DB2 9 XML performance characteristics

Use the TPoX benchmark to test the performance of a simulated brokerage scenario

Skill Level: Intermediate

Irina Kogan (ikogan@ca.ibm.com)
DB2/XML Performance
IBM Toronto Lab

Matthias Nicola (mnicola@us.ibm.com)
DB2/XML Performance
IBM Silicon Valley Laboratory

Berni Schiefer (schiefer@ca.ibm.com)
DB2 Distinguished Engineer
IBM Toronto Lab

22 Jun 2006

Updated 30 Jan 2007

Learn about the performance and scalability characteristics of a simulated securities brokerage transaction processing environment using DB2® 9 XML, IBM POWER5+, AIX 5.3, and TotalStorage DS8100. This scenario includes use of the FIXML schema, a financial industry standard. The Transaction Processing over XML (TPoX) database benchmark is used to obtain the results discussed in this article.

Introduction

Now that DB2 9 is released, it is time for a test drive with one of it's newest capabilities: pureXML®. A simulated brokerage environment is the selected stage. The processing profile there is characterized by:

- High transaction volume and concurrency
- Small transaction size
• Large number of small XML documents

• Variable XML document structure -- The tests include data compliant with FIXML, the financial industry's XML implementation of its Financial Information eXchange (FIX) standard.

Remember that XML applications generally fall into one of these groups:

• Data-oriented (higher volume, smaller size, used for this study)
• Document-oriented (variable volume, larger size)

Additionally, database applications involving XML are very diverse and can include the following:

• Publishing relational data as XML
• Content and document management with XML full-text search
• Consolidation of diverse data sources
• Forms processing
• Backend support for Web services and service-oriented architecture (SOA)
• Message-based transaction processing and XML-based online transaction processing (OLTP), especially in the financial industry

This article presents performance measurements in an XML-based transaction processing scenario that simulates a data-oriented financial application. The TPoX benchmark is used to mimic this scenario and to take the measurements. The test equipment includes the latest POWER5 server (p5 560Q) with AIX 5.3 and the TotalStorage DS8100 disk system.

DB2 9 and XML

The new XML support in DB2 9 includes pure XML storage, XML indexes, XQuery, SQL/XML, and sophisticated XML schema handling. "Pure" means that XML documents are stored and processed as type-annotated trees, unlike any previous technology in commercial relational databases. In particular, pureXML is significantly different from storing XML as large objects (BLOBS or CLOBs) or dividing XML into a set of relational tables. For more information, refer to the previous articles "What's new in DB2 Viper" (developerWorks, February 2006) and "Native XML Support in DB2 Universal Database."

Test scenario: Online brokerage

This test scenario models an online brokerage. We have experience in supporting financial companies in their adoption of XML. That experience helped us understand their data and processing characteristics. This scenario is purposefully simplified, yet
still representative in terms of documents, transactions, and XML schemas.

The main logical data entities in this scenario are as follows (see Figure 1):

- **Customer**: A single customer can have one or more **accounts**.
- **Account**: Each account contains one or more **holdings**.
- **Holding**: The number of shares of a **security**.
- **Security**: Identifier for a holding (for example, the name of a stock).
- **Order**: Each order buys or sells exactly one **security** for exactly one **account**.

**Figure 1. Data entities and their XML schema**

Document handling and size varies by type:

- For each customer there is a CustAcc document that contains all customer information, account information, and holding information for that customer. CustAcc document size ranges between 4KB and 20KB.
- Orders are represented using FIXML 4.4. FIXML is an industry standard XML schema for trade-related messages such as buy or sell orders (www.fixprotocol.org). Order document size is 1KB to 2KB. Order documents have many attributes and a high ratio of nodes to data.
- Security documents (fixed at 20833) represent the majority of US-traded stocks and mutual funds and use actual security symbols and names. Their size ranges between 3KB and 10KB.

The Toxgene data generator is used to produce instance documents for all three schemas. For more information on the Toxgene data generator, refer to ToXgene - the ToX XML Data Generator.

**TPoX benchmark**
The TPoX benchmark has recently become public as open-source at http://www.tpox.net. This benchmark simulates the financial scenario discussed in the previous section. It focuses on XQuery, SQL/XML, XML insert, update, and delete, XML indexes, and concurrency aspects. The workload driver written in Java runs the workloads and collects performance measurements. You can download the full benchmark package from the benchmark’s Web site. It contains a script called runall.ksh that allows you to generate the XML data and to repeat all the experiments discussed in this article. The script setenv.ksh makes it possible for you to configure TPoX execution for a specific machine.

Test equipment and configuration

Tests were run on the following equipment:

- **Processor:** IBM System p5 560Q using an eight-processor logical partition (LPAR) of a mid-range IBM System p5 560Q. The eight cores ran at 1.5GHz.
- **Memory:** 32GB
- **Operating System:** AIX 5L v5.3 TL04 (system type: 9116-561, two quad chip modules)
  - Simultaneous multi-threading presented 16 concurrent execution threads or logical processors.
  - A multi-pathing subsystem device driver (SDD) was installed. This feature improves storage server access through features such as enhanced data availability and dynamic I/O load balancing across the fiber channel adapters on the storage server.
- **Storage:** IBM TotalStorage DS8100 attached to the LPAR through four fiber channel adapters.

AIX configuration

During the installation of DB2, any required operating systems parameter adjustments are automatically made. The following virtual memory management parameters have been set to better control the amount of memory used for any file system caching:

```
  vmo -o minperm%=5
  vmo -o maxclient%=15
  vmo -o maxperm%=15
```

Before setting these parameters, it is recommended that you obtain a good understanding of the AIX virtual memory manager. These settings need to be considered in the context of other parameters, especially lru_file_repage. The use of these particular settings in this XML OLTP environment should not be considered a
general recommendation for all DB2 environments.

To specifically avoid attempts to cache the input files during data loading, the file system containing the raw XML input files was mounted with the JFS2 file system's concurrent I/O feature, using the \texttt{-o cio} option to the \texttt{mount} command.

Storage configuration

Standard configuration defaults were used for the TotalStorage DS8100. The DS8100 is essentially a POWER5 eServer p5 570 internally. Unlike its predecessor ESS with SSA loops, DS8100 disk interconnect is a Switched Fiber Channel Arbitrated Loop (FC-AL) for faster data access and high availability. The DS8100 was configured with a total of 128 disks on which 16 volumes were created. From this pool, eight volumes (64 disks) were assigned to this LPAR. Four volumes were sized at 388GB each using 6+Parity+Spare. The remaining four volumes were sized at 452GB each using 7+Parity. A single volume group (VG) was created that spanned all eight volumes. All storage components of the DB2 database were defined on this volume group, including tablespaces, log, and backups. Table 1 summarizes configuration aspects.

Table 1. Storage configuration

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Provisioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processors</td>
<td>Two processor complexes with pSeries POWER5 1.9 GHz two-way CEC each</td>
</tr>
<tr>
<td>Memory (cache)</td>
<td>32GB</td>
</tr>
<tr>
<td>Disk interconnect</td>
<td>Switched FC-AL</td>
</tr>
<tr>
<td>Number of disks</td>
<td>128 (only 64 used by the host LPAR)</td>
</tr>
<tr>
<td>Disk size/speed</td>
<td>73 GB, 15000 RPM</td>
</tr>
</tbody>
</table>

DB2 configuration

DB2 9 contains a number of new features, including new autonomic self-tuning capabilities. In this test, several of these autonomic capabilities are exploited, including:

- Automatic storage management
- Self-tuning memory management

Since DB2's self-tuning memory manager (STMM) was active, it was continuously refining the settings of a range of DB2 configuration parameters. Examples of some of the key DB2 configuration parameters managed and modified by STMM during the test runs are shown in Table 2. The important thing to realize is that the STMM autonomously changed these values depending on the type of workload you are running, such as insert-only, query-only, or a mixed workload.

Table 2. Database configuration, self-tuning

<table>
<thead>
<tr>
<th>DB Config Name</th>
<th>Initial Setting</th>
</tr>
</thead>
</table>
SELF_TUNING_MEM: ON (default)
DATABASE_MEMORY: AUTOMATIC (default)
SORTHEAP: 156
SHEAPTHRES_SHR: 10000
LOCKLIST: 53000
MAXLOCKS: 80
PCKCACHESZ: 27000

<table>
<thead>
<tr>
<th>Buffer Pool Name</th>
<th>Initial Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBMDEFAULTBP</td>
<td>1100000</td>
</tr>
<tr>
<td>CATBP</td>
<td>4000</td>
</tr>
<tr>
<td>TEMPBP</td>
<td>1000</td>
</tr>
</tbody>
</table>

This leaves only a handful of database configuration tasks for the DBA, which are summarized in Table 3.

### Table 3. Database configuration, manual

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Provisioning/Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Unicode. Automatic storage for all tablespaces. DB2 log on separate stripe</td>
</tr>
<tr>
<td>Memory</td>
<td>STMM enabled for all tests</td>
</tr>
<tr>
<td>Page size</td>
<td>16K (for tablespaces and bufferpools)</td>
</tr>
<tr>
<td>Tables and Indexes</td>
<td>Three tables: CustAcc, order, security. 24 XML Indexes: 10 on CustAcc, five on order, nine on security</td>
</tr>
<tr>
<td>Tablespaces</td>
<td>Six in total: one for each of the three tables, plus one for the indexes of each of the three tables. File system caching was disabled for all tablespaces.</td>
</tr>
<tr>
<td>Bufferpools</td>
<td>Three in total: the default, plus for catalog tablespace, and one for temp tablespace</td>
</tr>
</tbody>
</table>

### Workloads

Three XML workloads were designed, executed, and measured:

- Insert (write-only)
- Query (read-only)
- Mixed (read-write)

All of them are characterized by a large degree of concurrency. The workload is executed by a Java driver that spawns one to n concurrent threads. Each thread simulates a user that connects to the database and submits a stream of transactions.
without think times. Each stream is a weighted random sequence of transactions that are picked from a list of transaction templates. Each transaction is assigned a weight that determines the transaction's percentage in the workload mix. At run time, parameter markers in the transactions are replaced by concrete values drawn from configurable random value distributions and input lists.

**Insert workload: Write-Only**

The Insert workload populates the database with approximately 100GB of raw XML data:

- 6 million CustAcc documents
- 30 million orders
- 20,833 securities

First, all securities are inserted by 83 concurrent users. Then the CustAcc and Order documents are inserted in stages to verify that the insert performance is scalable. One hundred concurrent users are used at each of the stages shown in Table 4.

**Table 4: Database population in stages**

<table>
<thead>
<tr>
<th>Stage</th>
<th>Number of CustAcc documents in the database</th>
<th>Number of Order documents in database</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100,000</td>
<td>500,000</td>
</tr>
<tr>
<td>2.1</td>
<td>200,000</td>
<td>1,000,000</td>
</tr>
<tr>
<td>2.2</td>
<td>300,000</td>
<td>1,500,000</td>
</tr>
<tr>
<td>2.3</td>
<td>400,000</td>
<td>2,000,000</td>
</tr>
<tr>
<td>2.4</td>
<td>500,000</td>
<td>2,500,000</td>
</tr>
<tr>
<td>2.5</td>
<td>600,000</td>
<td>3,000,000</td>
</tr>
<tr>
<td>3.1</td>
<td>1,000,000</td>
<td>5,000,000</td>
</tr>
<tr>
<td>3.2</td>
<td>1,500,000</td>
<td>7,500,000</td>
</tr>
<tr>
<td>3.3</td>
<td>2,000,000</td>
<td>10,000,000</td>
</tr>
<tr>
<td>4.1</td>
<td>2,500,000</td>
<td>12,500,000</td>
</tr>
<tr>
<td>4.2</td>
<td>3,000,000</td>
<td>15,000,000</td>
</tr>
<tr>
<td>4.3</td>
<td>3,500,000</td>
<td>17,500,000</td>
</tr>
<tr>
<td>4.4</td>
<td>4,000,000</td>
<td>20,000,000</td>
</tr>
<tr>
<td>5.1</td>
<td>4,500,000</td>
<td>22,500,000</td>
</tr>
<tr>
<td>5.2</td>
<td>5,000,000</td>
<td>25,000,000</td>
</tr>
<tr>
<td>5.3</td>
<td>5,500,000</td>
<td>27,500,000</td>
</tr>
<tr>
<td>5.4</td>
<td>6,000,000</td>
<td>30,000,000</td>
</tr>
</tbody>
</table>

**Query workload: Read-Only**

After the insert workload has populated the database, a read-only workload was
executed against the database. This workload, consisting of seven XML queries, was executed with different degrees of concurrency using 25, 50, 75, 100, 125, and 150 concurrent users. The test duration was one hour for each of these six test runs.

The seven queries all have the following characteristics in common:

- They are written in standard-compliant SQL/XML notation, such as SQL with embedded XQuery, taking advantage of parameter markers. For more information, refer to Advancements in SQL/XML.
- They use the SQL/XML predicate XMLEXISTS to select XML documents based on one or multiple conditions which are expressed in XQuery notation.
- They use the SQL/XML function XMLQUERY to retrieve full or partial XML documents, or to construct new result documents that are different from the ones stored in the database.
- They use XML namespaces corresponding to the namespace in the XML data.
- They take advantage of one or multiple XML indexes to entirely avoid table scans.
- All seven queries are equally weighted in the workload.

Table 5 shows the seven queries in terms of their distinguishing characteristics and the tables they touch.

### Table 5: Summary of TPoX XML queries

<table>
<thead>
<tr>
<th>Q</th>
<th>Query Name</th>
<th>CustAcc</th>
<th>Security</th>
<th>Order</th>
<th>Characteristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>get_order</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>Return full order document without the FIXML root element.</td>
</tr>
<tr>
<td>2</td>
<td>get_security</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>Return full security document.</td>
</tr>
<tr>
<td>3</td>
<td>customer_profile</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Extract seven customer elements to construct a profile document.</td>
</tr>
<tr>
<td>4</td>
<td>search_securities</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Extract elements from some securities, based on four predicates.</td>
</tr>
<tr>
<td>5</td>
<td>account_summary</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Complex</td>
</tr>
</tbody>
</table>
Mixed workload: Read/Write

Similar to the read-only workload, the mixed workload was executed against the base population of 6 million CustAcc documents and 30 million orders, and with different degrees of concurrency using 25, 50, 75, 100, 125, and 150 concurrent users. The test duration was one hour for each of these test runs.

The mixed workload consists of:

- 70 percent read operations: queries
- 30 percent write operations: six percent updates, 12 percent document deletes, and 12 percent inserts.

While the queries were exactly the same as in the read-only workload above, the following observations were used in defining the update/delete/insert transactions:

- Customer accounts get updated to reflect trades (execution of orders), but not necessarily immediately after every order (three percent CustAcc updates)
- Order documents do not get updated in our scenario (hence no update order transaction)
- Security prices are updated regularly during a business day (three percent security updates)
- The turnover of customers is low (two percent CustAcc inserts, two percent CustAcc deletes)
- New orders arrive continuously, old orders get pruned from the system eventually and at the same rate (10 percent order insert, 10 percent order delete)
- The number of securities is fixed (no delete or insert transactions)

By combining and applying these objectives, the transaction mix shown in Table 6 was produced.

Table 6: Mixed workload transactions

<table>
<thead>
<tr>
<th></th>
<th>get_security_price</th>
<th>X</th>
<th>-</th>
<th>Extract the price of a security.</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>customer_max_order</td>
<td>-</td>
<td>X</td>
<td>Join CustAcc and orders to find the max of a customer's orders.</td>
</tr>
</tbody>
</table>
Update transactions first read a specific document based on an XQuery predicate, and then use it to update the original copy of that document in the database. In reality, the document would be modified between the read and the update step, but this is of low relevance for the purposes of this article and therefore avoided for simplicity.

Insertions are performed without XML schema validation.

Documents in the database are randomly selected for update and delete operations. Each newly inserted order and CustAcc document becomes immediately eligible for update or delete by a subsequent transaction.

## Results

The database setup and all of the workloads are executed in a single uninterrupted sequence. This results in a 23-hour non-stop system test as shown in Table 7.

### Table 7: Timing summary of a full test run with all workloads

<table>
<thead>
<tr>
<th>Task</th>
<th>Time Taken (mins)</th>
<th>Explanation/Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database creation and update</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>of db configuration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert workload</td>
<td>160</td>
<td>Combined from all stages</td>
</tr>
<tr>
<td>Run stats on table and indexes</td>
<td>340</td>
<td>The time is distributed as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22s - security</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2h45m - CustAcc</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2h54m - order</td>
</tr>
<tr>
<td>Database backup</td>
<td>23</td>
<td>-</td>
</tr>
</tbody>
</table>
Query and mixed workloads

<table>
<thead>
<tr>
<th>Workload</th>
<th>825</th>
<th>Each of the two workloads is run with 25, 50, 75, 100, 125, and 150 users.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database restore</td>
<td>17.5</td>
<td>-</td>
</tr>
<tr>
<td>Other</td>
<td>~15</td>
<td>Other miscellaneous tasks</td>
</tr>
<tr>
<td>Total</td>
<td>~1380</td>
<td>23 hrs total run time</td>
</tr>
</tbody>
</table>

**Insert workload results**

The total elapsed time to insert 36,020,833 documents was approximately 160 minutes, which produces an average throughput of 3770 inserts/sec. The throughput varied with size:

- Order documents (1K to 2K) were inserted at an average of 5320 inserts/sec
- Account documents (3K to 10K) were inserted at an average 1550 inserts/sec.

Both insert rates amount to roughly 30GB per hour. Figure 2 shows a nearly constant order insert rate as the number of orders grows to 30 million documents.

**Figure 2: Insert rate for Order documents**

![Insert rate for Order documents](image)

**Query workload results**

The performance of the query workload increased as the number of users increased.
and the CPUs were better utilized. Throughput flattens out as CPU utilization approaches 100 percent, as expected. The best throughput was achieved with 150 users and reached 5480 queries/sec with a CPU utilization of 96 percent, as shown in Figure 3.

Increasing the number of users to 175 did not produce a significantly higher throughput since the machine was already at capacity.

**Figure 3: Read-only query throughput, CPU utilization, and I/O wait time**

Mixed workload results

The best performance with a mixed workload was also achieved with 150 concurrent users, as shown in Figure 4. The throughput was 1980 transactions/sec. As expected, throughput of the mixed workload is lower than for the query-only and insert-only workloads.

**Figure 4: Mixed-workload throughput, CPU utilization, and I/O wait time**
Summary

The goal of this performance study was to demonstrate operating performance characteristics using the latest IBM server hardware, storage, AIX operating system, and DB2 9 software for an XML workload. The TPoX benchmark was used to achieve this goal. All tests used DB2's new STMM and automatic storage features.

We feel that the workload scenario is representative of a significant class of XML applications including message-based transaction processing as well as Web service application dealing with large numbers of small XML documents. We chose the financial industry due to our experience with it and its adoption of a mature, standardized XML schema, FIXML.

In summary:

- The overall test time was 23 hours, including database creation.
- The test data consisted of 6 million CustAcc documents, 30 million Order documents, and 20,833 Security documents.
- Tests were conducted with 25, 50, 75, 100, 125, and 150 concurrent users.
- Insert throughput (transactions per second or tps) varied with document size but was linear within a size. Throughput volume was steady at 30GB per hour regardless of document size:
• 1550 tps (for CustAcc documents, 4K to 20K)
• 5320 tps (for order documents, 1K to 2K)
• Query throughput scaled with concurrent users:
  • 2000 tps at 25 users
  • 5500 tps at 150 users (CPU maxed, wait-time-on-I/O approaching zero)
• Mixed transaction throughput also scaled with concurrent users and flattened at about 2000 tps:
  • 1000 tps at 25 users
  • 2000 tps at 150 users (~42 percent CPU, ~50 percent I/O wait).

Acknowledgments

The authors thank Sunil Kamath and Punit Shah for their contributions both to the overall success of this project and to this article.
Resources

Learn

- **Transaction Processing over XML (TPoX) benchmark**: Learn about TPoX, the benchmark based on the online brokerage scenario and making use of various new XML features in databases.

- **DB2 9 - pureXML and storage compression**: Learn about pureXML's seamless integration of XML with relational data speeds application development, and much more.


- **Toxgene Data Generator**: Learn about ToXgene, a template-based generator for large, consistent collections of synthetic XML documents.


- **IBM System p5 560Q**: The IBM System p5 560Q mid-range server implements outstanding price/performance, mainframe-inspired reliability and availability features, and scalability to 16-cores.

- **IBM Totalstorage DS8100**: The IBM TotalStorage DS8000 series offers high-capacity storage systems that are designed to deliver a generation-skipping leap in performance, scalability, resiliency and value.

- **"Redbook: The IBM TotalStorage DS8000 Series: Concepts and Architecture (SG24-6452-00)"**: Learn about the IBM TotalStorage DS8000 series of storage servers, its architecture, its logical design, hardware design and components, advanced functions, performance features, and specific characteristics.

- Visit the developerWorks Information Management zone to expand your skills on a wide variety of IBM Information Management products.

- Stay current with developerWorks technical events and webcasts.

Get products and technologies

- Build your next development project with IBM trial software, available for download directly from developerWorks.

Discuss

- Participate in the discussion forum for this content.

- Participate in developerWorks blogs and get involved in the developerWorks community.
About the authors

Irina Kogan
Irina Kogan is a software engineer at IBM Toronto Lab working on XML performance in DB2 with both developers and customers. Her areas of interest include XML storage and structural updates, XQuery, OLTP and benchmark development. Irina has B.Sc. and M.Sc. in computer science from York University. Her Masters degree focused on semantic integrity and good design principles in databases. Before joining IBM in 2004, she worked on XML and Java development at Siemens in Germany.

Matthias Nicola
Dr. Nicola is the technical lead for XML database performance at IBM's Silicon Valley Lab. His work focuses on all aspects of XML performance in DB2, including XQuery, SQL/XML, and all native XML features in DB2. Dr. Nicola works closely with the DB2 XML development teams as well as with customers and business partners who are using XML, assisting them in the design, implementation, and optimization of XML solutions. Prior to joining IBM, Dr. Nicola worked on data warehousing performance for Informix Software. He also worked for four years in research and industry projects on distributed and replicated databases. He received his doctorate in computer science in 1999 from the Technical University of Aachen, Germany.

Berni Schiefer
Berni Schiefer is a DB2 distinguished engineer. He has responsibility for DB2 performance benchmarking and solutions development, including the BCU. He joined the IBM Toronto Lab in 1985 and has worked on SQL/DS and the Starburst experimental relational database at the IBM Almaden Research Lab, prior to working on DB2. His current focus is on introducing advanced technology into DB2 with particular emphasis on processors, performance, XML, Linux, virtualization, and autonomies.