Benchmarking XML Database Systems – First Experiences

Timo Böhme Erhard Rahm
University of Leipzig, Germany
http://dbs.uni-leipzig.de

1 Introduction

We recently developed and published a scaleable multi-user benchmark called XMach-1 (*X*ML Data *Ma*nagement ben*ch*mark) for evaluating the performance of XML data management systems [1]. To our knowledge it is the first such benchmark. It aims at realistically evaluating the performance of individual systems as well as to allow for a performance comparison between different systems and architectures ranging from native XML data management systems to XML-enabled relational DBMS. Specifying and implementing the benchmark revealed a number of problems which are partly due to the lack of a standardized XML query language, the complexity of the XML format and the relative immaturity of current XML database software. After a brief review of XMach-1 we will discuss our experiences made so far.

2 XMach-1: Short Overview

The XMach-1 benchmark is based on a web application in order to model a typical use case of a XML data management system. The system architecture consists of four parts: the XML database, application servers, loaders and browser clients (Figure 1). Similar to TPC-W [2], the System under Test (SUT) for which response time and throughput performance is determined includes the database and application servers. Including both server types is necessary since XML database processing is typically spread across the database backend and application servers. The application servers are essential for improved throughput, scalability, load balancing and caching. The number of database and application servers is not predetermined but can be chosen according to the throughput goals.

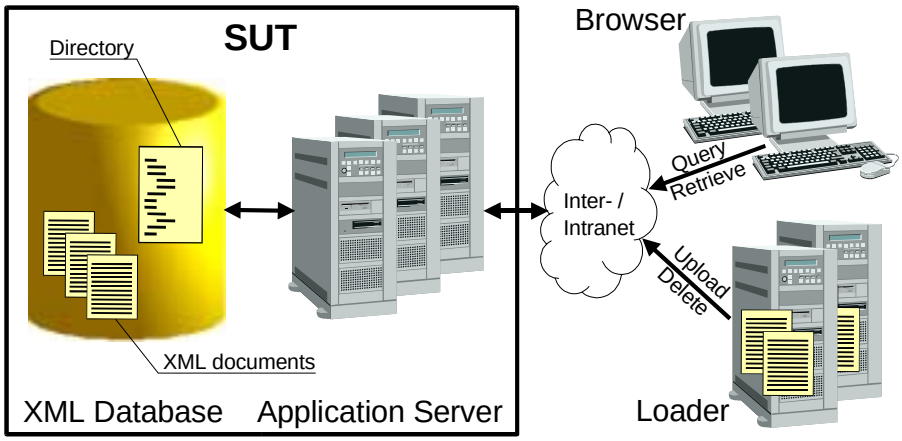

Figure 1: Components of benchmark architecture

The database contains a single directory structure and a certain number of text-oriented XML documents. The directory contains metadata about all other XML documents. It represents structured data as it is schema-based and holds element content only. The XML documents are generated synthetically to support almost arbitrary numbers of documents with well-defined contents and structure. Document size and structure are variable and skewed; the average size is about 18 KB. To support scaling to different system configurations, four database sizes are possible with an initial document number of 10.000, 100.000, 1.000.000 or 10.000.000. Due to insert operations the number of documents increases during benchmark execution.
Two benchmark variants are distinguished depending on whether the XML documents are schema-less or conform to schemas or DTDs. This allows us to run the benchmark with systems only supporting one of the two cases. If both variants are possible, we can evaluate the performance impact of having schema support.
The workload is defined by a mix of operations from 8 query types and 3 update types. These operations cover typical database functionality (join, aggregation, sort) as well as information retrieval and XML-specific features (document assembly, navigation, element access). All operations can be expressed by XML language proposals such as Quilt or XQuery [3]. The query and update workload is generated by emulated browsers and loaders (Fig. 1). The number of these clients is not predetermined but can be chosen according to the throughput goals. Interaction with the application server is via a HTTP interface.
The benchmark specifies the workload composition and response time limits per operation type. The main performance metric is query throughput measured in Xqps (XML queries per second) or – in the schema-less case – in $Xqps_{sl}$.

3 Experiences

We first discuss observations w.r.t. the specification of XMach-1. We then present first experiences from its implementation and adoption. Due to legal considerations we do not outline specific performance results.

3.1 Benchmark specification

We had two (somewhat conflicting) goals in mind when we defined the benchmark. Given the broad applicability of XML data management systems we did not want to limit ourselves to a specific usage form of XML data but strove for a domain-specific but rather comprehensive benchmark measuring the ability of a system to handle different types of XML data and a variety of operations. Second the benchmark architecture and specified operations should allow the implementation and execution on most XML data management systems ranging from native XML databases to relational DBMS augmented with XML functionality. We also tried to keep the benchmark as simple as possible to limit the effort for its implementation. We now discuss the implications of the two main goals.

3.1.1 Comprehensive Benchmark

Specifying a comprehensive benchmark means that we have to capture the main usage forms of XML with respect to both data structures and operations. In particular, text and non-text data has to be included, differing in content type, preservation of element order and element-content ratio. We therefore modelled a document repository containing XML text documents with different DTDs and a directory document holding metadata of the text documents. Moreover, we came up with a spectrum of query and update operations.
This design of the benchmark found positive reactions but also some criticisms which we will cite and comment:

- Some people would have preferred a more specific benchmark tailored to e-business (B2B) applications. While this is certainly an important application type for XML data we felt that the associated data management requirements would not be demanding enough to warrant a new benchmark in addition to (an XML-enabled version of) TPC-W. This is because B2B data is mostly uniformly structured and dominated by small data values. Our aim was to also evaluate the systems' ability to handle the more advanced features of XML compared to relational data structures including flexible support of schemas and variable structures.
- Another criticism was that the single directory document could become a bottleneck when running multiple concurrent queries. In our view this would only be a problem if the management systems treats the directory document in an unoptimized manner with locking and data allocation at the granularity of entire documents. XML allows the whole database to be contained in a single document making it necessary to perform critical database management tasks at finer granularities than documents, e.g. at the element level.

3.1.2 Executable on most systems

Currently available XML data management systems mostly support only a basic subset of XML features. Moreover, due to the lack of a standardized interface and query language there are large differences w.r.t. the supported query functionality. These limitations had to be considered in order to allow the adoption of XMach-1 to existing systems:

- We restricted the XML data to hierarchical documents with elements and attributes. Neither support for namespaces nor entities is assumed.
- According to the distinction in the XML specification of well-formed and valid documents we expected the systems to be able to handle optionally a DTD or some kind of XML schema. However current systems often either require a schema or ignore it completely. So it was necessary to support both kinds of systems which we do with the two variants of the benchmarks and different performance metrics.
- In order to not reduce the operation mix of the benchmark to very simple queries we needed a possibility to support systems with limited functionality. We thus allow implementing the query and update functionality at the application level, e.g. by programs executed on the application server.

3.2 Benchmarking XML Database Systems

The reference implementation of XMach-1 was done in Java since almost all XML data management systems provide a Java API. We developed a database loader and a generic workload generator to reduce the effort necessary to adapt the benchmark to a new system as much as possible. A specific module is to be provided to translate the generated operations to the specific system calls.

3.2.1 Database population

As stated above some XML data management systems require the definition of schemas prior to loading documents while other systems ignore schemas completely. So far we haven't seen a deep performance impact of using schemas. There is only small evidence that schema-based systems may scale better. On the other hand, systems requiring a schema tend to have problems with a large number of schemas.

XML data is often highly redundant so that XML data management systems should be able to limit the space requirements for document storage. However in our tests the databases occupied 3-8 times the space of the source data. Despite the space requirements of index structures there should be room for improvement.

3.2.2 Implementing database modules

While most operations of the XMach-1 workload mix can be expressed by a single statement of current XML language proposals we often had to implement them by programs consisting of multiple operations for specific systems. This was due to missing functionality such as for join or grouping. Furthermore some systems could not create new result elements or could only return and update complete documents.

3.2.3 Performance

We observed large differences in query response times between different systems even in single-user mode, e.g. ranging from 10 ms to several seconds on a small database. This was influenced by the fact that some operations had to be implemented at the application level. In multi user mode the 90th percentile response time limit which we set to 3 seconds for most queries was often missed. This means that only very modest throughput values are achieved, unless we change the specification to relax the response time requirements. Similar to TPC-C it could also be more appropriate to measure the throughput in Xqpm (XML queries per minute) instead of Xqps to more easily allow throughput values larger than 1.
An important performance factor in most systems is the definition of proper index structures. The systems mostly do not automatically create indexes but the user is responsible not only what to index but also how. A weak solution are systems which index all parts of the documents. Most evaluated systems do not output execution plans to inform the user if an index is used.

4 Conclusions

XMach-1 can be implemented for different XML data management systems. The implementation effort depends on the supported functionality. For currently available systems performance and scalability in a multi-user environment and with data corresponding to different DTDs are unsatisfying. Sufficient performance can be achieved for XML-formatted relational data or small data sets but not for large collections with millions of XML documents.

5 References

1. T. Böhme, E. Rahm: XMach-1: A Benchmark for XML Data Management. Proc. of 9th German database conference BTW01, Springer-Verlag, March 2001, http://dbs.uni-leipzig.de/en/projekte/XML/XmlBenchmarking.html
2. TPC-W Specification. http://www.tpc.org
3. XQuery: A Query language for XML. W3C Working Draft, Febr. 2001, http://www.w3.org/TR/xquery/