Tuning DB2 System Performance using DB2 Statistics Trace

John J. Campbell

DB2 for z/OS Development

Session Code: A14
3rd September 2010 at 12:30pm | Platform: DB2 for z/OS
Objectives

- Focus on key areas
  - System address space CPU, EDM pools, dataset activity, logging, lock/latch contention, DBM1 virtual and real storage, buffer pool, ...
- 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
- Dataset Open/Close
- Buffer Pools
- Lock/Latch
- Logging
- EDM Pool
- DBM1 Virtual Storage and Real Storage
- DDF
- Misc
DB2 System Parameters

- **SMFSTAT**
  - YES (default) starts the trace for the default classes (1, 3, 4, 5, 6)

- **STATETIME**
  - **Recommendation to set to 1 minute**
  - Only 1440 intervals per day
  - Highly valuable / essential to study evolulotional 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
    - Except if collecting STATS Class 8 – may then generate too much SMF data volume

- **SYNCVAL**
  - NO (default) specifies that statistics recording is not synchronized
  - 0 to 59 – DB2 stats interval is synchronized with the beginning of the hour (0 minute past the hour) or with any number of minutes past the hour up to 59
SMF Records

• DB2 Statistics Records are written as SMF 100 records
  • Except IFCID 225 (DBM1 Virtual Storage Usage – Summary)
    • IFCID 225 is written as an SMF type 102 record
  • V9 only with PK37354
    • IFCID 225 is being changed to an SMF type 100 record

• Recommendation to copy SMF 100 (and SMF 102 if do not have PTF for PK37354 applied to DB2 V9 or DB2 9) records, and to keep them separately
  • SMF 100 records represent a relatively small amount of the total SMF data volume
  • Improved elapsed time performance to post process
### System Address Space CPU Time

<table>
<thead>
<tr>
<th>CPU TIMES</th>
<th>TCB TIME</th>
<th>PREEMPT SRB</th>
<th>NONPREEMPT SRB</th>
<th>TOTAL TIME</th>
<th>PREEMPT</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM SERVICES AS</td>
<td>56.529340</td>
<td>0.000231</td>
<td>3:41.430105</td>
<td>4:37.959676</td>
<td>N/A</td>
<td>0.000561</td>
</tr>
<tr>
<td>DATABASE SERVICES AS</td>
<td>7:21.765947</td>
<td>13:45.738985</td>
<td>38:48.563767</td>
<td>59:56.068698</td>
<td>0.036406</td>
<td>0.007252</td>
</tr>
<tr>
<td>IRLM</td>
<td>0.017061</td>
<td>0.000000</td>
<td>3:28.923457</td>
<td>3:28.940518</td>
<td>N/A</td>
<td>0.000421</td>
</tr>
<tr>
<td>DDF AS</td>
<td>5.800377</td>
<td>7:21.291080</td>
<td>21.546607</td>
<td>7:48.638064</td>
<td>0.231706</td>
<td>0.000945</td>
</tr>
<tr>
<td>TOTAL</td>
<td>8:24.112725</td>
<td>21:07.030296</td>
<td>46:20.463936</td>
<td>1:15:51.606957</td>
<td>0.268111</td>
<td>0.009180</td>
</tr>
</tbody>
</table>

- **All TCB times should be low relative to MSTR and DBM1 SRB times**
- **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
- PREEMPT IIP SRB gives the portion of SRB time consumed on a zIIP processor
### TCB and SRB Times – Major Contributors

<table>
<thead>
<tr>
<th>TCB</th>
<th>SRB</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL processing</td>
<td>The same as in TCB case, but only in enclave preemptible SRB mode.</td>
</tr>
<tr>
<td>Lock requests</td>
<td>Reported in TCB instrumentation.</td>
</tr>
<tr>
<td>Buffer updates</td>
<td>Deferred write</td>
</tr>
<tr>
<td>Global lock requests*</td>
<td>Prefetch read</td>
</tr>
</tbody>
</table>

**ACCOUNTING**

- Application
- DBAS
- SSAS
- IRLM

**DATASET ACCESS**

- Dataset Open/Close
  - DBM1 Full System Contraction
  - Preformat
  - Extend

**STATISTICS**

- Archiving
- BSVD processing
- Error checking
- Management

- Local IRLM latch contention
- IRLM and XES global contention*
- Async XES request*
- P-lock negotiation*
- Deadlock detection

Delete Name = pageset close or pseudo-close to convert to non-GBP dependent

(*) Data Sharing specific
WLM Settings General Guidelines

- The subsystems/address spaces VTAM, IRLM, and RRS should be mapped to the service class SYSSTC.
- Prior to V9, the DB2 system address spaces (DBM1, MSTR, DIST, SPAS) and the WLM-managed stored procedure AS should be mapped to a user-defined service class which has an Importance 1 and a very high velocity goal (e.g. 85)
  - Do not recommend putting DBM1 and MSTR AS into SYSSTC because of risk of DB2 misclassifying incoming work.
- However with V9, move MSTR into service class SYSSTC
  - DB2 Service Monitor is built into and operates within MSTR.
  - DB2 Service Monitor must be running higher than the AS it monitors.
  - DB2 Service Monitor will be impacted if MSTR is pre-empted.
- If running at very high CPU utilisation with possibility of CPU starvation
  - Move all the DB2 system address spaces (DBM1, MSTR, DIST, SPAS) to service class SYSSTC.
WLM Settings General Guidelines ...

- No DB2 application workload should run higher than the DB2 system AS
- No DB2 application workload should run as discretionary
- For CICS and IMS, use 80th or 90th percentile response time goals as transactions are typically non-uniform
  - The transaction response time goals must be practically achievable
    - Use RMF Workload Activity Report to validate
-DISPLAY THREAD(*) SERVICE(WAIT)

- Identifies allied agents and distributed DBATs that have been suspended for more than 60 seconds or twice the IRLM resource timeout interval which ever is the greater
- If the thread is suspended due to IRLM resource contention or DB2 latch contention, additional information is displayed to help identify the problem

```
2004061 21:23:01.31 STC42353 DSNV401I -DT45 DISPLAY THREAD REPORT FOLLOWS -
  DSNV402I -DT45 ACTIVE THREADS -
  NAME ST A REQ ID AUTHID PLAN ASID TOKEN
  SERVER RA *  4 java LGB0031 DISTSERV 00C9 117718
  V490-SUSPENDED 04061-21:19:57.83 DSNTLSUS +00000046 14.21
  V437-WORKSTATION=kryten, USERID=1gb0031,
    APPLICATION NAME=java
  V445-G91E84C9.DC84.040305155824=117718 ACCESSING DATA FOR
    9.30.132.201
  CT45A T * 12635 ENTRU1030014 ADMF010 REPN603 00C6 90847
  V490-SUSPENDED 04061-19:30:04.62 DSNTLSUS +00000046 14.21
```

- Note that DISTSERV may return false positives for DBATs that have not been in use for more than the x seconds
-DISPLAY THREAD(*) SERVICE(WAIT) ...

- Will also attempt to dynamically boost priority (via WLM services) for any latch holder that appears to be stuck
- Can run with SCOPE(GROUP)
  - The command is forwarded onto all the other DB2 members in the group via notify message and each member runs it as if it was a local command
  - The output is piped back to the originating member via notify message
- Recommendations
  - V8: Strongly recommend to run at one-minute interval through automation
  - V9: Driven at one-minute interval by the internal DB2 System Monitor
    - New built-in monitor in DB2 9 (CM) effectively does the time-based automation
    - Does not display the output of the command
    - See DSNV507I message on -DISPLAY THREAD(*) TYPE(SYSTEM) output
WLM Blocked Workload

- Introduced on z/OS V1.9 (rolled back to 1.8 and 1.7 via APAR OA17735)
- Objective: Allow small amounts of CPU to be allocated to workloads that are CPU starved when system is running at 100% utilisation
- Controlled by two new parameters in IEAOPTxx parmlib member
  - BLWLINTHD – Specifies the threshold time interval for which a blocked address space or enclave must wait before being considered for promotion
    - Default value (after OA22443) is 20 seconds
  - BLWLTRPCT – Specifies how much of the CPU capacity is to be used to promote blocked workloads
    - Default value (after OA22443) is 5 (i.e. 0.5%)
  - For more information, see APARs OA17735, OA22443, and techdoc
- Recommendation: All customers should run with this function on
### Dataset Open/Close

<table>
<thead>
<tr>
<th>TOTAL BUFFERPOOLS</th>
<th>TOTAL</th>
<th>AVG/SEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER OF DATASET OPENS</td>
<td>18</td>
<td>0.30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OPEN/CLOSE ACTIVITY</th>
<th>TOTAL</th>
<th>AVG/SEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN DATASETS - HWM</td>
<td>18001</td>
<td></td>
</tr>
<tr>
<td>OPEN DATASETS</td>
<td>17803</td>
<td></td>
</tr>
<tr>
<td>DSETS CLOSED-THRESH.REACHED</td>
<td>100</td>
<td>1.64</td>
</tr>
</tbody>
</table>

- Frequent dataset opens are typically accompanied by high DBM1 TCB time and/or Accounting Class 3 Dataset Open wait 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
Bufferpool Page 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
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
Bufferpool Hit Ratio and Page Residency Time

- **Hit Ratios** = percentage of times the page was found in the BP
  - System HR = \( \frac{\text{Total getpages} - \text{Total pages read}}{\text{Total getpages}} \times 100 \)
  - Appli. HR = \( \frac{\text{Total getpages} - \text{Synchronous reads}}{\text{Total getpages}} \times 100 \)
- **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
  • Wait for for up to 40 changed pages for 4K BP (24 for 8K, 16 for 16K, 12 for 32K)
  • 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)
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)*
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 …

- 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 …

- In V8, 64bit IRLM is supported
  - PC=YES is forced
  - Reduced requirement for ECSA
- Make sure of high IRLM dispatching priority
  - Always 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

<table>
<thead>
<tr>
<th></th>
<th>TIMEOUT</th>
<th>DEADLOK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range allowed</td>
<td>1 to 3600sec</td>
<td>0.1 to 5sec</td>
</tr>
<tr>
<td>Default</td>
<td>60sec</td>
<td>1sec</td>
</tr>
<tr>
<td>Recommendation</td>
<td>30sec</td>
<td>0.5sec</td>
</tr>
</tbody>
</table>
IRLM Latch Contention

<table>
<thead>
<tr>
<th>LOCKING ACTIVITY</th>
<th>AVG/COMMIT</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCK REQUESTS</td>
<td>31.94</td>
<td>6258309</td>
</tr>
<tr>
<td>UNLOCK REQUESTS</td>
<td>7.39</td>
<td>1447822</td>
</tr>
<tr>
<td>CHANGE REQUESTS</td>
<td>1.03</td>
<td>201435</td>
</tr>
<tr>
<td>SUSPENSIONS (LOCK ONLY)</td>
<td>0.02</td>
<td>4417</td>
</tr>
<tr>
<td>SUSPENSIONS (IRLM LATCH)</td>
<td>1.33</td>
<td>260416</td>
</tr>
<tr>
<td>SUSPENSIONS (OTHER)</td>
<td>0.05</td>
<td>9401</td>
</tr>
</tbody>
</table>

- **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%
Thread Reuse

- Performance opportunity for high volume simple transactions
- Thread reuse can be monitored using the Accounting Trace
  - \((\#\text{COMMENTS}-\#\text{DEALLOCATION}) \times 100/\#\text{COMMENTS}\) is a good estimation of the level of thread reuse

<table>
<thead>
<tr>
<th>NORMAL TERM.</th>
<th>AVERAGE</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW USER</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>DEALLOCATION</td>
<td>0.33</td>
<td>212799</td>
</tr>
<tr>
<td>RESIGNON</td>
<td>0.67</td>
<td>429710</td>
</tr>
</tbody>
</table>

In this example, \((974827 - 212799) \times 100/974827 = 78\%\) of thread reuse
### Internal DB2 Latch Contention/Second

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

- **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
Log Statistics

<table>
<thead>
<tr>
<th>LOG ACTIVITY</th>
<th>AVG/COMMIT</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>READS SATISFIED-OUTPUT BUFF</td>
<td>2.3</td>
<td>1951.0K</td>
</tr>
<tr>
<td>READS SATISFIED-ACTIVE LOG</td>
<td>33.17</td>
<td>28132.0K</td>
</tr>
<tr>
<td>READS SATISFIED-ARCHIVE LOG</td>
<td>0.00</td>
<td>0</td>
</tr>
</tbody>
</table>

• “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 …

<table>
<thead>
<tr>
<th>LOG ACTIVITY</th>
<th>AVG/COMMIT</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNAVAILABLE OUTPUT LOG BUFF</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>OUTPUT LOG BUFFER PAGED IN</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>LOG RECORDS CREATED</td>
<td>14.58</td>
<td>4031.7K</td>
</tr>
<tr>
<td>LOG CI CREATED</td>
<td>0.68</td>
<td>188.2K</td>
</tr>
<tr>
<td>LOG WRITE I/O REQ (LOG1&amp;2)</td>
<td>2.06</td>
<td>569.7K</td>
</tr>
<tr>
<td>LOG CI WRITTEN (LOG1&amp;2)</td>
<td>2.35</td>
<td>649.9K</td>
</tr>
<tr>
<td>LOG RATE FOR 1 LOG (MB)</td>
<td>N/A</td>
<td>2.12</td>
</tr>
</tbody>
</table>

- **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
    \[= \frac{(\text{LOG CIs CREATED} \times 4\text{KB})}{(\text{LOG RECORDS CREATED})}\]
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 logging rate near ‘practical’ maximum
  - Try to 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
EDM Pools – V8 Picture

Global Dynamic Statement Cache (Dynamically Prepared Statements)

EDM Pool (CT/PT SKCT/SKPT)

- SKDS
- DBD
- CT
- PT
- SKCT
- SKPT

EDMSTMTC

EDMDBDC

EDMPOOL

DBM1 Address Space

Database Descriptors Pool (DBD)

2GB Bar
EDM Pools – V9 Picture

- **RDS Pool Above** (part of CT/PT that can be above)
- **Global Dynamic Statement Cache** (Dynamically Prepared Statements)
- **RDS Pool Below** (part of CT/PT that must be below)

Diagram:

- **DBM1 Address Space**
- **Skeleton Pool** (SKCT/SKPT)
- **Database Descriptors Pool** (DBD)
- **2GB Bar**

Diagram sections:

- **EDM_SKELETON_POOL**
  - **SKCT**
  - **SKPT**
- **EDMSTMTCC**
  - **SKDS**
- **EDMDBDC**
  - **DBD**
- **EDMPOOL**
  - **CT**
  - **PT**
## EDM Pool – Tuning V8

<table>
<thead>
<tr>
<th>EDM POOL</th>
<th>QUANTITY</th>
<th>/SECOND</th>
<th>/THREAD</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGES IN EDM POOL (BELOW)</td>
<td>70000.00</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>HELD BY CT</td>
<td>138.88</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>HELD BY PT</td>
<td>2492.74</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>HELD BY SKCT</td>
<td>559.90</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>HELD BY SKPT</td>
<td>63811.66</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FREE PAGES</td>
<td>2996.82</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>% PAGES IN USE</td>
<td>95.72</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>% NON STEAL. PAGES IN USE</td>
<td>3.76</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FAILS DUE TO POOL FULL</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

\[
% \text{NON-STEAL. PAGES IN USE} = \frac{(\text{HELD BY CT} + \text{HELD BY PT})}{\text{PAGES IN EDM POOL}} \times 100
\]

\[
\text{CT/PT HIT RATIO} = \frac{(\text{CT/PT REQUESTS} - \text{CT/PT NOT FOUND})}{\text{CT/PT REQUESTS}} \times 100
\]
EDM Pool – Tuning V8 …

- If EDM Pool is too small
  - Fewer threads can run concurrently
    - Race condition with CTHREAD/MAXDBAT
      - Queuing or resource unavailable ‘-904’
    - Need a balanced configuration
  - Increased response time due to loading of SKCT, SKPT
  - Increased I/O against SCT02 and SPT01

Increase EDM Pool size as needed to keep
- FAILS DUE TO POOL FULL = 0 ** Very serious condition **
- % NON-STEALABLE PAGES IN USE < 50%
- CT/PT HIT RATIO > 95 to 99%

Note: No need to tune for 50% FREE PAGES
## EDM Pool and Skeleton Pool – Tuning V9

<table>
<thead>
<tr>
<th>EDM POOL</th>
<th>QUANTITY</th>
<th>/SECOND</th>
<th>/THREAD</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGES IN RDS POOL (BELOW)</td>
<td>12576.00</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>HELD BY CT</td>
<td>84.14</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>HELD BY PT</td>
<td>309.74</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FREE PAGES</td>
<td>12182.12</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FAILS DUE TO POOL FULL</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>PAGES IN RDS POOL (ABOVE)</td>
<td>524.3K</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>HELD BY CT</td>
<td>0.00</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>HELD BY PT</td>
<td>137.15</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FREE PAGES</td>
<td>524.1K</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FAILS DUE TO RDS POOL FULL</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>PAGES IN SKEL POOL (ABOVE)</td>
<td>1280.00</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>HELD BY SKCT</td>
<td>61.85</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>HELD BY SKPT</td>
<td>1139.35</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FREE PAGES</td>
<td>78.81</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FAILS DUE TO SKEL POOL FULL</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>CT REQUESTS</td>
<td>221.1K</td>
<td>10.26</td>
<td>1.33</td>
<td>0.45</td>
</tr>
<tr>
<td>CT NOT FOUND</td>
<td>24899.00</td>
<td>1.16</td>
<td>0.15</td>
<td>0.05</td>
</tr>
<tr>
<td>CT HIT RATIO (%)</td>
<td>88.74</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>PT REQUESTS</td>
<td>4803.3K</td>
<td>222.99</td>
<td>28.90</td>
<td>9.69</td>
</tr>
<tr>
<td>PT NOT FOUND</td>
<td>540.1K</td>
<td>25.07</td>
<td>3.25</td>
<td>1.09</td>
</tr>
<tr>
<td>PT HIT RATIO (%)</td>
<td>88.76</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

% PAGES IN USE BY CT/PT BELOW = (HELD BY CT + HELD BY PT) / PAGES IN EDM POOL BELOW * 100

CT/PT HIT RATIO = (CT/PT REQUESTS – CT/PT NOT FOUND) / CT/PT REQUESTS * 100
EDM Pool and Skeleton Pool – Tuning V9 ...

- If EDM Pool Below 2GB is too small
  - Fewer threads can run concurrently
    - Race condition with CTHREAD/MAXDBAT
      - Queuing or resource unavailable “-904”
    - Need a balanced configuration

Increase size of RDS Pool Below 2GB as needed to keep

- FAILS DUE TO POOL FULL = 0 ** Very serious condition **
- % PAGES IN USE BY CT/PT BELOW < 50%

- If Skeleton Pool is too small
  - Increased response time due to loading of SKCT, SKPT
    - Increased I/O against SCT02 and SPT01

Increase Skeleton Pool size as needed to keep

- CT/PT HIT RATIO > 95 to 99%
Other EDM Recommendations – V8 and V9

- To keep the EDM Pool storage under control
  - Use BIND option RELEASE(COMMIT)
    - RELEASE(DEALLOCATE) should only be used for the most frequently executed plans/packages
- EDMBFIT=YES|NO?
  - Affects space search algorithm for all EDM Pools
    - EDMBFIT NO (‘first fit’ algorithm) >> Best for performance
    - EDMBFIT YES (‘best fit’ algorithm) >> Best for space utilisation
  - Use of EDMBFIT YES can be ‘worst fit’ algorithm
    - May lead to very high EDM LRU (LC24) contention on large pools
      - Resulting in non-linear DB2 performance
    - If LC24 contention rate is greater than 1000 per second
      - Switch to EDMBFIT=NO
  - General recommendation: Always use EDMBFIT=NO (default)
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 KEEPDYNAMIC(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 ...

<table>
<thead>
<tr>
<th>DYNAMIC SQL STMT</th>
<th>AVG/COMMIT</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREPARE REQUESTS</td>
<td>58.29</td>
<td>102.7K</td>
</tr>
<tr>
<td>FULL PREPARES</td>
<td>0.76</td>
<td>1335</td>
</tr>
<tr>
<td>SHORT PREPARES</td>
<td>57.60</td>
<td>101.5K</td>
</tr>
<tr>
<td>IMPLICIT PREPARES</td>
<td>6.65</td>
<td>11.7K</td>
</tr>
<tr>
<td>PREPARES AVOIDED</td>
<td>11.09</td>
<td>19.5K</td>
</tr>
<tr>
<td>CACHE LIMIT EXCEEDED</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>PREP STMT PURGED</td>
<td>0.00</td>
<td>0</td>
</tr>
</tbody>
</table>

- Global Dynamic Statement Cache hit ratio should be > 90-95%  
  \[ \text{[Short Prepares]} / \text{[Short + Full Prepares]} \]  
  \[ = \frac{57.60}{57.60 + 0.76} = 98.70\% \]
- Local Dynamic Statement Cache hit ratio should be > 70%  
  \[ \text{[Prepares Avoided]} / \text{[Prepares Avoided + Implicit Prepares]} \]  
  \[ = \frac{11.09}{11.09 + 6.65} = 62.51\% \]  
  - 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
DBD Pool

### Table

<table>
<thead>
<tr>
<th>EDM POOL</th>
<th>QUANTITY</th>
<th>/SECOND</th>
<th>/THREAD</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGES IN DBD POOL (ABOVE)</td>
<td>25600.00</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>HELD BY DBD</td>
<td>25458.84</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FREE PAGES</td>
<td>141.16</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FAILS DUE TO DBD POOL FULL</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>DBD REQUESTS</td>
<td>19197.4K</td>
<td>891.24</td>
<td>115.49</td>
<td>38.72</td>
</tr>
<tr>
<td>DBD NOT FOUND</td>
<td>52.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>DBD HIT RATIO (%)</td>
<td>100.00</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

- If DBD Pool is too small
  - Increased response time due to loading of DBD
  - Increased I/O against DBD01

- To keep the DBD Pool storage under control
  - Try to minimise the number of objects in a database
    - Especially in data sharing environment where multiple DBD copies can be stored in the DBD Pool due to DBD invalidation by other members
    - Smaller DBDs also help with contention (see Locking)
  - Compact DBD by REORG and MODIFY if many DROP TABLE in segmented tablespace

Increase DBD Pool size as needed to keep \( \text{DBD HIT RATIO} > 95 \text{ to } 99\% \)
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

Vertical scalability of DB2 member is limited
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: \texttt{VSTOR(D,xxxxDBM1)}
    - Produced by RMF Post Processor option: \texttt{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
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
    - New 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 Developer Works
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
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)

- 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
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
      - 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)
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
Basic Graphing

• Check the major components of DB2 storage to get an idea of the workload
  • Fixed
  • Getmained
  • Stack
  • Variable
  • Thread counts
Basic graphing of storage - leaky subsystem? 
First 7 days data (Mon-Sun)
Basic graphing – what happened next Mon-Wed
This DB2 took a full week to “warm up”
**Storage Overuse: DB2 Storage Contraction**

- When ‘running low’ on extended 31-bit virtual storage, 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
    - $= (\text{CTHREAD} + \text{MAXDBAT} + 1) \times 64K$ (Fixed, real value)
  - Storage reserved for open/close of datasets – QW0225MV
    - $= (\text{DSMAX} \times 1300) + 40K$ (Virtual number and no guarantee)
  - Warning to contract – QW0225SO
    - $= \text{Max (5\% of Extended Region Size, QW0225CR)}$
  - Storage Cushion = QW0225CR + QW0225MV + QW0225SO
How much storage is left in the DBM1 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
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>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Storage reserved for must complete (MB)</td>
<td>250</td>
<td>150</td>
<td>38</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>250</td>
<td>150</td>
<td>50</td>
</tr>
<tr>
<td>Storage Cushion (MB)</td>
<td>519</td>
<td>319</td>
<td>107</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 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 + 10% of Extended Region Size
    - 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>200</td>
</tr>
<tr>
<td>MVS extended region size (MB)</td>
<td>1700</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Storage Cushion (MB)</td>
<td>544</td>
<td>344</td>
<td>132</td>
</tr>
<tr>
<td>Basic Cushion (MB)</td>
<td>644</td>
<td>444</td>
<td>232</td>
</tr>
<tr>
<td>% of Extended Region Size</td>
<td>37%</td>
<td>44%</td>
<td>23%</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 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 …

- Remember to use the MAX impact value across all available data e.g. maximum system storage
- ‘Basic’ storage cushion (BC)
  - (BC) = QW0225CR + QW0225MV + QW0225SO + 10% of QW0225RG
- Calculate Max non-DB2 storage (ND)
  - (ND) = MAX(MVS 31 BIT EXTENDED HIGH PRIVATE QW0225EH – TOTAL GETMAINED STORAGE QW0225GM – TOTAL GETMAINED STACK STORAGE QW0225GS – TOTAL FIXED STORAGE QW0225FX – TOTAL VARIABLE STORAGE QW0225VR)
- Max. allowable storage (AS)
  - (AS) = QW0225RG – (BC) – (ND)
- Max. allowable storage for thread use (TS)
  - (TS) = (AS) – MAX(TOTAL AGENT SYSTEM STORAGE QW0225AS + TOTAL FIXED STORAGE QW0225FX + TOTAL GETMAINED STORAGE QW0225GM + MVS 31 BIT EXTENDED LOW PRIVATE QW0225EL)
- Average thread footprint (TF)
  - (TF) = (TOTAL VARIABLE STORAGE QW0225VR – MAX(TOTAL AGENT SYSTEM STORAGE QW0225AS) + TOTAL GETMAINED STACK STORAGE QW0225GS) / (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 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
- 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
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 active 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 # DSNLTDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I STATUS=STARTD
DSNL082I LOCATION LUNAME GENERICLU
DSNL083I STL715B USIBMSY.SYEC715B -NONE
DSNL084I TCP_PORT=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)

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

**INADBAT** – Current # of inactive DBAT – only applies to Private Protocol

**CONQUED** – Current # of queued connection requests that are waiting to be serviced by a DBAT

**DSCDBAT** – Current # of disconnected DBATs i.e. pooled DBATs available for reuse

**INACONN** – Current # of inactive connections

*(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 Locations

- **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: DRDA Remote Locs

<table>
<thead>
<tr>
<th></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 SWITCH</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>STATEMENTS BOUND AT SERVER</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>
**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 ALLOCATED</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

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

- 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

<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>

---- MISCELLANEOUS -----
BYPASS COL: 1585.00
Incremental BIND

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>PLAN/PACKAGE PROCESSING</th>
<th>QUANTITY</th>
<th>/SECOND</th>
<th>/THREAD</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>QXINCRB</td>
<td>INCREMENTAL BINDS</td>
<td></td>
<td>10138.00</td>
<td>2.82</td>
<td>3.77</td>
<td>0.33</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>SYN I/O MAX DELAY</th>
<th>CURRENT PAGES (VP)</th>
<th>CURRENT PAGES (HP)</th>
<th>NUMBER OF GETPAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP10</td>
<td>KAGURA24</td>
<td>TSP</td>
<td>23.35</td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>3433</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.01</td>
<td></td>
<td></td>
<td>78</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32.00</td>
<td></td>
<td></td>
<td>N/A</td>
<td>2868</td>
<td></td>
</tr>
<tr>
<td>BP11</td>
<td>KAGURA24</td>
<td>IDX</td>
<td>102.59</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>18991</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.08</td>
<td></td>
<td></td>
<td>35</td>
<td>74</td>
<td>245586</td>
</tr>
</tbody>
</table>

- **Count of Sync I/O per second**
- **Average Sync I/O (ms)**
John J. Campbell
DB2 for z/OS Development
CampbelJ@uk.ibm.com
A14
Tuning DB2 System Performance using DB2 Statistics Trace