



# Optimizing DB2 System Performance Using DB2 Statistics Trace

*John Campbell, DB2 for z/OS Development*

*Florence Dubois, DB2 for z/OS Development*



**GUIDE Share France**

Une Association Indépendante d'Utilisateurs IBM

Réunion du Guide DB2 pour z/OS France  
Vendredi 21 novembre 2008  
Tour Manhattan BMC, Paris-La Défense



# Optimizing DB2 System Performance Using DB2 Statistics Trace (première partie)

*John Campbell, DB2 for z/OS Development*

*Florence Dubois, DB2 for z/OS Development*



**GUIDE Share France**

Une Association Indépendante d'Utilisateurs IBM

Réunion du Guide DB2 pour z/OS France  
Vendredi 21 novembre 2008  
Tour Manhattan BMC, Paris-La Défense

# Objectives

- Focus on key areas
  - Dataset activity, buffer pool and GBP, lock/latch contention, system address space CPU, EDM pool, DBM1 virtual and real storage
- Identify the key performance indicators to be monitored
- Provide rules-of-thumb to be applied
  - Typically expressed in a range, e.g.  $< X-Y$ 
    - If  $< X$ , no problem - GREEN
    - If  $> Y$ , need further investigation and tuning - RED
    - Boundary condition if in between - AMBER
      - Investigate with more detailed tracing and analysis when time available
- Provide tuning advice for common problems



# Topics

- Statistics Time Interval
- Dataset Open/Close
- Buffer Pools and Group Buffer Pools
- Lock/Latch
- Log
- EDM Pool
- System CPU time
- DBM1 Virtual Storage and Real Storage
- Sort



# Statistics Time Interval

- Recommendation to set to 1 minute
  - Only 1440 intervals per day
  - Highly valuable / essential to study evolutionary trend that led to complex system problems, e.g. slowdowns etc.
  - Very small compared to Accounting Trace data volume
  - Very small increase in total SMF data volume
- Recommendation to copy SMF 100 and 102 records to keep separately
  - SMF 100 and 102 records represent relatively small amount of total SMF data volume
  - Improved elapsed time performance to post process



# Dataset Open/Close

| TOTAL BUFFERPOOLS       | TOTAL | AVG/SEC |
|-------------------------|-------|---------|
| NUMBER OF DATASET OPENS | 18    | 0.30    |

| OPEN/CLOSE ACTIVITY         | TOTAL | AVG/SEC |
|-----------------------------|-------|---------|
| OPEN DATASETS - HWM         | 18001 |         |
| OPEN DATASETS               | 17803 |         |
| DSETS CLOSED-THRESH.REACHED | 100   | 1.64    |

- Frequent dataset opens are typically accompanied by high DBM1 TCB time and/or Acctg Class 3 Dataset Open time
- Rule-of-Thumb:
  - NUMBER OF DATASET OPENS < 0.1 to 1/sec
- Could be caused by hitting DSMAX too frequently
  - Increase DSMAX in small incremental steps and study the effects on DBM1 virtual storage below the 2GB bar



# Dataset Open/Close ...

- Possible side effects of pseudo close activity (RW->RO switching)
  - Frequent dataset close and re-open
  - Expensive SYSLGRNX processing
  - Growth in SYSLGRNX pageset
  - Ping-pong in and out of GBP dependency
- Rule-of-Thumb
  - #DSETS CONVERTED R/W -> R/O < 10-15 per minute
- Recommendations
  - Take frequent system checkpoints
    - Set CHKFREQ=2-5 (minutes)
  - Adjust PCLOSEN/T to avoid too frequent pseudo closes
  - Use CLOSE(YES) as a design default



# CLOSE (YES) versus CLOSE (NO) data sets

- In V8 (and DB2 V7 after PQ69741), there is a significant difference between CLOSE(NO) and CLOSE(YES) for active data sharing
  - When there is GBP dependency (1 updater, 1 reader)

## CLOSE(YES)

When the next PCLOSET/N occurs for the reader, physical close of the object

(+) Data set ceases to be GBP-dependent. The remaining member can update the data set in its local buffer pool.

(-) Physical open on next access from reader

## CLOSE(NO)

*Old behaviour.* Same as CLOSE(YES)

*New behaviour.* When the next PCLOSET/N occurs for the reader, no physical close

(+) No physical open on next access from reader

(-) Data set stays GBP-dependent (SIX to readers IS) until it is stopped, or DB2 is stopped normally, or DSMAX is hit and all CLOSE(YES) objects have been closed



# Buffer Pool Tuning

- PREF DISABLED – NO BUFFER
  - 90% of buffers not available for steal, or running out of sequential buffers (VPSEQT with 80% default)
  - If non-zero
    - Increase BP size, and/or
    - Reduce deferred write threshold (VDWQT, DWQT)
    - Increase system checkpoint frequency (reduce CHKFREQ)
  - Can be much higher if prefetch intentionally disabled via VPSEQT=0
    - Interesting option for data in-memory BP
      - Avoid the overhead of scheduling prefetch engines when data is already in BP
    - No problem in that case
    - Consider using FIFO instead of LRU for PGSTEAL



# Buffer Pool Tuning ...

- DATA MGMT THRESHOLD
  - 95% of buffers not available for steal
  - Getpage can be issued for each row sequentially scanned on the same page -> potential large CPU time increase
  - If non-zero
    - Increase BP size, and/or
    - Reduce deferred write threshold (VDWQT, DWQT)
    - Increase system checkpoint frequency (reduce CHKFREQ)



# Buffer Pool Tuning ...

- PAGE-IN FOR READ OR WRITE
  - If BP is not entirely backed up by real storage there can be high PAGE-IN for READ or WRITE IO
    - BP may be too large for available real storage
    - Also by BP expansion via ALTER BPSIZE
  - Rule-of-Thumb in a steady-state:
    - PAGE-IN for READ <1-5% of pages read
    - PAGE-IN for WRITE <1-5% of pages written



# Buffer Pool Tuning ...

- Multiple buffer pools recommended
  - Dynamic performance monitoring much cheaper and easier than with performance trace
    - DISPLAY BPOOL for online monitoring
      - Dataset statistics via -DISPLAY BPOOL LSTATS (IFCID 199)
    - Useful for access path monitoring
  - Dynamic tuning
    - Full exploitation of BP tuning parameters for customised tuning
      - ALTER BPOOL is synchronous and effective immediately, except for Buffer pool contraction because of wait for updated pages to be written out
    - Prioritisation of buffer usage
    - Reduced BP latch contention
  - Minimum of 4 BPs: catalog/directory, user index, user data, work file



## Long Term Page Fix for Buffer Pool with Frequent I/Os

- DB2 BPs have always been strongly recommended to be backed up 100% by real storage
  - To avoid paging which occurs even if only one buffer is short of real storage because of LRU buffer steal algorithm
- Given 100% real storage, might as well page fix each buffer just once to avoid the repetitive cost of page fix and free for each and every I/O
  - New option: ALTER BPOOL(name) PGFIX(YES|NO)
    - Requires the BP to go through reallocation before it becomes operative
      - Means a DB2 restart for BP0
  - Up to 8% reduction in overall IRWW transaction CPU time



## Long Term Page Fix for Buffer Pool with Frequent I/Os ...

- Recommended for BPs with high buffer i/o intensity = [pages read + pages written] / [number of buffers]
  - Relative values across all BPs

| BPID  | VPSIZE | Read SPF | Read LPF | Read DPF | Read - Total | Written | IO Intensity |
|-------|--------|----------|----------|----------|--------------|---------|--------------|
| BP0   | 40000  | 0        | 0        | 6174     | 9134         | 107     | 0.2          |
| BP1   | 110000 | 5185     | 0        | 1855     | 19451        | 6719    | 0.2          |
| BP2   | 110000 | 19833    | 11256    | 9380     | 80951        | 5763    | 0.8          |
| BP3   | 75000  | 6065     | 0        | 14828    | 44865        | 7136    | 0.7          |
| BP4   | 80000  | 45933    | 3926     | 50261    | 122993       | 3713    | 1.6          |
| BP5   | 200000 | 0        | 0        | 0        | 0            | 0       | 0.0          |
| BP8K0 | 32000  | 11       | 0        | 11       | 31           | 169     | 0.0          |
| BP32K | 2000   | 873      | 0        | 6415     | 7981         | 38      | 4.0          |

*In this example: Best candidates would be BP32K, BP4, BP3, BP2  
No benefit for BP5 (data in-memory)*



# GBP Read Tuning

- Local BP search -> GBP search -> DASD I/O
- SyncRead (NF) = LBP (Local Buffer Pool) miss
- SyncRead (XI) = LBP hit with cross-invalidated buffer
  - Two reasons for cross invalidations
    - Directory entry reclaims – condition you want to tune away from
      - Consider XES AUTO ALTER
    - Perfectly normal condition in an active-active data sharing environment
  - Most data should be found in GBP
  - If not, GBP may be too small
- Consider enlarging GBP if
  - SyncReadXI miss ratio =  $(\text{SyncReadXI-nodata} / \text{SyncReadXI}) > 10\%$



# GBP Read Tuning ...

| GROUP BUFFERPOOL TOTAL      | AVG/COMMIT | TOTAL |
|-----------------------------|------------|-------|
| SYN.READ(XI)-DATA RETURNED  | 1.05       | 183K  |
| SYN.READ(XI)-NO DATA RETURN | 0.09       | 15K   |

- Example of GBP size check
  - SyncRead (XI) no data = 15K, SyncRead (XI) = 183K+15K
  - SyncRead (XI) miss ratio =  $15K / (183K + 15K) = 7.6\%$
  - Therefore, there is no GBP size shortage here

| GROUP BUFFERPOOL TOTAL        | AVG/COMMIT | TOTAL |
|-------------------------------|------------|-------|
| REG.PAGE LIST (RPL) REQUEST   | 0.84       | 147K  |
| NUMBER OF PAGES RETR.FROM GBP | 3.10       | 540K  |

- Register Page List request = sequential, list, or dynamic prefetch request
  - 1 CF request for each page found in GBP and 1 request for the remainder



# GBP Read Tuning ...

- GBPCACHE CHANGED is the default and the general recommendation
- GBPCACHE CHANGED|SYSTEM recommended for LOBs
  - SYSTEM is the default for LOBs
  - Do not use CHANGED for a LOB table space defined with LOG NO
- GBPCACHE NONE
  - Saves data transfer in GBP write and GBP access for castout
  - But updated pages must be written to DASD at or prior to commit
    - Set VDWT=0 to promote continuous deferred write
      - Minimises response time elongation at commit
  - Useful when an updated page is rarely re-accessed
    - e.g. GBP read/write ratio < 1%
    - Batch update sequentially processing a large table
    - Primarily to reduce CF usage



# GBP Write Tuning

| GROUP BUFFERPOOL TOTAL     | AVG/COMMIT | TOTAL |
|----------------------------|------------|-------|
| CHANGED PGS SYNC.WRTN      | 6.24       | 1087K |
| CHANGED PGS ASYNC.WRTN     | 0.06       | 11K   |
| PAGES CASTOUT              | 3.43       | 598K  |
| UNLOCK CASTOUT             | 0.13       | 23K   |
| CASTOUT CLASS THRESHOLD    | 0.01       | 1232  |
| GROUP BP CASTOUT THRESHOLD | 0          | 232   |

- Pages Sync Written to GBP - force write at
  - Commit
  - P-lock negotiation
- Pages Async Written to GBP
  - Deferred write
  - System checkpoint
- In GBP write, corresponding log record must be written first



# GBP Write Tuning ...

- Pages Castout / Unlock Castout is a good indicator of castout efficiency
  - Unlock Castout included in #Async Writes in GBP-dependent LBP stats
  - Pages Castout included in Pages Written in GBP-dependent LBP stats
- GBP castout thresholds are similar to local BP deferred write thresholds
  - Encourage Class\_castout (CLASST) threshold by lowering its value as it is more efficient than GBP\_castout threshold (notify to pageset/partition castout owner)
    - CLASST threshold check by GBP write
    - GBPOOLT threshold check by GBP castout timer (10sec default)
  - Default thresholds lowered in V8

|                          | V7  | V8  |
|--------------------------|-----|-----|
| VDWQT (dataset level)    | 10% | 5%  |
| DWQT (buffer pool level) | 50% | 30% |
| CLASST (Class_castout)   | 10% | 5%  |
| GBPOOLT (GBP_castout)    | 50% | 30% |



# GBP Write Tuning ...

- WRITE FAILED-NO STORAGE
  - GBP is full and no stealable data page
  - Potential outage if pages added to LPL list
  - Keep Write Failed < 1% of pages written by
    - Using bigger GBP, and/or
    - Smaller castout thresholds, and/or
    - Smaller GBP checkpoint timer



# Lock Tuning

- Lock avoidance may not be working effectively if Unlock requests/commit is high, e.g. >5/commit
  - BIND option ISOLATION(CS) with CURRENTDATA(NO) could reduce # Lock/Unlock requests dramatically
  - High Unlock requests/commit could also be possible from
    - Compressed or VL row update
      - Lock/Unlock of pointer if the update results in new row which can not fit in that page
    - Lock/Unlock in Insert to unique index when pseudo-deleted entries exist
    - Both can be eliminated by REORG
- V8 improvements which help to reduce locking cost
  - Non-cursor SELECT to try to avoid lock and unlock in ISOLATION(CS) even with CURRENTDATA(YES)
  - Overflow lock avoidance when an update of variable length row in data page results in new row which can not fit in that page



# Lock Tuning ...

- MAX PG/ROW LOCKS HELD from Accounting trace is a useful indicator of commit frequency
  - Page or row locks only
  - AVERAGE is for average of MAX, TOTAL is for max of MAX (of Accounting records)
    - So if transaction A had max. locks of 10 and transaction B had 20, then
      - AVERAGE (avg. of max.) = 15
      - TOTAL (max. of max.) = 20
  - In general, try to issue Commit to keep max. locks held below 100

| LOCKING                      | AVERAGE     | TOTAL      |
|------------------------------|-------------|------------|
| TIMEOUTS                     | 0.00        | 0          |
| DEADLOCKS                    | 0.00        | 0          |
| ESCAL.(SHARED)               | 0.00        | 0          |
| ESCAL.(EXCLUS)               | 0.00        | 0          |
| <b>MAX PG/ROW LOCKS HELD</b> | <b>2.37</b> | <b>576</b> |
| LOCK REQUEST                 | 40.99       | 949801     |
| UNLOCK REQUEST               | 6.12        | 141692     |
| QUERY REQUEST                | 0.00        | 0          |
| CHANGE REQUEST               | 1.32        | 30509      |
| OTHER REQUEST                | 0.00        | 0          |
| LOCK SUSPENSIONS             | 0.01        | 140        |
| IRLM LATCH SUSPENSIONS       | 0.13        | 3041       |
| OTHER SUSPENSIONS            | 0.00        | 4          |
| TOTAL SUSPENSIONS            | 0.14        | 3185       |



# Lock Tuning ...

- As a general rule, start with LOCKSIZE PAGE
  - If high deadlock or timeout, consider LOCKSIZE ROW
  - Not much difference between one row lock and one page lock request
  - However, the number of IRLM requests issued can be quite different
    - No difference in a random access
    - In a sequential scan,
      - No difference if 1 row per page (MAXROWS=1) or ISOLATION(UR)
      - Negligible difference if ISOLATION(CS) with CURRENTDATA(NO)
      - Bigger difference if ISOLATION(RR|RS), or sequential Insert, Update, Delete
      - Biggest difference if ISOLATION(CS) with CURRENTDATA(YES) and many rows per page
  - In data sharing, additional data page P-locks are acquired when LOCKSIZE ROW is used



# Lock Tuning ...

- DBD is locked
  - X by SQL DDL such as CREATE/DROP/ALTER (TS, TBL, IX) and some utility
  - S by BIND, DB2 utility, dynamic SQL without caching
- To minimise lock contention on DB2 Catalog/Directory pages and DBD
  - Assign unique authid and a private database to each user
    - Each Catalog page contains only one value for dbid or authid
  - Be careful about user query on Catalog tables
    - e.g. SQL LOCK TABLE or ISOLATION(RR)
  - One tablespace per database to minimise DBD lock contention, if necessary
    - Smaller DBD also minimises DBD page updates and logging volume
      - Example: Create Index with 1000 page DBD results in 2000 pages logged
    - Group all SQL DDLs within the same database in the same commit scope
      - Only one delete/insert of DBD rather than one per SQL DDL



# Lock Tuning ...

- In V8, 64bit IRLM is supported
  - PC=YES is forced
  - Reduced requirement for ECSA
- Make sure of high IRLM dispatching priority
  - Use WLM service class SYSSTC
- IRLM trace can add up to 25%
  - Can also increase IRLM latch contention
- MODIFY irlmproc,SET,DEADLOK= or TIMEOUT= to dynamically change deadlock and timeout frequency

|                | TIMEOUT      | DEADLOK     |
|----------------|--------------|-------------|
| Range allowed  | 1 to 3600sec | 0.1 to 5sec |
| Default        | 60sec        | 1sec        |
| Recommendation | 30sec        | 0.5sec      |



# Lock Tuning ...

- ZPARMS

- RRULOCK – U-lock on SELECT FOR UPDATE for ISOLATION(RR|RS)
  - To minimise deadlocks
  - Default is S-lock
- XLKUPDLT – X-lock for searched UPDATE/DELETE
  - Take X-lock immediately on qualifying rows/pages instead of S|U-lock
  - Good if most accessed objects are changed
  - V8 PQ98172 introduces a new option: TARGET
    - X-lock only for the specific table that is targeted by UPDATE or DELETE
    - S- or U-lock when scanning for rows/pages of other tables referenced by the query
- SKIPUNCI – Skip uncommitted inserts for ISOLATION(CS|RS)
  - New parameter in V8
  - Restricted to row level locking only



# IRLM Latch Contention

| LOCKING ACTIVITY         | AVG/COMMIT | TOTAL   |
|--------------------------|------------|---------|
| LOCK REQUESTS            | 31.94      | 6258309 |
| UNLOCK REQUESTS          | 7.39       | 1447822 |
| CHANGE REQUESTS          | 1.03       | 201435  |
| SUSPENSIONS (LOCK ONLY)  | 0.02       | 4417    |
| SUSPENSIONS (IRLM LATCH) | 1.33       | 260416  |
| SUSPENSIONS (OTHER)      | 0.05       | 9401    |

- Rule-of-Thumb:
  - #IRLM latch contention should be less than 1-5% of Total #IRLM Requests
- Example:
  - #IRLM latch contention = SUSPENSIONS (IRLM LATCH) = 1.33
  - #IRLM Requests = LOCK+UNLOCK+CHANGE = 31.94+7.39+1.03 = 40.36
  - #IRLM latch contention Rate =  $1.33 * 100 / 40.36 = 3.3\%$



# Data Sharing Lock Tuning

| DATA SHARING LOCKING        | AVG/COMMIT | TOTAL   |
|-----------------------------|------------|---------|
| SYNCH.XES - LOCK REQUESTS   | 8.62       | 1688167 |
| SYNCH.XES - CHANGE REQUESTS | 0.33       | 63924   |
| SYNCH.XES - UNLOCK REQUESTS | 8.20       | 1606128 |
| SUSPENDS - IRLM GLOBAL CONT | 0.15       | 29541   |
| SUSPENDS - XES GLOBAL CONT. | 0.40       | 78982   |
| SUSPENDS - FALSE CONTENTION | 0.08       | 15186   |

- Rule-of-Thumb:

- Global Contention Rate should be less than 3-5% of #XES IRLM Requests

- Example:

- #Global Cont. = SUSPENDS - IRLM+XES+FALSE = 0.15+0.4+0.08 = 0.63
- #XES IRLM Req. = SYNCH. XES - LOCK+CHANGE+UNLOCK + SUSPENDS - IRLM+XES+FALSE = 8.62+0.33+8.20+0.15+0.4+0.08 = 17.78
- Global Contention Rate =  $0.63 * 100 / 17.78 = 3.54\%$



# Data Sharing Lock Tuning ...

- Notes

- IRLM Cont. = IRLM resource contention
- XES Cont. = XES-level resource cont. as XES only understands S|X mode
  - e.g. member 1 asking for IX and member 2 for IS
  - Big relief in with Locking Protocol 2 when enabled after entry to V8(NFM) but increased overhead for pageset/partitions opened for RO on certain members, e.g.
    - -START DATABASE(...) SPACENAM(...) ACCESS(RO)
    - SQL LOCK TABLE statement
    - LOCKSIZE TABLESPACE or LOCKSIZE TABLE for segmented tablespace
    - Table scan with Repeatable Read isolation level
    - Lock escalation occurs
- False Cont. = false contention on lock table hash anchor point
  - Could be minimized by increasing the number of Lock entries in the CF Lock Table



# Data Sharing Lock Tuning ...

| DATA SHARING LOCKING        | AVG/COMMIT | TOTAL |
|-----------------------------|------------|-------|
| PSET/PART P-LCK NEGOTIATION | 0.01       | 1056  |
| PAGE P-LOCK NEGOTIATION     | 0.06       | 11650 |
| OTHER P-LOCK NEGOTIATION    | 0.01       | 1054  |

- Rule-of-Thumb:
  - #P-lock Negotiation should be less than 3-5% of #XES IRLM requests
- Example:
  - #P-lock Negotiation =  $0.01+0.06+0.01 = 0.08$
  - #XES IRLM Req. = 17.78 (see previous example)
  - #P-lock Negotiation Rate =  $0.08*100 / 17.78 = 0.5\%$



# Data Sharing Lock Tuning ...

- Notes

- P-lock contention and negotiation can cause IRLM latch contention, page latch contention, asynchronous GBP write, active log write, GBP read
- Breakdown by page P-lock type in GBP statistics
- Other P-lock negotiation for
  - Index tree P-lock (See LC06 contention in Latch section)
  - Castout P-lock
  - SKCT/SKPT P-lock

| GROUP BUFFERPOOL TOTAL | AVG/COMMIT |
|------------------------|------------|
| PAGE P-LOCK LOCK REQ   | 1.78       |
| - SPACE MAP PAGES      | 0.46       |
| - DATA PAGES           | 0.04       |
| - INDEX LEAF PAGES     | 1.28       |
| PAGE P-LOCK UNLOCK REQ | 1.71       |
| PAGE P-LOCK LOCK SUSP  | 0.15       |
| - SPACE MAP PAGES      | 0.05       |
| - DATA PAGES           | 0.00       |
| - INDEX LEAF PAGES     | 0.10       |
| PAGE P-LOCK LOCK NEG   | 0.06       |
| - SPACE MAP PAGES      | 0.03       |
| - DATA PAGES           | 0.00       |
| - INDEX LEAF PAGES     | 0.03       |



# Data Sharing Lock Tuning ...

- To reduce page p-lock and page latch contention on space map pages during heavy inserts into GBP-dependent tablespace
  - TRACKMOD NO
  - MEMBER CLUSTER
    - Option only available for partitioned tablespace
      - Switching to partitioned will likely result in extra getpages for true varying length rows and fixed length compressed
    - Increases the number of space map pages (199 data pages per space map instead of 10K per space map)
    - Only when MEMBER CLUSTER is used, LOCKSIZE ROW has the potential to reduce page p-lock and page latch contention on data pages
      - Without MEMBER CLUSTER, LOCKSIZE ROW will result in excessive page p-lock contention on data pages and space map pages, in addition to the extra locking protocol that comes with taking page p-lock



# Data Sharing Lock Tuning ...

- Effective lock avoidance is very important in data sharing
  - Long running unit of works can reduce the effectiveness of lock avoidance by stopping the GCLSN (global commit log sequence number) value from moving forward
- 50% of CLAIM\_REQ can be used as a very rough estimate of # tablespace/partition locks acquired when effective thread reuse with RELEASE(COMMIT), or no thread reuse
  - CLAIM\_REQ assumed once for index and once for tablespace/partition for a very rough estimation
  - These tablespace/partition locks can be eliminated with effective thread reuse with use of RELEASE(DEALLOCATE)



# Thread Reuse

- Performance opportunity for high volume simple transactions
- Thread reuse can be monitored using the Accounting Trace
  - $(\#COMMITTS-DEALLOCATION)*100/\#COMMITTS$  is a good estimation of the level of thread reuse

| NORMAL TERM. | AVERAGE | TOTAL  |
|--------------|---------|--------|
| NEW USER     | 0.00    | 0      |
| DEALLOCATION | 0.33    | 212799 |
| RESIGNON     | 0.67    | 429710 |

| HIGHLIGHTS   |        |
|--------------|--------|
| #OCCURRENCES | 642509 |
| #COMMITTS    | 974827 |

- In this example,  $(974827- 212799)*100/ 974827= 78\%$  of thread reuse



# Internal DB2 Latch Contention/Second

| LATCH CNT | /SECOND | /SECOND | /SECOND | /SECOND |
|-----------|---------|---------|---------|---------|
| LC01-LC04 | 0.00    | 0.00    | 0.00    | 0.00    |
| LC05-LC08 | 0.00    | 75.62   | 0.03    | 0.00    |
| LC09-LC12 | 0.00    | 2.47    | 0.00    | 4.52    |
| LC13-LC16 | 0.02    | 676.00  | 0.00    | 3.27    |
| LC17-LC20 | 0.00    | 0.00    | 105.50  | 0.00    |
| LC21-LC24 | 0.00    | 0.00    | 9.37    | 4327.87 |
| LC25-LC28 | 4.18    | 0.00    | 3.74    | 0.00    |
| LC29-LC32 | 0.00    | 2.26    | 4.46    | 2.71    |

- Rule-of-Thumb: Try to keep latch contention rate < 1K-10K per second
  - Disabling Acct Class 3 trace can help to reduce CPU time when high latch contention
- Typical high latch contention classes highlighted
  - LC06 = Index split latch
  - LC14 = Buffer pool LRU and hash chain latch
  - LC19 = Log latch
  - LC24 = Prefetch latch or EDM LRU chain latch



## Internal DB2 Latch Contention/Second ...

- LC06 for index tree P-lock by index split
  - Reduce index split via distributed freespace tuning, minimum index key size especially if unique index, NOT PADDED index for large varchar columns (V8)
  - A number of index splits in LEAFNEAR/FAR in SYSINDEXPART and RTS REORGLLEAFNEAR/FAR
  - X'46' in IFCID 57 performance trace
  - X'FE' index tree latch in non data sharing not in Class 6



# Internal DB2 Latch Contention/Second ...

- LC14 Buffer Pool latch
  - If many tablespaces and indexes, assign to separate buffer pools with an even Getpage frequency
  - If objects bigger than buffer pool, try enlarging buffer pool if possible
  - If high LC14 contention, use buffer pool with at least 3000 buffers
  - Use FIFO rather than LRU buffer steal algorithm if there is no read I/O, i.e. object(s) entirely in buffer pool
    - LRU = Least Recently Used buffer steal algorithm (default)
    - FIFO = First In First Out buffer steal algorithm
      - Eliminates a need to maintain LRU chain which in turn
        - » Reduces CPU time for LRU chain maintenance
        - » Reduces CPU time for LC14 contention processing



# Internal DB2 Latch Contention/Second ...

- LC19 Log latch
  - Minimise #log records created via
    - LOAD RESUME/REPLACE with LOG NO instead of massive INSERT/UPDATE/DELETE
    - Segmented tablespace if mass delete occurs
  - Increase size of output log buffer if non-zero unavailable count
    - When unavailable, first agent waits for log write
    - All subsequent agents wait for LC19
  - Reduce size of output log buffer if non-zero output log buffer paging
    - See Log Statistics section



# Internal DB2 Latch Contention/Second ...

- LC24 latch
  - X'18' in IFCID 57 performance trace for EDM LRU latch
    - Use EDMBFIT zparm of NO
    - Thread reuse with RELEASE DEALLOCATE instead of RELEASE COMMIT for frequently executed packages
  - X'38' for prefetch scheduling
    - Higher contention possible with many concurrent prefetches
    - Disable dynamic prefetch if unnecessary by setting VPSEQT=0
    - Use more partitions (one x'38' latch per data set)



# Disclaimer

© Copyright IBM Corporation [current year]. All rights reserved.

*U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.*

**THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE.**

IBM, the IBM logo, ibm.com, and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml)





# Optimizing DB2 System Performance Using DB2 Statistics Trace (seconde partie)

*John Campbell, DB2 for z/OS Development*

*Florence Dubois, DB2 for z/OS Development*



**GUIDE Share France**

Une Association Indépendante d'Utilisateurs IBM

Réunion du Guide DB2 pour z/OS France  
Vendredi 21 novembre 2008  
Tour Manhattan BMC, Paris-La Défense

# Objectives

- Focus on key areas
  - Dataset activity, buffer pool and GBP, lock/latch contention, system address space CPU, EDM pool, DBM1 virtual and real storage
- Identify the key performance indicators to be monitored
- Provide rules-of-thumb to be applied
  - Typically expressed in a range, e.g.  $< X-Y$ 
    - If  $< X$ , no problem - GREEN
    - If  $> Y$ , need further investigation and tuning - RED
    - Boundary condition if in between - AMBER
      - Investigate with more detailed tracing and analysis when time available
- Provide tuning advice for common problems



# Topics

- Statistics Time Interval
- Dataset Open/Close
- Buffer Pools and Group Buffer Pools
- Lock/Latch
- Log
- EDM Pool
- System CPU time
- DBM1 Virtual Storage and Real Storage
- Sort



# Log Statistics

| LOG ACTIVITY                | AVG/COMMIT | TOTAL    |
|-----------------------------|------------|----------|
| READS SATISFIED-OUTPUT BUFF | 2.3        | 1951.0K  |
| READS SATISFIED-ACTIVE LOG  | 33.17      | 28132.0K |
| READS SATISFIED-ARCHIVE LOG | 0.00       | 0        |

- “READ FROM ...” represents # of log records which are sent to Data Manager by Log Manager for log apply e.g., UNDO or REDO
  - Read from output log buffer is most efficient
  - Read from archive log dataset is least efficient



# Log Statistics

| LOG ACTIVITY                | AVG/COMMIT | TOTAL   |
|-----------------------------|------------|---------|
| UNAVAILABLE OUTPUT LOG BUFF | 0.00       | 0.00    |
| OUTPUT LOG BUFFER PAGED IN  | 0.00       | 0.00    |
| LOG RECORDS CREATED         | 14.58      | 4031.7K |
| LOG CI CREATED              | 0.68       | 188.2K  |
| LOG WRITE I/O REQ (LOG1&2)  | 2.06       | 569.7K  |
| LOG CI WRITTEN (LOG1&2)     | 2.35       | 649.9K  |
| LOG RATE FOR 1 LOG (MB)     | N/A        | 2.12    |

- Output Log Buffer size
  - Increase if #UNAVAIL OUTPUT LOG BUF > 0
    - More output log buffer space may help in RECOVER, Restart, DPROP, LOB but watch out for page in activity
  - Decrease if #OUTPUT LOG BUFFER PAGED IN > 1-5% of LOG RECORDS CREATED
- Approximate average log record size
  - = (LOG CIs CREATED \* 4KB)/(LOG RECORDS CREATED)



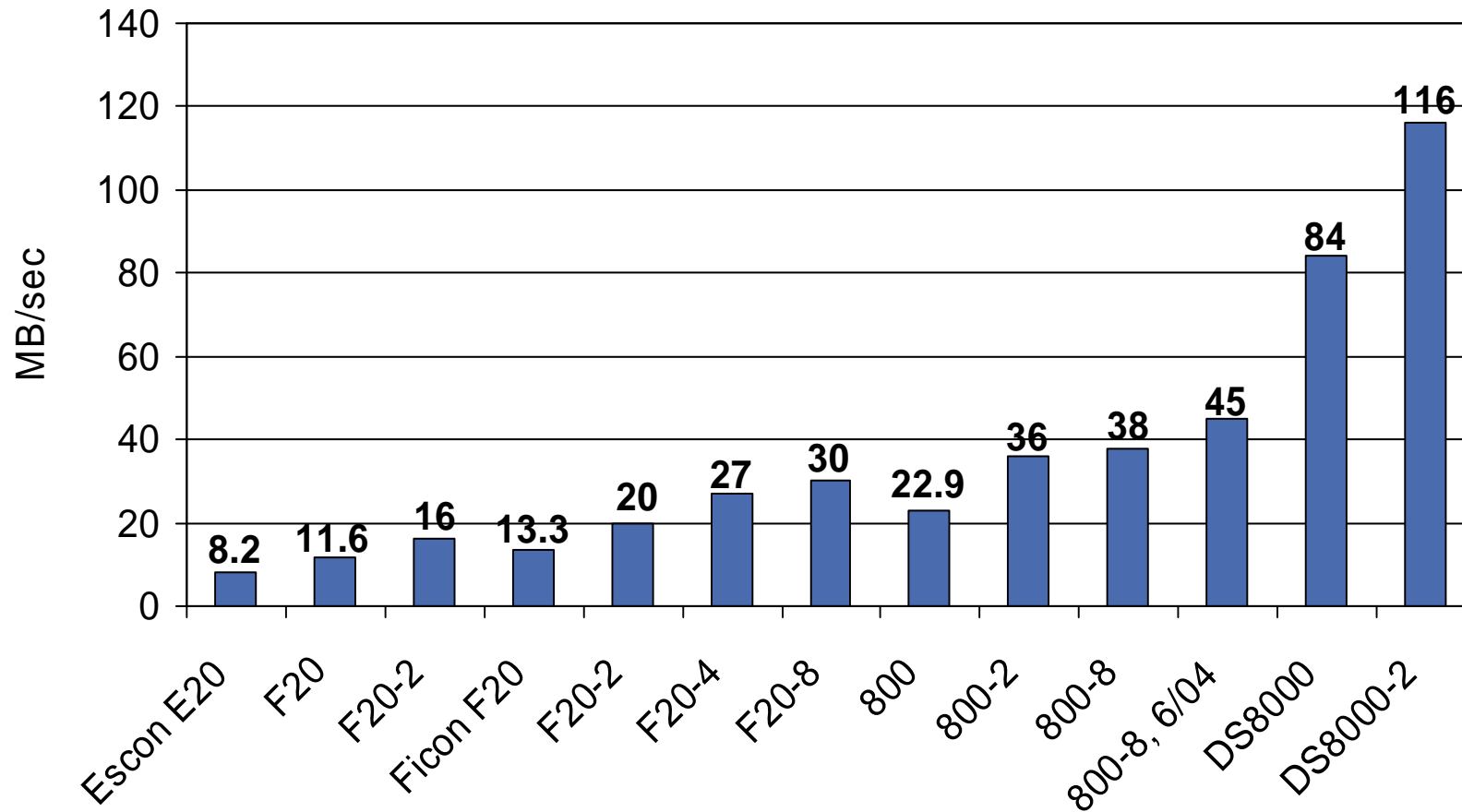
# Log Dataset I/O Tuning

- Avoid I/O interference among primary and secondary, active and archive, log read and write
- Log data rate =  $\#CIs\_created * 4KB / stats\_interval$
- Start paying attention if  $> 10MB/sec$  log rate
- If Log Rate near maximum,
  - Reduce log data volume
    - Variable-length record layout
    - Use of ESA Data Compression
    - Be aware of impact of DBD update
    - Use faster log device and/or IO striping



# Maximum Observed Rate of Active Log write

- First 3 use ESCON channel, the rest is FICON channel
- -2, -4, -8 indicate 2, 4, or 8 I/O stripes



# EDM Pool Tuning

- PAGES IN EDM POOL (BELOW) = Sum of pages for CT, SKCT, PT, SKPT, and Free pages
- Increase EDM Pool size as needed to keep:
  - % NON-STEALABLE PAGES IN USE (PTs, CTs) < 50%
  - FAILS DUE TO POOL FULL = 0
  - CT/PT HIT RATIO > 90 to 95%
    - Hit ratio = NOT FOUND/REQUESTS
- To keep EDM Pool size from getting too big use BIND option RELEASE (COMMIT) for all but most frequently executed plans/packages

| EDM POOL                  | TOTAL    |
|---------------------------|----------|
| PAGES IN EDM POOL (BELOW) | 87500    |
| HELD BY CT                | 145.94   |
| HELD BY PT                | 7706.12  |
| HELD BY SKCT              | 131      |
| HELD BY SKPT              | 55786.37 |
| FREE PAGES                | 23730.57 |
| % PAGES IN USE            | 72.88    |
| % NON STEAL. PAGES IN USE | 8.97     |
| FAILS DUE TO POOL FULL    | 0        |
| CT REQUESTS               | 763.2K   |
| CT NOT FOUND              | 1        |
| PT REQUESTS               | 16931.4K |
| PT NOT FOUND              | 1874     |



# EDM Pool Tuning ...

- EDMBFIT=YES|NO?
  - Trade-off between EDM Pool size constraint (i.e. DBM1 virtual storage constraint) and LC24 latch contention as discussed in Latch Monitoring/Tuning section
  - General recommendation always use EDMBFIT=NO (default)
- Smaller DBD helps (see Locking)
  - Especially in data sharing environment with multiple DBD copies in EDM Pool due to DBD invalidation by other members
  - Compact DBD by REORG and MODIFY if many DROP TABLE in segmented tablespace
  - DBD above 2GB in V8

| EDM POOL                   | TOTAL |
|----------------------------|-------|
| PAGES IN DBD POOL (ABOVE)  | 12500 |
| HELD BY DBD                | 1330  |
| FREE PAGES                 | 11170 |
| FAILS DUE TO DBD POOL FULL | 0     |
| DBD REQUESTS               | 63.2K |
| DBD NOT FOUND              | 1     |



# EDM Pool Tuning ...

- DB2 Statement Caching
  - Used by dynamic SQL applications to reuse and share prepared statements
    - Significant cost to fully prepare a dynamic SQL statement
  - Global Dynamic Statement Cache
    - Enabled by ZPARM CACHEDYN = YES
    - Prepared statements are kept in the EDM pool for reuse across all threads
    - REOPT(VARS) disables use of cache for that plan/package
  - Local Dynamic Statement Cache
    - Enabled by BIND option KEEP DYNAMIC(YES)
    - Prepared statements are kept in thread storage across COMMIT
    - MAXKEEPD limits #SQL statements across all threads and enforced at commit
    - CACHEDYN\_FREELOCAL > 0 limits #SQL statements across all threads and enforced at end of section



# EDM Pool Tuning ...

| DYNAMIC SQL STMT     | AVG/COMMIT | TOTAL  |
|----------------------|------------|--------|
| PREPARE REQUESTS     | 58.29      | 102.7K |
| FULL PREPARES        | 0.76       | 1335   |
| SHORT PREPARES       | 57.60      | 101.5K |
| IMPLICIT PREPARES    | 6.65       | 11.7K  |
| PREPARES AVOIDED     | 11.09      | 19.5K  |
| CACHE LIMIT EXCEEDED | 0.00       | 0      |
| PREP STMT PURGED     | 0.00       | 0      |

- Global Dynamic Statement Cache hit ratio should be > 90-95%  
= [Short Prepares] / [Short + Full Prepares]  
= 57.60/[57.60+0.76] = 98.70%
- Local Dynamic Statement Cache hit ratio should be >70%  
= [Prepares Avoided]/[Prepares Avoided + Implicit Prepares]  
= 11.09/[11.09+6.65] = 62.51%
  - Implicit Prepare can result in either Short or Full Prepare



# EDM Pool Tuning ...

- DB2 Statement Caching ...
  - Global Dynamic Statement Cache
    - Should be turned on if dynamic SQL is executed in the DB2 system
    - Best trade-off between storage and CPU consumption for applications executing dynamic SQL
  - Local Dynamic Statement Cache
    - Not recommended for storage constrained systems
    - Recommended for applications with a limited amount of SQL statements that are executed very often
    - Not recommended for applications with a large number of SQL statements that are executed infrequently



# System Address Space CPU Time

| CPU TIMES            | TCB TIME | SRB TIME    | TOTAL TIME  | /COMMIT  |
|----------------------|----------|-------------|-------------|----------|
| SYSTEM SERVICES AS   | 1.198663 | 1:44.737421 | 1:45.936084 | 0.000541 |
| DATABASE SERVICES AS | 6.260895 | 2:03.312609 | 2:09.573504 | 0.000661 |
| IRLM                 | 0.006539 | 5.420761    | 5.427300    | 0.000027 |
| DDF AS               | 0.004139 | 2.022798    | 2.026937    | 0.000010 |

- All TCB times and the IRLM SRB time should be low relative to MSTR and DBM1 SRB times
  - If not, needs further investigation
- If distributed application, DDF SRB is typically the highest by far as it includes Accounting TCB time also



# System Address Space CPU Time ...

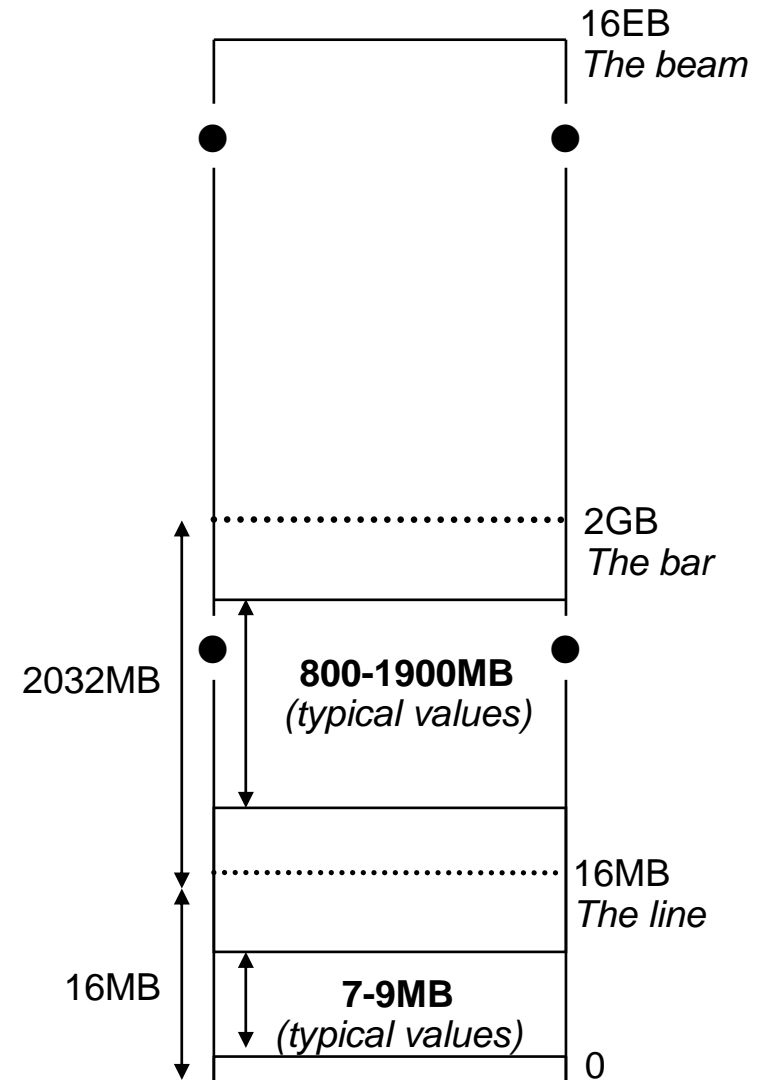
- Major MSTR SRB time
  - Physical log write, thread deallocation, update commit including page P-lock unlock
- Major DBM1 SRB time
  - Deferred write, prefetch read, parallel child tasks, Castout, async GBP write, P-lock negotiation, Notify exit, GBP checkpoint, Delete Name (pageset close or pseudo-close to convert to non GBP dependent)
- Major DBM1 TCB time
  - Dataset open/close, DBM1 Full System contraction
- Major IRLM SRB time
  - Local IRLM latch contention, IRLM and XES global contention, async XES request, P-lock negotiation

*Activities unique to data sharing are underlined*



# Fitting DB2 in the DBM1 Address Space

- DB2 DBM1 address space now has an addressing range of 16EB (“the beam”) based on 64 bit addressing but
  - Maximum of 16MB available “below the 16MB line”
  - Maximum of 2032MB available “above the 16MB line” and “below the 2GB bar”
- Practical maximum available to DB2 and specifically DBM1 AS is much less
  - Typical 7-9MB available “below the line”
  - Typical 800-1900MB available “above the line” and “below the 2GB bar”



# MVS Storage Overview

- EXTENDED REGION SIZE (MAX) – QW0225RG
  - Total theoretical amount DB2 has access to
- 31 BIT EXTENDED LOW PRIVATE – QW0225EL
  - DB2 uses a small amount of Low private (bottom up storage)
    - DB2 code itself
- 31 BIT EXTENDED HIGH PRIVATE – QW0225EH
  - DB2 mostly uses subpool 229 Key 7 (top down storage)
  - Other products also use address space storage
    - Dataset opens / DFP
    - SMF



# What Affects Max. Available Extended Region

- REGION = parm on JCL
  - No effect, DB2 uses high private
  - Region only affected low private storage
    - Some dataset open activity can be in trouble with a low REGION= parm
  - Usually REGION=0M is preferred
- ECSA – QW0225EC
  - Biggest factor
    - Large ECSA size would be 1 GB with typical sizes being 300-500MB
  - Some customers due to the needs of other products have huge ECSA leading to very small extended region size
    - Extensive use of ECSA by IMS TM across dependent regions
    - Generous over allocation for safety of ECSA and other extended common areas



# DB2 DBM1 Storage

## ■ Below 2GB

- DB2 Storage 31 bit / 24 bit
  - Getmained – QW0225GM
    - Part of EDM Pool
  - Variable – QW0225VR
    - Thread and system storage (AGL)
    - Part of the RID Pool
    - Local Dynamic Statement Cache
  - Fixed – QW0225FX
    - High performance fixed elements
  - Getmained Stack – QW0225GS
    - Program storage
- Non-DB2 Storage
  - Not tracked by DB2

## ■ Above 2GB

- DB2 Storage 64 bit
  - Getmained
    - Compression Dictionaries
    - DBD Pool
    - Global Dynamic Statement Cache
    - Part of CT/PT Pool (V9)
    - EDM Skeleton Pool (V9)
    - Fixed
    - Variable
  - Bufferpools
  - Buffer Control Blocks
  - Castout Buffers
- 64 bit Shared Memory (V9)



# Non-DB2 Storage

- Not tracked by DB2
- Non-DB2 storage is high private storage
  - $\text{TOTAL DBM1 STORAGE} = \text{TOTAL GETMAINED STORAGE QW0225GM} + \text{TOTAL GETMAINED STACK STORAGE QW0225GS} + \text{TOTAL FIXED STORAGE QW0225FX} + \text{TOTAL VARIABLE STORAGE QW0225VR}$
  - $\text{NON-DB2 STORAGE} = \text{MVS 31 BIT EXTENDED HIGH PRIVATE QW0225EH} - \text{TOTAL DB2 DBM1 STORAGE}$
- Used usually by MVS functions such as SMF
- SMF Type 30 records `DETAIL/DDCONS=YES` can cause this number to be large
  - The big hit to DB2 in this area is the DDNAME tracking: allocation does not realise that we have closed off a page set and reallocated it again
  - SMF Type 30 subtype 4 and 5 will track all the DDNAMES
  - Most environments do not need SMF Type 30 subtype 4 and 5
  - Recommend `DDCONS(NO)` and `NODETAIL`



# Storage Overuse: DB2 Storage Contraction

- When 'running low' on extended virtual, DB2 begins system contraction process which attempts to free any available segments of storage
  - Contraction can be:
    - Normal
    - A sign of a poorly tuned system
- 3 critical numbers for contraction:
  - Storage reserved for must complete (e.g. ABORT, COMMIT) – QW0225CR
    - =  $(CTHREAD+MAXDBAT+1)*\underline{64K}$  (Fixed, real value)
  - Storage reserved for open/close of datasets – QW0225MV
    - =  $(DSMAX*1300)+40K$  (Virtual number and no guarantee)
  - Warning to contract – QW0225SO
    - = Max (5% of Extended Region Size, QW0225CR)
  - Storage Cushion = QW0225CR + QW0225MV + QW0225SO



# Storage Overuse: DB2 Storage Contraction ...

- Examples:

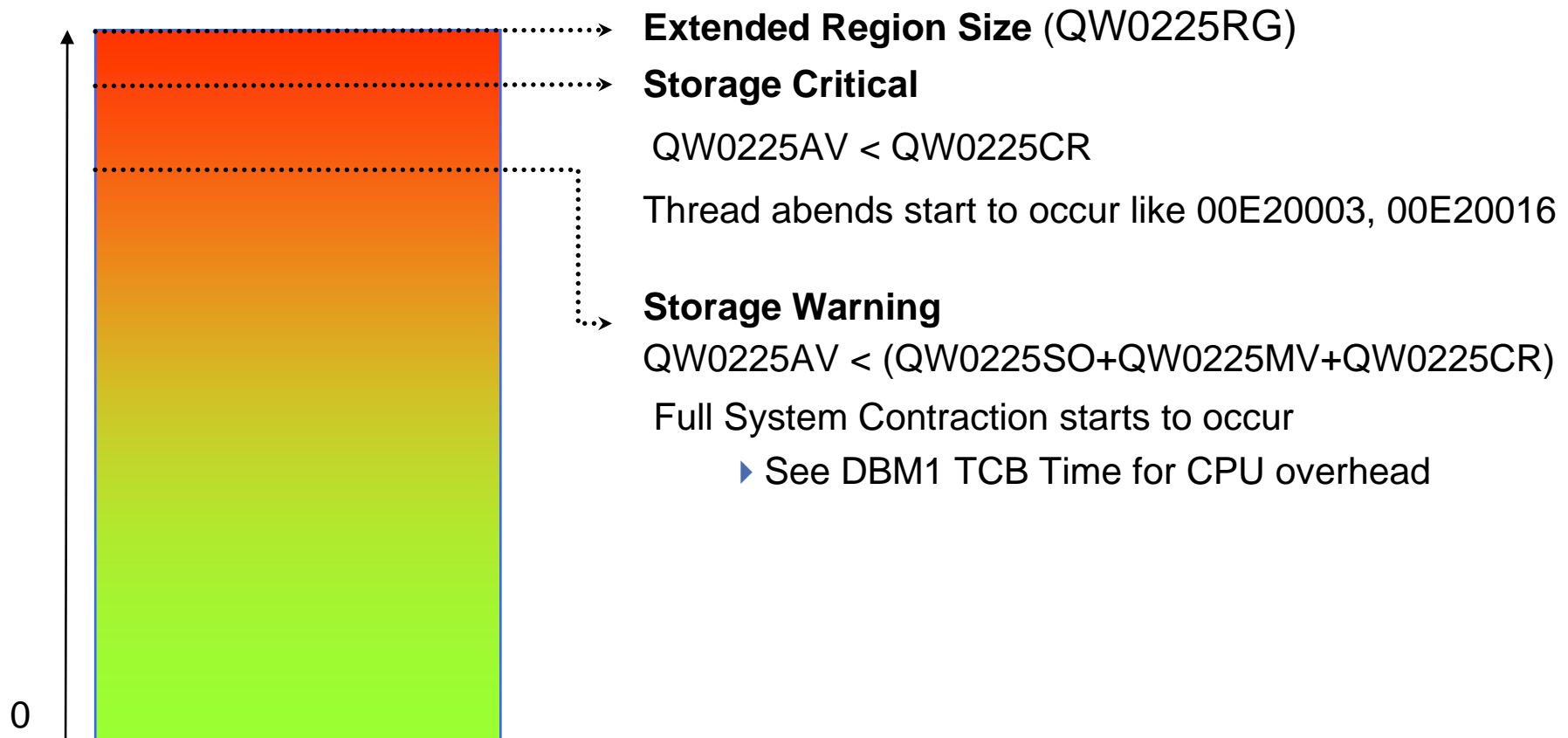
|   |            |            |            |
|---|------------|------------|------------|
| CTHREAD                                 | 2000       | 400        | 400        |
| MAXDBAT                                 | 2000       | 2000       | 150        |
| DSMAX                                   | 15000      | 15000      | 15000      |
| MVS extended region size (MB)           | 1700       | 1000       | 1000       |
| Storage reserved for must complete (MB) | 250        | 150        | 38         |
| Storage reserved for datasets (MB)      | 19         | 19         | 19         |
| Warning to contract (MB)                | 250        | 150        | 50         |
| <b>Storage Cushion (MB)</b>             | <b>519</b> | <b>319</b> | <b>107</b> |

**\*\*WARNING\*\* DO NOT SPECIFY CTHREAD + MAXDBAT TOO HIGH IN DB2 V8 OR THE CUSHION WILL BE VERY LARGE**



# Storage Overuse: DB2 Storage Contraction ...

- QW0225AV reports how much storage is available



# Storage Overuse: Large Contributors

- Stack use (QW0225GS)
  - Normal range is typically 300MB
  - Compressed only at full system contraction
- System agents (QW0225AS)
  - Some agents once allocated are never deallocated
    - For example: P-lock engine, prefetch engine
  - # engines: QW0225CE, QW0225DW, QW0225GW, QW0225PF, QW0225PL
    - If these counts are very low and system is on the brink of storage overuse, it is possible that the allocation of more engines could send the system into contraction
- User threads (QW0225VR-QW0225AS)
  - Typical user thread storage footprint can be 500KB to 10MB per thread depending on thread persistence, variety and type of SQL used
    - SAP Threads 10MB
    - CICS Threads 500KB
    - Number of threads obtained via QW0225AT + QDSTCNAT



# CONTSTOR

- Thread storage contraction turned on by zparm CONTSTOR = YES
  - Online changeable with immediate effect
- Associated CPU overhead
  - Benefit should be carefully evaluated before enabling
  - Ineffective for long-running persistent threads with use of RELEASE(DEALLOCATE)
- Compresses out part of Agent Local Non-System storage
  - Does not compress Agent Local System, Getmained Stack Storage, LDSC
- Controlled by two hidden zparms
  - SPRMSTH @ 1048576 and SPRMCTH @ 10
- Triggers
  - No. of Commits > SPRMCTH, or
  - Agent Local Non-System > SPRMSTH and No. of Commits > 5



# MINSTOR

- Best fit algorithm for thread storage turned on by zparm MINSTOR = YES
  - Online changeable, may not have an effect due to already cached pools
  - Restart recommended if this parm changed
- Changes the storage management of the user AGL POOL to “Best fit” rather than “First fit”
  - In order to find the best fit piece of storage, CPU cycles are used to scan and maintain ordered storage
  - In a POOL with low fragmentation, MINSTOR may not have a great effect but will cost CPU
- Only enable if fragmentation is a big issue
  - Only the SM=4 option of the DB2 Dump Formatter and a dump will really give you the definitive answer



# Protecting the System

- Plan on a 'Basic' storage cushion (free)
  - To avoid hitting short on storage and driving Full System Contraction
  - To provide some headroom for:
    - Tuning, some growth, Fast Log Apply, abnormal operating conditions
  - Basic cushion = Storage cushion + 5% of Extended Region Size
    - The Basic cushion should be less than 17.5% of the Extended Region Size, otherwise CTHREAD and/or MAXDBAT are probably set too high

|                               |      |      |      |
|-------------------------------|------|------|------|
| CTHREAD                       | 2000 | 450  | 450  |
| MAXDBAT                       | 2000 | 2000 | 200  |
| MVS extended region size (MB) | 1700 | 1000 | 1000 |
| Storage Cushion (MB)          | 519  | 319  | 107  |
| Basic Cushion (MB)            | 604  | 369  | 157  |
| 17.5% of Extended Region Size | 298  | 175  | 175  |



# Protecting the System ...

- Estimate the maximum number of threads that can be supported
  - Assuming the storage is proportional to the amount of threads, it is possible to predict a theoretical max. number of concurrent threads
  - It may be possible to run the system with more threads than the formula dictates, but there is the danger that the large threads may come in and cause out of storage conditions
- Set zparms CTHREAD and MAXDBAT to protect the system
  - CTHREAD and MAXDBAT are the brakes on the DB2 subsystem
    - Theoretical maximum:  $CTHREAD + MAXDBAT = 2000$
    - Practical maximum is much less (typical range 300-850)
  - Avoid over committing resources
  - Deny service and queue work outside the system to keep system alive



# Estimating Maximum Number of Threads

- Collect IFCID 225 since the start of DB2
  - Month end processing
  - Weekly processing
  - Utilities processing
  - Try to use a full application mix cycle
- Focus on time periods with
  - Increasing number of allied threads + active DBATs
  - Increasing use of getmained stack storage
  - Increasing use of AGL non-system
- Adjust the formula based on workload variations
- Protect the system by always using a pessimistic approach to formulating the numbers
  - Optimistic may mean a DB2 outage
- Always recalculate on a regular basis as new workloads and/or parameters are changed



# Estimating Maximum Number of Threads ...

- 'Basic' storage cushion (BC)
  - $(BC) = QW0225CR + QW0225MV + QW0225SO + 5\% \text{ of } QW0225RG$
- Calculate non-DB2 storage (ND)
  - $(ND) = \text{MVS 31 BIT EXTENDED HIGH PRIVATE } QW0225EH - \text{TOTAL GETMAINED STORAGE } QW0225GM - \text{TOTAL GETMAINED STACK STORAGE } QW0225GS - \text{TOTAL FIXED STORAGE } QW0225FX - \text{TOTAL VARIABLE STORAGE } QW0225VR$
- Max. allowable storage (AS)
  - $(AS) = QW0225RG - (BC) - (ND)$
- Max. allowable storage for thread use (TS)
  - $(TS) = (AS) - \text{TOTAL AGENT SYSTEM STORAGE } QW0225AS - \text{TOTAL FIXED STORAGE } QW0225FX - \text{TOTAL GETMAINED STORAGE } QW0225GM - \text{MVS 31 BIT EXTENDED LOW PRIVATE } QW0225EL$
- Average thread footprint (TF)
  - $(TF) = (\text{TOTAL VARIABLE STORAGE } QW0225VR - \text{TOTAL AGENT SYSTEM STORAGE } QW0225AS + \text{TOTAL GETMAINED STACK STORAGE } QW0225GS) / (\text{Allied threads } QW0225AT + \text{DBATs } QDSTCNAT)$
- Max threads supported =  $(TS) / (TF)$



# Virtual vs. REAL Storage

- Important subsystems such as DB2 should not be paging IN from auxiliary storage (DASD)
  - Recommendation to keep page in rates low (near zero)
  - Monitor using RMF Mon III
- V8 introduces very large memory objects that may not be backed by REAL storage frames
  - Virtual storage below 2GB bar is usually densely packed (as before in V7)
    - VIRTUAL=REAL is a fair approximation
  - Virtual storage above the bar number may be misleading
    - Backing rate is low for 64-bit storage
    - No need to back until first reference
  - For an LPAR with greater than 16GB of defined real storage, DB2 will obtain a minimum starting memory object above the bar of 16GB
    - This memory is sparsely populated
    - Virtual will not equal REAL



# Virtual vs. REAL Storage ...

- MEMLIMIT not observed by DB2 DBM1
  - Overridden by DB2 to 4TB
- Maintenance – For reading
  - PK19769 – Reset the above the bar pools with DISCARDATA
  - OA15666 – New keyword to DISCARDATA to force discard of the REAL frame without hitting AVQLow condition
    - KEEPREAL=NO
  - PK25427 – Support OA15666 changes in DB2
  - With this maintenance many customers saw a drop of 30-50% in the increase of REAL frame usage from DB2 V7 to DB2 V8
  - PK21237
    - Drop number of Write Engines back to pre V8 levels
    - Single 64-bit pool for all Buffer Manager engines



# Monitoring REAL Storage

- Real storage needs to be monitored as much if not more in DB2 V8 as Virtual storage
  - Need to pay careful attention to QW0225RL (Real frames in use by DBM1) and QW0225AX (Auxiliary frames)
    - Ideally QW0225RL should be significantly less than the amount of virtual consumed
- An indication of either (a) a DB2 code error or (b) an under provisioned system will see:
  - 100% real frames consumed
    - It will be important to know how much real is dedicated to a given LPAR
      - Although a physical machine may have 30GB real, a given LPAR may only have a fraction of this real dedicated
  - An extensive number of auxiliary frames in use
  - Performance degradation



# How to Limit REAL Storage

- New Hidden ZPARM SPRMRSMX
- Causes a DB2 outage when the limit hits
- Delivered in APAR PK18354
- Not widely broadcast
- Preferable to monitor the REAL storage numbers in IFCID 225 and generate alerts when large increase in AUX or REAL approaches max available



# RDS Sort Tuning

- Try to dedicate buffer pool for sort workfiles
  - Set VPSEQT=100%
  - Avoid following conditions by increasing buffer pool size as needed
    - Rules-Of-Thumb for workfile bufferpool
      - Merge Passes Degraded < 1 to 5% of Merge Pass Requested
      - Workfile Req. Rejected < 1 to 5% of Workfile Req. All Merge Passes
      - Workfile Prefetch Not Scheduled < 1 to 5% of Seq Pref Reads
      - Sync Reads < 1 to 5% of pages read by prefetch
        - » Number of physical workfile datasets may also need to be increased
      - Prefetch quantity of 4 or less



# RDS Sort Tuning ...

- **WORKFILE REQUESTS REJECTED** = # workfile requests rejected because bufferpool is too small to support concurrent sort activity
  - # logical workfiles, or runs, required for a given sort, with sufficient sort pool size =
    - 1 if rows already in sort key sequence
    - #rows\_sorted/32000 if rows in random sort key seq.
    - #rows\_sorted/16000 if rows in reverse sort key seq.
- **MERGE PASSES DEGRADED** = #times merge pass > 1 because workfile requests rejected
  - #merge passes =
    - 0 if #runs=1
    - 1 if #runs < max #workfiles in merge determined by bufferpool size
    - 2 else



# RDS Sort Tuning ...

- To reduce CPU and workfile I/O time
  - Minimise sort row size
  - Minimise key size
  - Minimise number of rows sorted
- To reduce workfile I/O time
  - Provide sufficient number of physical workfile datasets on different DASD volumes
    - 1, not multiple, physical workfile dataset per DASD volume
    - 1, not multiple, volume per STOGROUP



## RDS Sort Tuning ...

- To estimate sort data size, divide workfile Getpage
  - by 2 if #merge passes = 0
  - by 4 if #merge passes = 1
  - and then multiply by 4KB
- #merge passes is explained earlier
- Example: 100,000 row sort with 4000 workfile Getpage
  - Sort data size =  $4000 * 4KB / 4 = 4MB$
  - DASD space requirement estimate =  $4MB * 1.3$  Cyls



# Disclaimer

© Copyright IBM Corporation [current year]. All rights reserved.

*U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.*

**THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE.**

IBM, the IBM logo, ibm.com, and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml)

