IBM Software Group

Understanding J2C and Connection Polling in IBM WSAS

Jia Zhou WASCET Level 2 Engineer

e business on demand.

Updated February 21, 2012

AGENDA

- What is J2C
- Connection Pooling
- Common Problems & troubleshooting
- Understanding J2C trace
- Q/A

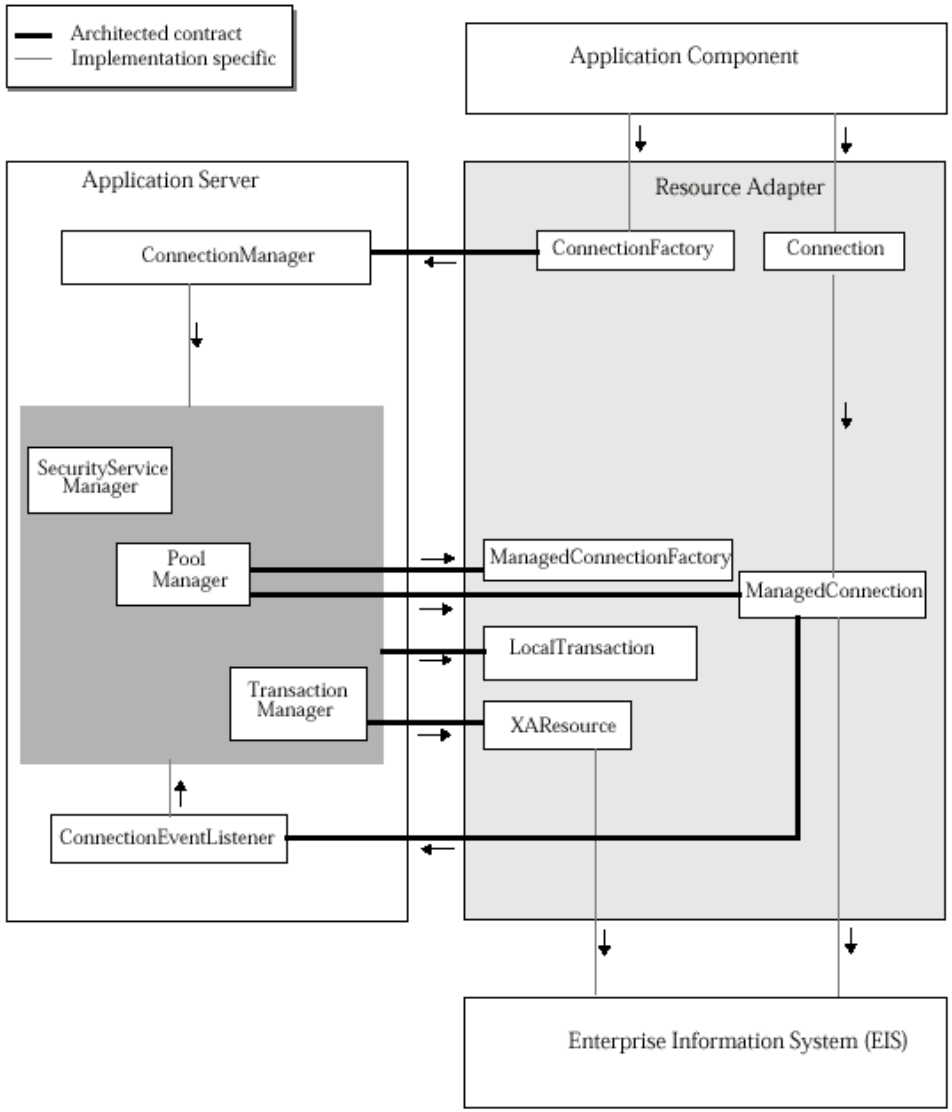
AGENDA

- What is J2C
- Connection Pooling
- Common Problems & troubleshooting
- Understanding J2C trace
- Q/A

What is J2C

- J2C is the implementation of JCA(Java Connector Architecture) 1.5 starting from WSAS v6.x
- JCA provides a standard mechanism to connect to EIS(Enterprise Information System) and Rational Database
  - Use JCA specification/Resource adapter to connect to EIS
  - Use JDBC specification/Data source to access to Database

Connection Management Architecture



Terminology

- Resource Manager- Logical front-end which manages the resources of an EIS (Enterprise Information System) or relational database
- Resource Adapter- Provides connectivity between an application component and EIS per the contracts specified in the JCA
- Connection Manager - Pools and manages connections within an application server
- Pool Manager- J2C Object which implements a connection pool.

Terminology (cont)

- MCF (ManagedConnectionFactory)- RA interface for the app server to match and create ManagedConnections.
- MC (ManagedConnection)- RA interface to represent an EIS Connection to the app server
- MCW (ManagedConnectionWrapper)- A wrapper object used by J2C to manage the state of MC objects
- An MCW can have the following states:

• STATE_NEW – initial state, before first use

• STATE_ACTIVE_FREE – not allocated to any user

• STATE_ACTIVE_INUSE – allocated to a user, but not currently used in a transaction

• STATE_TRAN_WRAPPER_INUSE – allocated to a user, and associated with a transaction

• STATE_INACTIVE – no longer associated with a MC or a connection pool, the MCW is parked in a MCW pool (distinct from any connection pool) for possible later reuse

AGENDA

- What is J2C
- Connection Pooling
- Common Problems & troubleshooting
- Understanding J2C trace
- Q/A

What is Connection Pooling

- The pooling of connections means that Connections, when an application is finished using them, are kept open with RM in a pool rather than actually being closed. A sub-sequent connection request might re-use a connection from a pool instead of constantly opened/closed
- Typically a J2C connection pool has 3 sub-pools:
  - Free pool
  - Shareable connection pool
  - Unshareable connection pool

Shareable connection

- Shareable Connection – Shareable connections allow for multiple getConnection requests by different parts of an application to each be provided with their own reference (or handle) to the same Connection simultaneously. This further reduces the Connection resources required by an application.

Shareable connection

- Sharing a connection can only happen within a sharing scope. The most common sharing scope is a transaction scope where multiple active connection handles can share the same physical connection.
  - A global transaction(user transaction) is either a user transaction initiated by the J2EE application, or a container global transaction initiated by the EJB container. Transaction is managed to external Transaction Manager
    - com.ibm.ws.Transaction.JTA.TransactionImpl@103a048#tid=61010
  - Local Transaction Containment (LTC) is always established by the container in the absence of a global transaction. Transaction is managed by internal Resource Manager
    - com.ibm.ws.LocalTransaction.LocalTranCoordImpl@3ba43ba4

Shareable connection

- Several conditions must be met for a shareable connection to actually be shared. If any of the following requirements is not met, two getConnection calls will return two distinct physical connections:
  - JNDI name
  - Resource authentication setting
  - Principal
  - Connection transaction isolation level property
  - Connection readOnly, catalog, and typaMap properties

Shareable connection

- In WAS, connections are shareable by default
- In Global transaction, connections are shared simultaneously
- In LTC, connections are used serially (get/use/close, get/use/close, etc)
- LTC behavior: if application closes a connection, the connection will remain in shareable pool to be shared within the same LTC, until the LTC ends

Typical LTC Usage - servlets

- LTC A begin
  - Servlet A begin
    - get Connection A //shareable
    - use Connection A
    - close Connection A
    - // Connection A remains in shared pool, associated with LTC A
    - include or forward Servlet B
    - LTC A suspend
    - LTC B begin
      - Servlet B begin
        - get Connection B //shareable
        - use Connection B
        - close Connection B
        - // Connection B remains in shared pool, associated with LTC B
      - Servlet B end
    - LTC B end //Connection B released to free pool
    - LTC A resume
    - // possible Connection A re-use (get/use/close)
  - Servlet A end
- LTC A end //Connection A released to free pool

Unshareable connection

- Unshareable connections are not shared with other components in the application
- The component using the unshared connection has full control of this connection
- You need to close unshareable connection properly in your code

AGENDA

- What is J2C
- Connection Pooling
- Common Problems & troubleshooting
- Understanding J2C trace
- Q/A

ConnectionWaitTimeoutException

- This error can happen when the connection pool is over used
- Search for J2CA0045E to see if this error is present
- Review the connection pool:
  - If the maximum connection size is set too low
  - If the connection wait timeout is set too low
  - If application doesn't close some connections(causing connection leak)
  - If shareable connection is used properly

Deadlock

- Application requires more than one concurrent connection per thread, and the database connection pool is not large enough for the number of threads.
- Troubleshooting – collect javacores/ Javadumps and j2c trace

StaleConnectionException

- Indicates the connection being held is currently not valid, often with returned SQL Exception and error code.
- Determine if it's happening with specific query
- Check if there's potential network problem
- http://www-01.ibm.com/support/docview.wss?uid=swg21207584

Connection Leak

- A connection leak occurs when the application uses a connection but it never explicitly calls the close() method on the connection object to return the connection back to the connection pool for reuse.
- Also happens if a servlet gets a connection, uses it, and closes it, but then remains in the servlet for a relatively long period of time, that connection remains unavailable for use by other threads

Connection Leak

```
Connection Leak Logic Information:
  MCWrapper id 385d385d  Managed connection WSRdbManagedConnectionImpl@56d556d5  State:STATE_TRAN_WRAPPER_INUSE Thread Id: 00000153 Thread Name: WebContain
  er : 104 Handle count 1
     Start time inuse Fri Nov 25 09:19:22 CST 2011 Time inuse 29 (seconds)
     Last allocation time Fri Nov 25 09:19:22 CST 2011
       getConnection stack trace information:
          com.ibm.ejs.j2c.ConnectionManager.allocateConnection(ConnectionManager.java:895)
          com.ibm.ws.rsadapter.jdbc.WSJdbcDataSource.getConnection(WSJdbcDataSource.java:668)
          com.ibm.ws.rsadapter.jdbc.WSJdbcDataSource.getConnection(WSJdbcDataSource.java:635)
          org.springframework.jdbc.datasource.DataSourceUtils.doGetConnection(DataSourceUtils.java:113)
          org.springframework.jdbc.datasource.DataSourceUtils.getConnection(DataSourceUtils.java:79)
          [redacted]getConnection(DbService.java:63)
          [redacted]executeQuery(DbService.java:108)
          [redacted]getQueryLevel3MovaUseInfoHTML(rptKpiStatus.java:12164)
          [redacted].doQueryLevel3MovaUseInfo(rptKpiStatus.java:11872)
          sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
          sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:60)
          sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:37)
          java.lang.reflect.Method.invoke(Method.java:611)
          [redacted].dispatch(CoreForm.java:405)
          [redacted].execute(CoreForm.java:311)
          [redacted]execute(DispatchOperationStep.java:68)
          [redacted].doExecute(Unknown Source)
          [redacted]execute(Unknown Source)
          [redacted]executeRequest(Unknown Source)
          [redacted]processRequest(Unknown Source)
          [redacted]processRequest(HtmlRequestHandler.java:60)
          [redacted]service(Unknown Source)
          javax.servlet.http.HttpServlet.service(HttpServlet.java:831)
          com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:1657)
          com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:1597)
```

Solution to connections held by LTC

- Make connection unshareable.
  - http://www14.software.ibm.com/webapp/wsbroker/redirect?version=compass&product=was-express-dist&topic=tdat_conpoolman
- Make the shareable connections attached to Global transaction.

Dubug Process

- Trace specification:
  - Standard trace if connecting to a database:
    *=info:WAS.j2c=all:RRA=all:Transaction=all
  - Connection leak trace:
    - Part of WAS.j2c=all trace
    - Provides stack dumps for connections that are “in-use” more than 10 seconds
    - Trace string: ConnLeakLogic=all

Dubug Process

- Useful configuration files
  - resources.xml – includes the definition of the connection factory/data source. Make sure to get the one for the correct scope(Cell, Node, Cluster, Server scope)
  - ibm-ejb-bnd.xmi or ibm-web-bnd.xmi – for resource reference data
  - ejb-jar.xml or web.xml – for res-auth value

AGENDA

- What is J2C
- Connection Pooling
- Common Problems & troubleshooting
- Understanding J2C trace
- Q/A

Trace Analysis – getConnection

```
[11/25/11 9:19:51:828 CST] 00000043 ConnectionMan > allocateConnection in cm 42ee42ee
Entry
[11/25/11 9:19:51:828 CST] 00000043 ConnectionMan 3 This CM is
[ConnectionManager]@42ee42ee
JNDI Name <jdbc/mpvs>
shareable <true>
[11/25/11 9:19:51:828 CST] 00000043 PoolManager > reserve Entry
[11/25/11 9:19:51:828 CST] 00000043 PoolManager 3 input parms...
subject = Subject:
        Principal:
defaultWIMFileBasedRealm/server:pdccsmpvsapp001Cell01_pdccsmpvsapp002Node01_MPV
SServer
affinity = com.ibm.ws.LocalTransaction.LocalTranCoordImpl@61d661d6;RUNNING;
Shared connection = true
Force new MC = true
Connection Request Information =
com.ibm.ws.rsadapter.spi.WSConnectionRequestInfoImpl@79a779a7
.....
 UserName     = null
 Password     = null
 Catalog/default= null
 IsReadOnly/def = null
 TypeMap/default= null
....
 Isolation    = READ COMMITTED (2)
 Support isolation switching =false
 oracle props  = null
 Handle type   = java.sql.Connection
```

Trace Analysis – return a shared connection

[11/25/11 9:19:51:829 CST] 00000043 PoolManager 3 reserve(), Pool contents ==>
PoolManager name:jdbc/mpvs
PoolManager object:1175209484
Total number of connections: 25 (max/min 25/10, reap/unused/aged 180/1800/0, connectiontimeout/purge 180/EntirePool)
(testConnection/inteval false/0, stuck timer/time/threshold 0/0/0, surge time/connections 0/-1)
Shared Connection information (shared partitions 200)
<connection details>

Total number of connection in shared pool: 10
Free Connection information (free distribution table/partitions 12/1)
<connection details>

Total number of connection in free pool: 15
UnShared Connection information
No unshared connections

[11/25/11 9:19:51:829 CST] 00000043 PoolManager > processPoolRequestStats Entry
[11/25/11 9:19:51:829 CST] 00000043 PoolManager < processPoolRequestStats Exit
[11/25/11 9:19:51:829 CST] 00000043 PoolManager 3 Searching for shared connection in partition 102
[11/25/11 9:19:51:829 CST] 00000043 SharedPool > getSharedConnection Entry
[11/25/11 9:19:51:829 CST] 00000043 PoolManager > processPoolRequestStats Entry
[11/25/11 9:19:51:829 CST] 00000043 PoolManager < processPoolRequestStats Exit
[11/25/11 9:19:51:829 CST] 00000043 SharedPool < getSharedConnection, returning mcWrapper Exit
MCWrapper id 5d905d90 Managed connection WSRdbManagedConnectionImpl@69fe69fe State:STATE_TRAN_WRAPPER_INUSE Thread Id: 00000043 Thread Name: WebContainer : 16 Handle count 0

Trace Analysis – return a shared connection

```
[11/25/11 9:19:51:830 CST] 00000043 PoolManager 3 reserved managed connection
WSRdbManagedConnectionImpl@69fe69fe
[11/25/11 9:19:51:830 CST] 00000043 PoolManager < reserve Exit
[11/25/11 9:19:51:830 CST] 00000043 ConnectionMan 3 Using MCWrapper@5d905d90
[11/25/11 9:19:51:830 CST] 00000043 ConnectionMan < allocateMCWrapper Exit
[11/25/11 9:19:51:831 CST] 00000043 ConnectionMan > involveMCInTran Entry
[11/25/11 9:19:51:831 CST] 00000043 MCWrapper 3 involvedInTransaction: true
[11/25/11 9:19:51:831 CST] 00000043 ConnectionMan < involveMCInTran Exit
[11/25/11 9:19:51:831 CST] 00000043 MCWrapper > getConnection Entry
[11/25/11 9:19:51:831 CST] 00000043 MCWrapper < getConnection: Exit
                com.ibm.ws.rsadapter.jdbc.WSJdbcConnection@18c618c6
[11/25/11 9:19:51:831 CST] 00000043 ConnectionMan < allocateConnection 18c618c6 Exit
```

Trace Analysis – closeConnection (LTC not complete)

- [11/25/11 9:19:51:954 CST] 000000b5 ConnectionEve > connectionClosed Entry
- [11/25/11 9:19:51:954 CST] 000000b5 ConnectionEve 3 ***Connection Close Request*** Handle Name: com.ibm.ws.rsadapter.jdbc.WSJdbcConnection@2fba2fba Connection Pool: jdbc/mpvs Details: : MCWrapper id 50b550b5 Managed connection WSRdbManagedConnectionImpl@68756875 State:STATE_TRAN_WRAPPER_INUSE Thread Id: 000000b5 Thread Name: WebContainer : 77 Handle count 1
- **[11/25/11 9:19:51:954 CST] 000000b5 MCWrapper 3 involvedInTransaction: true**
- [11/25/11 9:19:51:954 CST] 000000b5 ConnectionEve < connectionClosed Exit

Trace Analysis – release the connection

```
[11/25/11 9:19:52:143 CST] 00000155 LocalTransact > beforeCompletion Entry
[11/25/11 9:19:52:143 CST] 00000155 LocalTransact < beforeCompletion Exit
[11/25/11 9:19:52:143 CST] 00000155 LocalTransact > afterCompletion Entry
[11/25/11 9:19:52:143 CST] 00000155 LocalTransact 3 Using transaction wrapper@5efa5efa
[11/25/11 9:19:52:143 CST] 00000155 LocalTransact 3 Releasing the connection to the pool.
shareable = true handleCount = 0 isStale = false
[11/25/11 9:19:52:143 CST] 00000155 PoolManager > release Entry
[11/25/11 9:19:52:143 CST] 00000155 PoolManager 3 input parms...
MC = MCWrapper id 5ea55ea5 Managed connection
WSRdbManagedConnectionImpl@37c037c State:STATE_ACTIVE_INUSE Thread Id: 00000155
Thread Name: WebContainer : 106 Handle count 0

input affinity = com.ibm.ws.LocalTransaction.LocalTranCoordImpl@6f756f75;COMPLETED;
[11/25/11 9:19:52:143 CST] 00000155 PoolManager 3 release(), Pool contents ==>
PoolManager name:jdbc/mpvs
PoolManager object:1175209484
Total number of connections: 25 (max/min 25/10, reap/unused/aged 180/1800/0,
connectiontimeout/purge 180/EntirePool)
                (testConnection/inteval false/0, stuck timer/time/threshold 0/0/0, surge
time/connections 0/-1)
Shared Connection information (shared partitions 200)
    .....
    com.ibm.ws.LocalTransaction.LocalTranCoordImpl@6f756f75;COMPLETED; MCWrapper
id 5ea55ea5 Managed connection WSRdbManagedConnectionImpl@37c037c
State:STATE_ACTIVE_INUSE Thread Id: 00000155 Thread Name: WebContainer : 106 Handle
count 0
    ....
Free Connection information (free distribution table/partitions 12/1)
    ....
 Total number of connection in free pool: 16
UnShared Connection information
 No unshared connections
```

Trace Analysis – release the connection

```
[11/25/11 9:19:52:143 CST] 00000155 SharedPool > removeSharedConnection Entry
[11/25/11 9:19:52:143 CST] 00000155 SharedPool 3 Removed connection
[11/25/11 9:19:52:143 CST] 00000155 SharedPool < removeSharedConnection Exit
[11/25/11 9:19:52:143 CST] 00000155 FreePool > returnToFreePool, datasource: jdbc/mpvs
Entry
[11/25/11 9:19:52:143 CST] 00000155 MCWrapper > cleanup Entry
[11/25/11 9:19:52:143 CST] 00000155 MCWrapper 3 Clear the McWrapper handlelist for
the following MCWrapper: MCWrapper id 5ea55ea5 Managed connection
WSRdbManagedConnectionImpl@37c037c State:STATE_ACTIVE_INUSE Thread Id: 00000155
Thread Name: WebContainer : 106 Handle count 0
[11/25/11 9:19:52:143 CST] 00000155 MCWrapper 3 InstanceOf
DissociatableManagedConnection is true In ConnectionManager MCWrapper id 5ea55ea5
Managed connection WSRdbManagedConnectionImpl@37c037c State:STATE_ACTIVE_INUSE
Thread Id: 00000155 Thread Name: WebContainer : 106 Handle count 0

[11/25/11 9:19:52:143 CST] 00000155 MCWrapper 3 Calling mc.dissociateConnections()
[11/25/11 9:19:52:143 CST] 00000155 MCWrapper 3 Returned from
mc.dissociateConnections()
[11/25/11 9:19:52:143 CST] 00000155 MCWrapper < cleanup Exit
[11/25/11 9:19:52:144 CST] 00000155 FreePool < returnToFreePool Exit
[11/25/11 9:19:52:144 CST] 00000155 PoolManager 3 release(), Pool contents ==>
                  PoolManager name:jdbc/mpvs
PoolManager object:1175209484

Free Connection information (free distribution table/partitions 12/1)
.....
 (0)(0)MCWrapper id 5ea55ea5 Managed connection
WSRdbManagedConnectionImpl@37c037c State:STATE_ACTIVE_FREE
....

[11/25/11 9:19:52:145 CST] 00000155 PoolManager 3 released managed connection
WSRdbManagedConnectionImpl@37c037c
[11/25/11 9:19:52:145 CST] 00000155 PoolManager < release Exit
[11/25/11 9:19:52:145 CST] 00000155 LocalTransact < afterCompletion Exit
```

Q&A