Trace Records: The Answer to All DB2 for z/OS Performance Questions

Bart Steegmans

DB2 for z/OS L2 Performance

IBM

GSE DB2 for z/OS
March 23 2017
Please Note:

- IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

- Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

- The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

- The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user’s job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.
Outline

• DB2 Instrumentation Trace Facility
  ▪ IFCIDs
  ▪ Trace classes
  ▪ Trace destinations

• Which Data / Traces Should I Collect When?
  ▪ Traces you should have on all the time
    • Stats
    • Accounting
    ▪ More detailed trace to zoom in on a problem case

• Traces for specific problems (usage scenarios)
  ▪ Locking problem
  ▪ SQL performance problem
  ▪ Long service time suspensions (EU-switch)
Introducing DB2 Instrumentation Facility

- Aka DB2 tracing
  - Contain a wealth of information
- DB2 traces are grouped in **trace types**
- Trace types use different **trace classes**
- Trace classes consist of **one or more IFCIDs**
- IFCID = Instrumentation Facility Component ID
  - This is an individual trace record

```
1  3  4  5  6  7  8  9
  |     |     |     |
105  106 202 225
```

```
1  2  3  5  11
     |     |
     239
```

```
7  8  10
    |    |
    3
```

```
1  2
   |
105
```

```
3
```

```
4
```

```
5
```

```
6
```

```
7
```

```
8
```

```
9
```

```
10
```

```
Global
```

```
Monitor
```

```
Performance
```

```
Audit
```

```
Statistics
```

```
Accounting
```

```
Performance
```

```
Global
```

```
Monitor
```

```
Audit
```

```
Statistics
```

```
Accounting
```

```
Performance
```

```
Global
```

```
Monitor
```

```
Audit
```
Introducing IFCIDs

- IFCIDs can have their own trace record
  - Like most performance trace IFCIDs
    - IFCID 21 (lock detail) creates an IFCID 21 trace record
- IFCIDs can add fields to other IFCIDs
  - Like accounting IFCIDs
    - IFCID 232 (class2 times) adds the class 2 information to the IFCID 3 record
- Can do both
  - Like accounting IFCIDs when they are part of a performance trace
    - IFCID 226-227 (page latch start/end) adds this info to accounting class 3
    - Can also produce performance trace records as part of performance trace class(4)
- Same IFCID can be part of multiple trace types/classes
  - IFCID 172 (deadlock) is part of stats class(3) and performance class (6)
- IFCIDs can be part of no trace class
  - Always explicitly specified like IFCID 198 (getpage activity)
When are trace records written

- Vast majority is written *when event occurs*. Actually most of the time when the event is over, and it passes the filter criteria

- Sometimes *timer driven*
  - DB2 statistics info (`STATTIME`), or fixed intervals (like 1 min. stats)
  - DB2 dataset level statistics (`DSSTIME`) indicates when dataset level stats are to be reset for online monitors

- When a *thread de-allocates, is reused, goes inactive, rollup threshold is reached*, …
  - DB2 accounting

- When a *trace is started*
  - Eg. DB2 stats record is written when stats trace is started
  - (or modified which is an internal stop/start trace)

- *Never* written
  - Some trace records are never externalized; they serve as switches to add info to certain records, or are only written to an internal trace table that shows up in an SVC dump
Trace commands

• Starting a DB2 trace
  ▪ -START TRACE
  ▪ Or automatic at DB2 startup for some trace types (STA, ACCTG, MON, AUDIT, GLOBAL(probably not a good idea)

• Displaying a DB2 trace
  ▪ -DISPLAY TRACE

• Changing an existing trace
  ▪ -MODIFY TRACE
    • Performs a STO/STA trace under the covers

• Stop a trace
  ▪ -STOP TRACE
-START TRACE command

-START TRACE ( ACCTG )
  STAT
  PERFM
  MONITOR
  AUDIT
  GLOBAL
  DEST ( SMF, GTF, Opn, OPX, SRV )
  SCOPE ( LOCAL )
    GLOBAL
  CLASS (1,2,3,4...)
  AUTHID PLAN PKGLOC PKGCOL PKGPROG LOCATION
  USERID APPNAME WRKSTN CONNID CORRID ROLE
  XAUTHID XPLAN XPAR LOC
  XUSERID XAPPNAME XWRKSTN XCONNID XCORRID XROLE
  AUDTPLCY ( ... )
  TDATA ( COR, TRA, CPU, DIST )
  RMID ( Resource-Manager-ID )
  ASID ( x’yyyy’ )
  IFCID ( 63, 305,...)
  BUFSIZE ( OP-Buffer-Size )
  COMMENT ( ’........’ )
Trace Filtering

- In DB2 9 introduced extensive enhancement to instrumentation filtering
  - Filters now include PLAN, LOCATION, AUTHID, USERID, APPNAME, WRKSTN, PKGPROG, PKGLOC, PKGCOL, CONNID, CORRID, ROLE and exlude keywords for each (eg. XPLAN)
- Terminating and positional wildcards are allowed (eg. PLAN(DSNTEP*) , PLAN(PLAN_01))
  - More details in the appendix
- Trace filtering is important to reduce the trace volume (and trace overhead)
Using the IFCID keyword

- Is used to specify additional IFCIDs on top of the IFCIDs that are part of the trace class that you are starting
  - Eg. –STA TRA(P) C(1) IFCID(21)
  - This will trace IFCID 001,002, 031,042,043,076,077,078,079,102,103 105,107,106,153 (that make up perf class 1) and also IFCID 21
  - Using –STA TRA(P) C(1) IFCID(42,43) will still trace all the IFCIDs that are part of perf class 1. Specifying IFCID(42,43) is not needed as they are already part of perf class 1.

- If you only want to trace certain IFCIDs, use an “empty” trace class. That is a class that has no IFCIDs assigned to it by default.
  - For the performance trace, these are classes 30, 31, 32
  - Eg. To only trace IFCID 42,43 (system checkpoint sta/end), use - STA TRA(P) C(32) IFCID(42,43)
Using the TDATA keyword

- Is used to specify which trace headers you want DB2 to use when writing the trace record.
- The default trace headers depend on the trace type.
- Best practice: Always specify TDATA to make sure you have the headers you want.

  - **COR**: Correlation header
    - (Good stuff such as connection type, correlation-id, authid, plannname, workstation-name, … (not package name!))
  
  - **CPU**: CPU header
    - CPU time of the currently dispatched execution unit (TCB or SRB)
    - Includes (‘normalized’) CPU consumed on an specialty engine
    - Important to specify for performance traces
    - V11 zIIP time in separate counter
Using the DEST keyword

- **SMF**—System Management Facility
  - Typically used for ‘standard monitoring’, stats, acctg, and audit
- **GTF**—Generalized Trace Facility
  - Typically used for ‘detailed monitoring’, performance traces
- **OPn / OPX**—Destination for trace output
  - Used by monitor programs that issue IFI READA
- **[RES—Resident trace table]**
  - Used for global trace
  - Will show up in a dump (Size of trace table controlled by TRACTBL DSNZPARM )
- **[SRV—Serviceability routine]**
  (never seen it used other than DB2 internally)

- Always keep an eye out for:
  - **DSNW133I csect-name TRACE DATA LOST, dest NOT ACCESSIBLE RC=code**
Using SMF as trace destination

- Losing trace records usually not a problem
- Flooding could be a problem
  - As SMF is often used for charge back accounting
  - When using logstreams for SMF, it should be able to handle large trace volumes
- Usually used for stats and accounting - Not often for performance traces
  - Rollup accounting (ACCUMACC) to reduce the volume
  - Use SMFCOMP=YES to compress trace for SMF
- SMF record types used by DB2
  - SMF 100 = Mainly DB2 statistics (IFCID 1, 2, 202, 225, 230)
  - SMF 101 = DB2 accounting (IFCID 3, 239)
  - SMF 102 = Mainly DB2 performance (All other IFCIDs including IFCID 0 (global trace))
- SMF must be active and SMFPRMxx member allows 100-102 eg. SYS(TYPE(0:255),...)
Using GTF as trace destination

- Mainly used for performance traces
- Losing trace records can be a problem
- GTF particularities:
  - GTF data set is used in wrap-around
  - GTF data set cannot use extents
  - GTF can use multiple data sets in parallel
    - Sample JCL in appendix
  - GTF can use more buffers
  - Set up to only trace DB2 events
- To monitor via GTF use:
  - Start GTF proc
  - Start DB2 traces
  - Trace what is needed
  - Stop DB2 traces
  - Stop GTF proc
Displaying a DB2 trace

- IFCIDs and qualifications are displayed with -DIS TRA(*) DETAIL(*)

```
DSNW127I -DB9A CURRENT TRACE ACTIVITY IS -
TNO TYPE     CLASS        DEST QUAL IFCID
 01  STAT    01,03,04,05, SMF  NO
 01         06
 02  ACCTG   01,02,03,07, SMF  NO
 02         08
 03  ACCTG   01           OP1  NO
 04  PERFM   01           GTF   YES 198
**********END OF DISPLAY TRACE SUMMARY
DATA**********   DSNW143I -DB9A CURRENT TRACE
QUALIFICATIONS ARE -   DSNW152I -DB9A BEGIN TNO
01 QUALIFICATIONS:      DSNW152I -DB9A BEGIN TNO 01 QUALIFICATIONS:
NO QUALIFICATIONS     DSNW152I -DB9A BEGIN TNO 02 QUALIFICATIONS:
END TNO 01 QUALIFICATIONS
DSNW152I -DB9A BEGIN TNO 02 QUALIFICATIONS:
NO QUALIFICATIONS     DSNW152I -DB9A BEGIN TNO 03 QUALIFICATIONS:
END TNO 02 QUALIFICATIONS
DSNW152I -DB9A BEGIN TNO 03 QUALIFICATIONS:
NO QUALIFICATIONS     DSNW152I -DB9A BEGIN TNO 04 QUALIFICATIONS:
END TNO 03 QUALIFICATIONS
DSNW152I -DB9A BEGIN TNO 04 QUALIFICATIONS:
EXCLUDE PLAN: ABC*
END TNO 04 QUALIFICATIONS
DSNW148I -DB9A *****END OF DISPLAY TRACE
QUALIFICATION DATA*****
DSN9022I -DB9A DSNWVCM1 '-DIS TRACE' NORMAL COMPLETION
```
Tracing Details

- Trace overhead consists of:
  - Collecting trace data
  - Externalizing trace data
- When the IFCID is active, trace data is collected
- Before writing the trace record, the filter criteria are checked
  - When the trace record passes the filter criteria it is written out to the trace destination
  - If not, the trace record is discarded
  - So a restrictive filter saves in output not in trace collection overhead
- Filter criteria checked when record is written based on who is writing it
  - Prefetch I/O is not traced if PLAN(xxx) specified, as prefetch is done under a system thread, not under the thread running plan xxx
  - Using XPKGPROG(MYPACK) to avoid package info to be written for MYPACK only works if MYPACK is the active package when the record is written
Accounting is “special” in that starting additional trace classes doesn’t create extra trace records,

- They just add counters to existing records
- The result of that is that if
  - STA TRA(A) C(1) DEST(SMF)
  - STA TRA(A) C(1,2,3) DEST(OPX)
- The SMF records will also contain the class 2 and 3 data
- The record is only built once in storage

DB2 instrumentation facility is usually not effective to trace events during startup and shutdown

DB2 Trace records are described in:
- hlq.SDSNIVPD(DSNWMSGS)
- DSNDQxxx macros of hlq.SDSNMACS (most up to date)
DB2 Trace Types

- Statistics Trace
- Accounting Trace
- Audit Trace
- Performance Trace
- Monitor Trace
- [Global Trace]
Statistics Trace

- DB2 Statistics reports are typically based on IFCID 1,2, 225.
- Used as a prime indicator for **subsystem-related** problems
- Contains (an awful lot of) information about
  - SQL usage (DML, DCL, DDL, direct row access) and DB2 commands
  - Stored proc, triggers, UDFs
  - EDM pool
  - Subsystem services
  - Open/Close activity
  - Log activity
  - Plan/package processing
  - DB2 commands
  - RID list processing
  - Dynamic statement cache
  - Accelerator data
  - Authorization management
  - Locking activity / Data-sharing locking
  - Query parallelism
  - CPU times
  - DB2 IFI requests, IFC, and data capture
  - DB2 latch counters
  - Buffer pool and Group buffer pool activity
  - DDF activity
  - Storage statistics
  - Workfile database
  - Aggregated accounting statistics
Statistics Trace ...

- Stats class 1 - IFCID 1, 2,105,106, 202, 225
  - Since DB2 10 IFCID 1, 2, 202, (217), 225, and 230 no longer controlled by STATIME. Written at fixed, one-minute intervals.
- Stats class 3 – IFCID 172 , 196, 250, 258, 261, 262, 313, 330, 335, 337 (deadlock, timeout, long running threads)
- Stats class 4 – mainly DRDA exceptions and ASUTIME exceeded
- Stats class 5 - Data sharing statistics
- Stats class 7 – IFCID 365 (DDF location statistics)
- Stats class 8 - IFCID 199 (dataset I/O statistics)
  - Can be very useful to identify ‘hot’ objects
- Stats class 9 – IFCID 369 (Aggregated Accounting Statistics)
  - Only when accounting is also active
  - Summary info per connection type at 1 minute intervals
  - As piggybacks on accounting - very little additional overhead
Accounting Trace

- Accounting reports are based on accounting records (IFCID 3, 239)
- Contains local and distributed DB2 activity associated with a thread
- Used as a prime indicator for thread-related problems
- Class 1 data
  - Class 1 times
  - SQL statements counters
  - RID list processing
  - Buffer pool and group buffer pool activity
  - Regular and data sharing locking
  - Stored Procedures / triggers / UDFs
  - Claim and drain activity
  - Resource limit facility (RLF)
  - Distributed data facility (DDF)
  - Accelerator info
  - Dynamic statement cache
  - Logging
  - Query parallelism
Accounting Trace ...

- Class 2 data
  - Adds class 2 times to (existing) record

- Class 3 data
  - Activates all suspension counters and times (adds to existing record)

- Class 5 data
  - Activates IFI counters

- Class 7 data
  - Package level info (similar to class 2 info but at package level)

- Class 8 data
  - Package level wait counters

- Class 10 data
  - Detailed package accounting info (SQL stmt by type, locking, BP)

- [Class 11
  - To ‘separate’ writing IFCID 3 from 239 (V11 only) ]
DB2 Times Terminology - Accounting Class 1,2,3

Class 1
Class 2
(in DB2)
in Appl
1st SQL...
SQL...

Class 3
(susp time)
Agent, non-nested CPU
Agent, non-nested ET

in DB2
(Creating Thread)
SQL...
...Wait for I/O
...Commit
(Terminating Thread)

IRLM
...Wait for lock

Agent
Audit Trace

• More popular than ever before
• Some require AUDIT attribute at table to be turned on
  ▪ A lot of changes here in V10 with the introduction of audit policies
• Uses SMF 102 records when DEST(SMF) is used
• Overhead typically less than 5%
  ▪ Consider the frequency of certain events. For example, security violations are not as frequent as table accesses

<table>
<thead>
<tr>
<th>Trace class</th>
<th>Description</th>
<th>IFCIDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Access attempts denied due to inadequate authorization</td>
<td>140</td>
</tr>
<tr>
<td>2</td>
<td>Explicit GRANT and REVOKE</td>
<td>141</td>
</tr>
<tr>
<td>3</td>
<td>CREATE, ALTER, and DROP operations against audited tables</td>
<td>142</td>
</tr>
<tr>
<td>4</td>
<td>First change of audited object</td>
<td>143</td>
</tr>
<tr>
<td>5</td>
<td>First read of audited object</td>
<td>144</td>
</tr>
<tr>
<td>6</td>
<td>Bind time information about SQL statements that involve audited objects</td>
<td>145</td>
</tr>
<tr>
<td>7</td>
<td>Assignment or change of authorization ID</td>
<td>55,83,87,169,319</td>
</tr>
<tr>
<td>8</td>
<td>Utilities</td>
<td>23,24,25,219,220</td>
</tr>
<tr>
<td>10</td>
<td>Trusted context information</td>
<td>269,270</td>
</tr>
<tr>
<td>11</td>
<td>Audit administrative authorities</td>
<td>361 (*)</td>
</tr>
</tbody>
</table>
Monitor Trace

- Used by online monitors
  - **READA**
    - Using OPx trace buffers – that are then read by the monitor program
  - **READS**
    - Only use a return area (no OP buffer) – to obtain realtime data
    - Request waits until the requested data is available
    - Some IFCIDs are only available via READS
      - IFCIDs 124, 129, 147, 148, 149, 150, 185, 187, 197, 306

<table>
<thead>
<tr>
<th>Class</th>
<th>Description of class</th>
<th>Activated IFCIDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Standard accounting data.</td>
<td>0200</td>
</tr>
<tr>
<td>2</td>
<td>Entry or exit from DB2 event signaling.</td>
<td>0232</td>
</tr>
<tr>
<td>3</td>
<td>DB2 wait times</td>
<td>0006-0009, 0032, 0033,0044, 0045, 0117, 0118, 0127, 0128, 0170, 0171, 0174, 0175, 0213,0 214, 0215, 0216, 0226, 0227, 0242, 0243, 0321, 0322, 0378, 0379, 0382, 0383</td>
</tr>
<tr>
<td>5</td>
<td>Time spent processing IFI requests.</td>
<td>0187</td>
</tr>
<tr>
<td>6</td>
<td>Changes to tables created with DATA CAPTURE CHANGES.</td>
<td>0185</td>
</tr>
<tr>
<td>7</td>
<td>Entry or exit from DB2 event signaling for package accounting.</td>
<td>0200, 0232, 0240</td>
</tr>
<tr>
<td>8</td>
<td>Wait time for a package</td>
<td>0006-0009, 0032, 0033, 0044, 0045, 0051, 0052, 0056, 0057, 0117, 0118, 0127, 0128, 0170,171,174, 175, 213-216, 226, 227, 239, 241-243, 321, 322, 0378, 0379, 0382, 0383</td>
</tr>
<tr>
<td>9</td>
<td>Enables statement level accounting.</td>
<td>0124</td>
</tr>
<tr>
<td>10</td>
<td>Package detail for buffer manager, lock manager and SQL statistics</td>
<td>0239</td>
</tr>
</tbody>
</table>
## Performance Trace

- Great for problem identification
  - See DB2 command reference – STA TRA command

<table>
<thead>
<tr>
<th>Trace class</th>
<th>Description</th>
<th>IFCIDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Background events</td>
<td>1,2,31,42,43,76-79,102,103,105-107,153</td>
</tr>
<tr>
<td>2</td>
<td>Subsystem events</td>
<td>3,68-75,80-89,106,174,175</td>
</tr>
<tr>
<td>4</td>
<td>Reads to and writes from the buffer and EDM pools</td>
<td>6-10,29,30,105-107,127,128,226,227,321,322,357,358</td>
</tr>
<tr>
<td>5</td>
<td>Write to active archive log</td>
<td>32-41,104,106,114-120,228,229</td>
</tr>
<tr>
<td>6</td>
<td>Summary lock information</td>
<td>20,44,45,105-107,172,196,213,214,218,337</td>
</tr>
<tr>
<td>7</td>
<td>Detailed lock information</td>
<td>21,105-107,223</td>
</tr>
<tr>
<td>8</td>
<td>Data scanning detail</td>
<td>13-18,105-107,125,221,222,231,305,311,363</td>
</tr>
<tr>
<td>9</td>
<td>Sort detail</td>
<td>26-28,95-96,106</td>
</tr>
<tr>
<td>10</td>
<td>BIND, commands, and utilities detail</td>
<td>23-25,90,91,105-111,201,219,220,256,360</td>
</tr>
<tr>
<td>11</td>
<td>Execution unit switch and latch contentions</td>
<td>46-52,56,57,93,94,106,113</td>
</tr>
<tr>
<td>12</td>
<td>Storage manager</td>
<td>98-101,106</td>
</tr>
<tr>
<td>13</td>
<td>Edit and validation exits</td>
<td>11,12,19,105-107</td>
</tr>
<tr>
<td>14</td>
<td>Entry from and exit to an application</td>
<td>67,106,121,122</td>
</tr>
<tr>
<td>15</td>
<td>Distributed processing</td>
<td>157-163,167,183</td>
</tr>
<tr>
<td>16</td>
<td>Claim and drain information</td>
<td>211-216</td>
</tr>
<tr>
<td>17</td>
<td>Event-based console messages</td>
<td>197</td>
</tr>
<tr>
<td>18</td>
<td>Data set open/close</td>
<td>370,371</td>
</tr>
<tr>
<td>19</td>
<td>Data sharing coherency summary</td>
<td>249-251,256,257,261,262,267,268</td>
</tr>
<tr>
<td>20</td>
<td>Authorization exit parameters</td>
<td>314</td>
</tr>
<tr>
<td>21</td>
<td>LE Runtime diagnosis</td>
<td>327</td>
</tr>
<tr>
<td>22</td>
<td>Stored Procedure detail</td>
<td>380,499</td>
</tr>
<tr>
<td>23</td>
<td>Available for local use</td>
<td>None</td>
</tr>
</tbody>
</table>
Performance Trace ...

- **Performance overhead**
  - It depends
  - Typically 20~100 %
  - Perf class 1,2,3
    - Typically 5-30%
  - Tracing IFCID 198
    - Every getpage / release page / set write
    - Very expensive
      - Trace volume
      - CPU overhead
  - Filter as much as possible
    - Significantly reduces the overhead

<table>
<thead>
<tr>
<th>CLASS</th>
<th>EVENTS</th>
<th>OVERHEAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BACKGROUND EVENTS</td>
<td>LOW</td>
</tr>
<tr>
<td>2</td>
<td>SUBSYSTEM EVENTS</td>
<td>LOW</td>
</tr>
<tr>
<td>3</td>
<td>SQL EVENTS</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>4</td>
<td>BUFFER MANAGER + EDM I/O</td>
<td>HIGH</td>
</tr>
<tr>
<td>5</td>
<td>LOG MANAGER I/O</td>
<td>HIGH</td>
</tr>
<tr>
<td>6</td>
<td>LOCKING SUMMARY</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>7</td>
<td>LOCKING DETAIL</td>
<td>VERY HIGH</td>
</tr>
<tr>
<td>8</td>
<td>DATA MANAGER DETAIL</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>9</td>
<td>SORT DETAIL</td>
<td>HIGH</td>
</tr>
<tr>
<td>10</td>
<td>UTILITIES, BINDS, COMMANDS</td>
<td>LOW</td>
</tr>
<tr>
<td>11</td>
<td>EUS, LATCH CONTENTION</td>
<td>VERY HIGH</td>
</tr>
<tr>
<td>12</td>
<td>STORAGE MANAGER</td>
<td>VERY HIGH</td>
</tr>
<tr>
<td>13</td>
<td>EDIT AND VALIDATION</td>
<td>VERY HIGH</td>
</tr>
<tr>
<td>14</td>
<td>IN AND OUT OF DB</td>
<td>HIGH</td>
</tr>
<tr>
<td>16</td>
<td>REMOTE LOCATION EVENTS</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>17</td>
<td>CLAIM AND DRAIN</td>
<td>LOW</td>
</tr>
<tr>
<td>19</td>
<td>PHYSICAL OPEN CLOSE</td>
<td>LOW</td>
</tr>
<tr>
<td>20</td>
<td>DATA SHARING SUMMARY</td>
<td>LOW</td>
</tr>
<tr>
<td>21</td>
<td>DATA SHARING DETAIL</td>
<td>HIGH</td>
</tr>
<tr>
<td>22</td>
<td>AUTHORIZATION EXIT PARAMETERS</td>
<td>HIGH</td>
</tr>
<tr>
<td>24</td>
<td>STORED PROCEDURE DETAIL</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>30..32</td>
<td>None</td>
<td>none</td>
</tr>
</tbody>
</table>
DB2 Instrumentation Facility Interface - Summary

• Trace types
  ▪ Accounting
  ▪ Audit
  ▪ Performance
  ▪ Statistics
  ▪ Monitor
  ▪ Global

• Trace classes
  ▪ Multiple trace classes per trace type
  ▪ IFCID as basic unit of reporting
    ▪ Instrumentation Facility Component
      IDentifier

• Trace destinations
  ▪ SMF - Daily monitoring
  ▪ GTF - High volume
  ▪ OPx - Online monitoring
  ▪ (SRV - Exit to a user-written routine)

• Trace headers (TDATA)
  ▪ CORrelation header
  ▪ CPU header
  ▪ DISTRibuted header
  ▪ TRAce header

• Qualifications
  ▪ AUTHID
  ▪ PLAN
  ▪ LOCATION
  ▪ IFCID
  ▪ Many more since V9
Which Data / Traces to Collect and When to Collect them?
Which Data / Traces to Collect and When?

- Start at subsystem level (stats) and zoom in from there (top-down)

**Always active**

- **Statistics**
- **Subsystem**

**Accounting**

- Minimum 1 hour during peak time
- **Thread**

**Performance**

- **Statement**

**Activate when needed**

- **Standard tracing** - active all the time 😏
  - Statistics class 1,3,4,5,(7),(8), 9
  - Accounting 1,(2),3,(7,8) ,(1,10)

  - If you cannot afford accounting 1,2,3 to be active all the time, turn it on for one hour during peak time every day
Standard DB2 Traces - ‘Typical’ Overhead

- DB2 statistics
  - Negligible

- DB2 accounting
  - CLASS(1,3) (together with STAT(1,3,4,(6))): 2-5%
  - CLASS(3): Typically less than 1% overhead
    - Could be higher when system shows very high DB2 internal latch contention
  - CLASS(2): 1-10% (Higher for applications that go a lot back and forth between application and DB2, e.g. FETCH intensive applications)
  - CLASS(7,8) (Package accounting): Less than 5%
  - CLASS(10) (Package detail since PK28561): higher CPU overhead
DB2 Trace
Usage Scenarios
Trace Usage Scenarios

• Using DB2 Accounting data to zoom in on a locking problem
  ▪ Long (drain) lock suspension time
• Using DB2 Performances trace to zoom in on an SQL problem
  ▪ Runtime execution is not as expected
• Using DB2 Performance trace to zoom in on long service tasks waits
#1 Using Accounting info for a locking problem

- Accounting trace contains summarized information about locking
  - The types of lock/latch resource (suspension time and #suspensions)
    - DB2 internal latch
    - IRLM lock + IRLM latch
    - Page latch
    - Drain/claim
    - Global lock – P-lock and L-lock
  - The numbers to differ the suspend type
    - Local lock/latch suspend
      - IRLM lock or IRLM latch
    - Global lock suspend
      - False contention
      - XES contention
      - Real contention
      - Heuristic conversion: SYNC to ASYNC
Use Accounting Trace to Identify a Lock Issue

- The numbers:
  - IRLM lock + latch: 3
  - DB2 latch: 12
  - Drain: 1
  - Claim: 0
  - Page latch: 0
  - Global contention: 1

<table>
<thead>
<tr>
<th>CLASS 3 SUSPENSIONS</th>
<th>ELAPSED TIME</th>
<th>EVENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>---------------------</td>
<td>--------------</td>
<td>--------</td>
</tr>
<tr>
<td>LOCK/LATCH (DB2+IRLM)</td>
<td>0.002913</td>
<td>15</td>
</tr>
<tr>
<td>IRLM LOCK+LATCH</td>
<td>0.038008</td>
<td>3</td>
</tr>
<tr>
<td>DB2 LATCH</td>
<td>0.002913</td>
<td>12</td>
</tr>
<tr>
<td>DRAIN LOCK</td>
<td>6.955973</td>
<td>1</td>
</tr>
<tr>
<td>CLAIM RELEASE</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>PAGE LATCH</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>GLOBAL CONTENTION</td>
<td>0.518333</td>
<td>1</td>
</tr>
</tbody>
</table>
Use Accounting Trace to Identify a Lock Issue

<table>
<thead>
<tr>
<th>LOCKING</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMEOUTS</td>
<td>0</td>
</tr>
<tr>
<td>DEADLOCKS</td>
<td>0</td>
</tr>
<tr>
<td>LOCK REQUEST</td>
<td>136</td>
</tr>
<tr>
<td>UNLOCK REQST</td>
<td>2</td>
</tr>
<tr>
<td>QUERY REQST</td>
<td>0</td>
</tr>
<tr>
<td>CHANGE REQST</td>
<td>0</td>
</tr>
<tr>
<td>OTHER REQST</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL SUSPENSIONS</td>
<td>3</td>
</tr>
<tr>
<td>LOCK SUSPENS</td>
<td>0</td>
</tr>
<tr>
<td>IRLM LATCH SUSPENS</td>
<td>3</td>
</tr>
<tr>
<td>OTHER SUSPENS</td>
<td>0</td>
</tr>
</tbody>
</table>

- 3 IRLM lock + latch
  - (Real) lock suspend: 0
  - IRLM latch suspend: 3
    - 3 IRLM latch suspensions took 38 ms
Use Accounting Trace to Identify a Lock Issue

- What is the global contention for?
  - Parent L-lock
    - 1 parent L-lock suspension lasted 518ms

<table>
<thead>
<tr>
<th>GLOBAL</th>
<th>CONTENTION</th>
<th>L-LOCKS</th>
<th>ELAPSED TIME</th>
<th>EVENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-LOCKS</td>
<td></td>
<td>0.518333</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PARENT (DB,TS,TAB, PART)</td>
<td></td>
<td>0.518333</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>CHILD (PAGE, ROW)</td>
<td></td>
<td>0.000000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>OTHER</td>
<td></td>
<td>0.000000</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GLOBAL</th>
<th>CONTENTION</th>
<th>P-LOCKS</th>
<th>ELAPSED TIME</th>
<th>EVENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-LOCKS</td>
<td></td>
<td>0.000000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PAGESET/PARTITION</td>
<td></td>
<td>0.000000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>PAGE</td>
<td></td>
<td>0.000000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>OTHER</td>
<td></td>
<td>0.000000</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Use Accounting Trace to Identify a Lock Issue

• Is the 518 ms parent L-lock contention real IRLM lock contention?
  ▪ IRLM global contention: 0
  ▪ XES contention: 0
  ▪ Heuristic conversion: 0
  ▪ False contention: 2
• It’s false contention!
  ▪ 1 is the parent L-lock
• 2 false contention?
  ▪ What’s the other one?

<table>
<thead>
<tr>
<th>DATA SHARING</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>------</td>
</tr>
<tr>
<td>LOCK - XES</td>
<td>22</td>
</tr>
<tr>
<td>UNLOCK - XES</td>
<td>1</td>
</tr>
<tr>
<td>CHANGE - XES</td>
<td>0</td>
</tr>
<tr>
<td>SUSP - IRLM</td>
<td>0</td>
</tr>
<tr>
<td>SUSP - XES</td>
<td>0</td>
</tr>
<tr>
<td>CONV - XES</td>
<td>0</td>
</tr>
<tr>
<td>FALSE CONT</td>
<td>2</td>
</tr>
<tr>
<td>INCOMP. LOCK</td>
<td>0</td>
</tr>
<tr>
<td>NOTIFY SENT</td>
<td>0</td>
</tr>
</tbody>
</table>
Use Accounting Trace to Identify a Lock Issue

- Where is the drain/claim counted, local or global?
  - Local if it is local contention
    - OTHER SUSPENS 0
  - Global if it is global contention
    - FALSE CONT 2
  - It is also a false contention!

<table>
<thead>
<tr>
<th>CLASS 3 SUSPENSIONS</th>
<th>ELAPSED TIME</th>
<th>EVENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCK/LATCH (DB2+IRLM)</td>
<td>0.002913</td>
<td>15</td>
</tr>
<tr>
<td>IRLM LOCK+LATCH</td>
<td>0.038008</td>
<td>3</td>
</tr>
<tr>
<td>DB2 LATCH</td>
<td>0.002913</td>
<td>12</td>
</tr>
<tr>
<td>DRAIN LOCK</td>
<td>6.955973</td>
<td>1</td>
</tr>
<tr>
<td>CLAIM RELEASE</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>PAGE LATCH</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>GLOBAL CONTENTION</td>
<td>0.518333</td>
<td>1</td>
</tr>
</tbody>
</table>
Start a Locking Trace To Verify

• To double check that it is really false contention, collect performance trace
  
  ▪ `-START TRACE(P) CLASS(3,6,7,17) IFCID(226,227,259)
    TDATA(CPU,COR,TRA) DEST(GTF) PLAN(xx) AUTHID(xx)
    PKGPROG(xx)`

   • Class 3 – SQL activity trace
   • Class 6 – Lock summary (including suspensions)
   • Class 7 – Lock detail
   • Class 17 – Claim and drain
   • IFCID 226/227 – Page latch contention
     As lock contentions often go hand in hand with page latch contention
     it is usually a good idea to trace those as well
   • IFCID 259 – Page P-lock requests
Start a Locking Trace To Verify

- Locking trace confirms it is false contention

Lock suspend: 11:00:37.16219897
Lock resume: 11:00:44.11817197

Lock type: DRAIN CS
Lock resource: DB =NIGHTDB OB =SSSINCT

EVENT SPECIFIC DATA

<table>
<thead>
<tr>
<th>Event</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUSP.TIME</td>
<td>6.955973</td>
</tr>
<tr>
<td>DURATION</td>
<td>MANUAL</td>
</tr>
<tr>
<td>STATE</td>
<td>S</td>
</tr>
<tr>
<td>RESUME RSN</td>
<td>NORMAL</td>
</tr>
<tr>
<td>LOCAL CONTENTION</td>
<td>N</td>
</tr>
<tr>
<td>LATCH CONTENTION</td>
<td>N</td>
</tr>
<tr>
<td>IRLM QUEUED REQ</td>
<td>N</td>
</tr>
<tr>
<td>INTER-SYSTEM SYSTEM</td>
<td>Y*</td>
</tr>
<tr>
<td>NOTIFY MSG SENT</td>
<td>N*</td>
</tr>
<tr>
<td>NOTIFY MSG SENT</td>
<td>N*</td>
</tr>
</tbody>
</table>

FALSE CONTENTION
What’s Next?

- False contention time should be always very short, 6.9 seconds is too long, then what’s next?
  - Check DB2 statistics trace for more clues
    - IRLM CPU was very high at the problem time
  - Check RMF CF and XCF report

![1-minute interval statistics trace – address space CPU](chart.png)
#2 Runtime Execution is Not as Expected

- Access path looks ok in the PLAN_TABLE
- Runtime behavior is not ok
- Use DB2 accounting info (plan and/or package level) as a starting point
  - What is not as expected?
  - Elapsed time too high
    - Class 2 time (in DB2 or not)
    - Class 3 time
  - CPU time too high (in DB2 or not)
  - Locking behavior
  - #SQL calls
  - (High) BP activity
- Zoom in to the statement level
  - Detailed DB2 performance trace
  - READS IFCID 401 (static), 316-318 (dynamic) via MON trace class 29
  - EXPLAIN STMTCACHE ALL
Analyzing an SQL Performance Problem

- Accounting data is always a good starting point
  - Accounting class 1,2,3,7,8,10 was active
- Use package level info to demonstrate

<table>
<thead>
<tr>
<th>PROGRAM NAME</th>
<th>CLASS 7 ELAPSED TIME CONSUMERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNREXX</td>
<td>&gt; 1%</td>
</tr>
<tr>
<td>GLWJEMP1</td>
<td>=&gt; 2%</td>
</tr>
<tr>
<td>GLWJEMP2</td>
<td></td>
</tr>
<tr>
<td>DPTDEL</td>
<td>====&gt; 5%</td>
</tr>
<tr>
<td>DPTSEL</td>
<td></td>
</tr>
<tr>
<td>EMPDEL</td>
<td>=&gt; 2%</td>
</tr>
<tr>
<td>EMPFND</td>
<td></td>
</tr>
<tr>
<td>EMPQR2</td>
<td>&gt; 1%</td>
</tr>
<tr>
<td>EMPSEL</td>
<td>&lt;=--------------------------------&gt;</td>
</tr>
</tbody>
</table>

Overview by OMPE

Easy to find high CL7 ET or CPU

CLASS 7 info

manually added for easy reading
Analyzing an SQL Performance Problem

- Package class 8 and class 10 info
  - (Averages are per package occurrence in the acctg records)

### Class 8 info

<table>
<thead>
<tr>
<th>EMPSEL</th>
<th>AVERAGE TIME</th>
<th>AVG_EV</th>
<th>TIME/EVENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCK/LATCH</td>
<td>6.278443</td>
<td>49.1K</td>
<td>0.000128</td>
</tr>
<tr>
<td>IRLM LOCK+LATCH</td>
<td>4.656214</td>
<td>32.00</td>
<td>0.145507</td>
</tr>
<tr>
<td>DB2 LATCH</td>
<td>1.622229</td>
<td>49.0K</td>
<td>0.000033</td>
</tr>
<tr>
<td>SYNCHRONOUS I/O</td>
<td>0.012326</td>
<td>57.50</td>
<td>0.000214</td>
</tr>
<tr>
<td>OTHER READ I/O</td>
<td>0.048752</td>
<td>92.50</td>
<td>0.000527</td>
</tr>
<tr>
<td>OTHER WRITE I/O</td>
<td>0.000000</td>
<td>N/C</td>
<td>N/C</td>
</tr>
<tr>
<td>SERV.TASK SWITCH</td>
<td>0.000546</td>
<td>1.00</td>
<td>0.000546</td>
</tr>
<tr>
<td>ARCH.LOG(QUIESCE)</td>
<td>0.000000</td>
<td>N/C</td>
<td>N/C</td>
</tr>
<tr>
<td>ARCHIVE LOG READ</td>
<td>0.000000</td>
<td>N/C</td>
<td>N/C</td>
</tr>
<tr>
<td>DRAIN LOCK</td>
<td>0.000000</td>
<td>N/C</td>
<td>N/C</td>
</tr>
<tr>
<td>CLAIM RELEASE</td>
<td>0.000000</td>
<td>N/C</td>
<td>N/C</td>
</tr>
<tr>
<td>PAGE LATCH</td>
<td>0.000000</td>
<td>2.00</td>
<td>0.000003</td>
</tr>
<tr>
<td>NOTIFY MESSAGES</td>
<td>0.000000</td>
<td>N/C</td>
<td>N/C</td>
</tr>
<tr>
<td>GLOBAL CONTENTION</td>
<td>0.000000</td>
<td>N/C</td>
<td>N/C</td>
</tr>
<tr>
<td>TCP/IP LOB XML</td>
<td>0.000000</td>
<td>N/C</td>
<td>N/C</td>
</tr>
<tr>
<td>ACCELERATOR</td>
<td>0.000000</td>
<td>N/C</td>
<td>N/C</td>
</tr>
<tr>
<td>PQ SYNCHRONIZATION</td>
<td>0.000000</td>
<td>N/C</td>
<td>N/C</td>
</tr>
<tr>
<td><strong>TOTAL CL8 SUSPENS.</strong></td>
<td><strong>6.340073</strong></td>
<td><strong>49.2K</strong></td>
<td><strong>0.000129</strong></td>
</tr>
</tbody>
</table>

### Class 10 info

<table>
<thead>
<tr>
<th>EMPSEL</th>
<th>AVERAGE</th>
<th>TOTAL</th>
<th>EMPSEL</th>
<th>AVERAGE</th>
<th>TOTAL</th>
<th>EMPSEL</th>
<th>AVERAGE</th>
<th>TOTAL</th>
<th>EMPSEL</th>
<th>AVERAGE</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>8553.00</td>
<td>17106</td>
<td>BPOOL HIT RATIO (%)</td>
<td>100.00</td>
<td>N/A</td>
<td>TIMEOUTS</td>
<td>0.00</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INSERT</td>
<td>0.00</td>
<td>0</td>
<td>GETPAGES</td>
<td>51432.8K</td>
<td>102866K</td>
<td>DEADLOCKS</td>
<td>0.50</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UPDATE</td>
<td>0.00</td>
<td>0</td>
<td>BUFFER UPDATES</td>
<td>0.00</td>
<td>0</td>
<td>ESCAL.(SHARED)</td>
<td>0.00</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td>0.00</td>
<td>0</td>
<td>SYNCHRONOUS WRITE</td>
<td>0.00</td>
<td>0</td>
<td>ESCAL.(EXCLUS)</td>
<td>0.00</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIBE</td>
<td>0.00</td>
<td>0</td>
<td>SYNCHRONOUS READ</td>
<td>57.50</td>
<td>115</td>
<td>MAX PG/ROW LOCKS HELD</td>
<td>67.50</td>
<td>135</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PREPARE</td>
<td>0.00</td>
<td>0</td>
<td>SEQ. PREFETCH REQS</td>
<td>0.00</td>
<td>0</td>
<td>LOCK REQUEST</td>
<td>650.00</td>
<td>1300</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPEN</td>
<td>0.00</td>
<td>0</td>
<td>LIST PREFETCH REQS</td>
<td>0.00</td>
<td>0</td>
<td>UNLOCK REQUEST</td>
<td>30.50</td>
<td>61</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FETCH</td>
<td>0.00</td>
<td>0</td>
<td>DYN. PREFETCH REQS</td>
<td>4127.7K</td>
<td>8255484</td>
<td>QUERY REQUEST</td>
<td>0.00</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLOSE</td>
<td>0.00</td>
<td>0</td>
<td>PAGES READ ASYNCHR.</td>
<td>1308.50</td>
<td>2617</td>
<td>CHANGE REQUEST</td>
<td>0.00</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOCK TABLE</td>
<td>0.00</td>
<td>0</td>
<td>OTHER REQUEST</td>
<td>0.00</td>
<td>0</td>
<td>TOTAL SUSPENSIONS</td>
<td>32.00</td>
<td>64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALL</td>
<td>0.00</td>
<td>0</td>
<td>IRLM LATCH SUSPENS.</td>
<td>1.00</td>
<td>2</td>
<td>IRLM LATCH SUSPENS.</td>
<td>1.00</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Analyzing an SQL Performance Problem

- SQL Activity report BY STMTID
  - STMTID is a better identifier than the statement number

<table>
<thead>
<tr>
<th>EVENT</th>
<th>COUNT</th>
<th>TOT.ELAPS</th>
<th>TOTAL TCB</th>
<th>DETAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AET/EVENT</td>
<td>TCB/EVENT</td>
<td></td>
</tr>
<tr>
<td>PACKAGE</td>
<td></td>
<td>DB1B.GLWSAMP.EMPSEL.X'1977A1EC125C0A1C'</td>
<td>V1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACQUIRE(USE)</td>
<td>REOPT(N)</td>
<td>RELEASE(COMMIT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PREPARE(NODEFER)</td>
<td>KEEPDYNAMIC(NO)</td>
<td>PROTOCOL(DRDA)</td>
</tr>
</tbody>
</table>

6977 5702 0.082504 0.030489 SELECT
0.000014 0.000005

6979 5702 0.080681 0.036136 SELECT
0.0000014 0.0000006

6980 265 0.003914 0.001883 SELECT
0.000015 0.000007

6982 543 1:34.075850 53.127033 SELECT
0.017309 0.009775

STMT ID : 6982 STMT TYPE : STATIC
SORTS : 0
GET PAGES : 102819543 PARALLEL GRP CREATES : 0
SYNC BUFF READS : 98 BUFFER WRITES : 0
INDEX SCANS : 5435 TABLESPACE SCANS : 0
ROWS EXAMINED : 114931333 ROWS PROCESSED : 0
RID-LIMIT EXC. : 0 RID-NO STORAGE : 0
IN-DB2 ELAPSED : 1:34.026607 IN-DB2 CPU : 53.109795
GLOBAL LOCK : 0 DRAIN LOCK : 0.000000
LOCK/LATCH : 7.362120 LATCH : 3.244350
PAGE LATCH : 0.000012 CLAIM COUNT : 0.000000
SYNCHRON. I/O : 0.020706 UNIT SWITCH : 0.000000
READ-OTH. THREAD : 0.074695 WRITE-OTH. THREAD : 0.000000
LOG WRITER : 0.000000

IFCID 401 type info also present in IFCID 58

Avg 18900 getpages per execution!
Analyzing an SQL Performance Problem

- **Statement Type = STATIC**
  - We should be able to find it in the catalog

```
SELECT SUBSTR(COLLID,1,18) AS COLL_ID,
     SUBSTR(NAME,1,8) AS PACKAGE_ID,
     QUERYNO,
     STMT_ID,
     HEX(STMT_ID) AS HEX_STMT_ID,
     STATEMENT
FROM SYSIBM.SYSPACKSTMT
WHERE STMT_ID = 6982;
```

<table>
<thead>
<tr>
<th>COLL_ID</th>
<th>PACKAGE_ID</th>
<th>QUERYNO</th>
<th>STMT_ID</th>
<th>HEX_STMT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLWSAMP</td>
<td>EMPSEL</td>
<td>2610</td>
<td>6982</td>
<td>0000000000001B46</td>
</tr>
</tbody>
</table>

```
```
Analyzing an SQL Performance Problem

• Check the PLAN_TABLE (or use EXPLAIN(ONLY) to extract the info
  ▪ Use the QUERYNO to find the stmt

---

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>QBLOCKNO</th>
<th>PROGNAME</th>
<th>PLANNO</th>
<th>METHOD</th>
<th>CREATOR</th>
<th>TNAME</th>
<th>ACCESTYPE</th>
<th>MATCHCOLS</th>
<th>ACCESSCREATOR</th>
<th>ACCESSNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>2610</td>
<td>1</td>
<td>EMPSEL</td>
<td>1</td>
<td>0</td>
<td>GLWSAMP</td>
<td>GLWTEMP</td>
<td>I</td>
<td>1</td>
<td>GLWSAMP</td>
<td>GLXEMP3</td>
</tr>
</tbody>
</table>

INDEXONLY PREFETCH COLUMN_FN_EVAL SCAN_DIRECTION SORT(all)
N R F N

Access Path does not look too bad

-- Database=GLWSAMP
-- Index=GLWSAMP.GLWXEMP3 On GLWSAMP.GLWTEMP
--
CREATE UNIQUE INDEX GLWSAMP.GLWXEMP3
ON GLWSAMP.GLWTEMP
(EMP_NO ASC)
USING STOGROUP GLWG01
PRIQTY 300 SECQTY 100
ERASE NO
FREEPAGE 0 PCTFREE 10
GBPCACHE CHANGED
BUFFERPOOL BP16
CLOSE YES
COPY NO
PIECESIZE 2 G;

--
Analyzing an SQL Performance Problem

- What if we added another index?
  - Create index – Runstats – Rebind explain(yes)

```sql
-- Database=GLWSAMP
-- Index=GLWSAMP.GLWXEMP4 On GLWSAMP.GLWTEMP
---------------------------------------------------------------------
--
CREATE UNIQUE INDEX GLWSAMP.GLWXEMP4
  ON GLWSAMP.GLWTEMP
    (EMP_NO               DESC,
     MANAGER             ASC)
USING STOGROUP GLWG01
PRIQTY 300 SECQTY 100
ERASE NO
FREEPAGE 0 PCTFREE 10
GBPCACHE CHANGED
BUFFERPOOL BP16
CLOSE YES
COPY NO
PIECESIZE 2 G;
--
```

IX is used
MC=1 but now IXONLY

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>QBLOCKNO</th>
<th>PROGNAME</th>
<th>PLANNO</th>
<th>METHOD</th>
<th>CREATOR</th>
<th>TNAME</th>
<th>ACCESSTYPE</th>
<th>MATCHCOLS</th>
<th>ACCESSCREATOR</th>
<th>ACCESSNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>2610</td>
<td>1</td>
<td>EMPSEL</td>
<td>1</td>
<td>0</td>
<td>GLWSAMP</td>
<td>GLWTEMP</td>
<td>I</td>
<td>1</td>
<td>GLWSAMP</td>
<td>GLWXEMP4</td>
</tr>
</tbody>
</table>

INDEXONLY PREFETCH COLUMN_FN_EVAL SCAN_DIRECTION SORT(all)

Y R N
Analyzing an SQL Performance Problem

- Run again – Did it help?
  - CPU time / stmt execution 0.009775 -> 0.000007 (almost 1400x less)
  - 36570 executions (vs 5435 before) in the same time interval
  - Complete workload consists of many different tran
    - Overall workload throughput increased 7 times by just adding this index

<table>
<thead>
<tr>
<th>EVENT COUNT</th>
<th>TOT. ELAPS</th>
<th>TOTAL TCB</th>
<th>DETAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>6982</td>
<td>36570</td>
<td>0.466894</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.241616</td>
<td>SELECT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000013</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.000007</td>
<td></td>
</tr>
</tbody>
</table>

**STMT ID:** 6982
**STMT TYPE:** STATIC
**SORTS:** 0
**PARALLEL GRP CREATES:** 0
**BUFFER WRITES:** 0
**TABLESPACE SCANS:** 0
**ROWS PROCESSED:** 0
**RID-NO STORAGE:** 0
**IN-DB2 CPU:** 0.143823
**GLOBAL LOCK:** 0.000000
**DRAIN LOCK:** 0.000000
**CLAIM COUNT:** 0.000023
**UNIT SWITCH:** 0.000000
**WRITE-OTH. THREAD:** 0.000000

Avg 2 getpages per execution
#3 Analyzing long service task switches

- Accounting data is always a good starting point
  - Long open/close time observed

<table>
<thead>
<tr>
<th>TIMES/EVENTS</th>
<th>APPL(CL.1)</th>
<th>DB2 (CL.2)</th>
<th>IFI (CL.5)</th>
<th>CLASS 3 SUSPENSIONS</th>
<th>ELAPSED TIME</th>
<th>EVENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELAPSED TIME</td>
<td>1:55.93590</td>
<td>27.496244</td>
<td>N/P</td>
<td>LOCK/LATCH (DB2+IRLM)</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>NONNESTED</td>
<td>1:55.93590</td>
<td>27.496244</td>
<td>N/A</td>
<td>IRLM LOCK+LATCH</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>STORED PROC</td>
<td>0.000000</td>
<td>0.000000</td>
<td>N/A</td>
<td>DB2 LATCH</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>UDF</td>
<td>0.000000</td>
<td>0.000000</td>
<td>N/A</td>
<td>SYNCHRON. I/O</td>
<td>0.106740</td>
<td>72</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>0.000000</td>
<td>0.000000</td>
<td>N/A</td>
<td>DATABASE I/O</td>
<td>0.106740</td>
<td>72</td>
</tr>
<tr>
<td>CP CPU TIME</td>
<td>0.031744</td>
<td>0.028250</td>
<td>N/P</td>
<td>LOG WRITE I/O</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>AGENT</td>
<td>0.031744</td>
<td>0.028250</td>
<td>N/A</td>
<td>OTHER READ I/O</td>
<td>0.000171</td>
<td>1</td>
</tr>
<tr>
<td>NONNESTED</td>
<td>0.031744</td>
<td>0.028250</td>
<td>N/P</td>
<td>OTHER WRITE I/O</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>STORED PRC</td>
<td>0.000000</td>
<td>0.000000</td>
<td>N/A</td>
<td>SER.TASK SWTH</td>
<td>27.363426</td>
<td>3</td>
</tr>
<tr>
<td>UDF</td>
<td>0.000000</td>
<td>0.000000</td>
<td>N/A</td>
<td>UPDATE COMMIT</td>
<td>0.000828</td>
<td>1</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>0.000000</td>
<td>0.000000</td>
<td>N/A</td>
<td>SYSLGRNG REC</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>PAR.TASKS</td>
<td>0.000000</td>
<td>0.000000</td>
<td>N/A</td>
<td>EXT/DEL/DEF</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>SUSPEND TIME</td>
<td>0.000000</td>
<td>27.471342</td>
<td>N/A</td>
<td>ARC.LOG(QUIES)</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>AGENT</td>
<td>N/A</td>
<td>27.471342</td>
<td>N/A</td>
<td>LOG READ</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>PAR.TASKS</td>
<td>N/A</td>
<td>0.000000</td>
<td>N/A</td>
<td>DRAIN LOCK</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>STORED PROC</td>
<td>0.000000</td>
<td>N/A</td>
<td>N/A</td>
<td>CLAIM RELEASE</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>UDF</td>
<td>0.000000</td>
<td>N/A</td>
<td>N/A</td>
<td>PAGE LATCH</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>NOT ACCOUNT.</td>
<td>N/A</td>
<td>N/C</td>
<td>N/A</td>
<td>GLOBAL CONTENTION</td>
<td>0.001006</td>
<td>3</td>
</tr>
<tr>
<td>DB2 ENT/EXIT</td>
<td>N/A</td>
<td>70</td>
<td>N/A</td>
<td>COMMIT PH1 WRITE I/O</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>EN/EX-STPROC</td>
<td>N/A</td>
<td>0</td>
<td>N/A</td>
<td>ASYNCH CF REQUESTS</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>EN/EX-UDF</td>
<td>N/A</td>
<td>0</td>
<td>N/A</td>
<td>TCP/IP LOB XML</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>DCAPT.DESCRIPT</td>
<td>N/A</td>
<td>N/A</td>
<td>N/P</td>
<td>ACCELERATOR</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>LOG EXTRACT.</td>
<td>N/A</td>
<td>N/A</td>
<td>N/P</td>
<td>AUTONOMOUS PROCEDURE</td>
<td>0.000000</td>
<td>0</td>
</tr>
<tr>
<td>TOTAL CLASS 3</td>
<td>27.471342</td>
<td>79</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Analyzing long service task switches

- **EU-switches**
  - Used when DB2 has to perform a task that cannot run under the current Execution Unit (EU)
- **OPEN/CLOSE** is a good example when a EU-switch is needed
  - TRA(P) CLASS(19) to trace physical open/close (IFCID 370/371)
    - IFCID 105/107 does NOT trace physical open
  - **IFCID370/371** distinguishes between
    - Allocation time – basically z/OS part of the open (like SVC99)
    - Open time - basically Media Manager part of the open processing
    - Both are ok here – now what?

```
03:26:17.58460764 972270   1 370 OPEN DATA SET
N/P INFORMATION
------------------------------
DATABASE OPEN INFORMATION
DATA SET NAME    : RMTH.DSNDBC.THVCDM00.IXVC0562.I0001.A002            FLAGS     :    X'C7'
ACE ADDRESS      : X'1E3983A0'    DATABASE ID      :    670               OBID      :    1772
PART NUMBER      : X'00000002'    INSTANCE NUMBER  :    X'00000001'  DSMAX     :    70000
OPENED DATA SETS :    40126
ALLOCATION TIME  :    0.020264 OPEN TIME :    0.020337
```

© 2017 IBM Corporation
Check the complete EU switch processing

- Maybe we are out of services task to perform the open
- Trace the EU switch activity
  - Can be done as part of TRACE(P) C(11)
    - But that also traces internal DB2 latch suspensions (can be high volume)
  - Better to trace as TRACE(P) C(30) IFCID(46,47,48,49,50)
    - If you only want to know which EU switch trace IFCID170/171
  - Nope – start running on the OPEN TCB right away

<table>
<thead>
<tr>
<th>RECORD TIME</th>
<th>DESTNO</th>
<th>ACE</th>
<th>IFC</th>
<th>DESCRIPTION</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>03:25:50.24408312</td>
<td>965630</td>
<td>1</td>
<td>46</td>
<td>SERVICE RECORD ENTRVI8F</td>
<td></td>
</tr>
<tr>
<td>N/P</td>
<td></td>
<td></td>
<td></td>
<td>NETWORKID: RDNET</td>
<td>LUNAME: TD2DB2AT LUWSEQ: 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>QW0046AC X'1E3983A0'</td>
<td>QW0046FC X'0059'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>QW0046ID X'0A'</td>
<td></td>
</tr>
<tr>
<td>03:25:50.24415357</td>
<td>965633</td>
<td>1</td>
<td>49</td>
<td>SERVICE RECORD ENTRVI8F</td>
<td></td>
</tr>
<tr>
<td>N/P</td>
<td></td>
<td></td>
<td></td>
<td>NETWORKID: RDNET</td>
<td>LUNAME: TD2DB2AT LUWSEQ: 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>QW0049ID X'0A'</td>
<td>QW0049FC X'0059'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>03:26:17.58460764</td>
<td>972270</td>
<td>1</td>
<td>370</td>
<td>OPEN DATA SET ENTRVI8F</td>
<td></td>
</tr>
<tr>
<td>N/P</td>
<td></td>
<td></td>
<td></td>
<td>INFORMATION</td>
<td>DATABASE OPEN INFORMATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NETWORKID: RDNET</td>
<td>LUNAME: TD2DB2AT LUWSEQ: 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DATA SET NAME : RMTH.DSNDBC.THVCDM00.IXVC0562.I0001.A002</td>
<td>FLAGS : X'C7'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>... (see previous slide)</td>
<td></td>
</tr>
<tr>
<td>03:26:17.58475006</td>
<td>972271</td>
<td>1</td>
<td>50</td>
<td>SERVICE RECORD ENTRVI8F</td>
<td></td>
</tr>
<tr>
<td>N/P</td>
<td></td>
<td></td>
<td></td>
<td>NETWORKID: RDNET</td>
<td>LUNAME: TD2DB2AT LUWSEQ: 2</td>
</tr>
</tbody>
</table>
Check the z/OS side

- OPEN also needs to access the ICF catalog
- RMF ENQ postprocessor report (needs SMF type 77)
  - REPORTS(ENQ(DETAIL))
- Waiting to access the ICF catalog (UCAT DEVXR3 contains our dsn)

<table>
<thead>
<tr>
<th>ENQUEUE ACTIVITY</th>
<th>GRS MODE: STAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENQUEUE DETAIL ACTIVITY</td>
<td>-- JOBS AT MAXIMUM CONTENTION-- ...</td>
</tr>
<tr>
<td>-NAME-</td>
<td>---- CONTENTION TIME ----</td>
</tr>
<tr>
<td>MAJOR</td>
<td>MIN</td>
</tr>
<tr>
<td>MINOR</td>
<td>TOT</td>
</tr>
</tbody>
</table>

SYSIGGGV2
CATALOG.DEVXR3 (SYSTEMS)
0.000 62.617 178.15 0.019

1 CATALOG (S) 59 CATALOG (E) ...
RD01
RD02
CATALOG (S)
RD01

ENQUEUE DETAIL ACTIVITY

-NAME- | ...-----%QLEN DISTRIBUTION- AVG Q -REQUEST TYPE - --- CONTENTION ---
MAJOR | ... | MIN | 1 | 2 | 3 | 4+ | LNGTH | -EXCL- | -SHARE- | EVENT | --STAT CHNG- | MIN | MAX | MAX | TOTAL | TOTAL | %NODET |
MINOR | ... |

SYSIGGGV2
CATALOG.DEVXE3 (SYSTEMS)
...

57.7 25.5 9.9 6.8 2.17 0 7 0 59 9159 N/A N/A
How to find which service task invoked

- Performance trace with IFCID 46 thru 50
  - IFCID 46 for service task switch
  - 47 and 48 for Begin and End of SRB service task
  - 49 and 50 for Begin and End of TCB service task
  - Shows all nested service tasks

- Or use IFCID 170 and 171 for less output
  - Shows one (outermost) service task for each event in class 3 accounting

  EU switches are identified by the DB2 resource manager (RMID) and the function code (FC) that they invoke

- A long time gap between IFCID 46 and 47 (begin SRB task) or 49 (begin TCB task) can indicate that DB2 is waiting for a service task to become available
Commonly observed service tasks

- Commonly observed service tasks and RMID/FC identifiers as shown in IFCID 46, 47, 49, and 170
  - Update Commit x’03’/x’49’
  - Dataset Open x’0A’/x’59’
  - Dataset Close x’0A’/x’4F’
  - SYSLGRNX Update x’15’/x’44’
  - Dataset Extend x’12’/x’65’
  - Dataset Delete x’12’/x’6A’
  - Dataset Define x’12’/x’68’
  - Dataset Reset x’12’/x’6C’
  - Dataset Count x’12’/x’88’
Commonly Observed “OTHER” Service Tasks

- DB2 for z/OS application requestor using TCP/IP waiting for server response x'1B'/x'8F'

- Parallel query cleanup x'14'/x'77' or x'84'
  - Shows up in repeated execution of short-running queries in parallelism
  - Use higher SPRMPTH ZPARM value to replace this (default=120(ms))
  - Of course, for a long-running query, parallel query with degree ANY can result in many times reduction in elapsed time

- VSAM catalog update x'0A'/x'94'

- Peer BSDS read x'04'/x'41'
  - For programs reading the log via IFI 306 in a data sharing environment (like the capture task of replication products)
QUESTIONS

bart_steegmans@be.ibm.com
Thank You
Notices and Disclaimers

Copyright © 2015 by International Business Machines Corporation (IBM). No part of this document may be reproduced or transmitted in any form without written permission from IBM.

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

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY. IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

Any statements regarding IBM’s future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer’s responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer’s business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law.
Notices and Disclaimers (con’t)

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM’s products. IBM EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

- IBM, the IBM logo, ibm.com, Aspera®, Bluemix, Blueworks Live, CICS, Clearcase, Cognos®, DOORS®, Emptoris®, Enterprise Document Management System™, FASP®, FileNet®, Global Business Services®, Global Technology Services®, IBM ExperienceOne™, IBM SmartCloud®, IBM Social Business®, Information on Demand, ILOG, Maximo®, MQIntegrator®, MQSeries®, Netcool®, OMEGAMON, OpenPower, PureAnalytics™, PureApplication®, pureCluster™, PureCoverage®, PureData®, PureExperience®, PureFlex®, pureQuery®, pureScale®, PureSystems®, QRadar®, Rational®, Rhapsody®, Smarter Commerce®, SoDA, SPSS, Sterling Commerce®, StoredIQ, Tealeaf®, Tivoli®, Trusteer®, Unica®, urban{code}®, Watson, WebSphere®, Worklight®, X-Force® and System z® Z/OS, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.
Using GTF as trace destination

- Sample GTF PROC with multiple ("striping") datasets

```plaintext
//GTFDB2 PROC MEMBER=GTFDB2
//IEFPROC EXEC PGM=AHLGTF,
//    PARM='MODE=EXT,DEBUG=NO,TIME=YES,BLOK=1M,SIZE=8M,NOPROMPT',
//    REGION=0M
//GTFOUTxx DD  DSNAME=myprefix.GTFTRAxx,UNIT=SYSDA,
//    VOL=SER=xxxxxx,
//    DISP=(NEW,KEEP),SPACE=(CYL,(1000)),DCB=NCP=255
//*
//* repeated up to 15 times for all the other trace data sets !!
//*
//SYSLIB  DD   DSNAME=SYS1.PARMLIB(&MEMBER),DISP=SHR
```

- SYS1.PARMLIB(GTFDB2) contains

```plaintext
TRACE=USRP
USR=(FB9) <-- this is to limit the input only from DB2
END
```
Using GTF as trace destination

- GTF “striped” data MUST be merged (via IPCS) before usage

```
//GTFMERGE JOB (XXX),'BART.STEEGMANS',
   MSGLEVEL=(1,1),
   NOTIFY=&SYSUID,MSGCLASS=H,TIME=1440,REGION=256M
//IPCS EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=1500K
//IPCSDDIR DD DSN=myprefix.IPCSDDIR.DD,DISP=OLD
//TRACE1 DD DISP=SHR,DSN=myprefix.GTF01
//TRACE2 DD DISP=SHR,DSN=myprefix.GTF02
//TRACEALL DD DISP=(MOD,CATLG),
   DSN=myprefix.GTF.MERGE,
   DCB=(myprefix.GTF01),
   UNIT=(SYSDA,15),SPACE=(CYL,(500,500),RLSE)
//IPCSPRNT DD SYSOUT=* 
//SYSTSPRT DD SYSOUT=* 
//SYSTSIN DD *
   IPCS NOPARM
   SETDEF NOCONFIRM PRINT NOTERM
   DROPD RECORDS(ALL) DDNAME(TRACE1)
   DROPD RECORDS(ALL) DDNAME(TRACE2)
   COPYTRC TYPE(GTF) +
      INFILE(TRACE1,TRACE2) +
      OUTFILE(TRACEALL)
   SETDEF CONFIRM NOPRINT TERM
END
```
Trace Filtering Details

• DB2 9 introduced extensive enhancement to instrumentation filtering. Filters now include PLAN, LOCATION, AUTHID, userid, APPNAME, WRKSTN, PKGPROG, PKGLOC, PKGCOL, CONNID, CORRID, ROLE and eXclude keywords for each (eg. XPLAN)

• Terminating and positional wildcards are allowed (eg. PLAN(DSNTEP*) , PLAN(PLAN_01))

• For each affirmative filter, multiple values are allowed. Logic between them is OR. For example:
  ▪ -START TRACE(PERFM) CLASS(3) PLAN(A, B)
  ▪ Will write performance trace records if the PLAN = A OR PLAN = B
  ▪ Note: This will start effectively 2 traces so the total trace limit of 32 may be reached more rapidly. Consider the use of wildcards if possible.

• For each exclude filter, multiple values are allowed. Logic between these is AND. For example:
  ▪ -START TRACE(PERFM) CLASS(3) XPLAN(A,B)
  ▪ Will write performance trace records if the PLAN is not A AND is NOT B
Runtime behavior does not match plan_table

• This is a more general approach than the example provided earlier
• When DB2 accounting (plan and package level) and DB2 statistics info are not enough to identify the problem, zoom in at the SQL statement level
  ▪ Performance traces
  ▪ Dynamic Statement Cache (DSC) runtime statistics
  ▪ Static SQL statement level statistics via IFI 401
Obtain Detailed Runtime Execution Info

• Use **DB2 Performance Traces**
  ▪ Works for static and dynamic SQL
  ▪ -STA TRA(P) C(1,2,3) DEST(xxx)
    • Filter as much as possible (PLAN, AUTHID ...) to reduce the overhead
  ▪ If you suspect a sort related problem add CLASS 9
    • Be careful: It may seem that sort IFCID 95-96 took a long time but the time can also include the time to process the object itself DB2 kicks of the sort IFCID 95 (sort start) at beginning of the stmt, before there are rows to sort – then starts accessing the objects and as rows qualify they are handed to sort
  ▪ If need data manager detail add CLASS 8
    • Provides the order in which objects are accessed
    • Provides high level info about activity against each object
  ▪ If need I/O details add CLASS 4
    • Be careful with filtering for async read (and write) I/Os
  ▪ If you need detailed getpage info add IFCID(198)
    • Be careful, the trace volume becomes huge very quickly
Obtain Detailed Runtime Execution Info

• Use DB2 Performance Traces ...
  ▪ Verify that runtime access path (IFCID 22) is still the same than what you got in the PLAN_TABLE
    • Verify that the sequence in which objects are accessed matches with perf class 8 info
  ▪ To find CPU time used by an SQL stmt you need the CPU header
    • Subtract total CPU at stmt start from total CPU at stmt end
      – Most monitor programs will do this for you
    • CPU info also available in IFCID 58 when IFCID400 is active
  ▪ Sometimes the problem is related to the value of the input host variables at runtime
    • IFCID 247 can be used to trace those
  ▪ If you want to see the values that are being returned to the application (output host variables)
    • IFCID 248 can be used to trace those
Obtain Detailed Runtime Execution Info

• Obtain Statement level runtime info for **dynamic SQL**
  - IFI READS 316, 317, (318) for DSC
    - STA TRA(P) C(30) IFCID(316,317,318)
    - IFCID 316 - Info about statements in the dynamic statement cache (DSC)
    - IFCID 317 – Complete SQL statement text
    - IFCID 318 – (Total) runtime execution information at the SQL stmt level
  - EXPLAIN STMTCACHE ALL and IFCID318
    - Similar info to IFI READS – but via simple SQL statement
    - STA TRA(P) C(30) IFCID(318) is enough
    - Stopping IFCID318 will reset the execution stats
      So issue explain stmtcache all prior to stopping the trace
    - Need appropriate authority to use EXPLAIN STMTCACHE ALL
      – SQLADM authority / System DBADM authority / SYSADM authority
    - Populates user.DSN_STATEMENT_CACHE_TABLE
      – DDL in *.SDSNSAMP(DSNTESC)
    - Can use Data Studio to capture this info
Analyzing Dynamic SQL using the DSC Stats

• Statistics counters in the DSC are cumulative (totals) since IFCID 318 was activated
  ▪ Are zeroed when IFCID 318 is stopped so capture DSC before stopping the trace
  ▪ Make sure that no other trace with IFCID 318 is already running
    • In that case nothing happens if an additional trace for IFCID 318 is started and the STAT_TS will reflect the old TS (when the first trace with IFCID 318 was initially started)

• EXPLAIN STMTCACHE ALL only shows what is CURRENTLY in the cache
  ▪ (Important) Statements may have LRU-ed out
  ▪ In V10 IFCID 316 can be written to SMF/GTF when a statement is removed from the cache (eg. –STA TRA(P) C(30) IFCID(316,317,318) DEST(SMF) )

• Use EXPLAIN STMTCACHE STMID xxx to capture the access path into in the PLAN_TABLE (and other access path related tables)

• Find SQL statements that:
  ▪ Use most CPU/ET/.. in total
  ▪ Use most CPU/ET/.. on average

• Automate and execute on a daily basis during peak workload times
Obtain Detailed Runtime Execution Info

• Obtain statement level runtime info for **static SQL**
  ▪ IFI READS 401 (400)
    • Similar info as dynamic SQL stmts from DSC
    • ONLY available via IFI program (not via an SQL stmt)
  ▪ Perf trace records (like IFCID 58) will contain extra info if IFCID 400 is active
    • Stmt level info at end of the statement (or close cursor for cursor based SQL)

• IFCID 401 and 316 are available
  ▪ via IFI READS program
  ▪ Can be written to a trace destination (like SMF) but only when a stmt is evicted from the DSC or EDM pool
    • Eg. Invalidated in DSC because a RUNSTATS was run
Using Data Studio to capture DSC

![Image of Data Studio interface]

**Capture SQL from Statement Cache**

Create or select a filter for capturing SQL statements from the dynamic statement cache on the subsystem. Then, click Capture to capture statements. Select a statement to analyze, tune, or both and click Invoke Advisor and Tools.

- **Filter**
  - **Filter name:** Default_1

- **Enable Cache Trace**, **Disable Cache Trace**, **Disable cache trace after capturing**

**Captured Statements**

The number of captured statements is 196. Right-click a statement and select Invoke Advisor and Tools. If workload tuning is enabled, you can create a workload from all of the statements.
DSN_STATEMENT_CACHE_TABLE info

• General statement information

  ▪ **STMT_ID** - Statement ID - this value is the EDM unique token for the statement. Unique for a member. Can change if stmt is removed from the cache and later reinserted.

  ▪ **STMT_TOKEN** - Statement token you can provide an identification string on prepare

  ▪ **COLLID** - The collection ID: DSNDYNAMICSQNCACHE

  ▪ **PROGRAM_NAME** - The name of the package that performed the initial PREPARE for the statement

  ▪ **LINES** - The precompiler line number from the initial PREPARE of the statement

  ▪ **PRIMAUTH** - The primary authorization ID that did the initial PREPARE of the statement

  ▪ **CURSQLID** - The CURRENT SQLID that did the initial PREPARE of the statement

  ▪ **SCHEMA** - The value of the CURRENT SCHEMA special register

  ▪ **BIND_QUALIFIER** - The BIND qualifier. For unqualified table names, this is the object qualifier

  ▪ **BIND_ISO** - The value of the ISOLATION BIND option

  ▪ **BIND_CDATA** - The value of the CURRENTDATA BIND option

  ▪ **BIND_DYNRL** - The value of the DYNAMICRULES BIND option

  ▪ **BIND_DEGRE** - The value of the CURRENT DEGREE special register
DSN_STATEMENT_CACHE_TABLE info ...

- **BIND_SQLRL** - The value of the CURRENT RULES special register
- **BIND_CHOLD** - The value of the WITH HOLD attribute of the PREPARE for this stmt
- **BIND_RO_TYPE** - The current specification of the REOPT option for the statement
- **LITERAL_REPL** - Identifies cached statements where the literal values are replaced by the '& symbol
- **BIND_RA_TOT** - The total number of REBIND commands that have been issued for the dynamic statement because of the REOPT(AUTO) option
- **GROUP_MEMBER** - The member name of the DB2 that executed EXPLAIN
- **EXPANSION_REASON** - Applies to only statements that reference archive tables or temporal tables
- **STMT_TEXT** - The statement that is being explained (in CLOB).
- **INV_DROPALT** - Whether the statement has been invalidated by a DROP or ALTER
- **INV_REVOKE** - Whether the statement has been invalidated by a REVOKE
- **INV_LRU** - Whether the statement has been removed from the cache by LRU
- **INV_RUNSTATS** - Whether the statement has been invalidated by RUNSTATS
- **USERS** - The number of current users of the statement that have prepared or run the statement during their current unit of work
- **COPIES** - The number of copies of the statement that are owned by all threads in the system
DSN_STATEMENT_CACHE_TABLE info …

• Timestamps
  • **CACHED_TS** - When the statement was stored in the dynamic statement cache
  • **STAT_TS** - Timestamp when IFCID 318 is started
  • **EXPLAIN_TS** - The timestamp for when the statement cache table is populated.

• Statistical information
  • **STAT_EXECB** - The number of times this statement has been run. For a statement with a cursor, this is the number of OPENs
  • **STAT_ELAP** - The accumulated elapsed time that is used for the statement
  • **STAT_CPU** - The accumulated CPU time that is used for the statement
  • **STAT_SUS_SYNIO** - The accumulated wait time for synchronous I/O operations
  • **STAT_SUS_LOCK** - The accumulated wait time for lock requests
  • **STAT_SUS_SWIT** - The accumulated wait time for synchronous execution unit switch
  • **STAT_SUS_GLCK** - The accumulated wait time for global locks
  • **STAT_SUS_OTHR** - The accumulated wait time for read activity that is done by another thread
  • **STAT_SUS_OTHW** - The accumulated wait time for write activity done by another thread
**DSN_STATEMENT_CACHE_TABLE info**

- **STAT_SUS_LATCH** - The accumulated wait time for latch requests
- **STAT_SUS_PLATCH** - The accumulated wait time for page latch requests
- **STAT_SUS_DRAIN** - The accumulated wait time for drain lock requests
- **STAT_SUS_CLAIM** - The accumulated wait time for claim count requests
- **STAT_SUS_LOG** - Accumulated wait time for log writer requests
- **STAT_GPAGB** - The number of getpage operations that are performed
- **STAT_SYNRB** - The number of synchronous buffer reads that are performed
- **STAT_WRITB** - The number of buffer write operations that are performed
- **STAT_EROWB** - The number of rows that are examined
- **STAT_PROWB** - The number of rows that are processed
- **STAT_SORTB** - The number of sorts that are performed
- **STAT_INDXB** - The number of index scans that are performed
- **STAT_RSCNB** - The number of table space scans that are performed
- **STAT_PGRPB** - The number of parallel groups that are created
- **STAT_RIDLIMTB** - The number of times a RID list was not used because the number of RIDs would have exceeded DB2 limits
- **STAT_RIDSTORB** - The number of times a RID list was not used because there is not enough storage available to hold the list of RIDs