Maximizing offload to zIIP processors with DB2 9 for z/OS native SQL stored procedures

Richard Corrihons  
IBM Customer Center - PSSC  
Montpellier, France

Introduction
This document is based on what has been observed and measured during the benchmark of an OLTP banking application that makes extensive use of the new native SQL language stored procedures support in DB2 9® for z/OS. We conducted several tests from transaction unit test to multiple LPAR (horizontal) scalability tests to find out how native SQL stored procedures behave in a stress test environment. We compared JDBC® type 2 versus type 4 connections, in order to give recommendations for a production environment. As native SQL stored procedures are zIIP eligible under certain conditions, we also measured the exact zIIP offload.

Native SQL stored procedures
DB2 9 for z/OS introduces a number of new features including native SQL language stored procedures. Prior to DB2 9 for z/OS, SQL stored procedures had to run under the control of the z/OS WorkLoad Manager (WLM) component in managed address spaces. Running SQL procedures in a WLM address space (DB2 UDB® for z/OS Version 8) incurs the overhead of communication between the WLM address space and the DBM1 address space for each SQL call. For systems running heavy stored procedure workloads, composed of many short running stored procedures, at or near 100% CPU utilization, this added overhead could potentially inhibit the amount of work being processed.

In DB2 9 for z/OS, SQL language procedures can run natively in the DBM1 address space. Running SQL procedures in the DBM1 address space, avoids the overhead of starting new address spaces for stored procedure invocations, and the cost of the round-trip communication between the WLM and the DBM1 address space for each SQL call. Internal Throughput Ratio (ITR) improvements of between 0 to 80% have been observed as compared to running an external SQL procedure.

In DB2 9 for z/OS, an SQL procedure workload is zIIP eligible if driven by DDF requests arriving via DRDA®, provided it runs in the DBM1 address space under a DDF enclave SRB.

Work that runs on the zIIP does not incur software charges based on the service units consumed; therefore it is a very attractive lower-cost alternative to running workloads on a general purpose processor.
For information on native SQL procedure control statements refer to the *DB2 for z/OS SQL Reference*, SC18-9854. For a general description of SQL procedure new functionality, see *DB2 9 for z/OS Technical Overview*, SG24-7330.

**Test environment**
We conducted our tests on two System z z9™ machines configured with 4 LPARs and 2 Internal Coupling Facilities. In each LPAR we had one Websphere® Application Server (WAS) and one DB2 member of a 4 way data sharing group. We configured general purpose CPs, zIIPs and zAAPs in each LPAR, the purpose being to measure any offload to specialized engines. We tested JDBC type 2 and type 4 connections between WAS and DB2. The test was conducted on a 40 million customer account database.

**Transaction flow**
The WAS application is triggered by a Message-Driven Bean (MDB); the WAS application parses an XML message, and then issues one or two JDBC calls, depending on the transaction. Inside DB2, the business application logic processing is executed by nested native SQL stored procedures.

**Transactions tested**
The benchmark focused on 4 business transactions which are the most executed in a retail banking environment. They are balance inquiry (BI), cheque deposit (CQ), cash deposit (CD) and cash withdrawal (CW). In this section, we detail the SQL pattern for each transaction, the number of JDBC calls, the number of stored procedure calls and the average number of SQL per stored procedure call.

The SQL pattern of each transaction was:

![SQL pattern](image)

**Figure 1  SQL pattern per transaction**

There was one JDBC call for each BI or CQ transaction, and two JDBC calls for each CD or CW transaction. Because the tested application uses stored procedures, there is no need to have one JDBC call per SQL statement; this is one of the advantages of stored procedures.
With so few JDBC calls per transaction, we decided to compare JDBC type 2 versus type 4. The results of this test are shown in the “JDBC type 4 versus type 2 transaction unit test” section.

The number of stored procedure calls for each transaction was 3 for BI, 10 for CQ, 14 for CD and 17 for CW. This application uses nested stored procedures, so we have many more stored procedure calls than JDBC calls.

The average number of SQL statements per stored procedure call for each transaction was 1.66 for BI, 1.5 for CQ, 1.43 for CD and 1.41 for CW. These low values, suggest short running stored procedures; this is a situation where the duration of the stored procedure call itself might represent a non negligible overhead. Because native SQL stored procedures run in the same address space as the SQL engine this reduces the call duration, hence less overhead compared to external stored procedures. So this is an application where native stored procedures should be advantageous.

**JDBC type 4 versus type 2 transaction unit test**

The usual recommendation for performance is to use JDBC type 2 for local connections. Our transactions only have one or two JDBC calls, so use of JDBC type 4 should not dramatically impact the response time and CPU usage. In order to verify this assumption, we conducted tests transaction by transaction, using both connection types. Note: the application logic and SQL calls are almost the same for CD and CW transactions, so we only conducted transaction unit test for BI, CQ and CD.

We measured transaction response time at WebSphere level and CPU usage for each transaction. The injection rate was around 350 transactions per second.

![Graph](image)

**Figure 2 Transaction unit test JDBC type 2 versus JDBC type 4, response time and CPU**

The figure for CPU usage is relative. We took as our base value the CPU consumption of the BI Type 2 transaction (hence the value of 1), and the values for the other run types are relative to this value. So for example, the Cash Deposit transaction using Type 4 used almost four times more CPU than the Balance Inquiry using Type 2.
If we compare the results of Type 2 and Type 4 for the different operations:

- BI CPU usage was 5% higher for BI type 4
- CQ CPU usage was 5% lower for type 4
- CD CPU usage was 2% higher for type 4.

As expected, the transaction response time (WAS plus DB2) was quite similar with one millisecond more when using JDBC type 4 for CQ and CD when compared to type 2. Transaction response time was very low for BI, so it is not easy to draw conclusions.

Overall, type 2 and type 4 give similar results for response time and CPU usage, with a slight overhead for some transactions, so at this point there is no obvious reason to pick one particular connection type.

We can look at the processor usage breakdown, to see if it helps us to make a choice.

![JDBC Type 2 versus JDBC Type 4](image)

*Figure 3 Transaction unit test JDBC type 2 versus JDBC type 4, processor usage breakdown*

Figure 3 shows processor usage breakdown per processor type normalized to 100%. There is no zIIP usage for type 2 connections, while for type 4 connections we can see zIIP engagement. This is as expected, because type 4 connections are one way to engage zIIP, and native SQL stored procedures remain on the zIIP processor.

The use of JDBC type 4 connections is recommended for this application, because it provides key benefits (allows for zIIP exploitation and continuous availability), with very limited overhead. Normally, for a WAS/DB2 configuration in a parallel sysplex, IBM would recommend type 2 connection, because this eliminates the DRDA flow and TCP/IP overhead of type 4. However, with the use of nested stored procedures, the DRDA flows are minimal.

Continuous availability is demonstrated in the scenario when a DB2 member goes down due to a planned or unplanned outage. The surviving WAS on the LPAR can connect through DRDA type 4 to another DB2 member on a different LPAR. Therefore the capacity available on the LPAR, with the down DB2 member, can still run workload.

As JDBC type 4 was our recommendation, we conducted all the remaining tests using type 4 connectivity.
**JDBC type 4 transaction unit test**

For this test we focused on the zIIP offload. We conducted tests transaction by transaction. Note: the application logic and SQL calls are almost the same for CD and CW transactions, so we only conducted transaction unit test for BI, CQ and CD.

![Type 4 transaction unit test: DB2 zIIP usage %](image1.png)

**Figure 4  JDBC type 4 transaction unit test, DB2 zIIP usage**

Figure 4 shows processor usage breakdown per processor type normalized to 100%. We see zIIP engagement for all 3 transactions with a minimum of 42.71% for CD transaction.

**Scalability tests for transaction mix**

After transaction unit testing we switched to a mixed workload with the following transaction distribution: 50% BI, 10% CQ, 20% CD and 20% CW. The percentages were chosen in order to match what a bank would face in real life. With this distribution, 50% of the workload modified data, only the BI transaction was read only.

We conducted scalability test with an injection rate of 800 “mixed workload” transactions per second per LPAR. We used up to 4 LPARs, that is we injected a maximum of 3200 tps for the entire sysplex configuration.

![Transaction mix scalability test](image2.png)

**Figure 5  Transaction mix scalability test, transaction per second**
For this test we focused on DB2. We wanted to get response time and CPU usage at DB2 level. We also measured exact zIIP offload.

![Graph showing Transaction mix scalability test, DB2 response time and CPU]

In figure 6, DB2 CPU includes CPU for DDF enclave, MSTR, DBM1, DIST and IRLM address spaces. DB2 response time is DDF enclave duration. The tests with multiple LPARs show an increase in both CPU and response time compared to the test with one LPAR; this is due to the data sharing overhead. In data sharing, whatever the number of LPARs, the DB2 response time and CPU usage are stable, which demonstrates good horizontal scalability for this application.

Let’s now have a look at zIIP offload:

![Graph showing Transaction mix scalability test, DB2 zIIP usage]

Figure 7 shows processor usage breakdown per processor type normalized to 100%. We see zIIP offload of 47% for the data sharing runs. This definitely proves that using type 4 JDBC connections helps to reduce total cost of ownership (TCO).
**Conclusion**

We benchmarked an application that makes extensive use of native SQL stored procedure. Results from this benchmark show that this application performs and scales very well. We achieved 3193 business transactions per second with a DB2 response time of 13 ms.

We conducted tests to compare JDBC type 2 and type 4 connections. The use of type 4 connection is recommended for this application, because it provides key benefits with very limited overhead. Normally, for a WAS/DB2 configuration in Sysplex, IBM would recommend type 2 connection, because this eliminates the DRDA flow and TCP/IP overhead of type 4. However, with the use of nested stored procedures, the DRDA flows are minimal. Therefore using the type 4 connection is not a significant overhead and provides the following advantages:

- Allow for zIIP exploitation. Reviewing the test results show greater than 40% usage of zIIP processing
- Continuous availability

**Disclaimer and Trademarks**

Measurement data included in this document are obtained by the members of the System z benchmark center department at the IBM PSSC in Montpellier, France. The materials in this document are subject to enhancements at some future date, a new release of DB2, or a Programming Temporary Fix. The information contained in this document has not been submitted to any formal IBM review and is distributed on an "As Is" basis without any warranty either expressed or implied. The use of this information is a customer responsibility.

The following terms are trademarks or registered trademarks of the International Business Machines Corporation in the United States, other countries or both: DATABASE 2, DB2, DRDA, OS/390, MVS/ESA, SYSTEM/390, IBM, WebSphere, z/OS, zSeries, System z. The following terms are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both: JDBC. Other company, product or service names may be trademarks or service marks of others.