

Best Practices for Developing High Performing Java Applications for Db2 for z/OS

Maryela Weihrauch

IBM Distinguished Engineer WW Analytics on System z

weihrau@us.ibm.com

May 2020

Availability. References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

© **Copyright IBM Corporation 2012. All rights reserved.**

- **U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.**
- *Please update paragraph below for the particular product or family brand trademarks you mention such as WebSphere, Db2, Maximo, Clearcase, Lotus, etc*

IBM, the IBM logo, ibm.com, [IBM Brand, if trademarked], and [IBM Product, if trademarked] are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml

If you have mentioned trademarks that are not from IBM, please update and add the following lines:

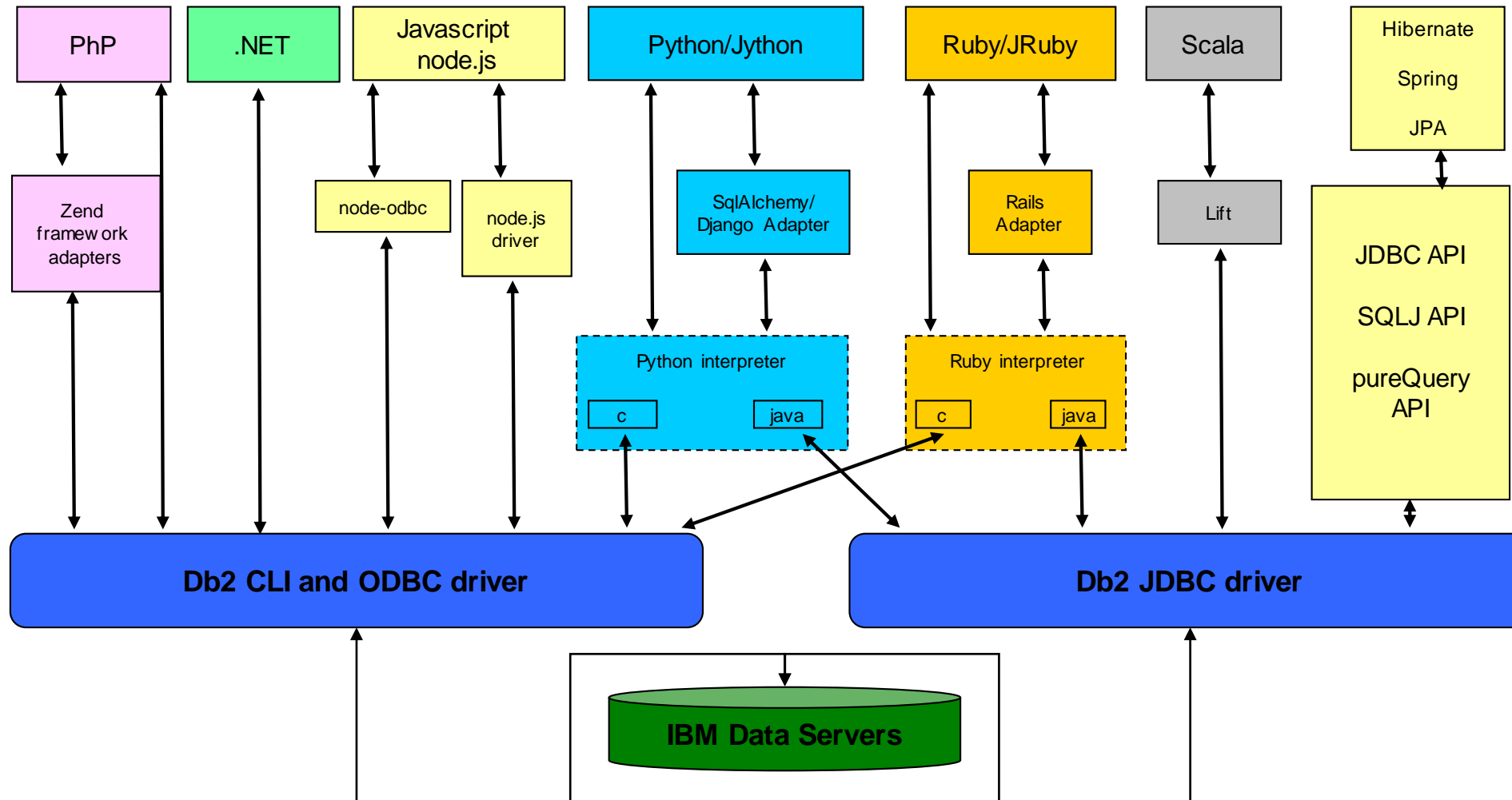
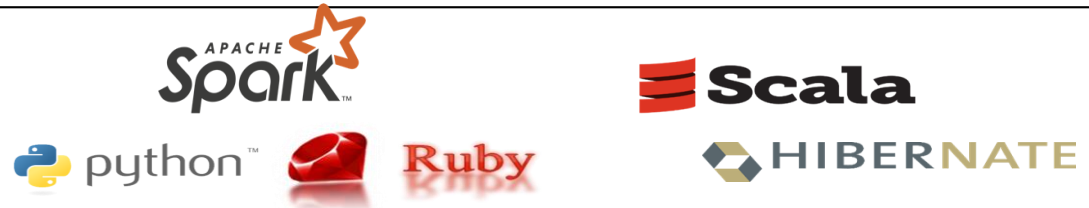
[Insert any special 3rd party trademark names/attributions here]

Other company, product, or service names may be trademarks or service marks of others.

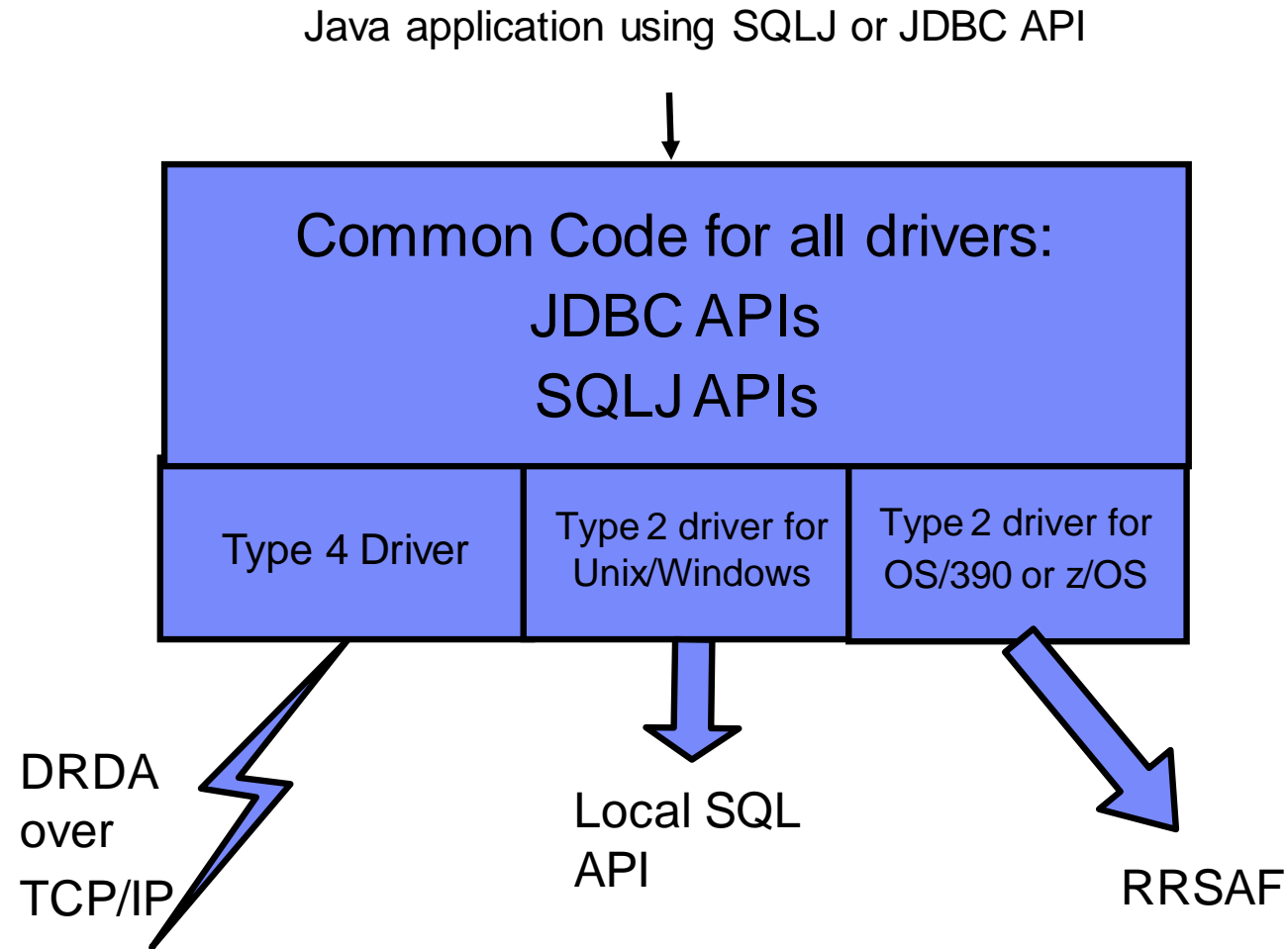
Agenda

- Db2 JDBC driver architecture and API details
- Connection Management
- Db2 Dynamic Statement Cache
- Best practice for SQL Execution

Db2 Drivers – Panoramic View



Data Server Driver for JDBC/SQLJ - Architecture



Getting a Connection

▪ java.sql.DriverManager API

- The actual driver type determined during runtime from the connection URL format:
 - Type 2 - "jdbc:db2:database"
 - Type 4 - "jdbc:db2://host:port/database" (default port number 446)

```
Class.forName("com.ibm.db2.jcc.DB2Driver");  
Connection con = DriverManager.getConnection("jdbc:db2://localhost:50000/sample",  
"username", "password");
```

- Disadvantage – Reduces portability due to class name and URL.

▪ javax.sql.DataSource API

- Logical name mapped to DataSource object via JNDI naming service
- logical name -> driver info, DB name, IP, port, user, password, etc.
- App servers used to manage & configure data sources
- Makes application portable

```
Context ctx=new InitialContext();  
DataSource ds=(DataSource)ctx.lookup("jdbc/sampledb");  
Connection con=ds.getConnection();
```

Driver and Connection Properties

- Global driver properties can be provided through Java system properties or a properties file.
- JDBC API defines a set of properties to identify and describe a DataSource implementation.
- Properties may be specified in any of three ways
 - JDBC 1 connection properties passed as a `java.util.Properties` object (consisting of key/value pairs) as argument to `DB2Driver.connect()` or `java.sql.DriverManager.getConnection()`
 - As part of the database URL itself
 - `jdbc:db2://server[:port]/databaseName[:propertyKey=value;...]`
 - `jdbc:db2://localhost:50000/TESTDB:user=foo;password=bar;`
 - Using `setXXX` methods
 - most properties are defined in the abstract `com.ibm.db2.jcc.DB2BaseDataSource` class
- Examples of properties - `ProgressiveStreaming`, **`securityMechanism`**, `loginTimeout`, `keepdynamic`, `deferPrepares`, **`enableSysplexWLB`**, `currentExplainMode`, `cursorSensitivity`, `maxTransportObjects`, `traceFile`, `traceLevel`, **`currentSchema`**, `currentSQLID`, `dumpPool`

Connection Optimization

- All DB resources hang off Connection objects, must be managed carefully
- Creating and terminating a connection is resource consuming, both in the driver and Db2
- Client obtaining a physical database connection requires multiple network requests to
 - Handshake on DRDA protocol
 - Validate client user credentials
 - Establish code page, packet sizes etc
- Get a connection object when needed
- Reuse a connection object for multiple Statement objects when possible, use connection pooling
- Close connections promptly, don't leave connection cleanup up to garbage collection

Use ClientInfo fields

- Can be used in WLM, RLF and profile definition and performance monitoring
- WebSphere Application Server supports explicit and implicit setting of client information

- Example how to call explicitly

...

```
WSConnection conn = (WSConnection) ds.getConnection();  
props.setProperty(WSConnection.CLIENT_ID, "user123");  
conn.setClientInformation(props);
```

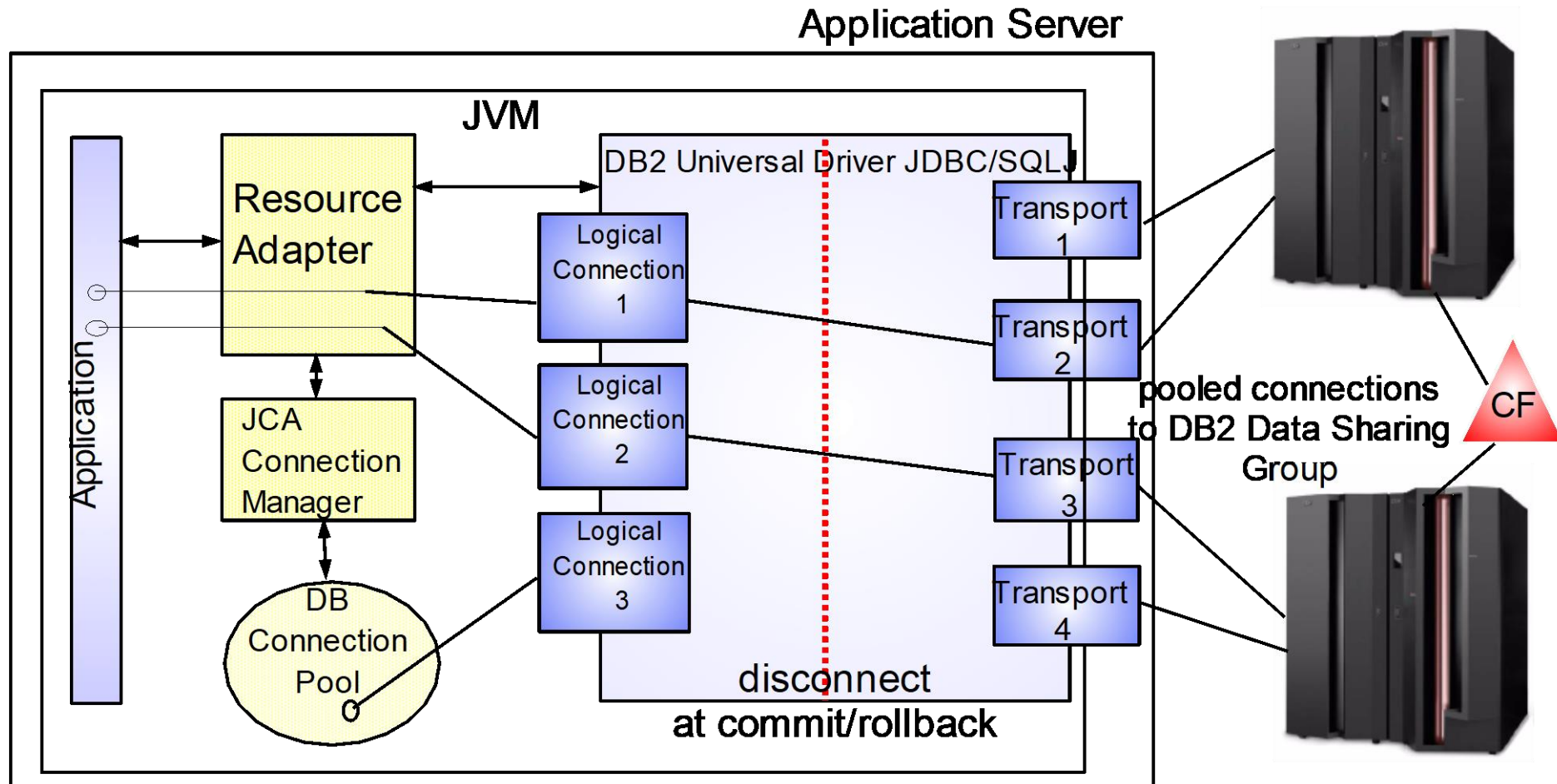
- Example how to call implicitly by turning on WebSphere Trace Group

WAS.clientinfo=all=enabled or

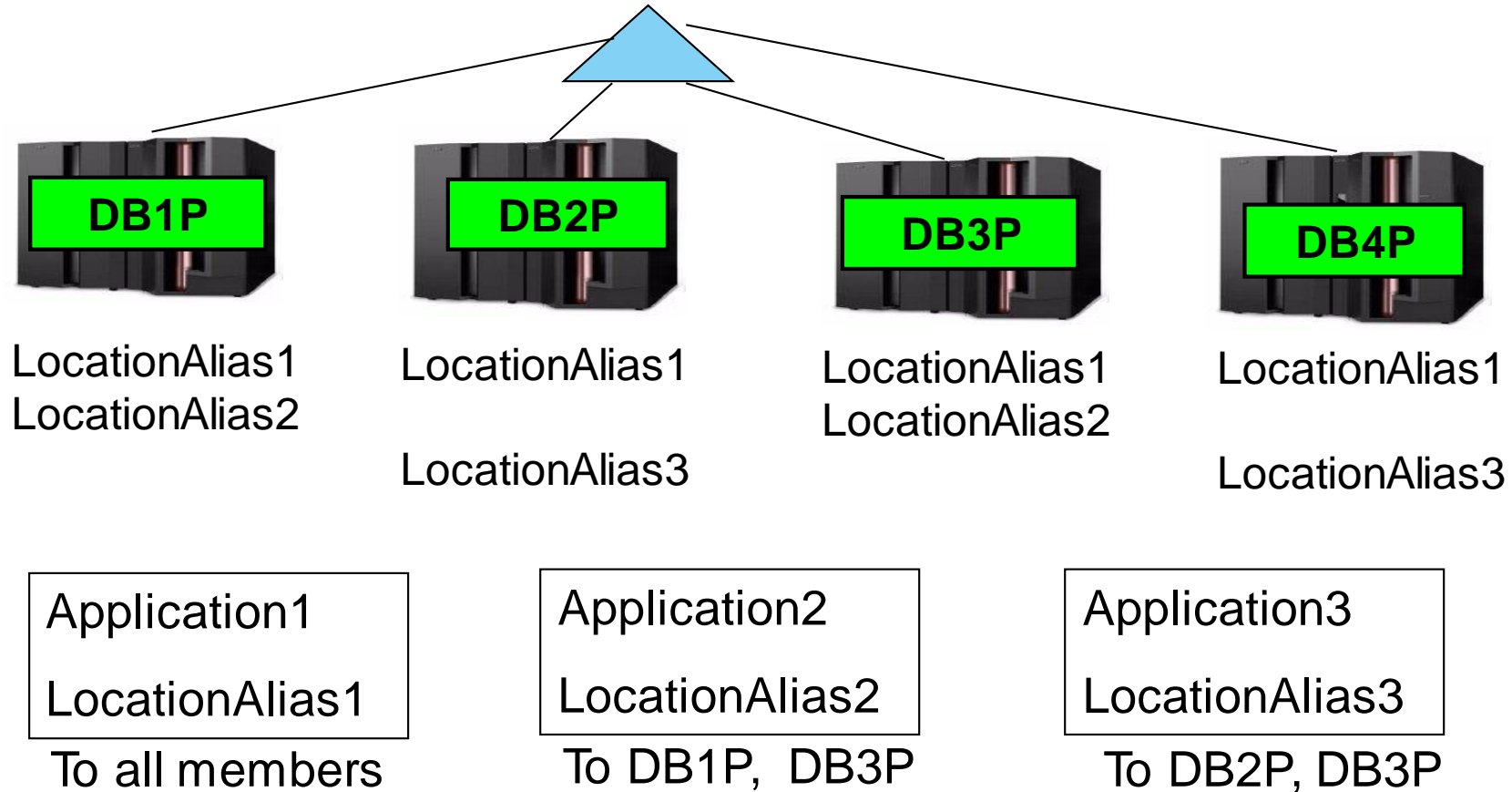
WAS.clientinfopluslogging=all=enabled

Sysplex Workload Balancing

- Data Server Driver Type 4 supports sysplex distribution and transparent failover at transaction boundary



Db2 Location Alias for Subgrouping



-MODIFY DDF command with the ***ALIAS*** keyword to configure and manage aliases dynamically without taking a Db2 or DDF outage.

Db2 High Performance DBAT

- **High Performance DBATs reduce CPU consumption by**
 - Supporting RELEASE(DEALLOCATE) to avoid repeated package allocation/deallocation
 - Avoids processing to go inactive and then back to active

- **Enable High Performance DBAT**
 - BIND client packages into different collection coll2 with RELEASE(DEALLOCATE)
 - BIND other frequently executed packages with RELEASE(DEALLOCATE)
 - In case of MDM REBIND trigger packages
 - Set -MODIFY DDF PKGREL(BNDOPT) to enable

- **In WAS datasource property definition point to new collection**
 - E.g. jdbcCollection=coll2

Db2 Connection Profile

- **Problem:** For distributed workloads, low priority or poorly behaving client applications may monopolize Db2 resources and prevent high-priority applications from executing.

 - **Solution:** Increased granularity of monitoring for system level activities
 - Number of connections
 - Number of threads
 - Idle thread timeout

 - Profiles specified in **SYSIBM.DSN_PROFILE_TABLE**
 - Db2 supports filtering and threshold monitoring of system related activities via keywords
 - Number of threads
 - Number of connections
 - Idle thread timeout
 - Scope filters
 - ROLE (available through Trusted Context support)
 - Product-specific identifier
- Db2 11 special register
 - Db2 12 Db2-provided global var

Prepared Statement Objects - Benefits

- PreparedStatement objects vs. Statement objects
 - 2 DB calls needed for fetch (describe, data)
 - PreparedStatement makes description calls at construction time, Statement makes them on every execution.
 - PreparedStatement enables Statement Pooling
 - Use Statement when SQL is not executed often

- PreparedStatement object pool
 - Client side optimization
 - Pool of PreparedStatement and CallableStatement objects, not active in a Connection
 - Reduced overhead of Java object creation and garbage collection
 - Pool exists for the life of an open connection, effectiveness depends on connection pooling.
 - No impact to application

- **No concern for SQLJ**

Statement or PreparedStatement

- Statement example

```
Statement stmt = con.createStatement();
stmt.executeUpdate("INSERT INTO EMPLOYEE VALUES( 'John', 123 )");
stmt.executeUpdate("INSERT INTO EMPLOYEE VALUES( 'Mary', 425 )");
```

- PreparedStatement example

```
PreparedStatement ps = con.prepareStatement("INSERT INTO EMPLOYEE VALUES(?, ?)");
ps.setString(1,"John");
ps.setInt(2,123);
ps.executeUpdate();
ps.setString(1,"Mary");
ps.setInt(2,425);
ps.executeUpdate();
```

Db2 Dynamic Statement Cache

- Dynamic statement prepared at run time
- Dynamic statement cache
 - To improve performance of dynamic SQL
 - Enabled by DSNZPARM CACHEDYN = YES
 - Allows applications (multiple threads) to reuse and share prepared statements
- Prepared stmt is saved in an in-memory cache
- Subsequent prepares of same stmt loads from cache if cache match criteria met (sql, authid, special regs, bind options etc.)
- Cache pool shared by different threads, plans and packages (“global cache”)
- Good cache hit rate produces significant performance benefits
- A Full Prepare can consume 10-100X more CPU than a Short Prepare!

Dynamic Statement Cache

- SQL can be EXPLAINED using the 'EXPLAIN STMTCACHE' feature.
 - Populates various explain tables with details on statements in the dynamic statement cache including access path information
- Use Dynamic SQL Stmt section of statistics to monitor the Global Cache Hit Ratio % to determine if the cache size needs to be increased.

| DYNAMIC SQL STMT | QUANTITY | /SECOND | /THREAD | /COMMIT |
|----------------------------|----------|---------|---------|---------|
| PREPARE REQUESTS | 5099.0K | 16.2K | N/C | 7.83 |
| FULL PREPARES | 0.00 | 0.00 | N/C | 0.00 |
| SHORT PREPARES | 5098.5K | 16.2K | N/C | 7.83 |
| GLOBAL CACHE HIT RATIO (%) | 100.00 | N/A | N/A | N/A |
| IMPLICIT PREPARES | 0.00 | 0.00 | N/C | 0.00 |
| PREPARES AVOIDED | 0.00 | 0.00 | N/C | 0.00 |
| CACHE LIMIT EXCEEDED | 0.00 | 0.00 | N/C | 0.00 |
| PREP STMT PURGED | 0.00 | 0.00 | N/C | 0.00 |
| LOCAL CACHE HIT RATIO (%) | N/C | N/A | N/A | N/A |
| CSWL - STMTS PARSED | 0.00 | 0.00 | 0.00 | 0.00 |
| CSWL - LITS REPLACED | 0.00 | 0.00 | 0.00 | 0.00 |
| CSWL - MATCHES FOUND | 0.00 | 0.00 | 0.00 | 0.00 |
| CSWL - DUPLS CREATED | 0.00 | 0.00 | 0.00 | 0.00 |

Literal Replacement for Global Dynamic Statement Cache

- Dynamic SQL with literals can be re-used in the cache
 - Literals replaced with &
(similar to parameter markers but not the same)
- To enable set the property `enableLiteralReplacement='YES'` in the JCC Driver
- Lookup Sequence
 - Original SQL with literals is looked up in the cache
 - If not found, literals are replaced and new SQL is looked up in the cache
 - Additional match on literal usability
 - Can only match with SQL stored with same attribute, not parameter marker
 - If not found, new SQL is prepared and stored in the cache
- **Db2 12 support as BIND option CONCENTRATESTMT on package**

Why SQLJ?

- Static SQL performance for Java applications
- Static SQL authorization model
- Monitoring/Manageability
 - Static SQL packages for accounting/monitoring
 - Static SQL locks in access path, so that access path changes do not occur without a conscious choice
- Measurements with the IRWW workload comparing JDBC vs SQLJ with the T2 driver

| | Throughput (ETR) | Normalized Throughput (ITR) | z/OS CPU Utilization | CL.1 CPU time |
|----------------|------------------|-----------------------------|----------------------|--------------------|
| JDBC T2 | 2636.83 | 3773.37 | 69.88 | 0.000672 |
| SQLJ T2 | 2694.80 (+2.20%) | 5174.35 (+37.13%) | 52.08 (-25.47%) | 0.000457 (-32.00%) |

Batching INSERT and SELECT

- INSERT – JDBC API

```
ps = conn.prepareStatement("INSERT INTO EMPLOYEE VALUES (?)")
ps.setInt(1)
ps.addBatch()
ps.setInt(2)
ps.addBatch()
Int[] returncodes = ps.executeBatch()
```

- SELECT - Db2 driver extension (executeDB2QueryBatch)

```
PreparedStatement ps = conn.prepareStatement("SELECT * from T1 where C1 = ?")
ps.setInt(1,1)
ps.addBatch()
ps.setInt(1,2)
ps.addBatch()
((com.ibm.db2.jcc.DB2PreparedStatement)pstmt).executeDB2QueryBatch();
While (ps.getMoreResults()) {
    Rs = ps.getResultSet()
    While (rs.next()){
    }
}
rs.close
```

Java Performance Problem Areas

➤ Java Application

- Autocommit(on) - default
- Mismatch of Java and Db2 data types
- Usage of String for numbers
- Retrieval of unused columns (select *)
- Transaction isolation REPEATABLE READ (default in WAS) or SERIALIZABLE
- Open cursor SELECT ... FOR UPDATE for locking semantics
 - Consider using WITH RS USE AND KEEP UPDATE LOCKS

➤ JDBC

- JDBC resources not closed (cursor, statements, connections)
- No usage of Parameter Markers
 - E.g. select c1, c2 FROM t1 WHERE c3=?
-> use literal replacement option
- Cursor are defined as hold by default
- Usage of Statement() instead of PreparedStatement() objects
 - No object caching in WAS

Locking and Concurrency

- If deadlocks and timeouts, turn on Db2 Performance Trace class(6)
 - Use LOCKSIZE ROW selectively for top reported tables
 - Combine with MEMBER CLUSTER if data sharing to reduces page P-lock and page latch contention on data pages
 - Can be defined via deferred ALTER/REORG

- Review indexes
 - Missing index causing table scan and deadlocks
 - Drop unused indexes and Ris

- Zparm SKIPUNCI – skip uncommitted inserts for ISOLATION(CS|RS)

Let's Practice

| AVERAGE | APPL (CL.1) | DB2 (CL.2) |
|--------------|-------------|------------|
| ----- | ----- | ----- |
| ELAPSED TIME | 0.278042 | 0.000280 |
| NONNESTED | 0.278042 | 0.000280 |
| STORED PROC | 0.000000 | 0.000000 |
| UDF | 0.000000 | 0.000000 |
| TRIGGER | 0.000000 | 0.000000 |
| CP CPU TIME | 0.000111 | 0.000080 |
| AGENT | 0.000111 | 0.000080 |
| NONNESTED | 0.000111 | 0.000080 |
| STORED PRC | 0.000000 | 0.000000 |
| UDF | 0.000000 | 0.000000 |
| TRIGGER | 0.000000 | 0.000000 |
| PAR.TASKS | 0.000000 | 0.000000 |
| SE CPU TIME | 0.000097 | 0.000071 |
| NONNESTED | 0.000097 | 0.000071 |
| STORED PROC | 0.000000 | 0.000000 |
| UDF | 0.000000 | 0.000000 |
| TRIGGER | 0.000000 | 0.000000 |

- Application uses Type 2 connectivity.
- User wants to change to type 4 connectivity to save CPU
- Db2 is not setup for distributed workloads which would create a considerable test effort to introduce in production
- Would you recommend to start the project to change the application to use type 4 connectivity?

Let's Practice

| AVERAGE | APPL (CL.1) | DB2 (CL.2) |
|--------------|-------------|------------|
| ----- | ----- | ----- |
| ELAPSED TIME | 0.278042 | 0.000280 |
| NONNESTED | 0.278042 | 0.000280 |
| STORED PROC | 0.000000 | 0.000000 |
| UDF | 0.000000 | 0.000000 |
| TRIGGER | 0.000000 | 0.000000 |
| CP CPU TIME | 0.000111 | 0.000080 |
| AGENT | 0.000111 | 0.000080 |
| NONNESTED | 0.000111 | 0.000080 |
| STORED PRC | 0.000000 | 0.000000 |
| UDF | 0.000000 | 0.000000 |
| TRIGGER | 0.000000 | 0.000000 |
| PAR.TASKS | 0.000000 | 0.000000 |
| SE CPU TIME | 0.000097 | 0.000071 |
| NONNESTED | 0.000097 | 0.000071 |
| STORED PROC | 0.000000 | 0.000000 |
| UDF | 0.000000 | 0.000000 |
| TRIGGER | 0.000000 | 0.000000 |

- Would you recommend to start the project to change the application to use type 4 connectivity?
- Type 4 connectivity would provide ~ 55% zIIP offload
- Type 2 connectivity in this report already get almost 50% zIIP offload
- ✓ Potential gCP saving is small, may not justify project cost

Let's Practice

| SQL DML | AVERAGE | TOTAL |
|----------|---------|--------|
| SELECT | 0.00 | 0 |
| INSERT | 0.00 | 0 |
| ROWS | 0.00 | 0 |
| UPDATE | 0.00 | 0 |
| ROWS | 0.00 | 0 |
| MERGE | 0.00 | 0 |
| DELETE | 0.00 | 0 |
| ROWS | 0.00 | 0 |
| | | |
| DESCRIBE | 0.94 | 50413 |
| DESC.TBL | 0.00 | 0 |
| PREPARE | 1.00 | 53571 |
| OPEN | 1.00 | 53571 |
| FETCH | 1.00 | 53571 |
| ROWS | 1.22 | 65594 |
| CLOSE | 0.00 | 0 |
| | | |
| DML-ALL | 3.94 | 211126 |

| NORMAL TERM. | AVERAGE | TOTAL |
|-----------------|---------|-------|
| NEW USER | 0.00 | 0 |
| DEALLOCATION | 0.00 | 7 |
| APPL.PROGR. END | 0.00 | 0 |
| RESIGNON | 1.00 | 53571 |
| DBAT INACTIVE | 0.00 | 0 |
| TYPE2 INACTIVE | 0.00 | 0 |
| RRS COMMIT | 0.00 | 0 |

- What performance optimization would you recommend for this Java application?
 - Consider SQL per commit
 - Consider thread management per commit

Let's Practice

| SQL DML | AVERAGE | TOTAL |
|----------|---------|--------|
| SELECT | 0.00 | 0 |
| INSERT | 0.00 | 0 |
| ROWS | 0.00 | 0 |
| UPDATE | 0.00 | 0 |
| ROWS | 0.00 | 0 |
| MERGE | 0.00 | 0 |
| DELETE | 0.00 | 0 |
| ROWS | 0.00 | 0 |
| | | |
| DESCRIBE | 0.94 | 50413 |
| DESC.TBL | 0.00 | 0 |
| PREPARE | 1.00 | 53571 |
| OPEN | 1.00 | 53571 |
| FETCH | 1.00 | 53571 |
| ROWS | 1.22 | 65594 |
| CLOSE | 0.00 | 0 |
| | | |
| DML-ALL | 3.94 | 211126 |

| NORMAL TERM. | AVERAGE | TOTAL |
|-----------------|---------|-------|
| NEW USER | 0.00 | 0 |
| DEALLOCATION | 0.00 | 7 |
| APPL.PROGR. END | 0.00 | 0 |
| RESIGNON | 1.00 | 53571 |
| DBAT INACTIVE | 0.00 | 0 |
| TYPE2 INACTIVE | 0.00 | 0 |
| RRS COMMIT | 0.00 | 0 |

- Mostly 1 single row select per Commit
 - Application uses auto commit
 - > Turn auto commit off
- Application could benefit from JDBC packages bound with RELEASE(DEALLOCATE) avoiding resignon
- Cost of avoidable thread management and commit is multiple of cost executing simple SQL

Configuration Options Accessing Data in Db2 for z/OS

P Program S Static SQL
C COBOL D Dynamic SQL

| Diagram | Hop # | Pay load | Network Protocol | SQL Statement | zIIP Offload | Dev. Skill |
|---------|-------|-----------------|----------------------------|----------------------------------|--|------------|
| | 3 | DB2 native data | TCPIP/ Sync Inbound | Coded in Client App. Dynamic SQL | DDF connection part only | Easy |
| | 4 | DB2 native data | TCPIP/ Sync Inbound | Coded in Client App. Dynamic SQL | DDF connection part only | Easy+ |
| | 4 | CTG native data | TCPIP/ Sync Inbound | Coded in CICS App. Static SQL | Some CTG portion | Mode rate |
| | 5 | MQ MSG | TCPIP/ Asyn In & Out bound | Coded in CICS App. Static SQL | All GCP | Mode rate+ |
| | 3 | REST/JSON | HTTP/ Sync Inbound* | Defined in DB2 z/OS. Static SQL | DDF connection partly. 90%+ on z/OSCEE | Super Easy |
| | 4 | REST/JSON | HTTP/ Sync Inbound* | Coded in CICS App. Static SQL | 90%+ on z/OSCEE | Easy |

Note (*) : equally applies to .NET and ODBC/CLI driver
 (**): equally applies to IMS TM

Summary

- Business critical Java applications with Db2 for z/OS as enterprise database server have been implemented commonly and successfully for many years now
- Going through installation checklist is highly recommended prior to each implementation to ensure success
 - Communication among WAS Administrator, Db2 System Programmer, and Application Architect
- No shortcuts in respect to setup for availability
 - User sees application availability and not Db2 system availability
- Monitor and react proactively and do not wait until user complains
 - Workload behavior changes over time

Useful links

- [http://www.bwdb2ug.org/PDF/Dynamic Statement Cache in a Nutshell.pdf](http://www.bwdb2ug.org/PDF/Dynamic%20Statement%20Cache%20in%20a%20N%20utshell.pdf)
- <http://www.ibm.com/systems/z/specialtyengines/>
- IBM Redbook - Db2 for z/OS and WebSphere Integration for Enterprise Java Applications –
<http://www.redbooks.ibm.com/abstracts/sg248074.html?Open>

Maryela Weihrauch
IBM
weihrau@us.ibm.com

Session code:E15



IDUG

Leading the Db2 User
Community since 1988

Please fill out your session
evaluation before leaving!

