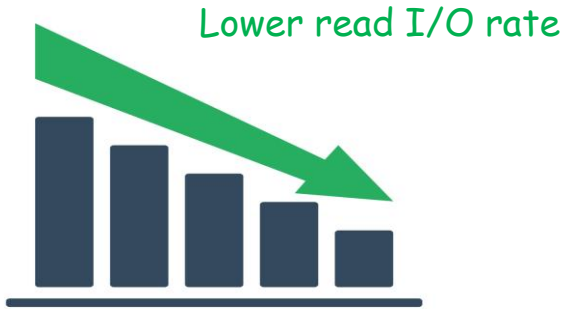
BP3 READ OPERATIONS	QUANTITY	/SECOND
SYNCHRONOUS READS	1676.1K	**465.56**
SEQUENTIAL PREFETCH READS	9912.00	**2.75**
LIST PREFETCH READS	191.00	**0.05**
DYNAMIC PREFETCH READS	26440.00	**7.34**

For this buffer pool, total read I/O rate is 465.56 + 2.75 + 0.05 + 7.34 = 475.7/second

Why the total read I/O rate is important

The primary aim in buffer pool tuning…

- Drive total read I/O rates as low as you can by enlarging pools with higher read I/O rates, without over-burdening system memory
  - Real storage is not over-burdened if z/OS LPAR's demand paging rate (available via z/OS monitor) is less than 1 per second
- Reducing read I/Os improves response time (less I/O wait time) and saves CPU (every I/O consumes some CPU time)



Driving down buffer pool read I/O rates

- Generally want to focus on pools with highest read I/O rates
  - If you have any buffer pools with read I/O rates > 1000/second, maybe try to get those below 1000/second
  - Some organizations that super-size buffer pools are aiming for read I/O rates that are < 100/second, or even < 10/second for each pool
- Some data points:
  - Read I/O rates for some real-world Db2 buffer pools exceed 12,000 per second (average over a 1-hour period)
  - At least one real-world Db2 subsystem has a total buffer pool configuration size (aggregate size of all pools) that is greater than 870 GB (this was seen on an LPAR with more than 1100 GB of real storage – demand paging rate was zero)
  - At least one real-world subsystem has an individual buffer pool > 290 GB in size

The GETPAGE rate and large real storage page frames

BP29 READ OPERATIONS	QUANTITY	/SECOND
GETPAGE REQUEST	192.1M	54.3K

- Especially when real storage not over-burdened (demand paging rate < 1/sec), every pool with GETPAGE rate > 1000/sec should be fully backed by large real storage page frames (1 MB or 2 GB frames)
- Large frames make page access more CPU-efficient, thanks largely to more-efficient virtual-to-real-storage address translation
- Busier the pool, greater the benefit – a pool with > 1000 GETPAGEs/sec is busy

Focusing on prefetch numbers

BP39 READ OPERATIONS	QUANTITY	/SECOND
DYNAMIC PREFETCH REQUESTED	1244.9K	351.68
DYNAMIC PREFETCH READS	295.5K	83.47
PREF.DISABLED-NO BUFFER	0.00	0.00
PREF.DISABLED-NO READ ENG	0.00	0.00

Nice to see significantly more prefetch requests versus prefetch reads – check for list and sequential prefetch, too (when all pages in a prefetch request are already in the buffer pool, there is no corresponding prefetch read I/O)

- Indicates number of times that prefetch could not be performed, due to lack of buffers that can hold prefetched pages (pages will be read synchronously, instead)
- Could be due a too-small buffer pool, or to a too-low VPSEQT value (VPSEQT is percent of buffers allowed to hold pages read into memory via prefetch – default is 80, but some people take it way down, to 20 or even less)

- Indicates number of times that there were more prefetch requests than Db2 prefetch tasks – for excess number of prefetch requests, pages will be read synchronously, instead
- Less likely to see non-zero value with Db2 12, as number of prefetch tasks doubled versus Db2 11
- If value non-zero, consider enlarging pool (if memory not over-burdened) and/or increasing VPSEQT value

How about buffer pool write activity?

BP20 WRITE OPERATIONS	QUANTITY	/SECOND
SYNCHRONOUS WRITES	326.00	0.09
ASYNCHRONOUS WRITES	943.00	0.27
HORIZ.DEF.WRITE THRESHOLD	0.00	0.00
VERTI.DEF.WRITE THRESHOLD	19.00	0.01
DM THRESHOLD	0.00	0.00

Generally speaking, want to see more aynch writes than synch writes – if that is not the case, might want to reduce VDWQT for the pool to trigger asynch write more frequently (caveat: if more synch writes than asynch writes, not a big deal if volume of writes really low – e.g., less than 10/second)

- Common for horizontal deferred write threshold to not be hit
- If VDWQT not hit at all, maybe reduce that value for pool (unless number of writes is really low – e.g., < 10/second)

- Important for the data manager threshold to always be zero
- Threshold hit when 95% of buffers in pool are non-stealable
- When DM threshold hit, GETPAGEs spike for pool, leading to more CPU usage
- Non-zero value here usually indicates that buffer pool is severely undersized

Group buffer pool activity (if Db2 data sharing environment)

GROUP BP29	QUANTITY	/SECOND
GBP SYN.READ(XI) HIT RATIO(%)	99.48	N/A
SYN.READ(XI)-DATA RETURNED	1064.8K	300.80
SYN.READ(XI)-NO DATA RETURN	5596.00	1.58
WRITE FAILED-NO STORAGE	0.00	0.00

GBP write failures due to lack of storage should always be zero – non-zero value usually means GBP is way undersized

- If number of synch GBP reads due to cross-invalidation (XI) is > 10/sec (it's 302/sec in this case) then check the XI read hit ratio for the GBP – if that's less than 90% and CF LPAR has available memory, enlarge GBP to get ratio over 90% (ideally, > 95%)
- Rationale: XI generally happens due to updates of pages in GBP-dependent objects, and that means the changed pages were written to GBP – larger GBP means the pages are more likely to still be there when looked for
- GBP XI read hits are good: > 100X faster than disk read
- If < 10 GBP XI reads/sec, volume too low to care about GBP hit rate

EDM pool: DBD and package caches (package cache called "skeleton pool")

EDM POOL	QUANTITY	/SECOND
PAGES IN DBD POOL (ABOVE)	150.0K	N/A
FREE PAGES	49076.25	N/A
FAILS DUE TO DBD POOL FULL	0.00	0.00
PAGES IN SKEL POOL (ABOVE)	200.0K	N/A
FREE PAGES	75860.15	N/A
FAILS DUE TO SKEL POOL FULL	0.00	0.00
DBD REQUESTS	11264.9K	3182.17
DBD NOT FOUND	12.00	0.00
DBD HIT RATIO (%)	100.00	N/A
PT REQUESTS	39556.5K	11.2K
PT NOT FOUND	107.00	0.03
PT HIT RATIO (%)	100.00	N/A

Definitely want zeroes here

- For both DBD cache and package cache, you want REALLY high ratio of "requests" to "not found" - like thousands to one ("not found" requires read from directory)
- If that ratio is not really high for one of these caches, make the cache larger (in ZPARM, it's EDMDBDC for DBD cache, and EDM_SKELETON_POOL)

Data sets: are you hitting the DSMAX limit?

- DSMAX value set years ago may be too small today
  - Result can be a high rate of physical closure of Db2 data sets
    - Not good – data set physical open/close operations are relatively expensive
  - If you see a lot of data set close activity due to the DSMAX threshold being reached, increase DSMAX
    - 200,000 is max value, but practical limit probably less due to below-the-bar DBM1 virtual storage consumption – up to 70,000 probably OK in most cases
    - Don't go overboard – good value is just large enough to make rate of data set close actions due to threshold reached either zero or very small

OPEN/CLOSE ACTIVITY	QUANTITY	/SECOND
OPEN DATASETS - HWM	13698.00	N/A
OPEN DATASETS	13313.75	N/A
DSETS CLOSED-THRESH.REACHED	0.00	0.00

- Zero is good
- A few per hour is OK
- Several per minute not so good

DDF activity

GLOBAL DDF ACTIVITY	QUANTITY	/SECOND
DBAT/CONN QUEUED-MAX ACTIVE	0.00	0.00
CONN REJECTED-MAX CONNECTED	0.00	0.00
DBATS CREATED (A)	77.00	N/A
DISCON (POOL) DBATS REUSED (B)	5567.2K	N/A
CUR ACTIVE DBATS-BND DEALLC	0.00	N/A
HWM ACTIVE DBATS-BND DEALLC	0.00	N/A

If this value > 0, you hit MAXDBAT limit during the report interval - may want to increase MAXDBAT value (unless general-purpose engines were all busy during DDF transaction surge, in which case letting more work in via higher MAXDBAT value could overwhelm system's processing resources)

If this value > 0, you hit CONDBAT limit during report interval - probably want to increase CONDBAT value (most connections will be inactive at any given time, and "footprint" of an inactive connection is very small, and CPU cost to switch an inactive connection to active state when needed is very small)

- Using fields labeled as A and B, DBAT reuse rate is B/(A+B) – it's > 99.99% in this case
- If DBAT reuse rate < 99%, may want to increase value of POOLINAC in ZPARM (# of seconds DBAT can stay in DBAT pool without being reused)

These fields – current and high-water mark – pertain to high-performance DBATs (high-perf DBAT functionality requires some use of RELEASE(DEALLOCATE) packages with DDF, and requires DDF PKGREL setting to be BNDOPT)

Dynamic statement cache activity

DYNAMIC SQL STMT		QUANTITY	/SECOND
PREPARE REQUESTS		10987.6K	3103.85
FULL PREPARES		44863.00	12.67
SHORT PREPARES		10942.7K	3091.17
CACHE HIT RATIO	(%)	99.59	N/A

Short prepares indicate "hits" in dynamic statement cache – you want very high percentage of these, because they are much less costly than full prepares

If dynamic statement cache hit ratio is < 95%, and if LPAR's real storage not over-burdened (i.e., demand paging rate < 1/second), increase cache size (EDMSTMTC in ZPARM) to get hit rate over 95% (ideally, over 99%)

DYNAMIC SQL STMT		QUANTITY	/SECOND
PREPARE REQUESTS		290.9K	1.69
FULL PREPARES		16576.00	0.10
SHORT PREPARES		274.3K	1.59
SHORT PREPARES		274.3K	1.59
BASED ON CACHE		274.3K	1.59
BASED ON CATALOG		0.00	0.00
LOOK-UP IN CATALOG		16714.00	0.10
CACHE HIT RATIO	(%)	94.30	N/A
CACHE+CATALOG HIT RATIO	(%)	94.30	N/A

New with Db2 12 and Dynamic Plan Stability

Accounting report – the application view of Db2 for z/OS performance

A word about aggregation of data in accounting report

- With most Db2 monitors, default is to aggregate data in an accounting report by auth ID within plan name
  - That's often not a very useful aggregation of data
  - For ongoing Db2 subsystem monitoring and tuning, often better to have data in accounting report aggregated at the connection type level
    - Result: overall report will contain several sub-reports – one for each connection type used during report interval (e.g., one showing all DDF activity, one showing all CICS-Db2 activity, one showing all TSO attach activity, etc.)
  - To get aggregation at connection type level in OMEGAMON for Db2 accounting long report, specify ORDER(CONNTYPE) in report control statement
    - For another Db2 monitor, specification may be ”group by”

What is largest component of a Db2 subsystem's workload?

- "Largest" refers to aggregate CPU cost of SQL statement execution, and "component" refers to connection type (DDF, CICS, TSO attach, etc.)
- Easy to answer this question with information from accounting long report with data aggregated at connection type level
- Example of doing this for DRDA connection type (DDF):

```
SUBSYSTEM: DBPA   ORDER: CONNTYPE   INTERVAL FROM: 10/05/18 09:00
                                              TO: 10/05/18 10:00

CONNTYPE: DRDA

AVERAGE        DB2 (CL.2)          HIGHLIGHTS
------------   ----------          --------------------------
                                   #OCCURRENCES     :   863270 (C)
CP CPU TIME      0.000686 (A)

SE CPU TIME      0.000861 (B)
```

- Total SQL statement CPU cost for this connection type is (A+B) X C, or 1335 CPU seconds
- Class 2 CPU time is in-Db2 CPU time, and "average" is "per occurrence" (per accounting record)
- SE CPU time is "specialty engine" (i.e., zIIP time) - be sure to check this for any connection type (not just DRDA)

Is the Db2 workload CPU-constrained?

- Is Db2 workload CPU-constrained?
- Check “not-accounted-for time” in accounting long report
  - This is in-Db2 elapsed time that is not CPU time, not “identifiable” wait time (i.e., “class 3” time)
  - Not-accounted for time generally associated with wait-for-dispatch time, which itself can indicate a CPU constraint situation

```
CONNTYPE: CICS

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

- For higher-priority, transactional workload such as CICS or DDF, generally want to see not-accounted-for time < 10% of in-Db2 elapsed time
- Often not unusual, and OK, to see not-accounted-for time > 10% of in-Db2 elapsed time for lower-priority batch workload

What if not-accounted-for time is too high?

- In other words, what if not-accounted-for time is > 10% of in-Db2 elapsed time for a higher-priority transactional workload (e.g., CICS or DDF)?
  - Could be a matter of adjusting WLM policy for the z/OS LPAR
    - If, for example, not-accounted-for time is 20% for DDF workload and 5% for CICS workload, and your organization considers both workloads to be equally important, does current WLM policy reflect that equivalence of importance?
  - If not-accounted for time is overly high for all components of your Db2 workload, quite likely that your system is CPU-constrained
    - Check “average MVS busy” for LPAR’s general-purpose engines in RMF CPU Activity report – if above 95%, wait-for-dispatch time likely to be a problem
    - In that case, if more capacity can’t be added, might be able to reduce CPU utilization through various system and/or application tuning actions

What zIIP offload are you getting for the DDF workload?

- zIIP offload for execution of SQL statements associated with DDF-using applications can be as high as 60%
  - Anything in range of 55-60% for DDF SQL zIIP offload is very good – what is the figure for your system?

```
SUBSYSTEM: DBPA   ORDER: CONNTYPE   INTERVAL FROM: 10/05/18 09:00
                                               TO: 10/05/18 10:00
CONNTYPE: DRDA

AVERAGE        DB2 (CL.2)
------------   ----------

CP CPU TIME      0.000686 (A)

SE CPU TIME      0.000861 (B)
```

zIIP offload for DDF workload is B/(A+B) - it's 56% in this case

What if zIIP offload for DDF workload is lower than desired?

- In other words, what if zIIP offload for DDF workload is well below 55%?
  - One possible cause: many calls of external Db2 stored procedures (i.e., stored procedure programs written in a language such as COBOL) by DDF applications
    - To be up to 60% zIIP-eligible, a SQL statement has to execute under a preemptable SRB in the DDF address space – an external stored procedure always runs under its own TCB in a stored procedure address space
    - Native SQL procedure always executes under task of its caller, and if caller is DDF-using application then task will be preemptable SRB in DDF address space, so native SQL procedure up to 60% zIIP-eligible when called via DDF
  - Another possible cause of lower-than-desired zIIP offload for DDF SQL: zIIP constraint situation (see next slide)

Is your system zIIP-constrained?

- Check the zIIP spill-over percentage, using information in the DRDA section of Db2 accounting long report with data aggregated at connection type level

```
SUBSYSTEM: DBPP      ORDER: CONNTYPE

CONNTYPE: DRDA

MEASURED/ELIG TIMES   APPL (CL1)
-------------------   ----------

CP CPU TIME              0.000762
 ELIGIBLE FOR SECP       0.000001  (A)

SE CPU TIME              0.000951  (B)
```

- Field B is average zIIP CPU time for the DDF workload
- Field labeled A is average CPU time for the DDF workload that was zIIP eligible but ended up getting consumed on a general-purpose engine (that happens when zIIP-eligible work is ready for dispatch but zIIP engines are busy with other work)
- Rate at which zIIP-eligible work "spills over" to general-purpose engines is A/(A+B) – it's 0.1% in this case
- A zIIP spill-over rate below 1% is great
- A zIIP spill-over rate > 5% is a sign that the LPAR should have more zIIP processing capacity

What is you system is zIIP-constrained?

- In other words, what if zIIP spill-over rate is greater than 5%?
  - Best responsive action is to add another zIIP engine (or engines) to LPAR
    - The more zIIP engines an LPAR has, the higher the utilization of those engines can be without driving the zIIP spill-over rate too high
    - Example: if 1 zIIP engine running at 50% busy, zIIP spill-over will likely be high; if 5 zIIP engines running at 50% busy, zIIP spill-over will likely be low
  - If another zIIP engine (or engines) cannot be added to LPAR, consider running zIIPs in SMT2 mode to reduce zIIP spill-over
    - SMT2: two pieces of work can be concurrently processed by one zIIP “core”
    - In SMT2 mode, processing of one piece of work not as fast as when zIIP engine is running in “uni-thread” mode – effective increase in throughput generally around 25-40%

How is your Db2 I/O performance?

- Average Db2 synchronous read times are getting to be really low (here, 204 microseconds)
- A time > 1 millisecond would concern me
  - Could indicate a performance issue – perhaps contention in the disk subsystem, or at a control block level, or at the processor level (overloaded processors could cause delay in re-dispatching suspended task following read I/O completion)

```
SUBSYSTEM: DBPP     ORDER: CONNTYPE

CONNTYPE: CICS

HIGHLIGHTS
--------------------------

SYNCH I/O AVG.   : 0.000204
```

Boosting application efficiency via thread reuse

- Tends to be most relevant for CICS-Db2 and IMS-Db2 workloads
  - CICS-Db2: drive thread reuse with protected threads for high-volume trans
  - IMS-Db2: boost thread reuse with WFI and/or pseudo-WFI regions
- CPU benefit of thread reuse maximized when high-use packages executed via reused threads are bound with RELEASE(DEALLOCATE)

```
SUBSYSTEM: DBPP      ORDER: CONNTYPE

CONNTYPE: CICS

NORMAL TERM.        AVERAGE      TOTAL
---------------   --------   --------
NEW USER               0.00          0
DEALLOCATION           0.00      21094

RESIGNON               1.00    4637747
```

- NEW USER: thread was reused, auth ID changed
- DEALLOCATION: thread was not reused
- RESIGNON: thread was reused, auth ID did not change
- In this example, thread reuse rate is 4,637,747 / (4,637,747 + 21,094) = 99.5%

Using accounting reports to gauge impact of tuning actions

- Suppose you take a performance tuning action, such as page-fixing a buffer pool, or implementing high-performance DBATs, or enlarging the dynamic statement cache – how do you measure the CPU effect?
  - Here's how: generate accounting long reports, with data aggregated by connection type, for one-hour period before and after the tuning action
    - Ideally, same hour of same day of week (e.g., 9-10 AM on Monday)
  - In those before and after reports, for the workload components (i.e., connection types) of interest, compare average class 2 CPU time (general-purpose + zIIP CPU time)
  - If that time is lower in the "after" report, you did good

Thank you!

Mark Rader
mrader@us.ibm.com