Optimising DB2 for z/OS System Performance Using DB2 Statistics Trace

John Campbell & Florence Dubois

IBM DB2 for z/OS Development

One-Day Seminar: Z03
Monday 2nd May 2011 | Platform: DB2 for z/OS
Disclaimer

© Copyright IBM Corporation 2011. 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.
Objectives

• Focus on key areas
  • System address space CPU, EDM pools, dataset activity, logging, lock/latch contention, DBM1 virtual and real storage, buffer pool and GBP,…

• 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

- Data Collection
- System Address Space CPU Time
- EDM Pools
- Dataset Open/Close
- Log Activity
- Lock/Latch
- DBM1 Virtual Storage and Real Storage
- Buffer Pools and Group Buffer Pools
- RDS Sort
- DDF Activity
- Miscellaneous
DBM1 Virtual Storage and Real Storage
Why is Storage Tuning Important?

- DB2 DBM1 AS 31-bit virtual storage constraint is the leading cause of DB2 outages
- If DBM1 AS 31-bit storage is limited/constrained then
  - Vertical scalability of DB2 member is limited
    - Must grow DB2 processing capacity horizontally
    - Extra DB2 members required in the DB2 data sharing group
  - Vertical scalability of standalone DB2 subsystem is limited
    - Must grow DB2 processing capacity horizontally
    - Migrate to DB2 data sharing configuration
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”
What is the Problem?

- Storage is allocated into different subpools which have unique characteristics
  - Storage acquired via MVS GETMAIN
  - Storage released by MVS FREEMAIN
- GETMAIN processing by DB2 components using DB2 Storage Manager
  - Requests may be conditional or unconditional to DB2 Storage Manager
  - "Short on Storage" condition can occur for both
  - DB2 recovery routines may be able to clean up
  - Individual DB2 threads (allied, DBAT) may abend with 04E/RC=00E200xx when insufficient storage available
    - e.g. 00E20003 & 00E20016
  - Eventually DB2 subsystem may abend with S878 or S80A due to non-DB2 subsystem component (e.g. DFP) issuing unconditional MVS getmain
    - DB2 getmains are MVS conditional getmains, so are converted to DB2 abends
      - e.g. 00E20016
Tracking DB2 Storage

- DB2 storage is mostly allocated in SP229 Key 7
- RMF for high level
  - Virtual Storage (VSTOR) Private Area Report
    - Interval data collected in SMF Type 78-2
    - Collected by RMF Monitor I session option: VSTOR(D,xxxxDBM1)
    - Produced by RMF Post Processor option: REPORTS(VSTOR(D,xxxxDBM1))
- IFC Records
  - IFCID 225
    - Storage Summary
    - Snapshot value as each DB2 Stats interval comes due (ZPARM = STATIME)
    - Now included in Statistics Trace Class 1
  - IFCID 217
    - Storage Detail Record at thread level
    - Effectively a dump SM=1 report but in IFC form
    - Available through Global Trace Class 10

Available through Global Trace Class 10
Tracking DB2 Storage …

- IFC Records …
  - First class support provided by OMEGAMON XE for DB2 PM/PE, DB2 PM and DB2 PE
    - Statistics Trace | Report
      - Includes FILE and LOAD data base table support as well as upgrade
        (ALTER TABLE ….) of already installed table DB2PM_STAT_GENERAL
    - Record Trace Report
  - SPREADSHEETDD subcommand option
    - Both DB2PE V2.1 & DB2PM V8.1 via APAR PK31073
    - OMEGAMON XE for DB2 PE V3 & V4 via APARs PK33395 & PK33406
- REXX Tools (MEMU2, MEMUSAGE)
  - Available for download from DB2 for z/OS Exchange
      - Click on 'View and download examples'
      - The file is tagged with 'memu2'
Tracking DB2 Storage …

- V8 APAR PK20800 8/07
  - -DISPLAY THREAD(*) SERVICE(STORAGE)
  - DSNV492I message that can be used by DB2 service for diagnostics

V91A  N  *  0 003.RCRSC 02 SYSOPR 0067 0
V490-SUSPENDED 07213-09:59:18.02 DSNRCRSC +00000230 01.51
V492-LONG 252 K VLONG 40 K 64 1028 K

- Includes Agent Local Non-System Storage usage
- Does not include Getmained Stack Storage usage
- The key values are the LONG storage pool and the VLONG storage pool values (252KB + 40KB = 292KB in the above example)
  - Reflect virtual storage consumption below the 2GB bar
  - May be used to identify poorly behaved applications or DB2 code issues
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 / reservation for pageset storage

- **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

<table>
<thead>
<tr>
<th>DBM1 and MVS Storage Below 2 GB</th>
<th>Continued</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 BIT LOW PRIVATE</td>
<td>(MB)</td>
<td>0.14</td>
</tr>
<tr>
<td>24 BIT HIGH PRIVATE</td>
<td>(MB)</td>
<td>0.47</td>
</tr>
<tr>
<td>31 BIT EXTENDED LOW PRIVATE</td>
<td>(MB)</td>
<td>130.86</td>
</tr>
<tr>
<td>31 BIT EXTENDED HIGH PRIVATE</td>
<td>(MB)</td>
<td>859.40</td>
</tr>
<tr>
<td>EXTENDED REGION SIZE (MAX)</td>
<td>(MB)</td>
<td>1296.00</td>
</tr>
<tr>
<td>EXTENDED CSA SIZE</td>
<td>(MB)</td>
<td>632.48</td>
</tr>
</tbody>
</table>
MVS Storage Overview ...

- ECSA – QW0225EC
  - Common storage area across all address spaces for a given LPAR
  - Large ECSA size would be 1GB with typical sizes being 300-500MB
  - Affects maximum available Extended Region
    - Biggest factor
  - Some customers due to the needs of other products have huge ECSA requirement leading to very small extended region size
    - Extensive use of ECSA by IMS across dependent regions
      - Mostly buffer pools, control blocks, data are in ECSA
      - Sizes are at user choice – For best performance they tend to be large
      - Not exploiting VSCR features of recent IMS releases
  - Generous over allocation for safety of ECSA and other common areas
  - Common LPAR image for Sysplex (best practice)
MVS Storage Overview …

- REGION parameter in JCL for ASID
  - MVS rule is high private cannot grow into low private
  - Region parameter controls how high low-private can grow
    - 0M means unlimited
    - No effect on getmained DB2 storage since DB2 only allocates high private storage
  - Region parameter can be used to save a few MB by forcing some low private growth into 24-bit low private (QW0225LO)
    - Some dataset open activity can be in trouble with a low REGION= parm
  - Usually REGION=0M is preferred

- MEMLIMIT
  - Not observed by DB2 DBM1
  - Overridden by DB2 to 4TB
DB2 DBM1 Address Space Storage

- 31-bit / 24-bit DB2 storage
  - Getmained
  - Variable
  - Fixed Storage
  - Stack storage

- Non-DB2 getmained
  - SMF
  - Dataset / pageset

<table>
<thead>
<tr>
<th>DBM1 AND MVS STORAGE BELOW 2 GB</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL DBM1 STORAGE BELOW 2 GB (MB)</td>
<td>616.67</td>
</tr>
<tr>
<td>TOTAL GETMAINED STORAGE (MB)</td>
<td>137.96</td>
</tr>
<tr>
<td>EDM POOL (MB)</td>
<td>120.00</td>
</tr>
<tr>
<td>TOTAL VARIABLE STORAGE (MB)</td>
<td>379.23</td>
</tr>
<tr>
<td>TOTAL AGENT LOCAL STORAGE (MB)</td>
<td>279.39</td>
</tr>
<tr>
<td>TOTAL AGENT SYSTEM STORAGE (MB)</td>
<td>187.10</td>
</tr>
<tr>
<td>NUMBER OF PREFETCH ENGINES</td>
<td>600.00</td>
</tr>
<tr>
<td>NUMBER OF DEFERRED WRITE ENGINES</td>
<td>300.00</td>
</tr>
<tr>
<td>NUMBER OF CASTOUT ENGINES</td>
<td>300.00</td>
</tr>
<tr>
<td>NUMBER OF GBP WRITE ENGINES</td>
<td>300.00</td>
</tr>
<tr>
<td>NUMBER OF P-LOCK/NOTIFY EXIT ENGINES</td>
<td>202.00</td>
</tr>
<tr>
<td>TOTAL AGENT NON-SYSTEM STORAGE (MB)</td>
<td>92.29</td>
</tr>
<tr>
<td>TOTAL NUMBER OF ACTIVE USER THREADS</td>
<td>101.12</td>
</tr>
<tr>
<td>RID POOL (MB)</td>
<td>1.13</td>
</tr>
<tr>
<td>PIPE MANAGER SUB POOL (MB)</td>
<td>7.97</td>
</tr>
<tr>
<td>LOCAL DYNAMIC STMT CACHE CNTL BLKS (MB)</td>
<td>0.99</td>
</tr>
<tr>
<td>THREAD COPIES OF CACHED SQL STMTS (MB)</td>
<td>16.24</td>
</tr>
<tr>
<td>IN USE STORAGE (MB)</td>
<td>0.61</td>
</tr>
<tr>
<td>STATEMENTS COUNT</td>
<td>82.77</td>
</tr>
<tr>
<td>HWM FOR ALLOCATED STATEMENTS (MB)</td>
<td>0.78</td>
</tr>
<tr>
<td>STATEMENT COUNT AT HWM</td>
<td>122.00</td>
</tr>
<tr>
<td>DATE AT HWM</td>
<td>02/02/09</td>
</tr>
<tr>
<td>TIME AT HWM</td>
<td>10:24:37.33</td>
</tr>
<tr>
<td>TOTAL FIXED STORAGE (MB)</td>
<td>2.61</td>
</tr>
<tr>
<td>TOTAL GETMAINED STACK STORAGE (MB)</td>
<td>96.87</td>
</tr>
<tr>
<td>TOTAL STACK STORAGE IN USE (MB)</td>
<td>49.35</td>
</tr>
<tr>
<td>STORAGE CUSHION (MB)</td>
<td>130.32</td>
</tr>
</tbody>
</table>
DB2 DBM1 Address Space Storage …

- Getmained - QW0225GM
  - V7 – EDM, Compression Dictionaries, Bufferpools and miscellaneous
  - V8 – Compression Dictionaries and Bufferpools are above the bar
  - Getmained should be much less in V8
    - If Compression is used
    - If Virtual Buffer Pools are used

- Variable Storage - QW0225VR
  - Most interesting from a tuning perspective
  - Variable length blocks
  - Thread pools (AGL)
    - Used by both System and User
  - Local Dynamic Statement Cache
  - V8 RDS OP POOL has gone, In V7 this could be significant

- Fixed Storage - QW0225FX
  - High performance storage
  - Fixed length blocks
  - Not usually so interesting from a tuning perspective
    - Small change in the great scheme of things

- Stack Storage - QW0225GS
  - Save areas
  - Working program variables
  - Small amounts of high speed storage allocations
  - Cached in the DB2 address space to allow greater performance
  - Compressed only at full system contraction
  - Stacks are bigger in V8
    - 64K instead of 20K
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
- Parameter DETAIL in SMFPRMxx can cause storage to creep and become very 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 NODETAIL
DB2 DBM1 Address Space Storage ...

- 64-bit DB2 storage
  - Getmained
    - Fixed
    - Variable
    - Compression Dictionaries
  - DBD Pool
  - Dynamic Statement Cache
  - RDS Pool Above (V9)
  - Skeleton Pool (V9)
- Buffer Pools
- Buffer Control Blocks
- Castout Buffers
- 64-bit Shared Private Storage (V9)

### DBM1 STORAGE ABOVE 2 GB

<table>
<thead>
<tr>
<th>DBM1 STORAGE ABOVE 2 GB</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIXED STORAGE (MB)</td>
<td>25.99</td>
</tr>
<tr>
<td>GETMAINED STORAGE (MB)</td>
<td>3951.66</td>
</tr>
<tr>
<td>COMPRESSION DICTIONARY (MB)</td>
<td>48.38</td>
</tr>
<tr>
<td>IN USE EDM DBD POOL (MB)</td>
<td>255.69</td>
</tr>
<tr>
<td>IN USE EDM STATEMENT POOL (MB)</td>
<td>93.98</td>
</tr>
<tr>
<td>IN USE EDM RDS POOL (MB)</td>
<td>0.09</td>
</tr>
<tr>
<td>IN USE EDM SKELETON POOL (MB)</td>
<td>96.61</td>
</tr>
<tr>
<td>VIRTUAL BUFFER POOLS (MB)</td>
<td>3460.16</td>
</tr>
<tr>
<td>VIRTUAL POOL CONTROL BLOCKS (MB)</td>
<td>0.91</td>
</tr>
<tr>
<td>CASTOUT BUFFERS (MB)</td>
<td>37.50</td>
</tr>
<tr>
<td>VARIABLE STORAGE (MB)</td>
<td>1255.45</td>
</tr>
<tr>
<td>THREAD COPIES OF CACHED SQL STMTS (MB)</td>
<td>126.59</td>
</tr>
<tr>
<td>IN USE STORAGE (MB)</td>
<td>1.08</td>
</tr>
<tr>
<td>HWM FOR ALLOCATED STATEMENTS (MB)</td>
<td>1.40</td>
</tr>
<tr>
<td>SHARED MEMORY STORAGE (MB)</td>
<td>2579.48</td>
</tr>
<tr>
<td>TOTAL FIXED VIRTUAL 64BIT SHARED (MB)</td>
<td>15.50</td>
</tr>
<tr>
<td>TOTAL GETMAINED VIRTUAL 64BIT SHARED (MB)</td>
<td>1194.41</td>
</tr>
<tr>
<td>TOTAL VARIABLE VIRTUAL 64BIT SHARED (MB)</td>
<td>1369.57</td>
</tr>
</tbody>
</table>
Storage Monitoring Using IFC225

- Graphing IFC 225 data basic elements to study evolutionary trend
Basic Graphing of Storage

- 7 days data (Mon-Sun) - Leaky subsystem?

DB2D Total Storage vs Threads
Basic Graphing of Storage

- Check the major components of DB2 storage to get an idea of the workload
  - Fixed
  - Getmained
  - Stack
  - Variable
  - Thread counts

- Does the previous graph show a storage leak?
  - Or does it?
Basic Graphing of Storage

- What happened next Mon-Wed: This DB2 took a full week to "warm up"
Lessons To Be Learned

- Wrong data time can lead to erroneous conclusion
  - Full week showed a possible leak
  - 10 days showed DB2 storage usage to be stable
- Ideal data is from DB2 startup to DB2 shutdown
  - If not possible, then get as much as you can
- Do not think you know the point where the maximum usage is – you may be way off the mark
- More data will lead you to a better result
How Much is Left Available in the Address Space

- QW0225AV – DB2 running total
  - Possibly inaccurate since DB2 storage manager has no idea about getmained storage obtained by other products directly from z/OS
    - DFP, SMF, etc
  - Number is re-evaluated when DB2 storage contraction occurs
  - Number is used to determine when DB2 storage contraction occurs

- What is really left
  - QW0225RG – (QW0225EL + QW0225EH)
  - These numbers obtained directly from z/OS
Storage Overuse: DB2 Storage Contraction

• When ‘running low’ on extended virtual, DB2 begins system contraction process which attempts to freemain 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)*64K (Fixed, real value) +25M
  • 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-25M)
  • Storage Cushion = QW0225CR + QW0225MV + QW0225SO
Storage Overuse: DB2 Storage Contraction

- Examples:

<table>
<thead>
<tr>
<th></th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTHREAD</td>
<td>2000</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>MAXDBAT</td>
<td>2000</td>
<td>2000</td>
<td>150</td>
</tr>
<tr>
<td>DSMAX</td>
<td>15000</td>
<td>15000</td>
<td>15000</td>
</tr>
<tr>
<td>MVS extended region size (MB)</td>
<td>1700</td>
<td>1700</td>
<td>1700</td>
</tr>
<tr>
<td>Storage reserved for must complete (MB)</td>
<td>275</td>
<td>178</td>
<td>63</td>
</tr>
<tr>
<td>Storage reserved for datasets (MB)</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>Warning to contract (MB)</td>
<td>225</td>
<td>128</td>
<td>85</td>
</tr>
<tr>
<td><strong>Storage Cushion (MB)</strong></td>
<td>519</td>
<td>325</td>
<td>166</td>
</tr>
</tbody>
</table>

**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

- **Extended Region Size** (QW0225RG)

- **Storage Critical**
  - QW0225AV < QW0225CR
  - Thread abends start to occur like 00E20003, 00E20016

- **Storage Warning**
  - QW0225AV < (QW0225SO+QW0225MV+QW0225CR)
  - Full System Contraction starts to occur
    - See DBM1 TCB Time for CPU overhead
Storage Overuse: Large Contributors

- Stack use (QW0225GS)
  - Normal range is typically 100-300MB
  - 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
  - Only compresses LONG storage (as per SERVICE(STORAGE))
  - Maximum of 1 compress every 5 commits so very cheap to implement
- 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
    - Local Dynamic Statement Cache
- 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 + 100MB
    - The Basic cushion should be less than 25% of the Extended Region Size, otherwise CTHREAD and/or MAXDBAT are probably set too high

<table>
<thead>
<tr>
<th></th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTHREAD</td>
<td>2000</td>
<td>450</td>
<td>450</td>
</tr>
<tr>
<td>MAXDBAT</td>
<td>2000</td>
<td>2000</td>
<td>150</td>
</tr>
<tr>
<td>MVS extended region size (MB)</td>
<td>1700</td>
<td>1700</td>
<td>1700</td>
</tr>
<tr>
<td>Storage Cushion (MB)</td>
<td>519</td>
<td>325</td>
<td>166</td>
</tr>
<tr>
<td>Basic Cushion (MB)</td>
<td>619</td>
<td>425</td>
<td>266</td>
</tr>
<tr>
<td>% of Extended Region Size</td>
<td>36%</td>
<td>25%</td>
<td>16%</td>
</tr>
</tbody>
</table>
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 getmainted 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…

- Remember to use the MAX impact value across all available data e.g. maximum system storage
- ‘Basic’ storage cushion (BC)
  - \[(BC) = QW0225CR + QW0225MV + QW0225SO + 100MB\]
- Calculate Max non-DB2 storage (ND)
  - \[(ND) = \text{MAX(MVS 31 BIT EXTENDED HIGH PRIVATE QW0225EH} - \text{TOTAL GETMAINED STORAGE QW0225GM} - \text{TOTAL GETMAINED STACK STORAGE QW0225GS} - \text{TOTAL FIXED STORAGE QW0225FX} - \text{TOTAL VARIABLE STORAGE QW0225VR})\]
- Max. allowable storage (AS)
  - \[(AS) = QW0225RG - (BC) - (ND)\]
- Max. allowable storage for thread use (TS)
  - \[(TS) = (AS) - (\text{MAX(TOTAL AGENT SYSTEM STORAGE QW0225AS)} + \text{MAX(TOTAL FIXED STORAGE QW0225FX)} + \text{MAX(TOTAL GETMAINED STORAGE QW0225GM)} + \text{MAX(MVS 31 BIT EXTENDED LOW PRIVATE QW0225EL)})\]
- Average thread footprint (TF)
  - \[(TF) = (\text{TOTAL VARIABLE STORAGE QW0225VR} - \text{TOTAL AGENT SYSTEM STORAGE QW0225AS} + \text{TOTAL GETMAINED STACK STORAGE QW0225GS}) / (\text{Allied threads QW0225AT + DBATs QDSTCNAT})\]
- 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
Monitoring REAL Storage

- Real storage needs to be monitored as much 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

<table>
<thead>
<tr>
<th>REAL AND AUXILIARY STORAGE</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL STORAGE IN USE (MB)</td>
<td>5958.66</td>
</tr>
<tr>
<td>AUXILIARY STORAGE IN USE</td>
<td>0.00</td>
</tr>
</tbody>
</table>

- 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

- V9 – Shared object storage can only be monitored at the LPAR level so it is only accurate for a single DB2 LPAR assuming no other exploiters of shared storage
Monitoring REAL Storage - Warning

• Excessive amounts of storage on AUX may cause long DUMP times and severe performance issues
  • Paging may become severe

• Make sure enough REAL storage is available in case DB2 has to take a DUMP
  • DUMP should complete in seconds to make sure no performance problems ensue
  • Once paging begins it is possible to have the DUMP take 10s of minutes
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
DB2 Service Monitor (V9 CM)

- Automatically issues console messages when DBM1 virtual storage below the 2GB bar reaches critical usage thresholds
  - 88, 92, 96, or 98 percent of available storage
- Identifies the agents that consume the most storage

```
DSNV508I -SE20 DSNVMON - DB2 DBM1 BELOW-THE-BAR
STORAGE NOTIFICATION
  91% CONSUMED
  87% CONSUMED BY DB2

DSNV510I -SE20 DSNVMON - BEGINING DISPLAY OF LARGEST
STORAGE CONSUMERS IN DBM1
DSNV512I -SE20 DSNVMON - AGENT 1: 094
NAME   ST A  REQ ID  AUTHID PLAN
-----  -- - --- -- ------- ----- 
SERVER RA *  18461 SE2DIA004 R3USER DISTSERV
  LONG 1720K VLONG 388K 64BIT 2056K

DSNV512I -SE20 DSNVMON - AGENT 2: 095
NAME   ST A  REQ ID  AUTHID PLAN
-----  -- - --- -- ------- ----- 
SERVER RA *   9270 SE2DIA001 R3USER DISTSERV
  LONG 1672K VLONG 388K 64BIT 2056K
```
New CPU AND STORAGE Metrics (V8 and V9)

- Introduced by PK62116
  - APAR PK66373 must be applied
  - APAR OA24404 for RMF must be applied
  - ZPARM ZOSMETRICS must be set to YES (default is NO)
  - RMF Monitor Type III must be running
- Adds the following metrics to IFCID 001 (QWOS)
  - Number of CPs
  - CPU utilisation
  - Storage values
  - Unreferenced Interval Count (UIC)
    - Indicator for real-storage constraints
- Support in Omegamon XE for DB2 PM/PE V4.20

<table>
<thead>
<tr>
<th>CPU AND STORAGE METRICS</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP LPAR</td>
<td>4.00</td>
</tr>
<tr>
<td>CPU UTILIZATION LPAR</td>
<td>39.20</td>
</tr>
<tr>
<td>CPU UTILIZATION DB2</td>
<td>0.00</td>
</tr>
<tr>
<td>CPU UTILIZATION DB2 MSTR</td>
<td>0.00</td>
</tr>
<tr>
<td>CPU UTILIZATION DB2 DBM1</td>
<td>0.00</td>
</tr>
<tr>
<td>UNREFERENCED INTERVAL COUNT</td>
<td>65535.00</td>
</tr>
<tr>
<td>REAL STORAGE LPAR (MB)</td>
<td>3071.00</td>
</tr>
<tr>
<td>FREE REAL STORAGE LPAR (MB)</td>
<td>1537.10</td>
</tr>
<tr>
<td>USED REAL STORAGE DB2 (MB)</td>
<td>194.65</td>
</tr>
</tbody>
</table>
Buffer Pools and Group Buffer Pools
Pages Classification and LRU Processing

- Pages in a BP are classified as either random or sequential
- Pages read from DASD
  - A page read in via synchronous I/O is classified as random
  - A page read in via any form of prefetch I/O is classified as sequential
- Pages already exist in the BP from previous work
  - A random page is never re-classified as sequential
  - A sequential page is re-classified as random when subsequently hit by a random getpage
    - V8 – Applies to getpages for pages just read by dynamic and list prefetch
    - V9 – Does not apply to getpages for pages just read by dynamic prefetch
Pages Classification and LRU Processing …

- DB2 has a mechanism to prevent sequentially accessed data from monopolizing the BP and pushing out useful random pages
  - Maintains two chains
    - LRU with all pages (random and sequential)
    - SLRU with only the sequential pages
  - Steals from the LRU chain until VPSEQT is reached, and then steals preferentially from the SLRU chain

- General recommendations
  - Set VPSEQT to 100 for the sort workfile BP
  - Set VPSEQT to 0 for data-in-memory BP to avoid the overhead of scheduling prefetch engines when data is already in BP
  - Unless you have done a detailed study, leave VPSEQT to 80 (default)
    - You may decide to set it lower (e.g. 40) to protect useful random pages
Hit Ratios and Page Residency Time

• Hit Ratios = percentage of times the page was found in the BP
  • System HR = (Total getpages – Total pages read) / Total getpages * 100
  • Appli. HR = (Total getpages – Synchronous reads) / Total getpages * 100

• Residency Time = average time that a page is resident in the buffer pool
  • System RT (seconds) = VPSIZE / Total pages read per second
  • Total getpages = random getpages + sequential getpages
  • Total pages read = synchronous reads for random getpages + synchronous reads for sequential getpages + pages read via sequential prefetch + pages read via list prefetch + pages read via dynamic prefetch
  • Synchronous reads = synchronous reads for random getpages + synchronous reads for sequential getpages
Deferred Writes

- VDWQT (Vertical Deferred Write Queue Threshold) based on number of updated pages at the data set level (% of VPSIZE or number of buffers)
  - DB2 schedules up to 128 pages for that data set, sorts them in sequence, and writes them out in at least 4 I/Os (page distance limit of 180 pages applies)

- DWQT (horizontal Deferred Write Queue Threshold) based on number of unavailable pages (updated + in-use) at the BP level (% of VPSIZE)
  - Write operations are scheduled for up to 128 pages per data set to decrease the number of unavailable buffers to 10% below DWQT
Deferred Writes …

- Setting VDWQT to 0 is good if the probability to re-write the page is low
  - DB2 waits for up to 40 changed pages for 4K BP (24 for 8K, 16 for 16K, 12 for 32K) and writes out 32 pages for 4K BP (16 for 8K, 8 for 16K, 4 for 32K)

- Setting VDWQT and DWQT to 90 is good for objects that reside entirely in the buffer pool and are updated frequently

- In other cases, set VDWQT and DWQT low enough to achieve a "trickle" write effect in between successive system checkpoints
  - But… Setting VDWQT and DWQT too low may result in poor write caching, writing the same page out many times, short deferred write I/Os, and increased DBM1 SRB CPU resource consumption
    - If you want to set VDWQT in pages, do not specify anything below 128
Critical Thresholds

- **Immediate Write Threshold**
  Checked at page update
  >> After update, synchronous write

- **Data Management Threshold**
  Checked at page read or update
  >> Getpage can be issued for each row sequentially scanned on the same page – potential large CPU time increase

- **Sequential Prefetch Threshold**
  Checked before and during prefetch
  90% of buffers not available for steal, or running out of sequential buffers (VPSEQT with 80% default)
  >> Disables Prefetch (PREF DISABLED – NO BUFFER)
Critical Thresholds …

Minimise  
PREF DISABLED - NO BUFFER (*)
Keep  
DATA MANAGEMENT THRESHOLD to 0

- 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

- 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 6 BPs: catalog/directory (4K and 8K), user index (4K), user data (4K), work file (4K and 32K)
Long-Term Page Fix for BPs 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

In a steady-state: PAGE-IN for READ / WRITE <1% of pages read / written

- 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 BPs with Frequent I/Os

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

<table>
<thead>
<tr>
<th>BPID</th>
<th>VPSIZE</th>
<th>Read Sync</th>
<th>Read SPF</th>
<th>Read LPF</th>
<th>Read DPF</th>
<th>Read -Total</th>
<th>Written</th>
<th>I/O Intensity</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP0</td>
<td>40000</td>
<td>2960</td>
<td>0</td>
<td>0</td>
<td>6174</td>
<td>9134</td>
<td>107</td>
<td>0.2</td>
</tr>
<tr>
<td>BP1</td>
<td>110000</td>
<td>12411</td>
<td>5185</td>
<td>0</td>
<td>1855</td>
<td>19451</td>
<td>6719</td>
<td>0.2</td>
</tr>
<tr>
<td>BP2</td>
<td>110000</td>
<td>40482</td>
<td>19833</td>
<td>11256</td>
<td>9380</td>
<td>80951</td>
<td>5763</td>
<td>0.8</td>
</tr>
<tr>
<td>BP3</td>
<td>75000</td>
<td>23972</td>
<td>6065</td>
<td>0</td>
<td>14828</td>
<td>44865</td>
<td>7136</td>
<td>0.7</td>
</tr>
<tr>
<td>BP4</td>
<td>80000</td>
<td>22873</td>
<td>45933</td>
<td>3926</td>
<td>50261</td>
<td>122993</td>
<td>3713</td>
<td>1.6</td>
</tr>
<tr>
<td>BP5</td>
<td>200000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>BP8K0</td>
<td>32000</td>
<td>9</td>
<td>11</td>
<td>0</td>
<td>11</td>
<td>31</td>
<td>169</td>
<td>0.0</td>
</tr>
<tr>
<td>BP32K</td>
<td>2000</td>
<td>693</td>
<td>873</td>
<td>0</td>
<td>6415</td>
<td>7981</td>
<td>38</td>
<td>4.0</td>
</tr>
</tbody>
</table>

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

- Two reasons for cross invalidations
  - Perfectly normal condition in an active-active data sharing environment
  - Directory entry reclaims – condition you want to tune away from
    - CPU overhead and I/O overhead if there is not enough directory entries
    - Extra CF access and Sync I/O Read

- DISPLAY GROUPBUFFERPOOL(*) GDETAIL(INTERVAL)

  DSNB787I - RECLAIMS
  FOR DIRECTORY ENTRIES = 0
  FOR DATA ENTRIES = 0
  CASTOUTS = 0

  DSNB788I - CROSS INVALIDATIONS
  DUE TO DIRECTORY RECLAIMS = 0
  DUE TO WRITES = 0
  EXPLICIT = 0

Cross Invalidations due to Directory Reclaims should be zero
Group Bufferpool Tuning …

- Recommend use of XES Auto Alter (autonomic)
  - Tries to avoid Structure Full condition
  - Tries to avoid Directory Entry Reclaim condition

- CFRM Policy
  - ALLOWAUTOALT(YES)
  - Set MINSIZE to INITSIZE
  - Set FULLTHRESHOLD = 80-90%
  - Set SIZE to 1.3-1.5x INITSIZE

- Periodic review and updated CFRM based on actual allocation and ratio
  - Especially when allocated size reaches SIZE
GBP Read Tuning

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QBGLXD</td>
<td>SYN.READ(XI)-DATA</td>
</tr>
<tr>
<td>QBGLXR</td>
<td>SYN.READ(XI)-NO DATA</td>
</tr>
<tr>
<td>QBGLMD</td>
<td>SYN.READ(NF)-DATA</td>
</tr>
<tr>
<td>QBGLMR</td>
<td>SYN.READ(NF)-NO DATA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group</th>
<th>BP14</th>
<th>QUANTITY /SECOND</th>
<th>/THREAD</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYN.READ(XI)-DATA RETURNED</td>
<td>1932.00</td>
<td>0.09</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>SYN.READ(XI)-NO DATA RETURN</td>
<td>39281.6K</td>
<td>1823.66</td>
<td>236.31</td>
<td>79.22</td>
</tr>
<tr>
<td>SYN.READ(NF)-DATA RETURNED</td>
<td>22837.00</td>
<td>1.06</td>
<td>0.14</td>
<td>0.05</td>
</tr>
<tr>
<td>SYN.READ(NF)-NO DATA RETURN</td>
<td>6955.8K</td>
<td>322.93</td>
<td>41.85</td>
<td>14.03</td>
</tr>
</tbody>
</table>

- Local BP search -> GBP search -> DASD I/O
- SYN.READ(NF) = Local Buffer Pool miss
- SYN.READ(XI) = Local Buffer Pool hit but cross-invalidated buffer
  - Most data should be found in GBP >> if not, GBP may be too small or pages have been removed because of directory entry reclaims

*Sync.Read(XI) miss ratio should be < 10%*

TOTAL SYN.READ(XI) = SYN.READ(XI)-DATA RETURNED + SYN.READ(XI)-NO DATA RETURN
Sync.Read(XI) miss ratio = SYN.READ(XI)-NO DATA RETURN / TOTAL SYN.READ(XI)
GBP Read Tuning ...

- **GBPCACHE CHANGED** - default and general recommendation
  - Do not use CHANGED for a LOB table space defined with LOG NO
- **GBPCACHE SYSTEM** - default for LOBs prior to V9
  - V9 default is CHANGED
- **GBPCACHE NONE**
  - GBP used only for Cross Invalidation
    - Reported as Explicit Cross Invalidation in GBP statistics
  - 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
    - Batch update sequentially processing a large table
    - GBP read/write ratio < 1%
    - Primarily to reduce CF usage
GBP Write Tuning

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QBGLWS</td>
<td>WRITE AND REGISTER</td>
</tr>
<tr>
<td>QBGLWM</td>
<td>WRITE AND REGISTER MULT</td>
</tr>
<tr>
<td>QBGLSW</td>
<td>CHANGED PGS SYNC.WRTN</td>
</tr>
<tr>
<td>QBGLAW</td>
<td>CHANGED PGS ASYNC.WRTN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Group BP14</th>
<th>CONTINUED</th>
<th>Quantity /Second</th>
<th>Thread /Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRITE AND REGISTER</td>
<td>54896.00</td>
<td>2.55</td>
<td>0.33</td>
</tr>
<tr>
<td>WRITE AND REGISTER MULT</td>
<td>255.5K</td>
<td>11.86</td>
<td>1.54</td>
</tr>
<tr>
<td>CHANGED PGS SYNC.WRTN</td>
<td>408.3K</td>
<td>18.96</td>
<td>2.46</td>
</tr>
<tr>
<td>CHANGED PGS ASYNC.WRTN</td>
<td>1713.4K</td>
<td>79.55</td>
<td>10.31</td>
</tr>
</tbody>
</table>

- 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 ...

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>GROUP BP14</th>
<th>QUANTITY</th>
<th>/SECOND</th>
<th>/THREAD</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>QBGLRC</td>
<td>PAGES CASTOUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QBGLUN</td>
<td>UNLOCK CASTOUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QBGLCT</td>
<td>CLASST THRESHOLD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QBGLGT</td>
<td>GBPOOLT THRESHOLD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QBGLCK</td>
<td>GBP CHECKPOINTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Pages Castout / Unlock Castout is a good indicator of castout efficiency
- GBP castout thresholds are similar to local BP deferred write thresholds
  - Encourage Class_castout (CLASST) threshold by lowering its value
    - 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

<table>
<thead>
<tr>
<th></th>
<th>V7</th>
<th>V8</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDWQT (dataset level)</td>
<td>10%</td>
<td>5%</td>
</tr>
<tr>
<td>DWQT (buffer pool level)</td>
<td>50%</td>
<td>30%</td>
</tr>
<tr>
<td>CLASST (Class_castout)</td>
<td>10%</td>
<td>5%</td>
</tr>
<tr>
<td>GBPOOLT (GBP_castout)</td>
<td>50%</td>
<td>30%</td>
</tr>
</tbody>
</table>
GBP Write Tuning ...

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QBGLWF</td>
<td>WRITE FAILED-NO STORAGE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GROUP BP14</th>
<th>QUANTITY /SECOND</th>
<th>/THREAD</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASTOUT ENGINE NOT AVAIL.</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>WRITE ENGINE NOT AVAILABLE</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>READ FAILED-NO STORAGE</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>WRITE FAILED-NO STORAGE</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

- WRITE FAILED-NO STORAGE
  - GBP is full and no stealable data page
  - Potential outage if pages added to LPL list

WRITE FAILED-NO STORAGE < 1% of TOTAL CHANGED PGS WRRTN

- Keep Write Failed < 1% of pages written by
  - Using bigger GBP, and/or
  - Smaller castout thresholds, and/or
  - Smaller GBP checkpoint timer
RDS Sort
Sort Phases

- Sort tree
  - Runs
  - Up to 64 M
  - One per thread

Sort Pool

- Work Files

Buffer Pool

- Work Files

Sorted Rows

• # 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 sequence
  • #rows_sorted / 16000 if rows in reverse sort key sequence

• #merge passes =
  • 0 if #runs=1
  • 1 if #runs < max #workfiles in merge determined by buffer pool size
  • 2 otherwise
### Sort Indicators

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QBSTWFF</td>
<td>MERGE PASS DEGRADED</td>
</tr>
<tr>
<td>QBSTWFD</td>
<td>WORKFILE_REQ. REJECTED</td>
</tr>
<tr>
<td>QBSTWKPD</td>
<td>WORKFILE PRF NOT SCHEDULED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>BP7</th>
<th>SORT/MERGE</th>
<th>QUANTITY</th>
<th>/SECOND</th>
<th>/THREAD</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX WORKFILES CONCURR. USED</td>
<td>8.22</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MERGE PASSES REQUESTED</td>
<td>7304.00</td>
<td>0.34</td>
<td>0.04</td>
<td>0.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MERGE PASS DEGRADED-LOW BUF</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WORKFILE REQ.REJCTD-LOW BUF</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WORKFILE REQ-ALL MERGE PASS</td>
<td>14610.00</td>
<td>0.68</td>
<td>0.09</td>
<td>0.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WORKFILE NOT CREATED-NO BUF</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WORKFILE PRF NOT SCHEDULED</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **MERGE PASSES DEGRADED** = number of times merge pass > 1 because workfile requests rejected because buffer pool is too small

**Merge Passes Degraded < 1 to 5% of Merge Pass Requested**

- **WORKFILE REQUESTS REJECTED** = number of workfile requests rejected because buffer pool is too small to support concurrent sort activity

**Workfile Req. Rejected < 1 to 5% of Workfile Req. All Merge Passes**

- **WORKFILE PREFETCH NOT SCHEDULED** = number of times sequential prefetch was not scheduled for work file because of buffer pool shortage

**Workfile Prefetch Not Scheduled < 1 to 5% of Seq Pref Reads**

(1) **ROT** assume VDWQT = 10 and DWQT = 50
Sort Indicators ...

<table>
<thead>
<tr>
<th>BP7 READ OPERATIONS</th>
<th>QUANTITY</th>
<th>/SECOND</th>
<th>/THREAD</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNCHRONOUS READS</td>
<td>5703.00</td>
<td>0.26</td>
<td>0.03</td>
<td>0.01</td>
</tr>
<tr>
<td>SYNCHRON. READS-SEQUENTIAL</td>
<td>8.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>SYNCHRON. READS-RANDOM</td>
<td>5695.00</td>
<td>0.26</td>
<td>0.03</td>
<td>0.01</td>
</tr>
<tr>
<td>SEQUENTIAL PREFETCH REQUEST</td>
<td>88102.00</td>
<td>4.09</td>
<td>0.53</td>
<td>0.18</td>
</tr>
<tr>
<td>SEQUENTIAL PREFETCH READS</td>
<td>124.00</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>PAGES READ VIA SEQ.PREFETCH</td>
<td>941.00</td>
<td>0.04</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>S.PRF.PAGES READ/S.PRF.READ</td>
<td>7.59</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DYNAMIC PREFETCH REQUESTED</td>
<td>693.9K</td>
<td>32.22</td>
<td>4.17</td>
<td>1.40</td>
</tr>
<tr>
<td>DYNAMIC PREFETCH READS</td>
<td>164.00</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>PAGES READ VIA DYN.PREFETCH</td>
<td>2290.00</td>
<td>0.11</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>D.PRf.PAGES READ/D.PRf.READ</td>
<td>13.96</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PREF.DISABLED-NO BUFFER</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>PREF.DISABLED-NO READ ENG</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>PAGE-INS REQUIRED FOR READ</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

- SYNCHRONOUS READS >> buffer pool shortage or too few physical work files

**ROT**

*Sync Reads < 1 to 5% of pages read by prefetch (1)*

- Prefetch quantity = PRF.PAGES READ/PRF.READ

**ROT**

*If prefetch quantity < 4, increase buffer pool size (1)*

(1) ROT assume VDWQT = 10 and DWQT = 50
RDS Sort Recommendations

- Use dedicated 4K and 32K buffer pools for sort workfile
  - Set VPSEQT=100
  - Two possible strategies
    - Tuning for optimal performance
      - Set write thresholds very high e.g. DWQT=90 and VDWQT=70
      - Design objective: Complete sort in memory and avoid I/O to the sort workfiles
      - This is OK for consistent number of small sorts
      - But increased risk of hitting buffer pool critical thresholds
        - Adversely affecting application performance and threatening system stability
    - Tuning for robust defensive performance
      - Set write thresholds to DWQT=50 and VDWQT=10
      - Design objective: Protect the system against unexpected surge of sort activity
        - Many parallel sorts or a few very large sorts
      - Best strategy if fluctuation in the amount of sort activity within and across members
RDS Sort Recommendations …

- Provide multiple physical workfiles placed on different DASD volumes
  - Sort workfile placement example
    - 4-way Data Sharing Group
    - Assume 24 volumes are available
    - Each member should have 24 workfile tablespaces on separate volumes
    - All members should share all 24 volumes (i.e. 4 workfiles on each volume)

- Sort pool size
  - ZPARM SRTPOOL determines the maximum size of the sort work area allocated for each concurrent sort user (i.e. on a per-thread basis)
    - In V8, most of the sort pool is moved above the 2GB bar
    - The default (2MB) is well suited for most operational workloads
      - For more detailed tuning, use IFCID 96
Temporary Space in DB2 9 for z/OS (CM)

- Declared Global Temporary Tables and Static Scrollable Cursors now use the WORKFILE database instead of the TEMP database

- Heavier use of 32K buffer pool and workfiles to help performance (CM)
  - If the workfile record length is less than 100 bytes, then a 4KB page tablespace is used
  - In all other cases, DB2 tries to use a 32KB page tablespace
    - If no 32KB page TS is available, DB2 uses a 4KB page TS – lost performance benefit

- Recommendations
  - Assign bigger 32KB page workfile buffer pool
  - Allocate bigger/more 32KB page workfile tablesaces
Temporary Space in DB2 9 for z/OS (CM) …

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QISTWFCU</td>
<td>Current total storage used, in MB</td>
</tr>
<tr>
<td>QISTWFMU</td>
<td>Maximum space ever used, in MB</td>
</tr>
<tr>
<td>QISTWFMX</td>
<td>Max. allowable storage limit (MAXTEMPS) for an agent, in MB</td>
</tr>
<tr>
<td>QISTWFNE</td>
<td># of times max. allowable storage limit per agent was exceeded</td>
</tr>
<tr>
<td>QISTWF04</td>
<td>Current total 4KB page tablespace storage used, in MB</td>
</tr>
<tr>
<td>QISTWF32</td>
<td>Current total 32KB page tablespace storage used, in MB</td>
</tr>
<tr>
<td>QISTW04K</td>
<td>Current total 4KB page tablespace storage used, in KB</td>
</tr>
<tr>
<td>QISTW32K</td>
<td>Current total 32KB tablespace storage used, in KB</td>
</tr>
<tr>
<td>QISTWFP1</td>
<td># of times a 32KB page tablespace was used when a 4KB page tablespace was preferable (but not available)</td>
</tr>
<tr>
<td>QISTWFP2</td>
<td># of times a 4KB page tablespace was used when a 32KB page tablespace was preferable (but not available)</td>
</tr>
</tbody>
</table>

- New instrumentation to monitor the usage of temp space
  - IFCID 2 updated with new counters
    - Including the number of times a 4KB page tablespace was used when a 32K page tablespace was preferable (but not available)
  - Support in Omegamon XE for DB2 PM/PE V4.20

**WORK FILE DATABASE**

<table>
<thead>
<tr>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX TOTAL STORAGE (KB) 5120</td>
</tr>
<tr>
<td>CURRENT TOTAL STORAGE (KB) 1600</td>
</tr>
<tr>
<td>STORAGE IN 4K TS (KB) 64</td>
</tr>
<tr>
<td>STORAGE IN 32K TS (KB) 1536</td>
</tr>
<tr>
<td>4K USED INSTEAD OF 32K TS 0</td>
</tr>
<tr>
<td>32K USED INSTEAD OF 4K TS 0</td>
</tr>
<tr>
<td>AGENT MAX STORAGE (KB) 0</td>
</tr>
<tr>
<td>NUMBER OF MAX EXCEEDED 0</td>
</tr>
</tbody>
</table>
Temporary Space in DB2 9 for z/OS (CM) …

- APAR PK70060 introduces a soft barrier between DGTT and non-DGTT space use in workfile database

- Recommendation: Define some tablespaces with zero secondary quantity and some with non-zero secondary quantity
  - For the DB2-managed workfile tablespaces with zero secondary quantity, make sure the primary quantity is a little bit less than 2GB to avoid DB2 allocating a new subsequent piece A002

- APAR PM02528 provides the choice to make it a hard barrier
  - New opaque zparm WFDBSEP=NO|YES
DDF Activity
ACTIVE vs. INACTIVE

- Two modes of running distributed threads (ZPARM CMTSTAT)
  - ACTIVE – Every connection is a DataBase Access Thread (DBAT), up until it is disconnected, even when waiting for new client transactions
  - INACTIVE – DBAT is pooled (DRDA) or goes inactive (Private Protocol) when the connection issues commit or rollback, and the following conditions are met
    - No WITH HOLD cursors are open
      - ODBC/CLI/JDBC/… clients have a default of WITH HOLD
        - Can be changed by setting CURSORHOLD=0 in db2cli.ini file
    - No Declared Global Temporary Tables exist on the connection
    - No LOB locators are held
    - No package (stored procedure, trigger, UDF, or non-nested task) with KEEPDYNAMIC YES bind option has been accessed
Inactive Connection

- DRDA connections use the Inactive Connection support (previously called type 2 inactive thread)
  - Upon commit, the DBAT is marked in DISCONN state (pooled) and the connection becomes inactive
    - The DBAT can be reused by any inactive connection or any new connection
  - New UOW (Unit of Work) request will be dispatched on a pooled DBAT, if one exists. If all the DBATs are currently in use, DB2 will create a new DBAT. If MAXDBAT is reached, the request is queued.
  - After 200 state switches, DBAT is purged
  - After POOLINAC of time in pool, DBAT is purged
    - Default 120 seconds
    - Best for resource utilisation
      - A small number of threads can typically be used to service a large number of connections
Inactive DBAT

- Private protocol connections still use the old Inactive DBAT support (previously called type 1 inactive thread)
  - Upon commit, DBAT memory footprint is reduced and the DBAT goes inactive
  - New UOW request would cause DBAT to be returned to active set of DBATs and memory footprint expanded
  - Inactive DBAT tied to user’s connection – no thread pooling
  - Potentially requires a large number of threads to support a large number of connections
  - MAXTYPE1 controls how many DBATs using private protocol can go inactive
    - 0 = any DBAT which uses private protocol will stay active (includes any DRDA DBAT which hopped out to another server via private protocol)
    - nnn = maximum number of DBATs using private protocol which can be inactive concurrently (DBAT/connection is aborted if number is exceeded)
-DISPLAY DDF DETAIL command

```
DSNL080I  # DSNLTDFF DISPLAY DDF REPORT FOLLOWS:
DSNL081I  STATUS=STARTD
DSNL082I  LOCATION     LUNAME     GENERICLU
DSNL083I  STL715B     USIBMSY.SYEC715B  -NONE
DSNL084I  TCPPORT=446  RESPORT=5001
DSNL085I  IPADDR=9.30.115.135
DSNL085I  IPADDR=2002:91E:610:1::5
DSNL086I  SQL   DOMAIN=v7ec103.stl.ibm.com
DSNL086I  RESYNC  DOMAIN=v7ec103.stl.ibm.com
DSNL090I  DT=I CONDBAT= 900 MDBAT= 450
DSNL092I  ADBAT= 112 QUEDBAT= 0 INADBAT= 0 CONQUED= 0
DSNL093I  DSCDBAT= 51 INACONN= 540
DSNL099I  DSNLTDDF DISPLAY DDF REPORT COMPLETE
```

**DT** – DDF is configured with DDF THREADS ACTIVE (A) or INACTIVE (I)

**CONDBAT** – Max # of inbound connections as defined in ZPARM CONDBAT

**MDBAT** – Max # of DBATs as defined in ZPARM MAXDBAT

**ADBAT** – Current # of DBATs (assigned + disconnected DBATs back in the pool \(^1\))

**QUEDBAT** – Cumulative counter incremented when MAXDBAT is reached (reset at DDF restart)

**INADBAT** – Current # of inactive DBAT – only applies to Private Protocol \(^1\)

**CONQUED** – Current # of queued connection requests that are waiting to be serviced by a DBAT \(^1\)

**DSCDBAT** – Current # of disconnected DBATs i.e. pooled DBATs available for reuse \(^1\)

**INACONN** – Current # of inactive connections \(^1\)

\(^1\) *Only applies if DDF INACTIVE support is enabled*
Global DDF Activity

CURRENT ACTIVE DBATS
Current # of DBATs (assigned and pooled)

CURRENT DBATS NOT IN USE
Current # of pooled DBATs available for reuse

CUR TYPE 2 INACTIVE DBATS
Current # of inactive connections (!)

CUR QUEUED TYPE 2 INACT THR
Current # of connections requests queued for DBAT

DBAT QUEUED-MAXIMUM ACTIVE
# of times MAXDBAT was reached

CONV.DEALLOC-MAX.CONNECTED
# of times CONDBAT was reached

ACC QUEUED TYPE 2 INACT THR
# of resumed connection requests

DBATS CREATED vs. POOL DBATS REUSED
Indicator of DBAT pooling efficiency

<table>
<thead>
<tr>
<th>GLOBAL DDF ACTIVITY</th>
<th>QUANTITY</th>
<th>/SECOND</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBAT QUEUED-MAXIMUM ACTIVE</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>CONV.DEALLOC-MAX.CONNECTED</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>COLD START CONNECTIONS</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>WARM START CONNECTIONS</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>RESYNCHRONIZATION ATTEMPTED</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>RESYNCHRONIZATION SUCCEEDED</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>CUR TYPE 1 INACTIVE DBATS</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>TYPE 1 INACTIVE DBATS HWM</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>TYPE 1 CONNECTIONS TERMINAT</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>CUR TYPE 2 INACTIVE DBATS</td>
<td>4</td>
<td>N/A</td>
</tr>
<tr>
<td>TYPE 2 INACTIVE DBATS HWM</td>
<td>16</td>
<td>N/A</td>
</tr>
<tr>
<td>ACC QUEUED TYPE 2 INACT THR</td>
<td>14746</td>
<td>0.68</td>
</tr>
<tr>
<td>CUR QUEUED TYPE 2 INACT THR</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>QUEUED TYPE 2 INACT THR HWM</td>
<td>2</td>
<td>N/A</td>
</tr>
<tr>
<td>CURRENT ACTIVE DBATS</td>
<td>10</td>
<td>N/A</td>
</tr>
<tr>
<td>ACTIVE DBATS HWM</td>
<td>26</td>
<td>N/A</td>
</tr>
<tr>
<td>TOTAL DBATS HWM</td>
<td>30</td>
<td>N/A</td>
</tr>
<tr>
<td>CURRENT DBATS NOT IN USE</td>
<td>4</td>
<td>N/A</td>
</tr>
<tr>
<td>DBATS NOT IN USE HWM</td>
<td>20</td>
<td>N/A</td>
</tr>
<tr>
<td>DBATS CREATED</td>
<td>269</td>
<td>N/A</td>
</tr>
<tr>
<td>POOL DBATS REUSED</td>
<td>29325</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Note: All HWM values are since the start of DDF – reset only at DDF restart
DRDA Remote Locs

- **Block fetch**
  - DB2 groups the rows that are retrieved by an SQL query into as large a "block" of rows as can fit in a message buffer
  - Can significantly decrease the number of messages sent across the network
  - Block fetch is used only with cursors that do not update or delete data
    - Open cursor SELECT ... FOR UPDATE disables block fetch

- **ROWS vs. BLOCKS**
  - Indicator of block fetch efficiency

<table>
<thead>
<tr>
<th>DRDA REMOTE LOCS</th>
<th>SENT</th>
<th>RECEIVED</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSACTIONS</td>
<td>778.00</td>
<td>1513.00</td>
</tr>
<tr>
<td>CONVERSATIONS</td>
<td>778.00</td>
<td>1513.00</td>
</tr>
<tr>
<td>CONVERSATIONS QUEUED</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>SQL STATEMENTS</td>
<td>2389.00</td>
<td>283.0K</td>
</tr>
<tr>
<td>SINGLE PHASE COMMITS</td>
<td>0.00</td>
<td>6635.00</td>
</tr>
<tr>
<td>SINGLE PHASE ROLLBACKS</td>
<td>0.00</td>
<td>349.00</td>
</tr>
<tr>
<td>ROWS</td>
<td>250.8k</td>
<td>8733.00</td>
</tr>
<tr>
<td>MESSAGES</td>
<td>342.2k</td>
<td>342.3K</td>
</tr>
<tr>
<td>BYTES</td>
<td>134.0M</td>
<td>79906.4K</td>
</tr>
<tr>
<td>BLOCKS</td>
<td>135.0K</td>
<td>23.00</td>
</tr>
<tr>
<td>MESSAGES IN BUFFER</td>
<td>250.1K</td>
<td></td>
</tr>
<tr>
<td>CONT-&gt;LIM.BLOCK FETCH SWTCH</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>STATEMENTS BOUND AT SERVER</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>
Miscellaneous
# RID List Processing

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QISTRLLM</td>
<td>TERMINATED-EXCEED RDS LIMIT</td>
</tr>
<tr>
<td>QISTRPLM</td>
<td>TERMINATED-EXCEED DM LIMIT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RID LIST PROCESSING</th>
<th>QUANTITY</th>
<th>/SECOND</th>
<th>/THREAD</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAX RID BLOCKS ALLOCATED</td>
<td>8469.00</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>CURRENT RID BLOCKS ALLOCAT.</td>
<td>47.28</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>TERMINATED-NO STORAGE</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>TERMINATED-EXCEED RDS LIMIT</td>
<td>515.00</td>
<td>0.07</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>TERMINATED-EXCEED DM LIMIT</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>TERMINATED-EXCEED PROC.LIM.</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

- RID list processing failures may cause unnecessary CPU resource consumption and possibly unnecessary I/O, as in most cases, DB2 reverts to tablespace scan
  - TERMINATED-EXCEED DM LIMIT
    - Number of RID entries > physical limit (approx. 26M RIDs)
  - TERMINATED-EXCEED RDS LIMIT
    - Number of RIDs that can fit into the guaranteed number of RID blocks > maximum limit (25% of table size)

- Most common reasons
  - Inaccurate or incomplete statistics
    - e.g. old statistics, inadequate or missing distribution statistics collection
  - Use of the LIKE operator in SQL statements
  - Use of host variables or parameter markers for range predicates on SQL statements (BETWEEN, >, <)

- Identify offending applications and SQL statements with accounting reports and/or IFCID 125
Phantom or Orphaned Trace

<table>
<thead>
<tr>
<th>IFC DEST.</th>
<th>WRITTEN</th>
<th>NOT WRTN</th>
<th>BUF.OVER</th>
<th>NOT ACCEP</th>
<th>WRT.FAIL</th>
<th>IFC RECORD COUNTS</th>
<th>WRITTEN</th>
<th>NOT WRTN</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMF</td>
<td>18779.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>SYSTEM RELATED</td>
<td>179.00</td>
<td>0.00</td>
</tr>
<tr>
<td>GTF</td>
<td>1048.1K</td>
<td>491.00</td>
<td>N/A</td>
<td>491.00</td>
<td>0.00</td>
<td>DATABASE RELATED</td>
<td>120.00</td>
<td>0.00</td>
</tr>
<tr>
<td>OP1</td>
<td>1261.8K</td>
<td>56.00</td>
<td>N/A</td>
<td>56.00</td>
<td>N/A</td>
<td>START TRACE</td>
<td>981.7K</td>
<td>536.4K</td>
</tr>
<tr>
<td>OP2</td>
<td>0.00</td>
<td>0.00</td>
<td>N/A</td>
<td>0.00</td>
<td>N/A</td>
<td>STOP TRACE</td>
<td>1.00</td>
<td>3.00</td>
</tr>
<tr>
<td>OP3</td>
<td>0.00</td>
<td>0.00</td>
<td>N/A</td>
<td>0.00</td>
<td>N/A</td>
<td>SYSTEM PARAMETERS</td>
<td>107.00</td>
<td>65.00</td>
</tr>
<tr>
<td>OP4</td>
<td>0.00</td>
<td>0.00</td>
<td>N/A</td>
<td>0.00</td>
<td>N/A</td>
<td>SYS.PARMS-BPOOLS</td>
<td>62.00</td>
<td>0.00</td>
</tr>
<tr>
<td>OP5</td>
<td>0.00</td>
<td>0.00</td>
<td>N/A</td>
<td>0.00</td>
<td>N/A</td>
<td>AUDIT</td>
<td>11.00</td>
<td>0.00</td>
</tr>
<tr>
<td>OP6</td>
<td>0.00</td>
<td>0.00</td>
<td>N/A</td>
<td>0.00</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OP7</td>
<td>0.00</td>
<td>1260.3K</td>
<td>N/A</td>
<td>1260.3K</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OP8</td>
<td>0.00</td>
<td>0.00</td>
<td>N/A</td>
<td>0.00</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES</td>
<td>0.00</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>2328.6K</td>
<td>1260.8K</td>
<td>1260.8K</td>
<td>0.00</td>
<td></td>
<td></td>
<td>982.2K</td>
<td>536.4K</td>
</tr>
</tbody>
</table>

- **IFC RECORD COUNTS NOT WRTN**
  - Phantom or orphaned trace because monitoring (e.g. vendor tool) stopped but the corresponding DB2 trace didn’t
- Same CPU overhead as real trace
- Display Trace to check
- V9 (CM) tries to eliminate orphaned trace records
Within each collection (e.g. “COL_a.*, COL_b.*, COL_c.*”), efficient matching index access to find the package, but DB2 goes serially through the PKLIST entries

Success rate (%) = PACKAGE ALLOC. SUCCESS / PACKAGE ALLOC. ATTEMPT * 100

Impact of long PKLIST search
- Additional CPU resource consumption, catalog accesses, and elapsed time
- Can aggravate DB2 internal latch (LC32) contention

Recommendations
- Reduce the number of collections on the PKLIST
  - Scrub all inactive or unused collections on PKLIST
  - Fold in and collapse the number of collections on PKLIST
- Ruthlessly prioritise and reorder the collection sequence on PKLIST based on frequency of access
- Use SET CURRENT PACKAGESET special register to direct the search to a specific collection
Disabled SPROCs

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QISTCOLS</td>
<td># OF COLUMNS (rows x columns) FOR WHICH AN INVALID SPROC WAS ENCOUNTERED</td>
</tr>
</tbody>
</table>

- Many plans/packages have SPROCs for fast column processing
- As a result of invalidation, DB2 has to build SPROCs dynamically at execution time
  - e.g. V7 to V8 migration, V8 to V9 migration
  - Typical CPU performance impact in 0 to 10% range
- Non-zero value for BYPASS COL indicator of problem
- IFCID 224 identifies plans and packages that need rebinding to re-enable SPROCs
Incremental BIND

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QXINCRB</td>
<td>INCREDMENTAL BINDS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PLAN/PACKAGE PROCESSING</th>
<th>QUANTITY /SECOND</th>
<th>/THREAD</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>INCREDMENTAL BINDS</td>
<td>10138.00</td>
<td>2.82</td>
<td>3.77</td>
</tr>
</tbody>
</table>

- Items that can cause Incremental Bind include
  - Static plan or package with VALIDATE(RUN) and bind time failure
  - Static SQL with REOPT(VARS)
  - Private Protocol in requestor
  - SQL referencing Declared Global Temp Table
  - Possibly DDL statements
Dataset Statistics for I/O Tuning

- Statistics class 8 (IFCID 199)

<table>
<thead>
<tr>
<th>BPOOL</th>
<th>DATABASE</th>
<th>TYPE</th>
<th>SYNCH I/O AVG</th>
<th>ASYNC I/O AVG</th>
<th>SYN I/O AVG DELAY</th>
<th>CURRENT PAGES (VP)</th>
<th>CHANGED PAGES (VP)</th>
<th>CURRENT PAGES (HP)</th>
<th>NUMBER OF GETPAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BP10</td>
<td>KAGURA24</td>
<td>TSP</td>
<td>23.35</td>
<td>0.01</td>
<td>8</td>
<td>3433</td>
<td>0</td>
<td>N/A</td>
<td>2868</td>
</tr>
<tr>
<td></td>
<td>TETHIS</td>
<td>N</td>
<td>32.00</td>
<td></td>
<td>78</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BP11</td>
<td>KAGURA24</td>
<td>IDX</td>
<td>102.59</td>
<td>4.04</td>
<td></td>
<td>18991</td>
<td>74</td>
<td>N/A</td>
<td>245586</td>
</tr>
<tr>
<td></td>
<td>TETHISX1</td>
<td>N</td>
<td></td>
<td>5.98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Count of Sync I/O per second

Average Sync I/O (ms)
John Campbell & Florence Dubois
IBM DB2 for z/OS Development
CampbelJ@uk.ibm.com
fldubois@uk.ibm.com

Session
Optimising DB2 for z/OS System Performance Using DB2 Statistics Trace