DB2 Universal Java Client: Managing Enterprise Class Java Applications

Curt Cotner,
IBM Distinguished Engineer
cotner@us.ibm.com

Enabling Your
On Demand DB2 World

INTERNATIONAL DB2 USERS GROUP
Independent • Not-for-Profit • User Run

Systems Development Life Cycle

- Strategy
  - establish business objectives
- Analysis
  - identify user requirements
- Design
  - system flow, screen layouts, database schema, etc.
- Build
  - write the code
- Test
  - QA process - validate code/performance
- Deploy
  - put the system into production, monitor, etc.

Build -- application development in Java

- DB2 Universal Java Driver
  - support for JDBC 3.0 and SQLJ 3.0
  - type 2 and type 4 drivers
  - improved deployment and ease of use
  - integration with WebSphere
- SQLJ -- static SQL for Java application development
  - more concise syntax
  - easier to code/understand
  - better security/authorization characteristics
  - better performance
- WSAD -- integrated development environment for both SQLJ and JDBC
  - significant improvements with WSAD 5.1
  - more improvements coming...

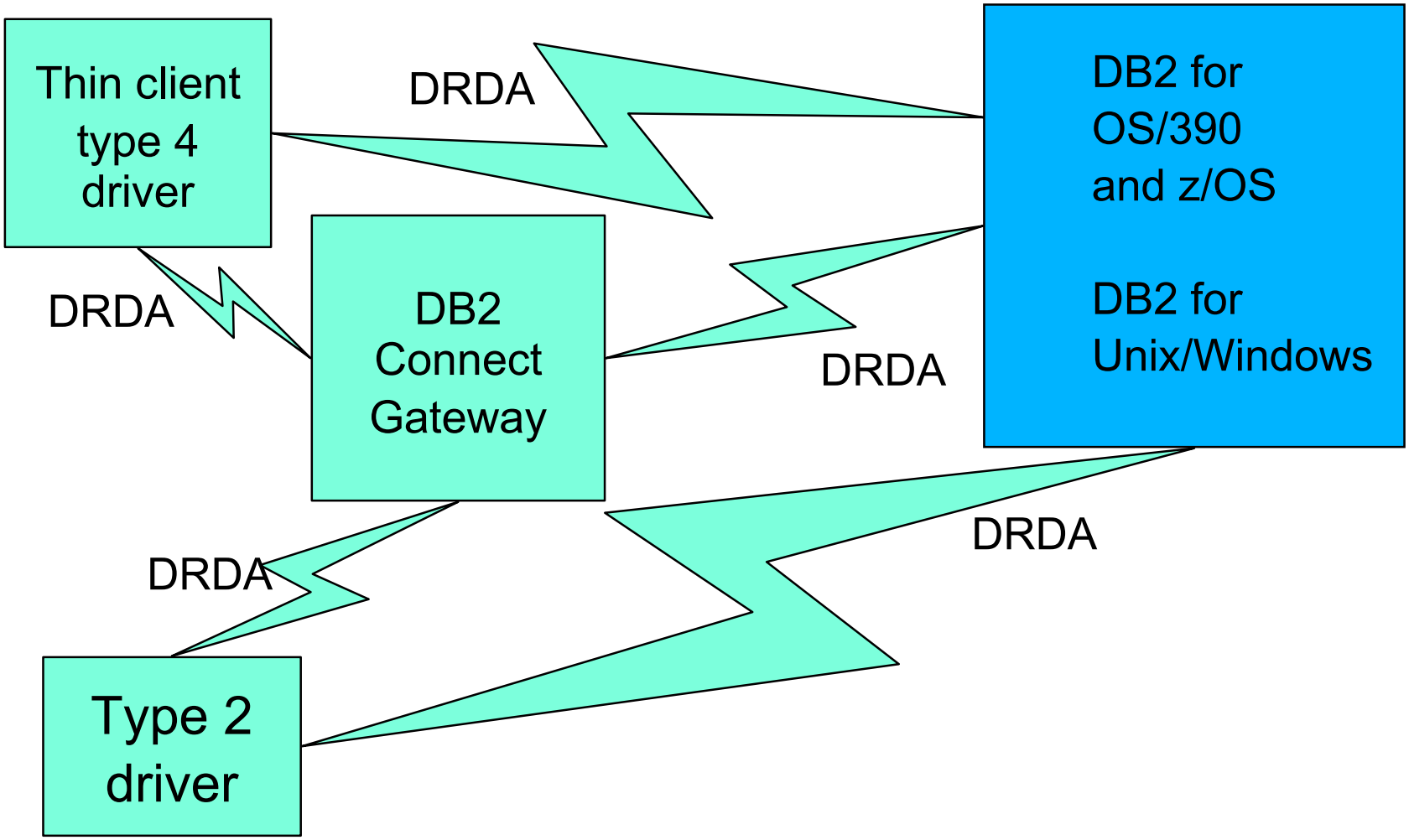
What is the DB2 Universal Java Client?

- Significant reengineering of Java support for DB2 Connect, CAE, and DB2 for OS/390 and z/OS client software
- Uses DRDA protocols for all client communication
  - eliminates DB2RA and net driver protocols
  - traditional DB2 client configuration (CCA) is optional
  - much better handling of different client/server software levels
- Improved DB2 Connect consistency/performance
  - much higher percentage of common code
  - fewer unique code paths for specific hardware configurations
- Several significant improvements to DRDA
  - support for long SQL names and statements
  - DRDA query block sizes can now be up to 2M bytes
  - server-supplied stored procedures for SQL error messages, metadata
  - many internal performance improvements

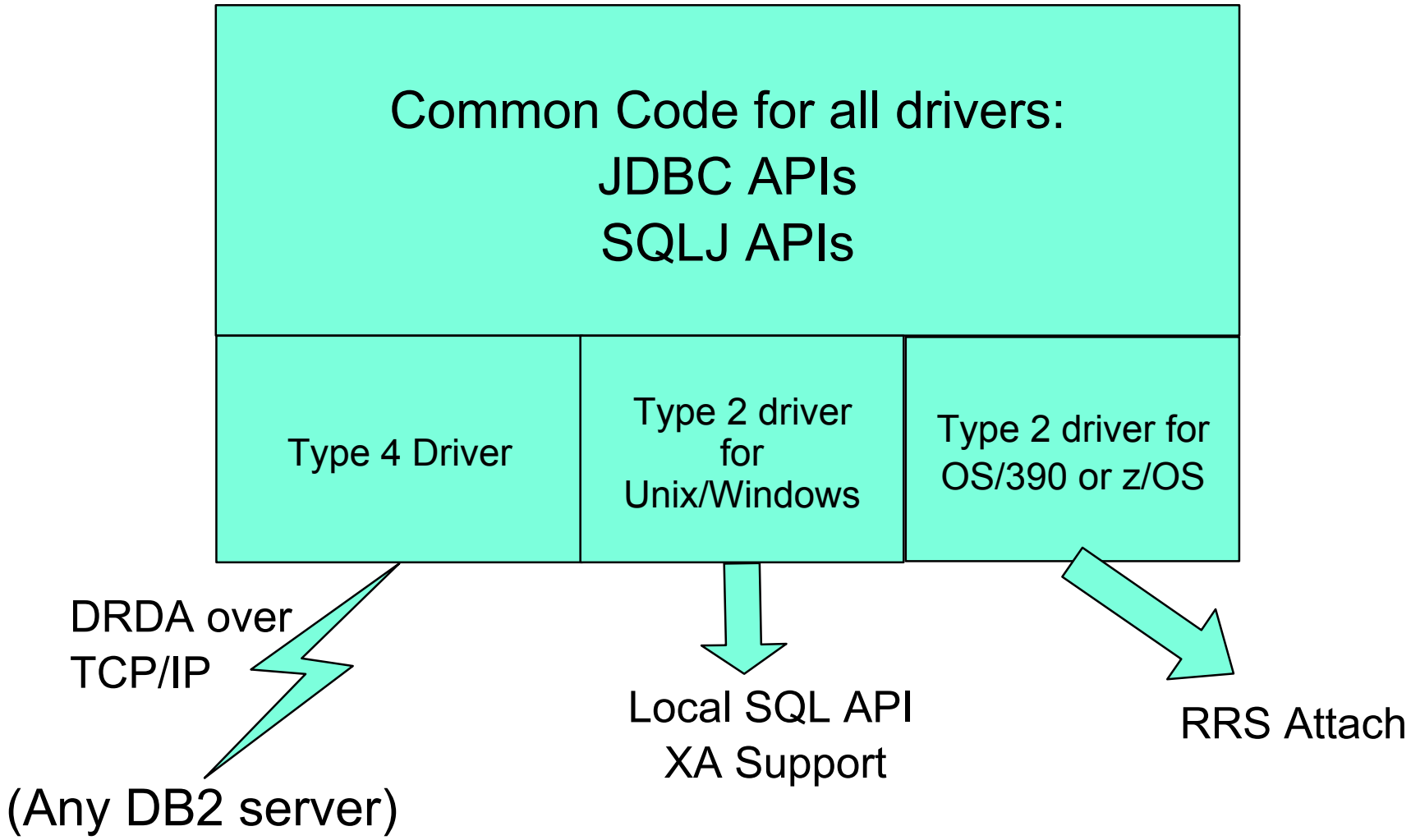
Key Java Improvements

- Type 2 and type 4 driver
  - Updated to support JDBC/SQLJ 3.0 standard
    - savepoint support
    - new metadata for PreparedStatements
    - return autogenerated keys
    - multiple open ResultSets for a single stored procedure
    - WITH HOLD cursors
    - improved BLOB/CLOB support

New DB2 UDB Universal Java Client



DB2 Java Universal Client Internal Architecture



Universal Java Driver Support

- WebSphere on Unix/Windows
  - Type 4 – WAS v5.0.1, v4.0.6
  - Type 2 – WAS v5.0.2, v4.0.7
- WebSphere on z/OS v5.0.2 (both Type 2 and Type 4 supported)
- DB2 Servers:
  - DB2 for iSeries (v5.1)
  - DB2 for z/OS
    - Type 4 (v7)
    - Type 2 (v6)
  - DB2 for Windows, UNIX, Linux (v8.1)
  - Cloudscape v5.1

Why use SQLJ?

- Static SQL performance for Java applications
  - less code written by the application programmer
  - resulting code is easier to maintain
  - significant performance advantage over JDBC
- Static SQL authorization model
  - provides Java with a stronger authorization model
- Monitoring/managability
  - static SQL packages for accounting/monitoring
  - static SQL locks in access path, so that access path changes don't occur without a conscious choice

Retrieve a single row from DB2

SQLJ:

```
#sql [con] { SELECT ADDRESS INTO :addr FROM EMP
             WHERE NAME=:name };
```

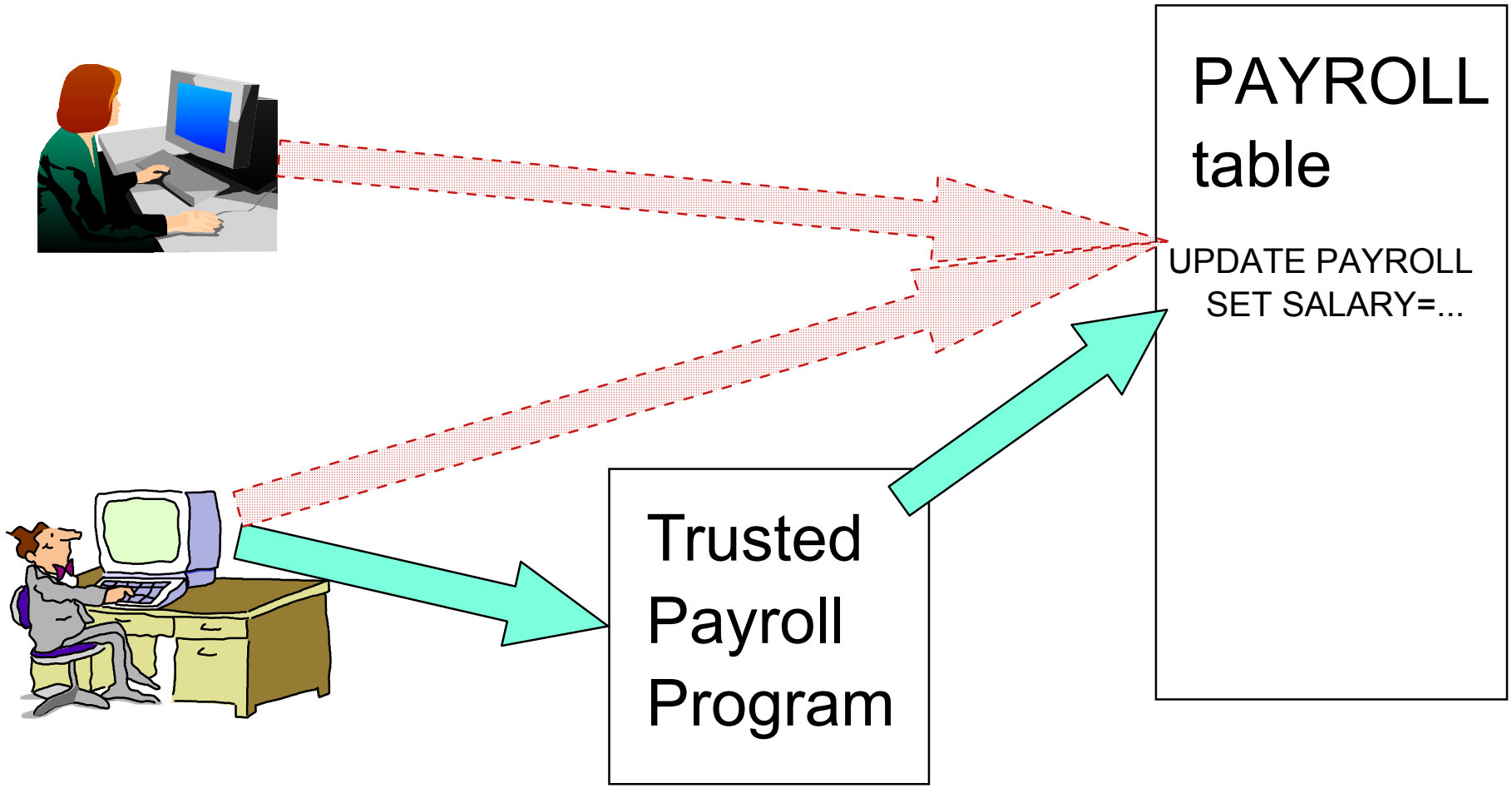
JDBC:

```
java.sql.PreparedStatement ps = con.prepareStatement(
      "SELECT ADDRESS FROM EMP WHERE NAME=?");
ps.setString(1, name);
java.sql.ResultSet names = ps.executeQuery();
names.next();
addr = names.getString(1);
names.close();
```

-- concise
-- strong typing

-- portable across platforms and DBMSs
-- compile/bind time schema checking
-- static SQL performance and authorization!!!

Authorization semantic issues



Static SQL Authorization

- Static SQL is associated with "program"
  - plans/packages identify "programs" to DB2
  - program author's table privileges are used
  - end users are granted EXECUTE on program
- Dynamic SQL is associated with "user"
  - no notion of "program"
  - end users must have table privileges
  - BIG PROBLEM FOR A LARGE ENTERPRISE!!!

SQLJ Application Development

- 100% Java application process
  - eliminates DBRM files and .bnd files
- New SQLJ serialized profile format
  - fully portable to all platforms -- user can deploy on any server platform without running db2profc on the target system.
  - contains information needed for all BIND operations, without having to recustomize on each BIND
  - allow multiple class file to be bound into a single DB2 package
- Simplifies deployment of applications, but does require changes in existing procedures used by SQLJ users.

WSAD 5.1 Tooling

- Support for generating SQLJ for CMP beans
  - includes static singleton select for improved performance
- SQLJ programs are fully supported by the WSAD workbench
  - .sqlj and .ser files are first class objects now
- Support for access intent has been added
  - better control over isolation level
  - automatically generates KEEP UPDATE LOCKS for JDBC/SQLJ access to DB2 for OS/390 when required
- Built-in SQLJ profile translation and customization tool
- SQLJ editor
- SQLJ debugger

Test -- validate correctness/performance

- Improvements in testing program correctness:
  - Improved JDBC and SQLJ trace
  - Trace integrated with WebSphere trace
  - APIs for reporting SQLCA contents
  - Server generated SQL error message text
- Improvements to validate consistent performance
  - SQLJ and statically bound packages
  - ability to bind multiple .ser files into a single set of packages

New Java Universal Driver Trace

- **Trace activation**
  - External API of com.ibm.db2.jcc.DB2Connection
  - Dynamically turn trace on/off
  - Multiple levels of trace detail
  - Helpful for analyzing method flows, DRDA buffer
  - Calling example
    ((DB2Connection) con).setJCCLogWriter(
    java.io.PrintWriter logWriter, int tracelevel);
- **Example of trace output**

```
[ibm][db2][jcc][Thread:main][Connection@50b9ee8a]setAutoCommit(false) called
[ibm][db2][jcc][Thread:main][Connection@50b9ee8a]prepareStatement(SELECT
  FKEY FROM WRKTB01 WHERE (FKEY >= ?) OPTIMIZE FOR 1 ROW ) called
[ibm][db2][jcc][Thread:main][Connection@50b9ee8a]prepareStatement () returned
  PreparedStatement@ee32e8a
[ibm][db2][jcc][Thread:main][PreparedStatement@ee32e8a]setShort (1, 400) called
[ibm][db2][jcc][Thread:main][PreparedStatement@ee32e8a]executeQuery () called
[ibm][db2][jcc][Thread:main][ResultSetMetaData@2b98ae8a]BEGIN
  TRACE_RESULT_SET_META_DATA
[ibm][db2][jcc][Thread:main][ResultSetMetaData@2b98ae8a]Result set meta data for
  statement Statement@136bee8a
[ibm][db2][jcc][Thread:main][ResultSetMetaData@2b98ae8a]Number of result set
  columns: 12
```

Improved SQL Error Information

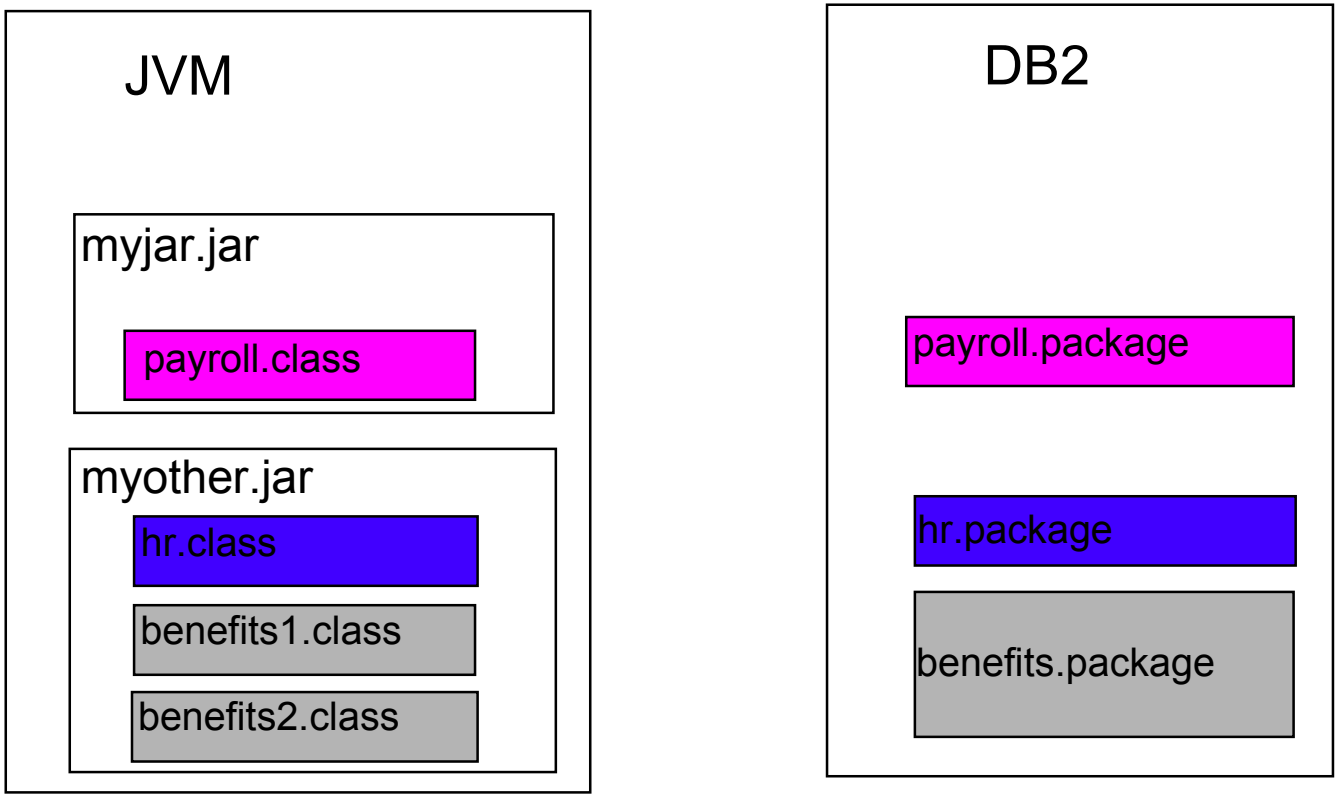
- DB2Diagnosable class for reporting contents of the SQLCA and SQL error message text
  - getSQLCode()
  - getSQLErrmc()
  - getSQLErrp()
  - getSQLErrd()
  - getSQLState()
  - getSQLWarn()
  - getSQLErrorMessage()
- Information is accessible for both JDBC and SQLJ whenever an SQL exception is thrown

Native DB2 Server SQL Error Messages

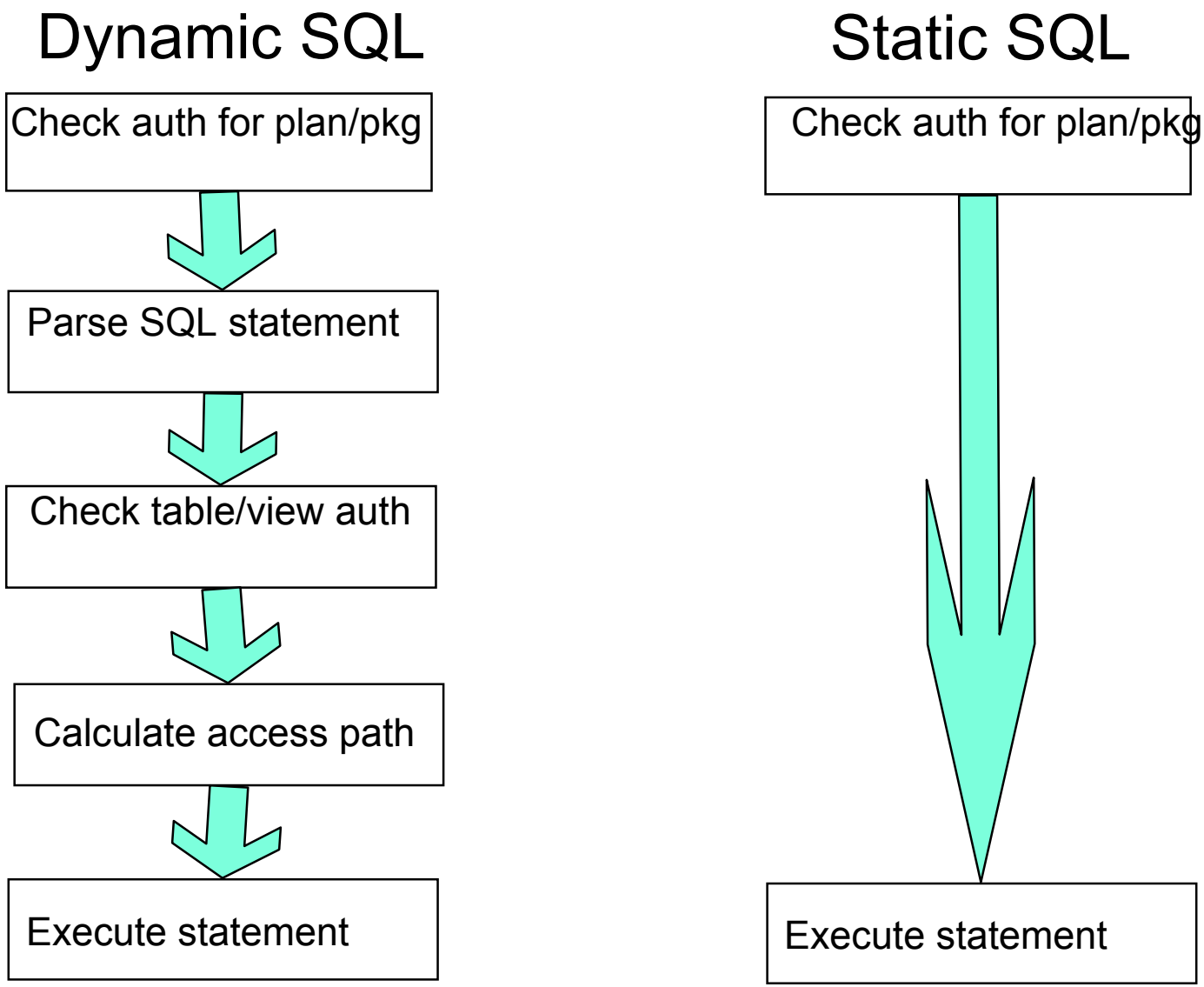
- "Error Message" stored procedures are provided by each DB2 server (including DB2 for OS/390 V6 and V7)
- Allows DB2 client to return "native" error message text for the target DB2 server
- Native error message is only returned when explicitly requested
  - getSQLErrorMessage()

SQLJ packages -- consistent performance

-- SQL statements are recorded in the DB2 catalog
-- SQL access paths are pre-bound for each SQL statement
-- access paths don't change until next REBIND
-- package names visible to online monitoring tools (DB2PM, etc.)
-- EXPLAIN data can be saved during program deployment



Static SQL is FASTER!!!



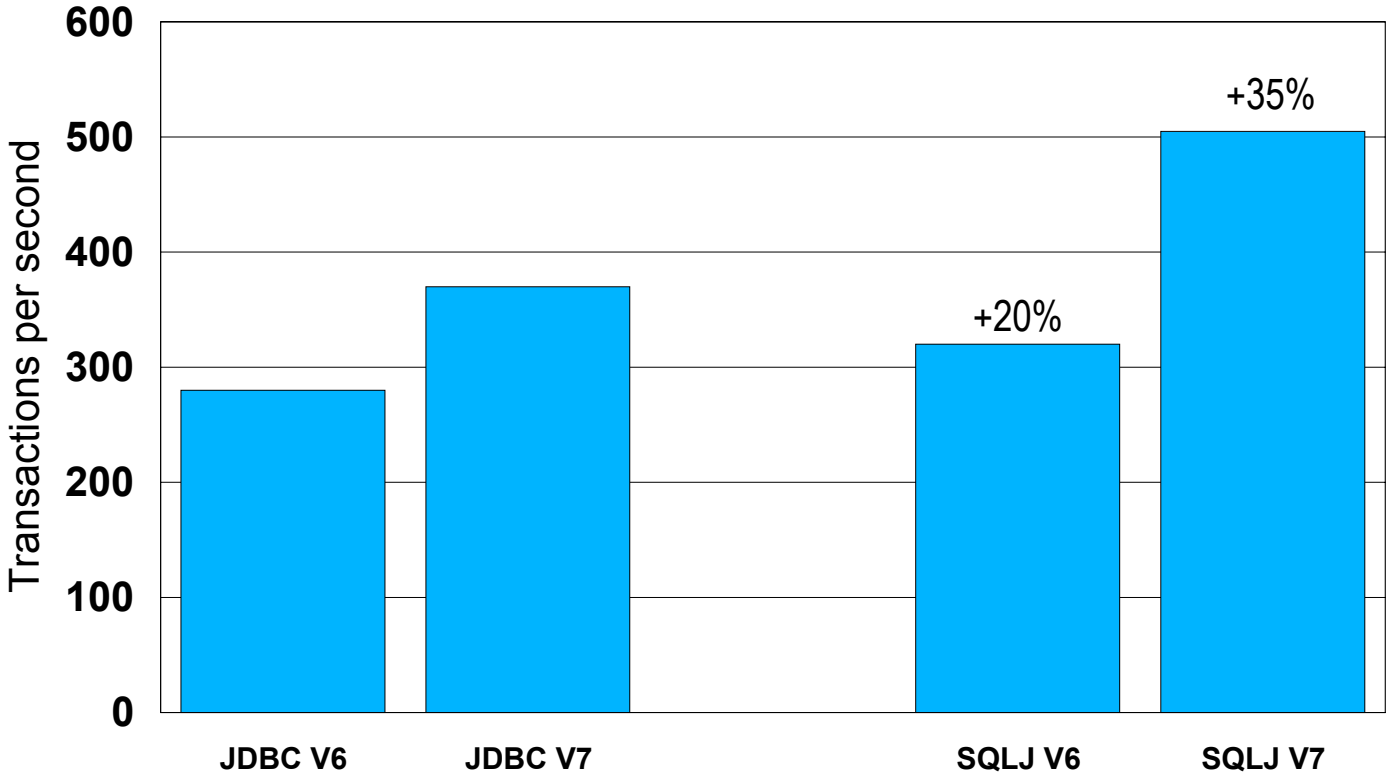
SQL API Comparison

Unix Client

Major Financial Firm

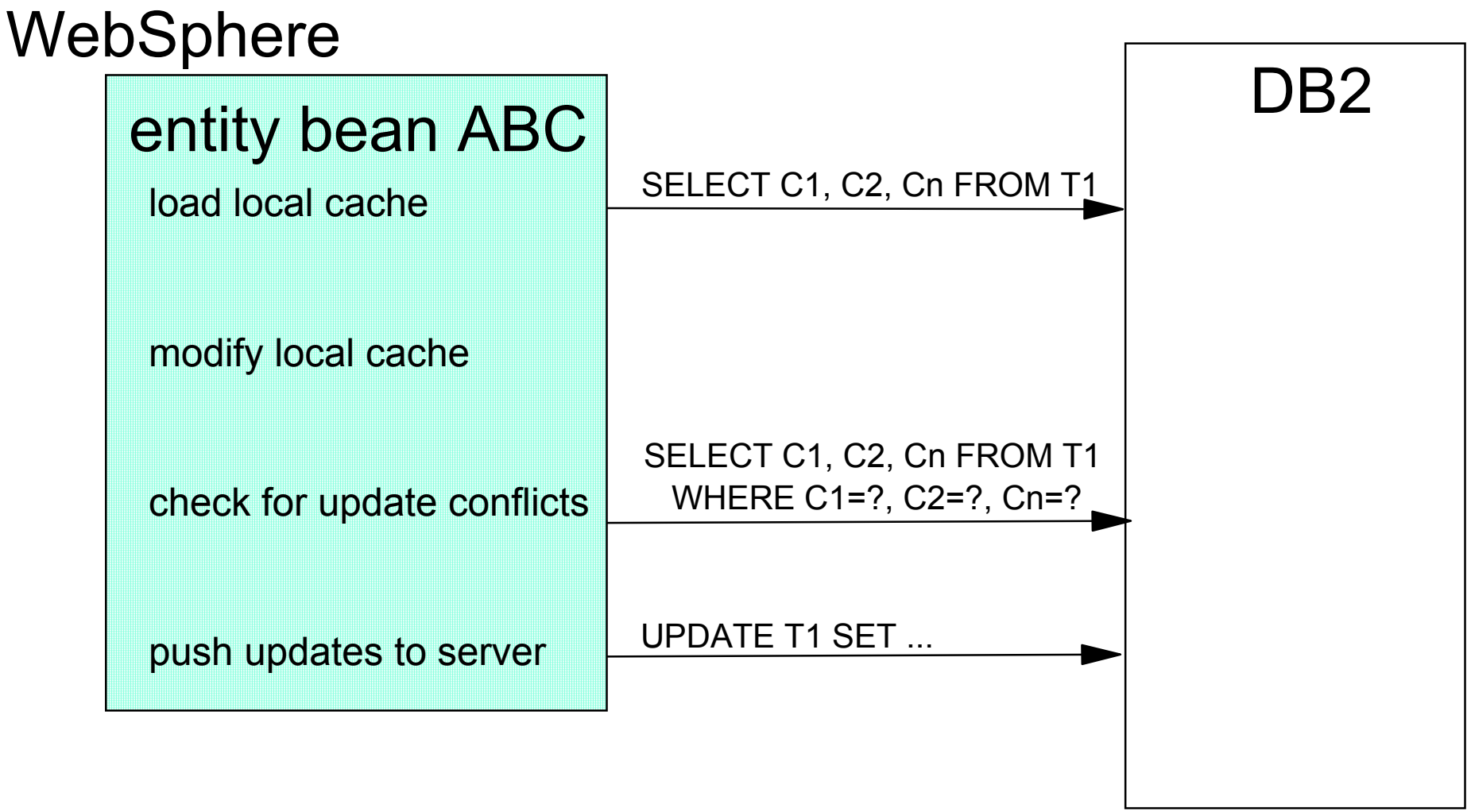
Driver	Elapsed Time	CPU Time (Class 1)	Network Messages
Embedded C	0.929	0.092	87
V7 JDBC T2	1.798	0.134	147
V7 SQLJ T2	6.253	0.341	611
V8 JDBC T4	2.004	0.138	147
V8 SQLJ T4	0.790	0.091	87

Java API Performance Comparisons



Normalized throughput for zSeries G7 with 3 engines with 100% cache hit for JDBC.
SQLJ advantage increased from 20% to 35% when Java overhead was reduced.

WebSphere EJB Persistence

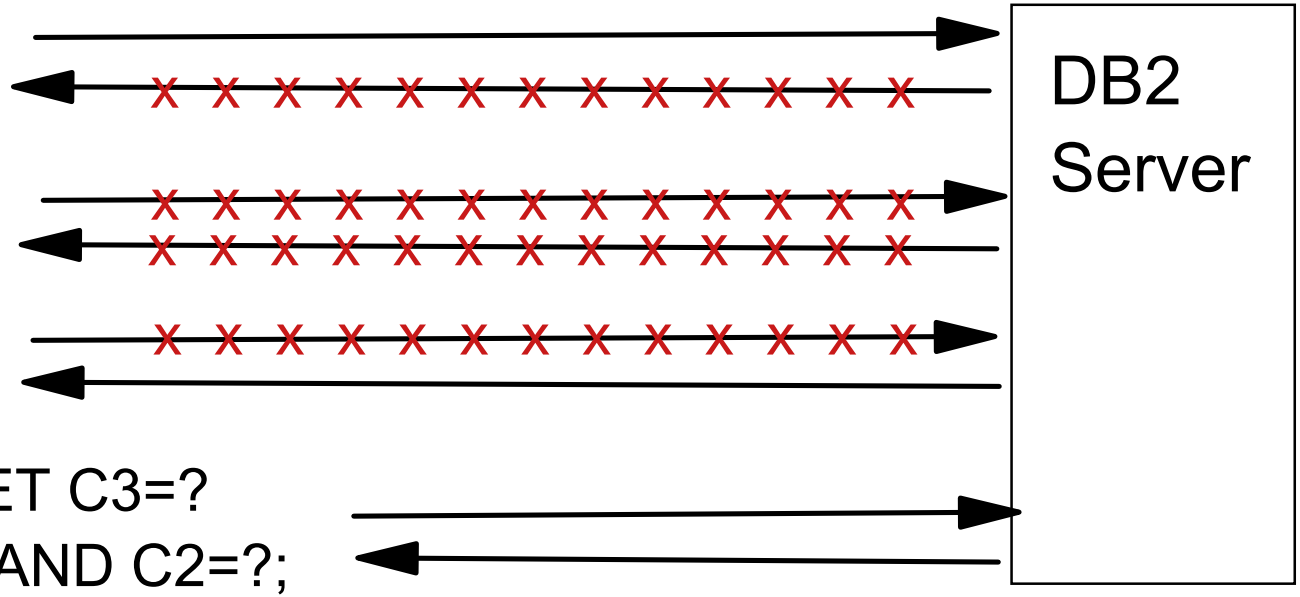


READ ONLY USING UPDATE LOCKS

- Allows WebSphere persistence layer to minimize network traffic when using searched update and pessimistic locking.

```
PREPARE "SELECT C1, C2, C3 FROM T1
    WHERE  C1=? AND C2=? AND C3=? FOR READ ONLY
    KEEP UPDATE LOCKS"
OPEN
FETCH
CLOSE

UPDATE T1 SET C3=?
WHERE C1=? AND C2=?;
```



SQLJ for WebSphere Persistence

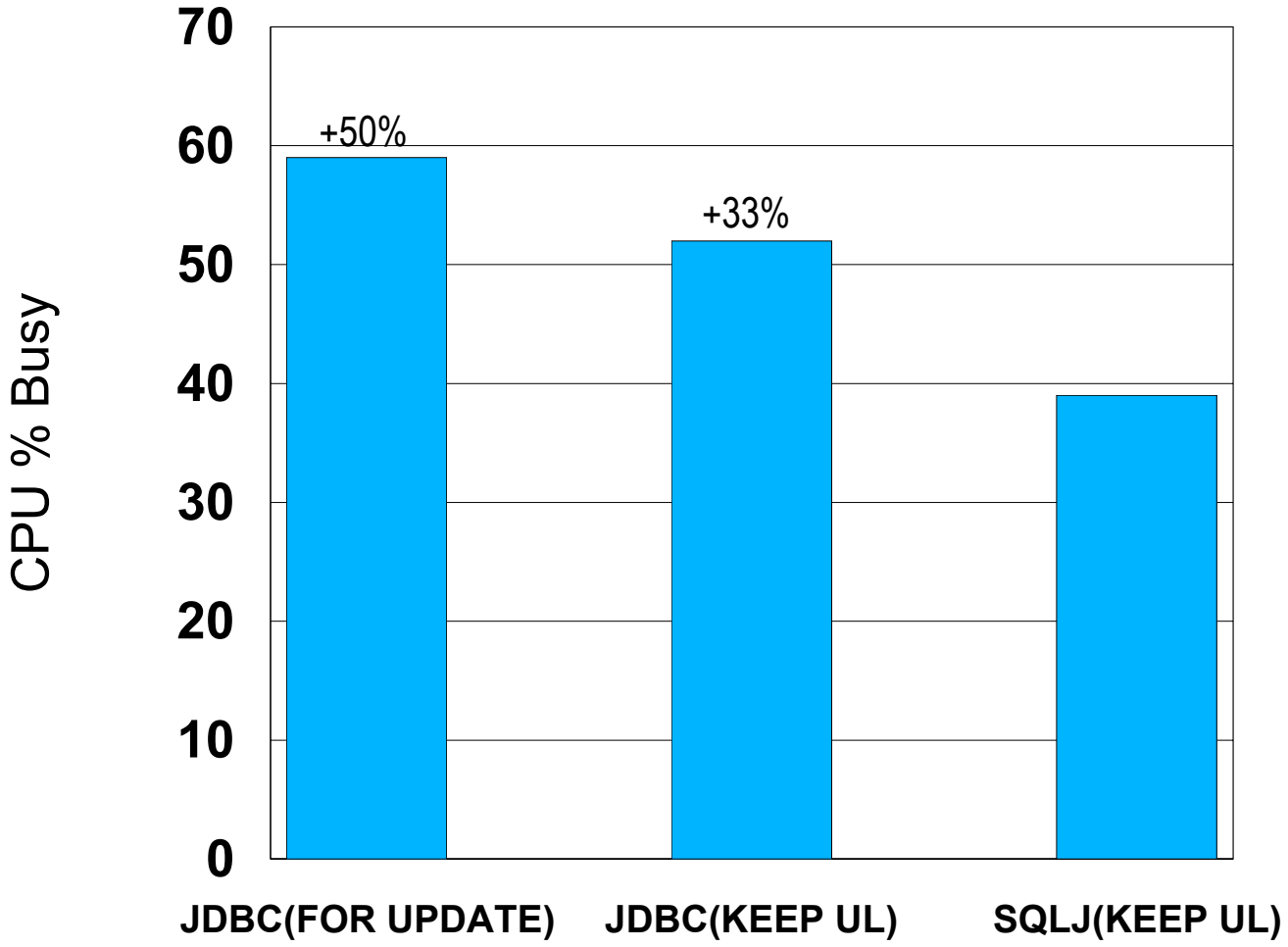
JDBC -- 10 API calls

- Load data into cache
  - PREPARE "SELECT..."
  - OPEN
  - FETCH
  - CLOSE
- Make local cache updates
- Check for changes at server and lock row
  - PREPARE "SELECT... KEEP UPDATE LOCKS WHERE..."
  - OPEN
  - FETCH
  - CLOSE
- Push update to server
  - PREPARE "UPDATE..."
  - EXECUTE

SQLJ -- 3 API calls

- Load data into cache
  - SELECT INTO...
- Make local cache updates
- Check for changes at server and lock row
  - SELECT INTO ... KEEP UPDATE LOCKS WHERE..."
- Push update to server
  - UPDATE ... WHERE ...

Java API Performance Comparisons
FOR READ ONLY KEEP UPDATE LOCKS



IS NOT DISTINCT FROM

- SQL uses three-valued logic where any given comparison can return: TRUE, FALSE, or NULL
- Applications can use IS NOT DISTINCT FROM to obtain a TRUE result instead of NULL when a comparing NULL values

```
SELECT C1 FROM T1 WHERE
        C1 IS NOT DISTINCT FROM :hv;
```

C1 value	:hv value	RESULT
NULL	'ABC'	FALSE
NULL	NULL	TRUE
'ABC'	'ABC'	TRUE
'ABC'	NULL	FALSE
'ABC'	'DEF'	FALSE

Deploy -- production usage/monitoring

- Production applications must be reliable
  - application should perform/behave consistently unless an explicit change was authorized
  - an access path change can be an outage if it happens to the wrong SQL statement in a critical application!!!
- Production monitoring
  - which applications are running?
  - how much resource is application XYZ using?
  - which programs caused the deadlock or timeout?
  - which user is running on a given DB2 thread?
  - what is causing increased elapsed time in application ABC?

Default monitoring for JDBC

-D71B DIS THREAD(*)

```
NAME     ST A  REQ ID               AUTHID    PLAN       ASID  TOKEN
SERVER   RA *   952 db2jccThread USRT001  DISTSERV 004A   11
 V445-G91E81C5.G49D.00F330EEE11F=122 ACCESSING DATA FOR 9.30.129.197
SERVER   RA *   112 db2jccThread USRT001  DISTSERV 004A   123
 V445-G91E81C5.G49D.00F330DF1111=222 ACCESSING DATA FOR 9.30.129.197
SERVER   RA *   432 db2jccThread USRT001  DISTSERV 004A   432
 V445-G91E81C5.G49D.00F330DF736F=432 ACCESSING DATA FOR 9.30.129.197
SERVER   RA *   772 db2jccThread USRT001  DISTSERV 004A   21
 V445-G91E81C5.G49D.00F330DF736F=382 ACCESSING DATA FOR 9.30.129.197
```

-- All programs typically have the same package name...

-- Access paths can change on any given PREPARE

Instrumentation APIs for JDBC or SQLJ

- New Java methods for existing Set Client Information API
  - setClientUser("maryela1")
  - setClientWorkStation("9.30.11.123")
  - setClientApplicationInformation("payment")
  - setClientAccountingInformation(String)

- For all users: provides additional monitoring information

```
-D71B DIS THREAD(*)
NAME     ST A   REQ ID           AUTHID   PLAN     ASID TOKEN
SERVER   RA *   952 db2jccThread USRT001  DISTSERV 004A   432
 V437-WORKSTATION=9.30.11.123, USERID=maryela1,
      APPLICATION NAME=payment
 V445-G91E81C5.G49D.00F330DF736F=432 ACCESSING DATA FOR
9.30.129.197
```

- For DB2 for z/OS: Strings are included in IFC records

```
   LOCATION: SYPEC15A                          DB2 PERFORMANCE
MONITOR
      GROUP: N/P                                 ACCOUNTING REPORT
  SUBSYSTEM: V71A                                     ORDER: TRANSACT
                              #OCCURS  ... CLASS1 EL.TIME CLASS2 EL.TIME
TRANSACT                      #DISTRS  ... CLASS1 CPUTIME CLASS2 CPUTIME
----------------------------- -------      -------------- --------------
payment                             4            1.000902       0.005561
                                    4            0.003573       0.002457
```

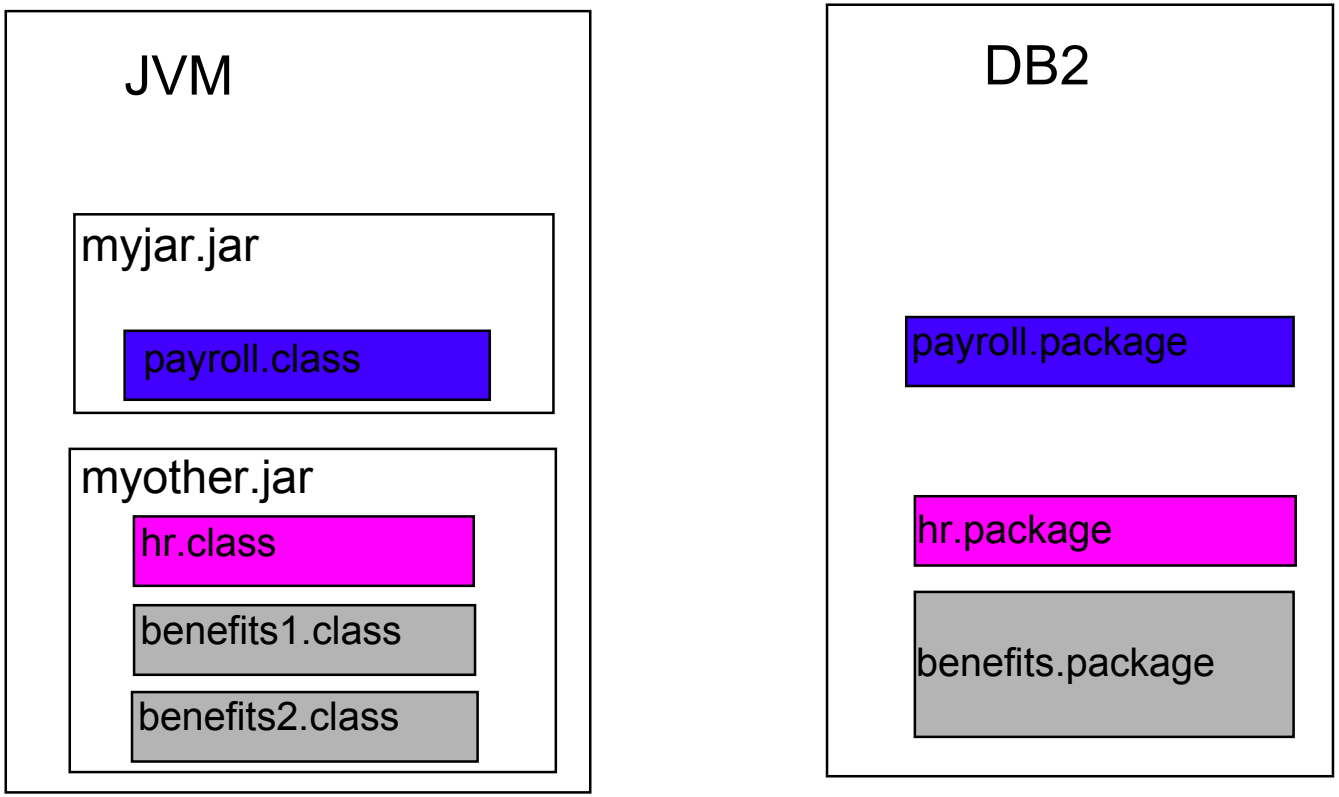
Enhanced monitoring for JDBC or SQLJ

```
-D71B DIS THREAD(*)

NAME      ST A  REQ ID              AUTHID    PLAN       ASID   TOKEN
SERVER   RA *   952 db2jccThread USRT001  DISTSERV 004A   432
 V437-WORKSTATION=9.30.129.202, USERID=SALLY,
     APPLICATION NAME=payment
 V445-G91E81C5.G49D.00F330DF7111=222 ACCESSING DATA FOR 9.30.129.197
SERVER   RA *   952 db2jccThread USRT001  DISTSERV 004A   432
 V437-WORKSTATION=9.30.129.214, USERID=JOE,
     APPLICATION NAME=accounts_payable
 V445-G91E81C5.G49D.00F330DF7222=332 ACCESSING DATA FOR 9.30.129.197
SERVER   RA *   952 db2jccThread USRT001  DISTSERV 004A   432
 V437-WORKSTATION=9.30.129.111, USERID=SAM,
     APPLICATION NAME=hr_appl1
 V445-G91E81C5.G49D.00F330DF753F=442 ACCESSING DATA FOR 9.30.129.197
```

SQLJ packages greatly simplify management/monitoring

-- programs involved in locking issues (deadlock/timeout)
-- SQL activity by program
-- program-level performance monitoring (CPU time, I/O operations, getpages, etc.)
-- content of SQL statements issued by each program
-- report SQL access paths for each SQL statement in each program
-- access paths don't change until next REBIND
-- package names visible to online monitoring tools (DB2PM, Omegamon, etc.)



DB2 Correlation IDs

- DB2 server side main log (db2diag.log)

  2003-04-29-12.27.43.791070 Instance:db2inst1 Node:000
  PID:2706(db2agent (ICMNLSDB)) TID:8192 Appid:G916625D.NA8C.068149162729
  access plan manager sqlra_sqlC_dump Probe:25 Database:ICMNLSDB

- DB2 Trace (server side)

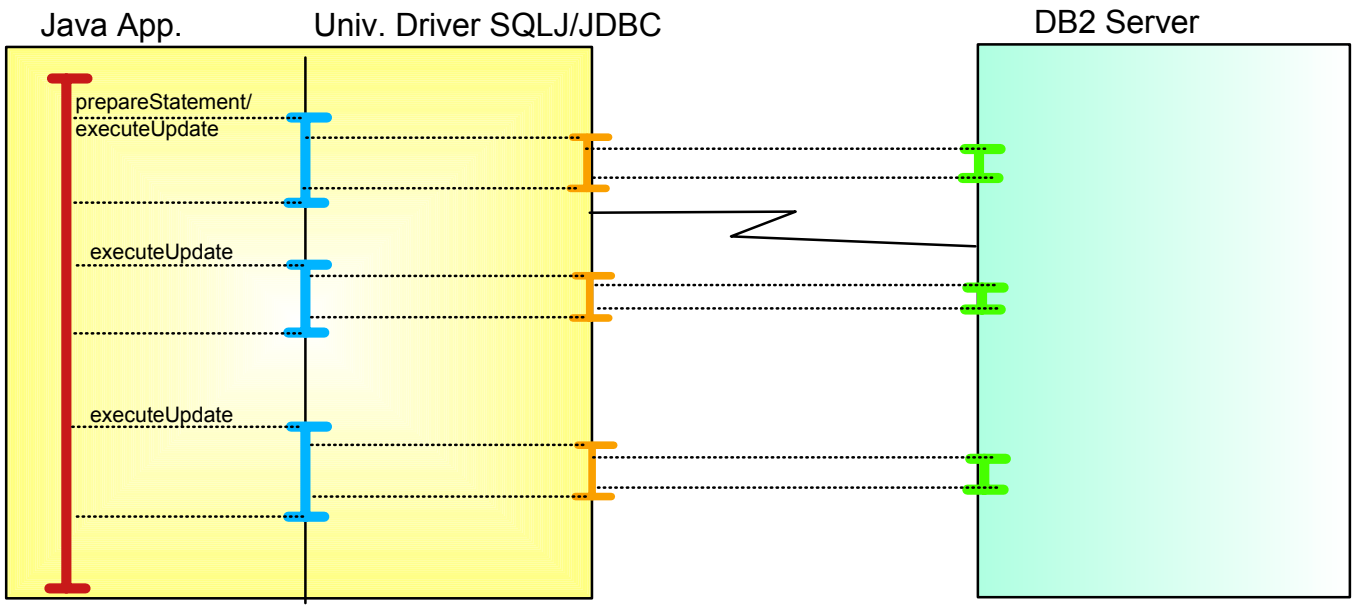
  3571 mbt_scb DB2 common communication sqlccgetapplid cei (3.3.43.10.2.1)
  pid 1188018 tid 1 cpid -1 node 0 sec 0 nsec 16431127 probe 10
  marker name: PD_SQLT_MARK_APPID
  Description: Correlator identifier (TCP/IP connection, JDBC type 4) bytes 26
  appID: G916625D.NA8C.068149162729

- Universal JDBC driver trace (client side)

ibm][db2][jcc][time:1050540951783][thread:main][Connection@8385e3] Database product version: SQL08012
[ibm][db2][jcc][time:1050540951783][thread:main][Connection@8385e3] Driver name: IBM DB2 JDBC Universal Driver
[ibm][db2][jcc][time:1050540951783][thread:main][Connection@8385e3] Driver version: 1.3.7 Test Build
[ibm][db2][jcc][time:1050540951783][thread:main][Connection@8385e3] DB2 Correlator: G916625D.NA8C.068149162729
[ibm][db2][jcc][time:1050540951783][thread:main][Connection@8385e3] END TRACE_CONNECTS

Java API for application monitoring

- DB2SystemMonitor monitor=
  ((DB2Connection)conn).getDB2SystemMonitor();
- monitor.enable(true);
- monitor.start(com.ibm.db2.jcc.DB2SystemMonitor.RESET_TIMES);
- monitor.stop();
- monitor.getServerTime()
- monitor.getNetworkIOTime()
- monitor.getCoreDriverTime()
- monitor.getApplicationTime()



Summary

- New DB2 Universal Java driver -- better performance and portability for Java applications
- SQLJ -- improved performance and managability for critical e-business applications
- WebSphere and DB2 -- integration has improved significantly, more coming in the future