

The XMLBench Project: Comparison of fast, multi-platform XML libraries

Suren Chilingaryan¹

Forschungszentrum Karlsruhe, Hermann-von-Helmholtz-Platz-1, 76344,
Eggenstein-Leopoldshafen, Germany,
Suren.Chilingarian@ipe.fzk.de,
WWW home page: <http://xmlbench.sourceforge.net>

Abstract. The XML technologies have brought a lot of new ideas and abilities in the field of information management systems. Nowadays, XML is used almost everywhere: from small configuration files to multi-gigabyte archives of measurements. Many network services are using XML as transport protocol. XML based applications are utilizing multiple XML technologies to simplify software development: DOM is used to create and navigate XML documents, XSD schema is used to check consistency and validity, XSL simplifies transformation between different formats, XML Encryption and Signature establishes secure and trustworthy way of information exchange and storage. These technologies are provided by multiple commercial and open source libraries which are significantly varied in features and performance. Moreover, some libraries are optimized to certain tasks and, therefore, their actual library performance could significantly vary depending on the type of data processed. XMLBench project was started to provide comprehensive comparison of available XML libraries in their functionality and ability to sustain required performance. The main target was fast C and C++ libraries able to work on multiple platforms. The applied tests compare different aspects of XML processing and are run on few auto-generated data sets emulating library usage for different tasks. The libraries tested: Expat XML Parser, QTSoftware XML Module, Apache XML Project (C and Java versions), Gnome XML Toolkit, Oracle XDK (C and Java versions), and Intel XML Software Suite. The XML processing aspects benchmarked: XML parsing (DOM and SAX models), DOM manipulations, Schema Validation, XSL Transformation and XML Security. Data-sets used: a scalable generator producing XML files with simple structure emulating configuration files, a scalable generator emulating exchange of data with OPC XML-DA server, XML generator modeling an auction website provided by Xmark project, and, finally, RDF data set. The details of test setup and achieved results will be presented.

1 Introduction

The rapid spread of XML technologies in the information management systems resulted in appearance of multiple XML toolkits and standalone tools aimed

to automate various parts of XML processing. Event driven and object model based parsers are implementing DOM (Document Object Model [1]) and SAX (Simple API for XML [2]) specifications in order to provide applications with general access to XML data. Some of them are implementing DTD (Document Type Definition [3]), XSD (XML Schema Definition [4]), and Relax-NG [5] to provide consistence and type conformance checking. Few tools implementing XML Encryption [6] and XML Signature [7] specifications are used to protect data from unauthorized access and guarantee data integrity. XPath [8] and XML Query [9] based tools are used to search and extract certain information from XML documents. XSL (XML Stylesheet Language [10]) formatters are able to present original data in multiple ways using different formats. SOAP (Simple Object Access Protocol [11]) is a core component of Web Service architectures and used to establish message interchange between remote components. OPC XML-DA (Open Process Control, XML Data Access [12]) and OPC UA Open Process Control, Unified Architecture [13]) servers and clients are used to provide access to control data in process automation tasks.

Being used nowadays at almost all stages of data processing, underlying XML software is greatly affecting overall stability and performance of complete informational system. This makes obvious that selection of appropriate tools is extremely important. To make proper selection, the available tools should be analyzed in their ability to cover required subset of XML specifications and stay in sync with rapidly developing technologies. The performance and memory consumption on different considered data sets should stay stable over long execution times and fit parameters of available hardware.

The plenty of various comparisons and benchmarks of various XML tools are already published on the web. The XML Parser Benchmarks by Matthias Farwick and Michael Hafner are comparing SAX and DOM performance of few C and Java parsers [14]. Unfortunately, the interest of authors is mainly in Java parsers and very important Xerces-C, Intel XSS are not covered. Dennis Sosnoski has published very interesting results providing multi-aspect comparison of XML software [15]. The individual results are presented for XML parsing, document construction, document serialization, and etc. The memory consumption is presented as well. However, the benchmark is quite outdated and only Java parsers are covered. Some vendors of the XML software (for example, Intel [16]) provide benchmarking tools with their software. However, they as well include very limited set of tested software and such results should be treated with caution.

Java performance is highly improved in last years and in many tasks could compete with performance of C/C++ applications. The advantage of the later is much higher potential in exploring features of modern hardware like, for example, streaming SIMD extensions. Another important aspect is finer grained control on memory allocation which allows significant reduction of memory consumption while working with in-memory representations of big XML documents. Finally, there are many embedded applications with simple hardware which is only capable of C code execution.

The XMLBench project was started to provide comprehensive measurement of available and actively developed XML libraries written in C and C++ languages. Only software available on multiple platforms was considered. The tests are trying to exploit broad range of XML processing tasks using data sets ranging from few kilobytes up to hundreds of megabytes.

Chapter 2 describes the considered XML toolkits. The third section contains details on the benchmark setup, presents results, and proposes few ideas for performance improvement. Final chapter summarizes obtained results and includes discussion on their reliability.

2 XML Toolkits

This section contains list of tested toolkits including information about latest version, supported platforms and languages, and provided subset of XML features. The latest versions of all presented toolkits were used during benchmarks if not stated otherwise in this chapter.

2.1 Apache XML Project

The Apache XML Project is maintained by the Apache Foundation. The project consists of several libraries each having C++ and Java versions: Apache Xerces [17, 18] is validating XML parser. Apache Xalan [19, 20] provides implementation of XPath and XSL transformations. Apache Axis [21] provides Web Services infrastructure. Apache XML Security [22] implements XML Signature and Encryption specifications. Apache FOP [23] implements XSL-FO. Finally, XPath2 and XQuery are provided by 3rd party project: XQilla [24].

Version	Xerces 3.0 (from 29.09.2008), Xalan 1.10, XML Security 1.4.0
License	Open Source Apache License 2.0
Supported Systems	32/64 bit versions; Windows, Linux, BSD, OS X, Solaris, AIX, HP-UX
Native Language	C++ and Java
Language Bindings	Perl
XML Version	XML 1.0 and 1.1 (partly), Namespace support, XInclude support
API Supported	SAX 1 and 2, DOM levels 2 and 3
Network	FTP, HTTP
Validation	DTD, Schema 1.0
Security	Canonical XML 1.0, XML Signature, XML Encryption
Query	XPath 1.0 and 2.0
Stylesheets	XSLT 1.0, XSL-FO 1.1 (Java Only)
Web Services	SOAP 1.1 and 1.2, attachments, binary serialization, WSDL 1.1, JSON

Unfortunately, the Xerces-C 3.0 is only supported by Xalan-C 1.11 which is still not released. In turn Apache XML Security depends on Xalan-C and even latest snapshot do not include stable support of Xalan-C 1.11 prereleases. Therefore, most of the tests were executed with Xerces-C 3.0 and preliminary version of Xalan-C 1.11. The Encryption and Signature benchmarks were run with Xerces-C 2.8.0, Xalan-C 1.10, and Apache XML Security 1.4.0.

2.2 Gnome XML Library

The Gnome XML is a set of libraries developed by various people for Gnome Project. However, it has widely spread outside of Gnome since then. The core of the toolkit is a Gnome XML and XSLT C Libraries, also known as LibXML and LibXSLT correspondingly [25]. XML Security is provided by XMLSec Library [26]. DOM API is provided by GDome library [27], SOAP using CSOAP [28], and XSL-FO by xmlroff [29].

Version	LibXML 2.7.3 (from 18.01.2009), LibXSLT 1.1.24, XmlSec 1.2.11, GDome 0.8.1
License	Open Source MIT License
Supported Systems	32/64 bit versions; Windows, Linux, BSD, OS X, Solaris, QNX
Native Language	C
Language Bindings	Perl, PhP, Python, Ruby, Tcl/Tk
XML Version	XML 1.0 5th Ed., Namespace support, XInclude support
API Supported	SAX 1 and 2, Native DOM style, DOM level 2, Push Parser Mode
Network	FTP, HTTP
Validation	DTD, Schema 1.0 (partly), Relax-NG
Security	Canonical XML 1.0, XML Signature, XML Encryption
Query	XPath 1.0
Stylesheets	XSLT 1.0, EXSLT, XSL-FO 1.1 (partly)
Web Services	SOAP 1.1

Due to unresolved problems, 2.7.3 version of LibXML was considerably slower in creation/serialization of big DOM documents. For that reason, in DOM creation benchmark, the results achieved with LibXML 2.6.32 are presented. The LibXML 2.7.3 was used to obtain all other results.

2.3 Expat

The Expat [30] is a very simple and fast XML parser widely used in various open-source software including Mozilla project. Although it only implements SAX API, the Sablotron [31] and Arabica [32] libraries are implementing DOM, XPath and XSLT specifications at the top of Expat.

Version	Expat 2.0.1 (from 05.06.2007), Arabica Oct2008, Sablotron 1.0.3
License	Open Source MIT License
Supported Systems	32/64 bit versions; Windows, Linux, BSD, OS X, Solaris, QNX, HP-UX, AIX
Native Language	C
Language Bindings	C++, Lua, OCaml, Perl, PhP, Python, Ruby, Tcl/Tk
XML Version	XML 1.0, Namespace support, XInclude support
API Supported	SAX 1 and 2, DOM level 2
Network	-
Validation	DTD only
Security	-
Query	XPath 1.0
Stylesheets	XSLT 1.0
Web Services	-

2.4 QTSoftware XML Module

QT [33] is a popular cross-platform application framework. Since version 3 it includes module providing some XML processing capabilities.

Version	Qt 4.4.3 (from 27.09.2008)
License	Commercial, Open Source GPL (Switch to LGPL is announced [34]) License
Supported Systems	32/64 bit versions; Windows, Linux, BSD, OS X, Solaris, QNX, HP-UX, AIX
Native Language	C++
Language Bindings	Python
XML Version	XML 1.0, Namespace support
API Supported	SAX 2, DOM level 2
Network	FTP, HTTP
Validation	DTD only
Security	-
Query	XQuery 1.0, XPath 2.0
Stylesheets	-
Web Services	-

2.5 Intel XML Software Suite

Intel has developed its XML Software Suite (XSS [35]) using all capabilities of their newest processors. XSS is heavily optimized and utilizing full SSE instruction set including SSE4. As well, the XSS is only tested toolkit which can

benefit from multiple cores while performing manipulations on a single document. However, using dual core machine and current benchmark set, the results have not shown any performance benefit and, therefore, multi-core results are not included in this article.

Version	Intel XSS 1.2 (from 19.01.2009)
License	Commercial, sources are not available
Supported Systems	32/64 bit versions; Windows, Linux, HP-UX
Native Language	C++/Java
Language Bindings	-
XML Version	XML 1.0 5th Ed., Namespace support, XInclude support
API Supported	SAX 2, DOM level 2 and partial level 3, StAX parser (Java only)
Network	FTP, HTTP
Validation	DTD, XML Schema 1.0
Security	-
Query	XPath 1.0
Stylesheets	XSLT 1.0
Web Services	-

2.6 Oracle XDK

The Oracle XML Development Kit (XDK [36]) is developed by the Oracle Corp. to provide XML capabilities in their database solutions.

Version	Oracle XDK 10g 10.2.0.2.0 (from 07.04.2006)
License	OTN, sources are not available
Supported Systems	32 bit only; Windows, Linux, HP-UX, AIX
Native Language	C/C++/Java
Language Bindings	-
XML Version	XML 1.0, Namespace support, XInclude support
API Supported	DOM level 2 and 3, StAX (Java Only), SAX ¹
Network	FTP, HTTP
Validation	DTD, XML Schema 1.0
Security	-
Query	XPath 1.0, XPath 2.0 in Java version
Stylesheets	XSLT 1.0, XSLT 2.0 in Java version
Web Services	-

¹ Implemented on top of DOM according to benchmark results

3 Benchmark Results

3.1 System Setup

The hardware platform was Fujitsu-Siemens Celsius W350 workstation.

- Intel Core Duo E6300 (1.86GHz, 2MB L2 cache)
- Fujitsu-Siemens D3217-A, Intel Q965 chipset (ICH8R, 1066 MHz bus)
- 4GB DDR2-800 Memory

All presented results were obtained using 64 bit version of Gentoo Linux.

- Kernel 2.6.25
- GNU C Library 2.6.1
- GNU C Compiler 4.2.4
- Java SE Runtime Environment 1.6.0_07 (64-bit)
- *-O2 -march=nocona* optimization flags were used to build all open source libraries

The XML data was always pre-generated and loaded into the memory before running benchmark. Therefore, the hard drive performance is not affecting results. A few DOM benchmarks on big data sets were the only exclusion. In that case some of the parsers have utilized more than 4GB of memory and, therefore, disk swapping was happening. However, such results in any case were high above the rejection threshold and anyway were scored with worst value.

3.2 Data Description

Four different data sets were used in benchmarking process.

- *RDF* - A big RDF (Resource Description Framework [37]) document from DMoz.org project describing various web resources.
- *XMLGen* - Scalable data generator producing very simple XML content: four levels of deepness, no namespaces, very limited amount of different XML nodes.
- *XMark* - Another scalable data generator provided by XMark Project [38]. It produces XML documents modeling an auction website. The XML have slightly more complicated structure: more levels, higher variety of XML elements. The namespaces are still not used.
- *OPCGen* - A data generator emulating behavior of OPC XML-DA server [12]. Various SOAP messages used in data exchange are generated. Size of some of these messages is scaled by scaling parameter, others are staying constant. The XML elements and attributes are belonging to few different namespaces.

3.3 Benchmark Setup

All tests were run in a single user mode without any system services running. To eliminate any possibility of negative interference for each task, for each type of data, and for each XML toolkit a standalone test application were executed. The toolkit initialization, if possible, was performed before data processing and time spent in initialization is not counted in benchmark scores. Then the test was executed continuously on slightly changing data. The scaling parameters have remained the same during the test execution, but values of text nodes and attributes were generated randomly. The execution time of a single test was about 5 to 10 minutes. It was slightly longer on big data sets to provide at least 10 repetitive executions. The full test set has approximately taken two days for completion.

For each toolkit the time required to process data is measured. This time, then, is divided by the time required by library from *Gnome XML Toolkit* to accomplish the same task on the same data (The LibXML, LibXSLT, or XMLSec timings are used as a reference values depending on benchmark). The resulting value is called performance index and shown on performance charts below (better results are represented by smaller values of the indexes). To prevent poisoning of overall result by a single failed test, the maximal value of the performance index is limited by 10 (and 15 for DOM parsing benchmark, see explanation below).

3.4 Parsing Benchmark

The parsing benchmark evaluates the time required to parse an XML document and provide encompassed information to the application. The parsing should be performed before any other action could be done on XML data. Hence, the parsing performance is significantly contributing to overall performance of any application implicating usage of XML data.

Currently, two main approaches of parsing are widely used.

- The DOM (Document Object Model [1]) parsers are evaluating XML data and construct in-memory DOM representation. This representation may be described as linked tree of objects (nodes). Each object provide a set of methods to obtain and set values of associated properties and navigate to neighboring elements.
- The SAX (Simple API for XML [2]) is a signal based parsing interface. The application provides a set of callbacks which are called when correspondent data is encountered in XML stream.

Basically, the DOM representation is much easier to handle and it can be used to alter content of XML document. The SAX parsers are normally faster and do not have any limitation on the size of XML document.

To show relative performance of SAX and DOM parsers, the timings achieved by LibXML SAX parser are used to calculate performance indexes in both, SAX and DOM, cases. The maximal allowed index of DOM parser is raised to

15 to acknowledge fact what most of DOM parsers are slower than their SAX counterparts.

Figure 1 presents results of parsing benchmark. The DTD and other means of validity checking were not used.

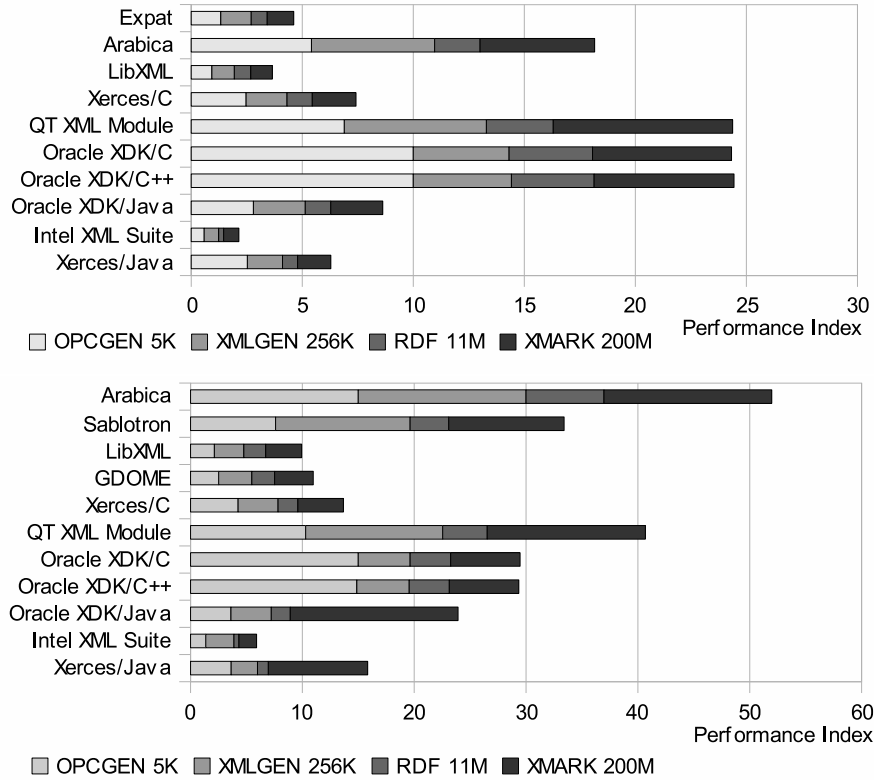


Fig. 1. Benchmark results of SAX (top) and DOM (bottom) mode parsers

3.5 Schema Validation Benchmark

The Figure 2 compares time needed by toolkits to validate XML documents against schema [4]. The data is generated by *XMLGen* and *OPCGen* generators. The *XMLGen* schema is very simple, do not include any type checking and consists only of 20 lines. OPC data have a complex structure and is described by rather complex schema definition considering checking of multiple XSI types. To get pure validation time, the non-validated parsing time is subtracted from complete time spent on parsing and validation. The validation context is created only once at the initialization stage.

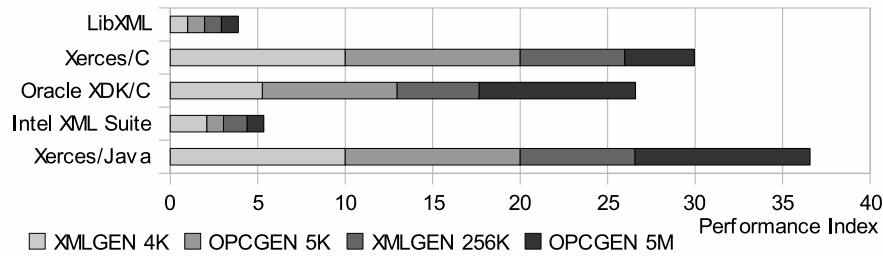


Fig. 2. Results of XML schema validation benchmark

3.6 DOM Manipulation Benchmark

This benchmark evaluates ability of XML toolkits to manipulate data in DOM representation. First *XMLGen* style document is created using DOM API and, then, serialized into the UTF-8 string. In four tests presented on Figure 3 the resulting string sizes are varied from 5 KB to 90 MB.

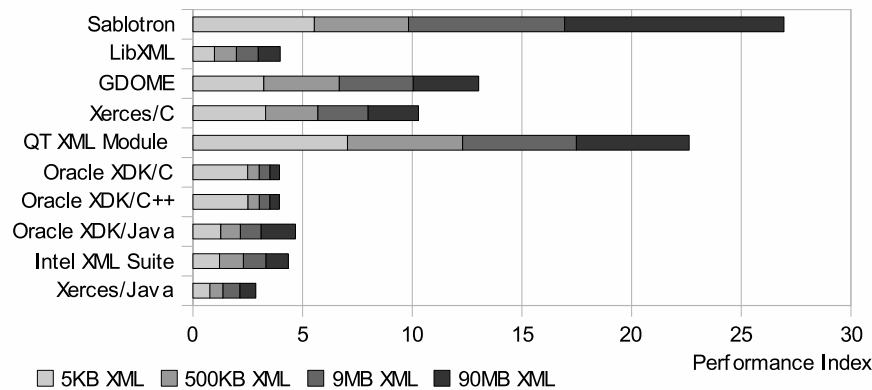


Fig. 3. Results of DOM manipulation benchmark

3.7 XSL Transformation Benchmark

This benchmark evaluates performance of XSL transformation engines [10]. The *XMLGen* and RDF samples are converted to HTML. Rather big ODF (Open Document Format) document is converted to MediaWiki format. Both, *XMLGen* and RDF, are processed using tiny and simple XSL stylesheets. However, stylesheet used to process RDF document demands reference lookup over a big node-set. And the stylesheet used to process ODF document is taken from OpenOffice 3 distribution and have much more sophisticated structure.

The performance charts are depicted on the Figure 4. The transformation engine is created only once at initialization stage and the time needed to parse original documents is excluded from presented results.

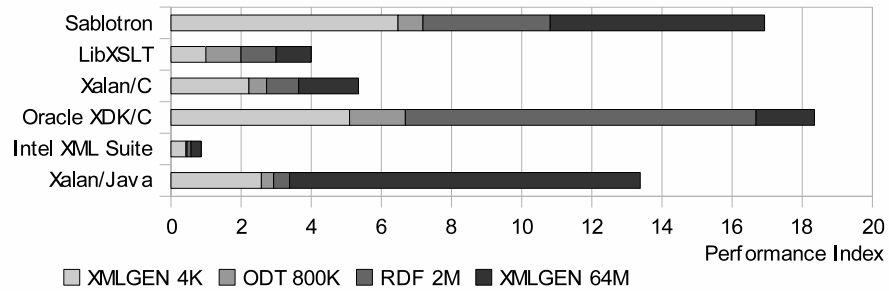


Fig. 4. Results of XSL transformation benchmark

3.8 XML Security Benchmark

The security capabilities are measured by two independent tests.

- First one evaluates time required to sign XML document with digital signature and, then, verify this signature.
- Second - time used to encrypt XML document and, then, decrypts it back.

Results of both tests are presented on Figure 5. The key generation and initialization of security contexts are performed during test initialization and not affecting test results. The time needed to parse original document is not included as well.

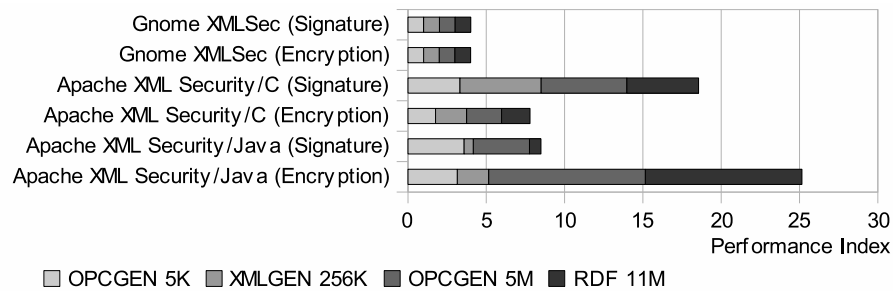


Fig. 5. Results of XML security benchmark

3.9 Software Optimization

A few things can be done to optimize speed of XML applications linked with open source libraries. The performance is significantly dependent on the compiler and optimization flags used to build stack of XML libraries. However, in most Linux distributions the software is compiled with safe but not always optimal flags. Considerable performance increase could be achieved with recompilation of certain libraries using better optimization flags. Upgrading system to 64 bit version of operating system and installing latest version of *Gcc* compiler and *Glibc* system library could improve performance as well. A basic estimation of performance impact of different compilers and optimization options could be found on Figure 6. However, this is highly dependent on libraries and hardware used and should be evaluated for each specific case.

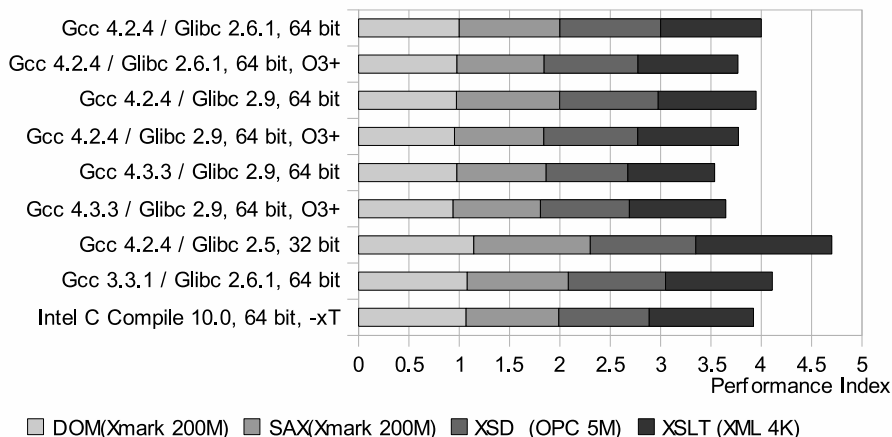


Fig. 6. Impact of used compiler on a performance of Apache Xerces and Xalan libraries. The timings of *Gcc 4.2.4 / Glibc 2.6.1* (with standard optimization flags: *-O2 -march=nocona*) were used as a reference values to calculate performance indexes (This combination were utilized to build libraries for the rest of tests). *O3+* optimization flags include: *-O3 -unroll-loops -mfpmath=sse,387 -march=nocona*.

If XML data includes text messages written in non-Latin alphabets, it becomes very important to select proper character encoding. The Figure 7 illustrates this fact. *LibXML* is treating data in UTF-8 encoding internally. The performance is significantly decreased if UTF-16 encoding is used for the data. Vice-versa *Xerces/C* has UTF-16 internal data representation and performance drops if UTF-8 encoded document is passed.

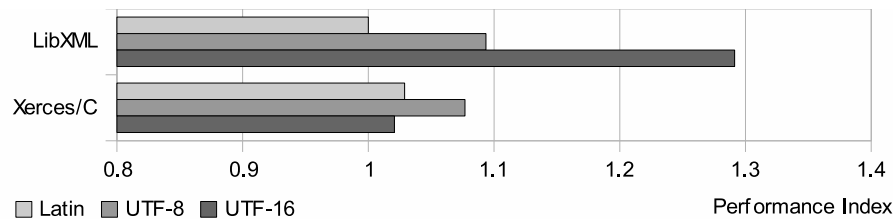


Fig. 7. Performance of Xerces/C and LibXML libraries depending on Unicode encoding used to store Cyrillic characters. The time needed by LibXML to process Latin document was used to calculate performance indexes in this test.

4 Discussion

As with any other benchmark results, presented scores should be treated cautiously. As it could be seen from the charts, the performance is highly dependent on the type of data and on task performed. Especially good this point is illustrated by XSL transformation benchmark presented on Figure 4. The minimal and maximal indexes of *Intel XML Suite* differ for more than ten times. The *Xalan/C* is in some tests up to two times faster than *LibXSLT* and in others two times slower. Figure 7 indicates what parsing performance is dependent on encoding used to code Unicode characters. The system, compiler and optimization flags used to compile toolkits could have significant impact on performance as well. Figure 6 proves that only proper selection of compiler and optimization flags may give extra 15% of performance. Finally, non-performance issues should be considered. Not all toolkits are implementing specifications completely and properly. Various defects and gaps in implementation are not easy to detect using benchmarks evaluating only a few standard test cases.

Taking into account aforesaid, the fastest among open source toolkits is clearly *Gnome XML Toolkit*. It have relatively rich set of features and shows good performance in most of the tests. The main problems are low performance in some of XSL transformation tests and incomplete implementation of XML Schema specification. The *Apache XML Toolkit* is slightly slower in average and especially in tests related to XML security. However, it has the best list of supported features and its schema implementation is much more complete. The top performance is obviously has *Intel XML Suite*. Thoroughly using hardware optimizations it provides best results in most of the benchmarks. In the rest, actually few DOM and Schema tests, it is only a little behind the winning *LibXML* and *Xerces/Java* (correspondingly). Especially impressive performance is demonstrated in XSL transformation benchmark. The *Intel XML Suite* has managed to convert the Open Office document to MediaWiki format ten times faster than any competing tool. In other XSLT tests Intel has also outperformed all competitors from two to eight times. The drawbacks: it is a commercial product and it is only supported on Intel and HP platforms. The XML Security implementation is still missing.

The Java version of *Apache XML Toolkit* has shown very good results as well. In most cases it has performance comparable with fastest C libraries and it even got a best score in the DOM manipulation benchmark. The main problem was memory consumption. In few tests involving processing of big documents the Java have failed to perform required actions using available system memory and disk swapping was happening.

Finally, I want to reference two research projects which can be useful in the case if performance is an ultimate goal.

Developers at IBM have implemented a non-validating high-performance XML parser on *IBM Cell* platform. The parser consists of a front-end that provides a SAX application interface, and eight back-end parse threads on the CELL SPEs. According to published results the parser is 2.3 times as fast as *LibXML* on an Intel Pentium M processor [39].

AsmXml is open source high performance XML parser. It has very limited set of capabilities (even namespaces are not supported), requires schema file describing syntax of XML documents which are going to be parsed and, for that reasons, was not included in the benchmarks. However, it is implemented in pure assembler and have very efficient memory model [40]. The results showed on Figure 8 indicate that it is three to ten times faster than *Intel XML Suite* and *LibXML* in parsing *XMLGen* documents.

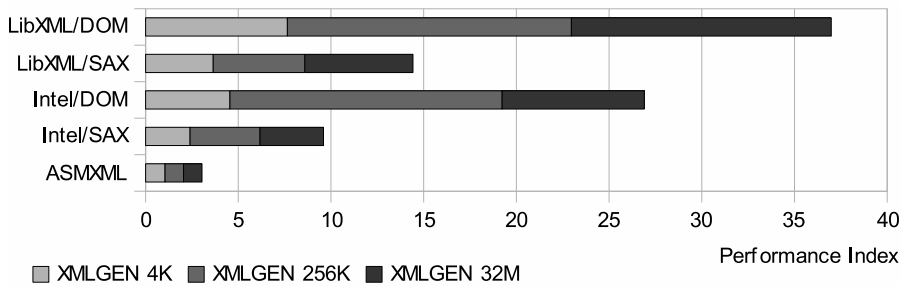


Fig. 8. Performance of AsmXml parser compared to Intel XML Suite and LibXML. AsmXml timings were used to calculate performance indexes in this test.

References

1. W3C: Document object model
Available as <http://www.w3.org/TR/DOM-Level-2-Core/>
2. Megginson, D.: Simple api for xml (sax)
Available as <http://www.saxproject.org>
3. W3C: Extensible markup language (xml) 1.0
Available as <http://www.w3.org/TR/REC-xml>

4. W3C: Xml schema part 0: Primer
Available as <http://www.w3.org/TR/xmlschema-0/>
5. OASIS: Relax ng, iso/iec 19757-2:2003
Available as <http://www.oasis-open.org/committees/relax-ng/spec-20011203.html>
6. W3C: Xml encryption syntax and processing
Available as <http://www.w3.org/TR/xmlenc-core/>
7. W3C: Xml signature syntax and processing
Available as <http://www.w3.org/TR/xmlsig-core/>
8. W3C: Xml path language (xpath)
Available as <http://www.w3.org/TR/xpath>
9. W3C: Xquery 1.0: An xml query language
Available as <http://www.w3.org/TR/xquery/>
10. W3C: Xsl transformations
Available as <http://www.w3.org/TR/xslt>
11. W3C: Soap version 1.2 part 0: Primer
Available as <http://www.w3.org/TR/2003/REC-soap12-part0-20030624/>
12. OPC Foundation: Opc xmlda 1.01 specification
Available as <http://opcfoundation.org>
13. OPC Foundation: Opc ua part1 - overview and concepts 1.01 specification
Available as <http://opcfoundation.org>
14. Farwick, M., Hafner, M.: Xml parser benchmarks
Available as <http://www.xml.com/pub/a/2007/05/16/xml-parser-benchmarks-part-2.html>
15. Sosnoski, D.: Xmlbench document model benchmark
Available as <http://www.sosnoski.com/opensource/xmlbench/index.html>
16. Intel: Xml benchmark tool
Available as <http://software.intel.com/en-us/articles/intel-xml-software-products/>
17. The Apache Software Foundation: Apache xerces-c
Available as <http://xerces.apache.org/xerces-c/>
18. The Apache Software Foundation: Apache xerces-j
Available as <http://xerces.apache.org/xerces2-j/>
19. The Apache Software Foundation: Apache xalan-c
Available as <http://xml.apache.org/xalan-c/>
20. The Apache Software Foundation: Apache xalan-j
Available as <http://xml.apache.org/xalan-j/>
21. The Apache Software Foundation: Apache axis
Available as <http://ws.apache.org/axis2/>
22. The Apache Software Foundation: Apache xml security
Available as <http://santuario.apache.org/index.html>
23. The Apache Software Foundation: Apache fop (formatting objects processor)
Available as <http://projects.apache.org/projects/fop.html>
24. XQilla Team: Xqilla
Available as <http://xqilla.sourceforge.net/>
25. Veillard, D.: The xml c parser and toolkit of gnome
Available as <http://xmlsoft.org/>
26. Sanin, A.: Xmlsec library
Available as <http://www.aleksey.com/xmlsec/>
27. Casarini, P.: Gnome dom engine
Available as <http://gdome2.cs.unibo.it/>

28. Ayaz, F.: Client/server soap library in pure c
Available as <http://csoap.sourceforge.net/>
29. XMLroff Team: Xmlroff xsl formatter
Available as <http://xmlroff.org/>
30. Expat Team: Client/server soap library in pure c
Available as <http://csoap.sourceforge.net/>
31. Ginger Alliance: Sablotron: Xslt, dom and xpath processor
Available as <http://www.gingerall.org/sablotron.html>
32. Higgins, J.: Arabica xml and html processing toolkit
Available as <http://www.jezuk.co.uk/arabica>
33. QT Software: Qt cross-platform application framework
Available as <http://www.qtsoftware.com/>
34. Nokia: Nokia to increase adoption of qt with additional licensing option
Available as <http://www.nokia.com/A4136001?newsid=1281996>
35. Intel: Intel xml software suite
Available as <http://software.intel.com/en-us/articles/intel-xml-software-suite/>
36. Oracle: Oracle xml developer kit 10g
Available as <http://www.oracle.com/technology/tech/xml/xdkhome.html>
37. W3C: Resource description framework
Available as <http://www.w3.org/TR/rdf-syntax-grammar/>
38. Schmidt, A.R., Waas, F., Kersten, M.L., Carey, M.J., Manolescu, I., Busse, R.: Xmark: A benchmark for xml data management. In: Proc. of Int. Conf. on Very Large Databases (VLDB), Hong Kong, China. (2002) 974–985
Available as <http://www.xml-benchmark.org/>
39. Letz, S., Zedler, M., Thierer, T., Schuetz, M., Roth, J., Seiffert, R.: Xml offload and acceleration with cell broadband engine (2006)
Available as <http://xtech06.usefulinc.com/schedule/paper/27>
40. Kerbiquet, M.: Asmxml
Available as <http://mkerbiquet.free.fr/asm-xml/>