DB2 for z/OS Distributed Best Practices

Adrian Burke
DB2 for z/OS SWAT team
agburke@us.ibm.com
Agenda

- DB2 11
- Gateway
- High Availability
- Driver and Connection Settings
- Protecting DB2
- Identifying Threads
DB2 11 and DB2 Connect

- Any in-service level of DB2 Connect drivers or DB2 Connect server should work with DB2 V11 CM and V11 NFM (V10.5 fp2 recommended for all new functions)
  > DB2 Connect in service versions are V9.5 and later
  > DB2 Connect V9.5 out of service at end April-2014

- Seamless upgrade available with DB2 Connect V9.7 FP6 and V10.1 FP2 Driver or later
  > DB2 Connect drivers seamlessly handle migration from V10, V11 CM, V11 NFM
  > Applications will continue to function as members are migrated one by one (without any exceptions seen)
    – While in V11 CM, applications will continue to see V10 function level
    – While fully at V11 NFM, new connections will see V11 function level using APPLCOMPAT
  > DB2 Connect Server is not seamless, application will get connection error and must reconnect
DB2 11 Features (Distributed workload vs. V10)

- **Sync Receive CM mode**
  - z/OS 1.13 via APAR (PM80004) or z/OS 2.1
  - Saves SRB switch when data becomes available
  - Async used previously in order to detect socket failure, no longer necessary with improvements in z/OS

- **Non-Stored procedures performance**
  - CL1 elapsed time down 3-6%
  - Total CPU time/commit reduced 4-6%

- **Stored proc workloads performance**
  - CL2 elapsed time down 4%
  - Total CPU time/commit reduced 3-5%
  - With implicit commit 8-20% CL1 elapsed improvement
    - Implicit Commit for stored procedures
      - CM but need Connect V10.5 FP 2
      - Relies on AUTOCOMMIT being on, but bundles commits in single message flow
Cancel thread and Interrupt statement …

- Pre DB2 11 DDF CANCEL THREAD may not work in some cases
  - Historically "soft", or reactive in nature
  - Cancel detection point may not be encountered in a timely manner
- DB2 11 New FORCE keyword on CANCEL THREAD command
  - Purges thread of a remote connection in the DB2
  - FORCE option accepted only after CANCEL THREAD issued without the FORCE option
  - Available in CM
- Purge running DDF SRB using new z/OS purge SRB function
  - Requires new z/OS CALLRTM TYPE=SRBTERM service
    - Available in z/OS 1.13 or later via APAR OA39392

Syntax

```
>>-CANCEL --+THREAD(token)------------------------+-----------+--------+
        --DDF THREAD(+-luwid-+)+ -DUMP+ -LOCAL+ -NOBACKOUT+ --FORCE--
        +-token+
```
BIND/DDL Break-in

- **DB2 10 DDF threads**
  - To enable
    - Rebind packages with RELEASE(DEALLOCATE)
    - Issue command –MODIFY DDF PKGREL(BINDOPT)
  - To disable
    - Wait 200 commits (existing DBATs)
    - –MODIFY DDF PKGREL(COMMIT), which only effects DBATs created after the command
  - There is no solution for non-DDF threads in DB2 10

- **DB2 11 DDF threads**
  - Enabled same as was done in DB2 10
  - To disable
    - Automatically done on next COMMIT if waiter on a package lock

- **DB2 11 non-DDF threads**
  - Automatically done on next COMMIT if waiter on a package lock > ½ the IRLM timeout value
  - Idle thread break-in (PM95929, PM96001, PM96004)
Client Information and Block Fetch

- New (v11) CLIENT_CORR_TOKEN special register - client Information field in NFM
  - Allows applications to correlate their business processes (beyond the scope of DB2) across the enterprise
  - Special register value available in the accounting correlation header record and in the enhanced thread-info message (IP address, port, unique 12 character ID)
    - Reported in DISPLAY THREAD DSNV402I message V442 section
    - Can be SET via various APIs: Sqleseti/ JDBC/ SET_CLIENT_ID Function/ SIGNONs
  - This special register cannot be used for classifying DDF work within WLM

- DB2 11 supports Package based continuous block fetch called Blasting (DB2 z/OS → DB2 z/OS)
  - Great for large result sets, no requests for next query block, no movement between I/O buffers
  - Requester package is required to be bound with DBPROTOCOL(DRDACBF)
  - Separate TCP/IP socket and DB2 DBAT opened for each read-only statement
  - Statement is implicitly closed and thread pooled when query is exhausted
Why Would I use DB2 Connect Gateway

- Two Phase Commit
  - Many transaction managers still require a DB2 Connect Server for two phase commit if they use a dual transport model
    - For example, Tuxedo, Encina
  - MS DTC is a transaction manager that does not require a DB2 Connect Server

- Federation
  - Homogeneous Federation is possible with DB2 Connect Server
    - Use of nicknames to access other DB2 and Informix data servers

- Licensing
  - DB2 Connect Enterprise Edition with concurrent user licensing requires DB2 Connect Server configurations only
  - Ensure you have license for MSUs and Host (based on # of subsystems or D.S. group)
High Availability…why not Connect Gateway

- Availability is the key advantage to the IBM Data Server Driver
  - **Automatic Client Reroute** (ACR) - only available in driver
    - Connection to DB2 drops, seamless reroute connection at transaction boundary to other DB2 in data sharing group
    - If DB2 crashed then inflight tran is rolled back
    - Next SQL statement re-driven under the covers to another member
      - JAVA applications will **not** get -30108 during re-connection to another DB2
  - **Sysplex WLB** - transaction level balancing only available in driver
    - Every 10 seconds WLM refreshes list of optional DB2 members with relative weights
    - At transaction boundary next SQL can be routed to another member due to
      - LPAR CPU
      - DB2 health
      - VSTOR
      - MAXCONQN / MAXCONQW

- Gateway balances and fails over at the connection level, not per COMMIT
Sysplex Workload Balancing…failover

- Availability aspect is arguably more important as if we lose a DB2 or connection, we can **re-drive the SQL** request across an existing connection to another member of the group without the application’s knowledge – Automatic Client Reroute only exists in the driver, not Connect Gateway
- Even without failure WLM will feed back results so transactions, not just connections are balanced across the sysplex

![Diagram of Sysplex Workload Balancing](image-url)
Non-Seamless Failover

- **Affinities**
  - "client affinity" – used with pureScale as opposed to letting WLM provide workload balancing information to our drivers, a manual configuration is applied at the driver level that provides an "affinity" to a specific member. The remaining members in the group are specified as failover members - if the main member goes down, and can also be configured to fail back to the main member when it comes back up.

  - This might be used for VUE licensing to force connections to one member unless there is an outage
  - You can add DSN aliases in the .config file (XML) so that different requestors are assigned different client accounting fields

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>enableClientAffinitiesList</td>
<td>DB2BaseDataSource.YES (1)</td>
</tr>
<tr>
<td>clientRerouteAlternateServername</td>
<td>host1, host2, host3</td>
</tr>
<tr>
<td>clientRerouteAlternatePortNumber</td>
<td>port1, port2, port3</td>
</tr>
<tr>
<td>maxRetriesForClientReroute</td>
<td>3</td>
</tr>
<tr>
<td>retryIntervalForClientReroute</td>
<td>2</td>
</tr>
</tbody>
</table>
T2 High Availability

- Resource Adapter Failover feature starting in WebSphere Application Server V8
  - If DB2 goes down locally it reverts to remote DB2 using T4 attach
    - Could also have another member of same group on that LPAR for T2 attach, but why?
  - Not seamless, no transaction level routing
  - Non-seamless failback
Tuning Data Connections
## Key DB2 DDF Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Possible values</th>
<th>Default – as of V8*</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDF</td>
<td>NO / AUTO / COMMAND</td>
<td>No</td>
<td>DDF Startup</td>
</tr>
<tr>
<td>CMTSTAT</td>
<td>ACTIVE / INACTIVE</td>
<td>INACTIVE*</td>
<td>Thread Pooling - Pool inactive threads</td>
</tr>
<tr>
<td>CTHREAD</td>
<td>1-2,000 (V10 – 19,999)</td>
<td>200*</td>
<td>Max users - allied (local) threads RRSAF or CAF (CICS, IMS, TSO, Batch attach, SPUFI, Classic QMF, etc)</td>
</tr>
<tr>
<td>MAXDBAT</td>
<td>0-1,999 (V10 – 19,999)</td>
<td>200*</td>
<td>Max remote active DDF Threads - DBM1 Address Space</td>
</tr>
<tr>
<td>CONDBAT</td>
<td>0-15,000 (V10 – 150,000)</td>
<td>10,000*</td>
<td>Max remote connections - DDF Address Space</td>
</tr>
<tr>
<td>MAXTYPE1</td>
<td>0-CONDBAT</td>
<td>0</td>
<td>Max inactive DBATs, these are used for private protocol. DRDA uses inactive connections.</td>
</tr>
<tr>
<td>POOLINAC</td>
<td>0-9,999</td>
<td>120</td>
<td>Approximate time, in seconds that an inactive/unused DBAT can remain idle in the pool before it is terminated. DBAT deleted after being used 200 times also.</td>
</tr>
<tr>
<td>IDTHTOIN</td>
<td>0-9,999</td>
<td>120*</td>
<td>The IDLE THREAD TIMEOUT (IDTHTOIN) parameter specifies the time (in seconds) that an active DBAT can remain idle before it is canceled. – Should set a “little” higher than TCPKPALV</td>
</tr>
<tr>
<td>TCPKPALV</td>
<td>ENABLE / DISABLE / 1-65534</td>
<td>120*</td>
<td>TCP/IP keep alive (Goes hand-in-hand with IDTHTOIN)</td>
</tr>
<tr>
<td>TCPALVER</td>
<td>NO (server), YES, CLIENT, SERVER, SERVER_ENCRYPT</td>
<td>NO</td>
<td>whether DB2 is to accept TCP/IP connection requests that contain only a user ID (no password, RACF PassTicket, or Kerberos ticket).</td>
</tr>
</tbody>
</table>
JDBC driver properties

- **Auto Commit**
  - Turn autoCommit off unless you cannot control it in the reference architecture
    - Example: conn.setAutoCommit(false);
    - Disable using non-java driver Autocommit configuration keyword

- **Cursor Hold**
  - Driver default is to open result set cursors using the WITH HOLD option
  - WITH HOLD cursor result sets persist across COMMITs
    - Java driver can override default using setResultHoldability=2
    - Non-java driver can override default using CursorHold=0

- **Implicit Close**
  - Specifies whether result set cursors are closed automatically after all rows are fetched
  - Java driver queryCloseImplicit property
  - Non-java driver CursorTypes set to static forward only cursors
  - Disabling leaves result set cursors open on DB2 after all rows are fetched

- **Controlling the size of a query block returned per network request**
  - Set query block size up to 256K
    - Control size using non-java FET_BUF_SIZE configuration keyword
    - Control size using java queryDataSize property
Driver properties cont..

- Interrupt processing for SQL cancel [Statement.cancel()] and timeouts - we want to close the socket when this is triggered to avoid hung threads
  - JAVA – interruptProcessingMode = CLOSE_SOCKET(2)
    - Similar setting for queryTimeoutInterruptProcessingMode ➔ CLOSE_SOCKET(2)
  - Non-JAVA – Interrupt=2

- For non-JAVA driver how do I maintain standards?
  - db2dsdriver.cfg provided – XML File
  - ODBC, CLI, OLE-DB and .NET or open source (Perl, PHP)

- DB2 10 high performance database access threads
  - Provides additional CPU benefits in DB2 – reintroduce RELEASE(DEALLOCATE) to DDF
  - Similar performance benefits as KEEPDYNAMIC – but don’t work together
  - Can be altered off by –MODIFY DDF command; -DIS DDF shows you which option is on
  - Requires rebinding of DB2 Connect client packages
    - Should use the set currentPackageSet in driver for ‘good’ applications
  - Controlled by POOLINAC, not IDTHTOIN
    - Thread recycled after 200 executions
  - DB2Binder utility defaults to releasePackageResourcesAtCommit(false) by default changes for DB2 10 packages (9.7 FP 3a)
Driver properties for Sysplex WLB

- **V9.7FP6 / V10.1FP2** and above recommended for Sysplex support
  - Algorithm changed to avoid driver flooding DB2 with connection attempts
- Set `enableSysplexWLB` to true (enable transports)
- Better defaults for **Global properties**
  - `db2.jcc.maxTransportObjects`
    - Max # of connections to DB2 server across all datasources
    - Default value of **1000** (previous default -1, unlimited transports)
  - `db2.jcc.maxTransportObjectIdleTime`
    - Maximum elapsed time in seconds before an idle transport is dropped
    - Default value is 60 sec
    - This helps eliminate inactive connections
  - `db2.jcc.maxTransportObjectWaitTime`
    - Number of seconds the client waits for a transport to become available
    - Default value is 1 sec (previous default = -1, unlimited wait)
Connection Pool Properties

- **Statement Cache Size** *(Default: 10)*
  > Number of statements to keep in prepared Statement Cache

- **Connection Timeout** *(Default: 180)*
  > How long to attempt connection creation before timeout

- **Max Connections** *(Default: 10)*
  > Max connections from JVM instance

- **Min Connections** *(Default: 1)*
  > Lazy minimum number of connections in pool

- **Reap Time** *(Default: 180)*
  > How often cleanup of pool is scheduled in seconds

- **Unused Timeout** *(Default: 1800)*
  > How long to let a connection sit in the pool unused

- **Aged Timeout** *(Default: 0)*
  > How long to let a connection live before recycling (affects number of inactive connections seen in DB2)

- **Purge Policy** *(Default: EntirePool)*
  > After StaleConnection, does the entire pool get purged or only individual connection

---

**Connection Pool counters:**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>PercentUsed</td>
<td>The average percent of the pool that is in use</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>connectionPoolModule.percentUsed</td>
<td>Per connection pool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PercentMaxed</td>
<td>The average percent of the time that all connections are in use</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>connectionPoolModule.percentMaxed</td>
<td>Per connection pool</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Max Connections in application server should not be ignored

- This setting should be based on testing and profiling the application, and also depends on application connection timeout behavior
- It is basis for determining min number of transports necessary to support application workload
Connection flowchart on DB2 for z/OS

- Connection and/or Sign on processing
  - CMTSTAT=INACTIVE
    - YES: CONDBAT limit reached?
    - NO: MAXDBAT limit reached?
      - YES: Authorization
      - NO: Request queued until an unused DBAT is available
- DSNL092I QUEDBAT+1
- DSNL030I message
- Reject connection
- Enclave created at first SQL
How do the thread pool settings relate to the database connection pool settings?

- Ideally an application would be tested for throughput and an appropriate Max Connections set… but otherwise start by setting the Max Connections value to the sum of all of the max threads possible in the thread pools used by the application which might access DB2.

- Set the number of Minimum Connections to the sum of all of the min threads in the thread pools used by the application.

- Multiply the Max Connection values by the number of application servers going against that particular DB2 member or group. This gives you the real number of total connections that could be made to the database at anytime.

- If the total max connections going to the database (DB2 on z/OS) is larger than CONDBAT then CONDBAT should be assessed to absorb the overage, and avoid the Connection Timeout popping –
  - Should WAS hold all the available connections?
  - CONDBAT +1 means –SQL code and DB2 appears unavailable.
How many possible connections

- **Scope of defined resources** = CELL $\rightarrow$ NODE $\rightarrow$ SERVER
  - More granular setting overrides more general one – need to account for resources SERVERs consume
    - So if maxConnections at cell is 50, and there are 10 servers (or servant regions), then total connections is really 500
So when you say connections.....??

- General logic: I have more app server threads than I need connections to DB2
  - I have more app server ‘connections’ than transports (real connection to DB2)
    - I have more connections to DB2 than active threads in DB2
      - Thus we funnel the most abundant resources down the most finite = =
      - **How many concurrent threads can I support on DB2??**
General Recommendations… but defaults shown

- So for WAS relating to DB2 timeouts.. *without sysplex WLB*
  - TCPKPALV (Enable=720) < IDTHTOIN (120) - we want to ping the socket prior to having DB2 cancel a thread for being idle, incase we have lost the internet connection
  
  - IDTHTOIN (120) > Aged Timeout (0) - This may be the most controversial one, having aged timeout lower will recycle the connections after so many seconds (minutes) in order to 1) reduce the possible number of inactive connections seen in the DB2 member 2) in cases where sysplex workload balancing is not actually set up, it will allow the connection to be re-driven to another member
    - The downside to this would be the small overhead in re-establishing a new connection, and thread reuse when the connection is closed
  
  - Unused Timeout (1800) > Reap Time (180) - means we kill unused connections at the server to free up unused resources, and that no setting should be less than the Reap time, since it is the daemon that goes out looking for other timeouts that have popped
    - But if you have a minimum connection count then those will not be affected by Unused timeout
  
  - IDTHTOIN > TCPKPALV > Aged Timeout > Unused Timeout > Reap - this the end-to-end picture with all 4 of the major timeout values
  
  - Connection Timeout (Network Timeout) – how long can the application handle not getting a connection, this should be a very small number of seconds with sysplex distributor connecting to a data sharing group
Recommendations WITH Sysplex WLB

- The general idea here is to take the WebSphere timeout settings out of the picture and allow the driver properties to control the connections/transports to DB2 z/OS
  - Max Connections = based on concurrency tests and SLAs for application**
  - Aged timeout = 0 back to default
  - Unused =0
  - Purge Policy = FailingConnectionOnly

- maxTransportObjects
  - Max number of connections at driver level (not data source level)
  - TOTAL # transports < CONDBAT * DB2 Members
    - But MAXDBAT limits concurrency of execution

- maxTransportObjectIdleTime = 60 seconds (default)
  - This is how you get rid of inactive connections (like Agedtimeout in WAS)
  - As long as this is less than IDLTHTOIN then we are okay

- maxTransportObjectWaitTime, default is 1 second
  - SQLCODE -4210, SQLSTATE 57033 for connection waiting for a transport

- maxRefreshInterval = 30 seconds (default) → 10 seconds (for WLM)
Thread Monitoring:

DSNL080I  +DB2P DSNLTDDDF DISPLAY DDF REPORT FOLLOWS:
DSNL081I STATUS=STARTD
DSNL082I LOCATION LUNAME GENERICLU
DSNL083I LCNDB2P -NONE -NONE
DSNL084I TCPPORT=9001 SECPORT=0 RESPORT=9007
  IPNAME=DSNDB2P
DSNL085I IPADDR=:192.168.132.5
DSNL086I SQL DC
DSNL086I RESYNC DOMIAN IBM.COM
DSNL089I MEMBER IPADDR=:192.168.132.5
DSNL090I DT=I CONDBAT= 3000 MDBAT= 500
DSNL091I MCONQN= 0 MCONOW= 0
DSNL092I ADBAT= 53 QUEDBAT= 17422 INADBAT= 0 CONQUED= 0
DSNL093I DSCDBAT= 35 INACONN= 1848
DSNL094I WLMHEALTH=100 CLSDCONQN= 0 CLSDCON
DSNL106I PKGREL = COMMIT

How many threads are currently doing work

What you think you can handle.

You hit max DBAT at some point and requests were queued

How many threads are lounging in the pool

Type 2 inactive thread/ inactive connection

RELEASE(COMMIT)
Profile enhancements in DB2 10

- Catalog table holds profiles
- Can limit:
  - Active threads
  - Idle thread timeout
  - Connections
- Limit based on
  - IP address
  - Product ID
  - Role and AUTHID
  - Collection ID and package
  - Client Accounting info.
- Warning or Exception
  - 1 attribute could be warning (just once or for each threshold), another could be exception where incoming connections are queued or failed
  - DSNT771I reason code 00E3050x
Protecting DB2 (using Sysplex WLB)

- DBAs want to control the connection queue depth and the wait time for a queued connection to get a DBAT, exceeding which the connection should be terminated, AND have DB2 health decrease as connections approach CONDBAT
  - Two new zP ARMs
    - MAXCONQN (depth) and
    - MAXCONQW (wait time) are supported. -DISPLAY DDF
  - DSNL074I issued when > 80% CONDBAT reached and DB2 health lowered
- DETAIL will show the current configured values and connections closed because limit exceeded (ACR used). This info is also recorded in stats and accounting

**DSNL094I WLMHEALTH=100 CLSDCONQN= 0 CLSDCONQW= 0**

- APAR number is PM43293 (V9 & V10).
Protecting DB2 - continued

- Prior to PM43292 WLM weights were not updated until PI index interval of 10 seconds, however with PM43923 DB2 Health is pushed out if CONDBAT is encroached upon
- How do you know where connection will end up?
  > Assume Member 1’s health = 75 / Member 2’s = 25
  - (Member’s currently active transactions) / (Total active transactions for the group) <= (Member priority) / (Total of all member priorities)
  - Member 1 has 80 active transactions (in a UoW)
  - Member 2 has 20 active transactions
  - Current ratios -
    • Member 1  = active transactions / total transactions = 80 / 100 = 0.8 (which is > 0.75 target ratio)
    • Member 2  = 20 / 100 = 0.20 (which is < 0.25 target ratio)
  - Next transaction would be assigned to member 2
Types of location aliases

- **Static location aliases**

  You can use DSNJU003 (change log inventory) utility to define and modify as many as 8 static location aliases. Changes to these aliases require you to stop both DDF and DB2, thus requiring an outage.

- **Dynamic location aliases (New in v10)**

  You can use the MODIFY DDF command with the ALIAS option to define and manage as many as 40 location aliases dynamically. You can start, stop, cancel, modify, and delete dynamic location aliases without stopping either DDF or DB2. These aliases cannot be defined or managed by the DSNJU003 utility, and the DSNJU004 utility does not print any information about them. You have to use the DISPLAY DDF command to find information about these aliases. Before you can define dynamic location aliases, DB2 must be started, but DDF may or may not be started.
DB2 z/OS Server Static LOCATION Aliases

- BSDS for each DB2 member can include multiple LOCATION names for the DB2 server
- Each LOCATION name is registered with WLM separately for WLM sysplex workload balancing information
- Useful for:
  - consolidating DB2 systems
  - workload routing within sysplex

DB2 Client

CONNECT TO DB2LOC1
CONNECT TO DB2LOCA
Dynamic Location Aliases

- **Business Challenge**: Block incoming traffic from certain app servers (connection storm) without disrupting other traffic. Setting MAXDBAT=0 blocks all inbound traffic.
  - **Solution**: You can associate a location alias with an app server and use the –MODIFY DDF command to dynamically stop the alias on a member which blocks incoming traffic from only those app servers that are connected to that alias without affecting other inbound or any outbound traffic.

- **Business Challenge**: Need the ability to manage poorly performing applications without taking a DB2 server outage.
  - **Solution**: You can associate a location alias with an app server and use the –MODIFY DDF command to dynamically exclude poorly behaved applications from sysplexWLB (change PORT).

- **Business Challenge**: Stopping DDF causes all the app servers connected to a member to failover, but it also disrupts all outbound traffic. DBAs are looking for a more granular failover where only certain app servers fail over while others don’t.
  - **Solution**: You can associate a location alias with an app server and use the –MODIFY DDF command to dynamically stop the alias which only causes the app servers connected to that alias to failover, without affecting others.
Dynamic Location Aliases - Continued

*Examples:*

- **MODIFY DDF ALIAS(alias1) ADD**
  
  Alias1 is created and is stopped by default.

- **MODIFY DDF ALIAS(alias1) PORT(9000)**
  
  Alias1 is associated with port 9000.

- **MODIFY DDF ALIAS(alias1) IPv4(2.2.2.2)**
  
  Alias1 is associated with IP address 2.2.2.2

- **MODIFY DDF ALIAS(alias1) START**
  
  DDF will accept requests for alias1 on port 9000. When a client connects to alias1, IP address 2.2.2.2 is returned in the server list.

- **MODIFY DDF ALIAS(alias1) STOP**

  Alias1 is stopped and will not accept new requests. Existing requests will be allowed to complete.
Displaying location alias-specific information

- DISPLAY DDF ALIAS(alias1 ) DETAIL

DSNL080I @ DSNLTDDF DISPLAY DDF (alias1) REPORT FOLLOWS:
DSNL087I ALIAS PORT SEC PORT STATUS
DSNL088I ALIAS1 9000 5005 STARTED
DSNL089I MEMBER IPADDR=:2.2.2.2
DSNL089I MEMBER IPADDR=2002:91E:610::1
DSNL096I ADBAT= 100 CONQUED= 1000 TCONS= 1000
DSNL100I LOCATION SERVER LIST:
DSNL101I WT IPADDR IPADDR
DSNL102I 32 ::2.2.2.2 2002:91E:610::1
DSNL102I 32 ::1.2.3.4
DSNL099I DSNLTDDF DISPLAY DDF REPORT COMPLETE

ADBAT : The number of active data base access threads (DBATs) that are currently processing requests on behalf of the specified alias.
CONQUED : The number of connection requests that are currently queued and waiting to be serviced on behalf of the specified alias.
TCONS : The total number of remote connections that are currently associated with the alias.
Identifying threads

- Problem determination
- Root cause analysis
- Resource usage limits
- Test/dev ability to profile applications
- Auditing and authorization investigations
Setting Client info

- In data source
  - All applications sharing data source appear the same to DB2
  - Need source per application to change extended data source properties
- In the Resource Reference (**Simple and granular**)
  - Specific to the application
  - Create extended data source properties (same keywords)
- Calling stored procedure – WLM_SET_CLIENT_INFO
  - Requires application add a call to proc and populate the information
- Having application set it
  - setClientInfo() method to set correlation and accounting info
- Create a wrapper from incoming getConnection() string that dynamically picks up program name and IDs
  - Can use Hibernate or Spring class to populate their intermediary config file
  - Could use a wrapper from Websphere that uses getConnection() and WSSubject class to pull the information out of the incoming request to populate client info

Ease of implementing
But static

Requires coding, but flexible
Data Source Definition in WebSphere

- Set them in the data source definition
  - clientWorkstation
  - clientApplicationInformation – Process name
  - clientUser
  - ClientAccountingInformation
  - clientProgramName - CORRID

- Using a Resource Reference for the application
  - clientWorkstation
  - clientApplicationInformation - Process name
  - clientUser
  - clientAccountingInformation
  - clientProgramName - CORRID
How do I identify applications more. . .

- Client application information can be set at the driver level:
  - `connectionProperties.put("clientProgramName", "agb_v9")` → sets Correlation ID
  - `connectionProperties.put("clientApplicationInformation", "AdrianBurke")` → sets process name/application name

- Starting with JDBC 4.0 the `setClientUser` API is public
  - `conn.setClientInfo("ClientUser","JDBC40API_clientuser");`
  - `conn.setClientInfo("ClientHostname","JDBC40API_clientworkstation");`
  - `conn.setClientInfo("ApplicationName","JDBC40API_clientapplication");` → sets process name/application name
  - `conn.setClientInfo("ClientAccountingInformation","JDBC40API_clientaccounting");`

- What would this look like… next slide
Enclave 10C00007A55 on System DEMOMVS

Subsystem type: DDF  Plan name: DISTSERV
Subsystem name: DB1S  Package name: SYSLN200
Priority:  Connection type: SERVER
Userid: DNET060  Collection name: NULLID
Transaction name: DNET060  Correlation: agb_v9
Transaction class:  Procedure name: DB2_DRDA
Netid:  Function name: 
Logical unit name: Performance group: 
Subsys collection: DSNSG  Scheduling env: 
Process name: AdrianBurke  
Reset: NO
References

- DB2 for z/OS and WebSphere Integration for Enterprise Java Applications

- Subsystem and Transaction Monitoring and Tuning with DB2 11 for z/OS

- WebSphere z/OS – The Value of Co-Location
The End!