Distributed access to DB2 for z/OS

MONITORING & TROUBLESHOOTING

Cristian Molaro
Belgium 2010

DISCLAIMER

PLEASE BE AWARE THAT THE ACTUAL PROGRAMMING TECHNIQUES, ALGORITHMS AND ALL NUMERICAL PARAMETERS USED IN EXAMPLES GIVEN IN THIS PRESENTATION ARE SUBJECT TO CHANGE AT SOME FUTURE DATE EITHER BY A NEW VERSION OF DB2, A NEW RELEASE, A SMALL PROGRAMMING ENHANCEMENT (SPE) OR A PROGRAMMING TEMPORARY FIX (PTF).

THE INFORMATION CONTAINED IN THIS PRESENTATION HAS NOT BEEN SUBMITTED TO ANY FORMAL REVIEW AND IS DISTRIBUTED ON AN "AS IS" BASIS WITHOUT ANY WARRANTY EITHER EXPRESS OR IMPLIED. THE USE OF THIS INFORMATION OR THE IMPLEMENTATION OF ANY OF THESE TECHNIQUES IS A CUSTOMER RESPONSIBILITY AND DEPENDS ON THE CUSTOMER'S ABILITY TO EVALUATE AND INTEGRATE THEM INTO THE CUSTOMER'S OPERATIONAL ENVIRONMENT. WHILE EACH ITEM MAY HAVE BEEN REVIEWED FOR ACCURACY IN A SPECIFIC SITUATION, THERE IS NO GUARANTEE THAT THE SAME OR SIMILAR RESULTS WILL BE OBTAINED ELSEWHERE. CUSTOMERS ATTEMPTING TO ADAPT THESE TECHNIQUES TO THEIR OWN ENVIRONMENTS DO SO AT THEIR OWN RISK.

DB2 IS A TRADEMARK OF INTERNATIONAL BUSINESS MACHINE CORPORATION. THIS PRESENTATION USES MANY TERMS THAT ARE TRADEMARKS. WHEREEVER WE ARE AWARE OF TRADEMARKS THE NAME HAS BEEN SPELLED IN CAPITALS.
**Keep it simple**

- Avoid complex configurations
  - Difficult problem determination
  - Performance slows down
  - Problems may result from the various clients/servers being at different levels
- Use direct connections when no added value on using DB2 Connect Server
- Linux on z and Hipersockets
  - Probably today’s best option for a DB2 Connect server

--

**Analyze it simple**

- DRDA configurations can be complex
- Divide the analysis into 3 basic components:
  - The client
  - The network
  - The server
Problem determination

- Typical DRDA problem areas
  - Connectivity
  - Network
  - Performance degradation
  - Data conversion issues
  - Abends

- Eliminate non distributed elements as a root cause
  - Application issues?
  - User error?

- Narrow scope
  - The client?
  - The network?
  - The server?

Problem determination: Survey

- Have you experience in:
  - DB2 for LUW troubleshooting & monitoring?
  - DB2 for z/OS troubleshooting & monitoring?
  - Network troubleshooting & monitoring?
  - DB2 for z/OS + LUW troubleshooting & monitoring?
  - DB2 for z/OS + LUW + Network troubleshooting & monitoring…?

- The challenge? To combine multi-disciplinary expertise in monitoring and problem determination
**End to End Monitoring**

- **Basic requirement:** to identify the Response Time components

  ![Diagram showing the components of Response Time](image)

- **Application RT** = \((4 - 1)\)
- **Time in DB2** = \((3 - 2)\)
- **Network**
  - \((2 - 1) + (4 - 3)\) → if clocks in synch
  - \((4 - 1) - (2 - 3)\) → if clocks not in synch
- Averages Request and Response Network times

---

**db2 ping**

- Tests the network response time of the underlying connectivity between a client and a database server
- Can simulate ≠ packages sizes (bytes) for REQUEST and RESPONSE

```plaintext
C:\..\SQLLIB\BIN>db2 ping proddb request 100 response 32000 10 times
Elapsed time: 4256 microseconds
Elapsed time: 4507 microseconds
Elapsed time: 4240 microseconds
Elapsed time: 5034 microseconds
Elapsed time: 3998 microseconds
Elapsed time: 4099 microseconds
Elapsed time: 4030 microseconds
Elapsed time: 4071 microseconds
Elapsed time: 4096 microseconds
Elapsed time: 4053 microseconds
```

- Requires to be connected to the target database
  - For testing if you can reach the server, use the **ping** and/or the **traceroute** OS commands
Network protocol analyzers

- Network time can be a major component of the End to End RT
- Network analyzers can provide valuable information for problem determination and performance analysis
- Can be hardware or software based
- Wireshark
  - Freeware network analyzer
  - Supports and format the DRDA protocol packages
  - http://www.wireshark.org/

Getting distributed detailed information

- Use DIAGLEVEL=4 at client side
  - `db2 update dbm cfg using diaglevel 4`
  - Capture all possible diagnostic information written by DB2

Other settings

- DFT_MON_STMT=ON → Start/stop time statements ids
- DFT_MON_UOW=ON → Start/end times, completion status
- DB2CONNECT_IN_APP_PROCESS=NO → monitor clients local to DB2 Connect Server
### Traces available on distributed components

<table>
<thead>
<tr>
<th>Client / Driver</th>
<th>Available traces</th>
<th>What the trace contains?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Data Server Driver for JDBC and SQLJ (type 4)</td>
<td>JCC Trace</td>
<td>JCC trace contains both JCC driver trace and DRDA trace only when TRACE_ALL is specified</td>
</tr>
<tr>
<td>IBM Data Server Driver for ODBC and CLI</td>
<td>CLI trace, db2trc, db2drdat</td>
<td>CLI trace contains the driver trace. db2trc contains db2 client side buffers and DRDA buffers. (db2drdat available from 9.5 FPack 4)</td>
</tr>
<tr>
<td>All other Data Server Clients, DB2 Connect, DB2 ESE and so forth</td>
<td>CLI trace, db2trc, db2drdat</td>
<td>CLI trace + db2trc + db2drdat. db2drdat contains only DRDA buffers.</td>
</tr>
</tbody>
</table>

**Tip:** Get experience on collecting and analyzing distributed component traces  
**Refer to DB2 9 for z/OS: Distributed Functions SG24-6952 for more details and examples**

---

### JDBC traces

- Records all function calls made by the DB2 JDBC drivers to a log file  
- Very useful for problem determination  
- Warning:  
  - May produce a huge amount of information  
  - Be careful with overhead when enabling them in a production environment  
- Refer to Redbook SG24-6952 for activation procedure examples  
- WAS 6.1:  
  - Data sources > SNAME > WebSphere Application Server data source properties > Connection pools > Custom properties > traceLevel
JDBC trace example

```java
[jcc] BEGIN TRACE_DRIVER_CONFIGURATION
[jcc] Driver: IBM DB2 JDBC Universal Driver
[jcc] Driver Architecture 3.57.82
[jcc] Using global properties:
[jcc] os.name = Windows XP, system
[jcc] os.arch = x86, system
[jcc] Dumping all system properties: {}
[jcc] Dumping all file properties: {}
[jcc] END TRACE_DRIVER_CONFIGURATION
```

```java
[jcc] BEGIN TRACE_CONNECTS
[jcc] Attempting connection to svr1:3322/DB2PLOC
[jcc] Using properties: {}
[jcc] traceLevel=-1, clientRerouteAlternateServerName=null
[jcc] END TRACE_CONNECTS
```

```
[jcc] [time:2010-05-30-09:51:20.146] [thread:WebContainer : 4] [tracepoint:315] creating a socket to svr1 at 10.50.1.30
[jcc] [time:2010-05-30-09:51:20.146] [thread:WebContainer : 4] [tracepoint:1] [Request.flush]
[jcc] SEND BUFFER: EXCSAT
```

---

**DB2 JDBC/JCC Driver Versions**

---

**db2drdat - DRDA trace command**

- Capture the DRDA data stream exchanged between a client and the server
- Most often used for problem determination...
- And it can provide great information for performance tuning
  - By determining how many sends and receives are required to execute an application
  - To confirm connection reuse, blocking is active, etc

```
C:\Program Files\IBM\SQLLIB\BIN>db2drdat on
Trace is turned on

C:\Program Files\IBM\SQLLIB\BIN>db2drdat off
Trace is turned off
```

---

**Capture the DRDA data stream exchanged between a client and the server**

**Most often used for problem determination**

**And it can provide great information for performance tuning**

- By determining how many sends and receives are required to execute an application
- To confirm connection reuse, blocking is active, etc
db2drdat - DRDA trace command, example

- Trace information is written in ./db2drdat.dmp by default

**SEND BUFFER (AR):**
- EXCSAT RQSDSS
- ASCII: 0 1 2 3 4 5 6 7 8 9 A B C D E F
- EBCDIC: 0123456789ABCDEF

**RECEIVE BUFFER (AR):**
- EXCSATRD OBJDSS
- ASCII: 0 1 2 3 4 5 6 7 8 9 A B C D E F
- EBCDIC: 0123456789ABCDEF

**EXCSAT:** exchange server attributes
- Is the first command that is sent from a requester to a server
- The AR communicates the minimum level of support that it requires from the AS

**EXCSATRD:** exchange server attributes reply data
- 00D300F2: indicates reply from an incompatible server

More info? DB2 Version 9.1 for z/OS: DDM command support

---

**DRDA levels**

- Communication will be done using the lowest DRDA level supported by the Clients / Server

**Working with down-level clients?**
- An old client will work but probably with a subset of the DRDA capabilities of the DB2 server
- Clients and servers are supported independently

**BUT:** feedback from IBM DDF Level 2 Support area:
- Typical problem: distribution protocol errors or errors with certain DDM code points
- Special register settings not taking effect after connection reuse
- Many (sometimes undetermined) problems solved after updating clients

Keep clients up to date
Use INACTIVE threads

- Enable inactive thread support
  - CMSTAT=INACTIVE
  - Allows DB2 for z/OS pooling:
    - Reduction on CPU utilization
    - Reduction on Memory utilization
- Allow DDF threads to become INACTIVE
  - Avoid holding resources
    - WITH HOLD cursors not closed
    - DTT not dropped
    - Application using packages bound using KEEPDYNAMIC
  - Resources held across a COMMIT would prevent the connection and associated DB2 thread from being pooled
  - Origin of another common problem: idle thread timeout

DB2 for z/OS Timeouts

- Example: a remote client connects to DB2, does some work, and then does not go inactive:

```
STC09109  DSNL027I  _DBXP SERVER DISTRIBUTED AGENT WITH
            LUWID=GA65B414.PA09.1111C8065156=43494
            THREAD-INFO=TOTO123:MACBOOK01:TOTO123:javaw.exe
            RECEIVED ABEND=04E
            FOR REASON=00D3003B
```

- IDTHTOIN
  - Time, in seconds, an active server thread remains idle before it is cancelled
  - Inactive connections are not subject to idle thread timeout
  - In general, default works well
- Client may receive SQL30081N which would indicate that the remote connection was terminated
Client timeouts

- **QueryTimeoutInterval** is the client side equivalent of **IDTHTOIN**
- The **QueryTimeoutInterval** value specifies the interval between checks for expired queries
- Controls the timeouts of threads that are connected and active but not doing any work

Can be disabled by adding **QueryTimeoutInterval** = 0 in the db2cli.ini file in the common section

Set to a value that is larger than **SQL_ATTR_QUERY_TIMEOUT**

-DIS DDF

DSNL080I -DSNL080I DSNLTDFF DISPLAY DDF REPORT FOLLOWS:
DSNL081I STATUS=STARTD
DSNL082I LOCATION LUNAME GENERICLU
DSNL083I DB2P DB2P.LU1 -NONE
DSNL084I TCP PORT=5136 SEC PORT=5137 RES PORT=5138 IP NAME=-NONE
DSNL085I IP ADDR=192.168.1.1
DSNL086I SQL DOMAIN=WWW.HELLWORLD.BE
DSNL089I DT=I CONDBAT= 1000 MDBAT= 200
DSNL092I ADBAT= QUEDBAT= 0 INADBAT= 0 CONQUED= 0
DSNL093I DSCDBAT= INACONN= 0
DSNL099I DSNLTDFF DISPLAY DDF REPORT COMPLETE

- **DT=I** --> DDF configured with INACTIVE threads
- **CONDBAT** --> MAX REMOTE CONNECTED
- **MDBAT** --> MAX REMOTE ACTIVE
- **ADBAT** --> Current # of DBATs, active and disconnected
- **QUEDBAT** --> Count # times MDBAT was reached, only reset at restart
- **INADBAT** --> Current # of inactive DBATs, DISPLAY THREAD TYPE(INACTIVE)
- **CONQUED** --> Current # of queued connections
- **DSCDBAT** --> Current # of disconnected DBAT= DBAT pool threads
- **INACONN** --> Current # of inactive connections
DB2 for z/OS: Tracing distributed events

- DB2 accounting records are created separately at the requester and each server.
- Events are recorded in the accounting record at the location where they occur:
  - The Class 2 TCB time at the requester does not include processing time at the server.
  - To get the Total TCB: add Class 2 time at the AR to the Class 2 time at the AS.
  - Same for getpage, prefetch, locking and I/O counts.
- **Important:**
  - Private protocol → SQL activity is counted at both the requester and server.
  - DRDA → SQL activity is counted only at the server.
  - … but are not using PP anymore, right?

DB2 Accounting for distributed workloads

- DB2 writes an accounting record when a DDF thread:
  - Ends.
  - Is made inactive.
  - Does not go inactive because using KEEPDYNAMIC(YES).
  - Or a sign-on occurs for an RRSAF thread.
- This has the potential to create a huge amount of records.
- zParm ACCUMACC controls whether and when DB2 accounting data is accumulated for DDF and RRSAF threads:
  - ACCUMACC=NO, default no effect.
  - ACCUMACC = n, n defines the accumulation interval.
- zParm ACCUMUID defines the aggregation criteria:
  - Value from 0 to 17.
  - ACCUMUID=1 → End user ID.
DDF Rollup

- ACCUMAC and ACCUMUID can be changed dynamically
- Allows to get detailed accounting information if needed
- Change back to original value after problem reproduction

![Graph: Impact of ACCUMAC on accounting](image)

Distributed unit of work (DUOW)

- DB2 Connect support multisite updates with two-phase commit
  - Updates data in multiple remote database servers with guaranteed integrity
  - Involves more than 1 database server
- DUOW is available for applications developed using:
  - Regular SQL
  - TP monitors that implement the XA interface specification
    - IBM TxSeries CICS
    - IBM Message and Queuing Series
    - IBM Component Broker Series
    - Microsoft Transaction Server (MTS)
    - BEA Tuxedo and several others

![Image of distributed unit of work](image)
Distributed unit of work (DUOW)

A distributed transaction can update any mix of supported database servers within a single transaction

Supported by:
- DB2 for Linux, UNIX and Windows Version 8 or later
- DB2 for z/OS Version 7 or later
- DB2 for IBM i

Exploits two-phase commit technology centrally controlled by a coordinator
- DB2 or TP monitor

DUOW problem determination can be a challenge!

Scenario:
- DUOW involving DB2 for z/OS as one of the participants
- Timeouts in DB2 for z/OS

Problem analysis

Isolated problem with big business impact
System was not under stress
No indication of z/OS issues
Extract of DB2 MSTR address space

```
DSN376I   DBP1 PLAN=DISTSERV WITH
       CORRELATION-ID=db2jcc_app11
       CONNECTION-ID=SERVER
       LUW-ID=GA647215.0622.C5BAAC8177FD=72267
       THREAD-INFO=USERID1:wassrv01:USERID1:db2jcc_application
       IS TIMED OUT. ONE HOLDER OF THE RESOURCE IS PLAN=DISTSERV
       WITH
       CORRELATION-ID=db2jcc_app11
       CONNECTION-ID=SERVER
       LUW-ID=GA647214.F72A.C5BAAC83BE46=111082
       THREAD-INFO=USERID2:wassrv03:USERID2:db2jcc_application
       ON MEMBER DBP2

Holder LUW-ID = GA647214.F72A.C5BAAC83BE46
```
Problem resolution

- Analysis using DSN1LOGP
  - DSN1LOGP allows to specify 1 to 10 LUW-IDs to include information in a summary report

<table>
<thead>
<tr>
<th>URID</th>
<th>LSRN</th>
<th>Type</th>
<th>Subtype</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>135137F4C22</td>
<td>C5BAAC83C4C7</td>
<td>(OR CONTROL)</td>
<td>BEGIN UR</td>
<td>2010-05-25-06.53.02.490736</td>
</tr>
<tr>
<td>135137F4C22</td>
<td>C5BAAC83CDA1</td>
<td>(OR CONTROL)</td>
<td>BEGIN COMMIT1</td>
<td></td>
</tr>
<tr>
<td>135137F4C22</td>
<td>C5BAAC9D5CA6</td>
<td>(OR CONTROL)</td>
<td>END COMMIT2</td>
<td></td>
</tr>
</tbody>
</table>

Commit takes too long! ≈ 27 seconds

- Typical values in this environment
  - DUOW → 0.03 secs / two-phase commit
  - BATCH → 0.005 secs / two-phase commit (RRS)

- LSRN to TS: SELECT TIMESTAMP(X'C5BAAC83C4C7'||X'0000') FROM SYSIBM.SYSDUMMY1;

WLM: client set info

- DB2 server systems allow requester systems to influence certain accounting and monitoring information using the EXCSQLSET command

- DB2 server systems have implemented the concepts of:
  - End user IDs
  - End user workstation names
  - End user application names
  - Accounting data

- Much of this information is externalized in various forms:
  - The DSNV437I message of the DISPLAY THREAD command
  - THREAD-INFO data in various messages such as DSNT375I
  - The QWHC trace record correlation header
  - The QMDA section of DB2 accounting trace records
How to identify a client?

- Method depends on API used:
  - CLI
  - C program
  - Java program
  - WebShpere Application Server

- APAR PK74330:
  - Stored procedure SYSPROC.WLM_SET_CLIENT_INFO
  - Allows a DB2 for z/OS AR to set client information associated with the current connection at the DB2 for z/OS server
  - Changes the following DB2 for z/OS client special registers
    - CURRENT CLIENT_ACCTNG
    - CURRENT CLIENT_USERID
    - CURRENT CLIENT_WRKSTNNAME
    - CURRENT CLIENT_APPLNAME

How to identify an ODBC client?

- Special registers cannot be set through SQL
- Information can be used by WLM classification identifier AI position ≥ 56
- Exposed in RMF report
  
*Enclave Classification Data*

```
DSNL027I  -PRD1 SERVER DISTRIBUTED AGENT WITH 778
LUWID=C9DE5919.F7D7.C5C2D6F15029=636
THREAD-INFO=CRIS:TotoMac:Toto:TestFromMac:*:*:*:*
RECEIVED ABEND=04E
FOR REASON=00D3003B
DSNL028I  -PRD1 C9DE5919.F7D7.C5C2D6F15029=636 779
ACCESSING DATA FOR
LOCATION ::10.50.1.12
IPADDR ::10.50.1.12
```
SUMMARY

Agenda

- DRDA CONFIGURATIONS
- METHODOLOGY
- END TO END MONITORING
- TOOLS AND TECHNIQUES
- COMMON PROBLEMS
- SUMMARY AND CONCLUSIONS
QUESTIONS?

CRISTIAN MOLARO
cristian@molaro.be

THANKS!

CRISTIAN MOLARO
cristian@molaro.be
IDUG EUROPE 2010

Registration Now Available

Threads
- DB2 10 for z/OS
- DB2 for z/OS Lessons Learned
- DB2 9.7 for LUW in Depth
- DB2 for LUW Lessons Learned
- SQL & XML
- DB2 Performance

5 One Day Seminars

Experts panel + Special Interest Groups and more

The Belgian speakers this year
- Davy Goethals
- Kurt Struyf
- Frédéric Bothy
- Cristian Molaro

Check www.idug.org for cost Saving Opportunities
- Early Bird Discount
- Multiple Delegate Discount
- Mentor program

Questions or more information?
- Offline