

Benchmark Setup

1. Hardware

1. Intel Core Duo E6300 (1.86GHz, 2MB L2 cache)
2. Fujitsu-Siemens D3217-A, Intel Q965 chipset (ICH8R, 1066 Mhz bus)
3. 4 GB DDR2-800 Memory
4. Mirroring raid of 2 x Seagate Baracuda 7200.10 ST3500630AS (500GB)

2. Software

1. Gentoo Linux 64bit, Single User Mode
2. Kernel 2.6.25
3. GNU C Library 2.6.1
4. GNU C Compiler 4.2.4
5. Java SE Runtime Environment 1.6.0_07 (64 bit)

3. Libraries

1. Expat 2.0.1, Sablotron 1.0.3, Arabica Oct2008
2. Gnome LibXML 2.7.3, GDome 0.8.1, LibXSLT 1.1.24, XMLSec 1.2.11
3. Apache Xerces/C++ 3.0, Xalan/C++ 1.10, XML Security/C++ 1.4.0
4. Apache Xerces/Java 2.9.1, Xalan/Java 2.7.1, XML Security/Java 1.4.2
5. Trolltech QT 4.4.3
6. Intel XML Software Suite for C++ 1.2
7. Intel XML Software Suite for Java 1.2
8. Oracle XML Developers Kit for C 10.2.0.2.0
9. Oracle XML Developers Kit for Java 10.2.0.2.0

4. XML Files

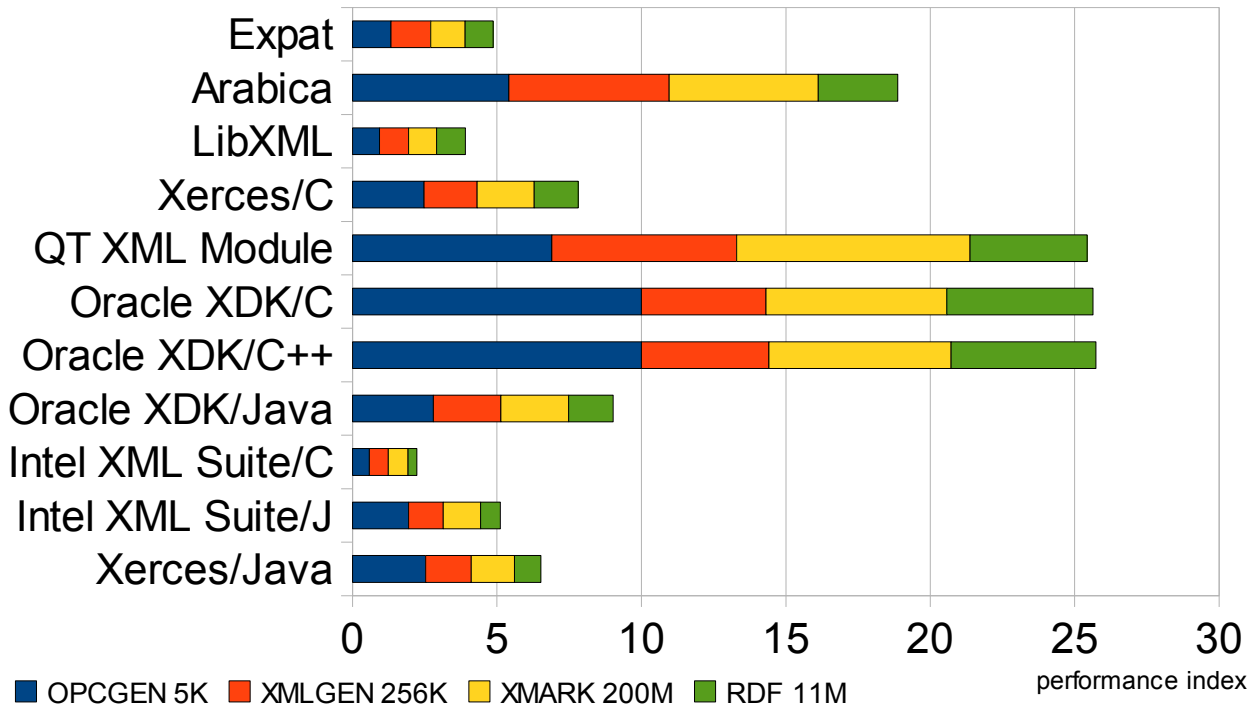
1. **RDF** - A big RDF document from DMoz.org project describing various web resources. It includes nodes from few namespaces and has constant depth of 3 levels.
2. **XMLGen** - Scalable data generator producing very simple XML content: 4 levels of depth, no namespaces, very limited amount of different XML nodes.
3. **XMark** - Another scalable data generator provided by XMark Project . It produces XML documents modeling an auction website. The XML have slightly more complicated structure: up to 8 levels of depth, higher variety of XML elements. The namespaces are still not used.
4. **OPCGen** - A data generator emulating behavior of OPC XML-DA server . The SOAP messages used in data exchange are generated. Size of some of these messages is scaled by scaling parameter, others are staying constant. The nodes and attributes are belonging to 3 different namespaces. The depth of generated XML is depending on the message type and varies from 1 to 5.

5. Performance Indexes

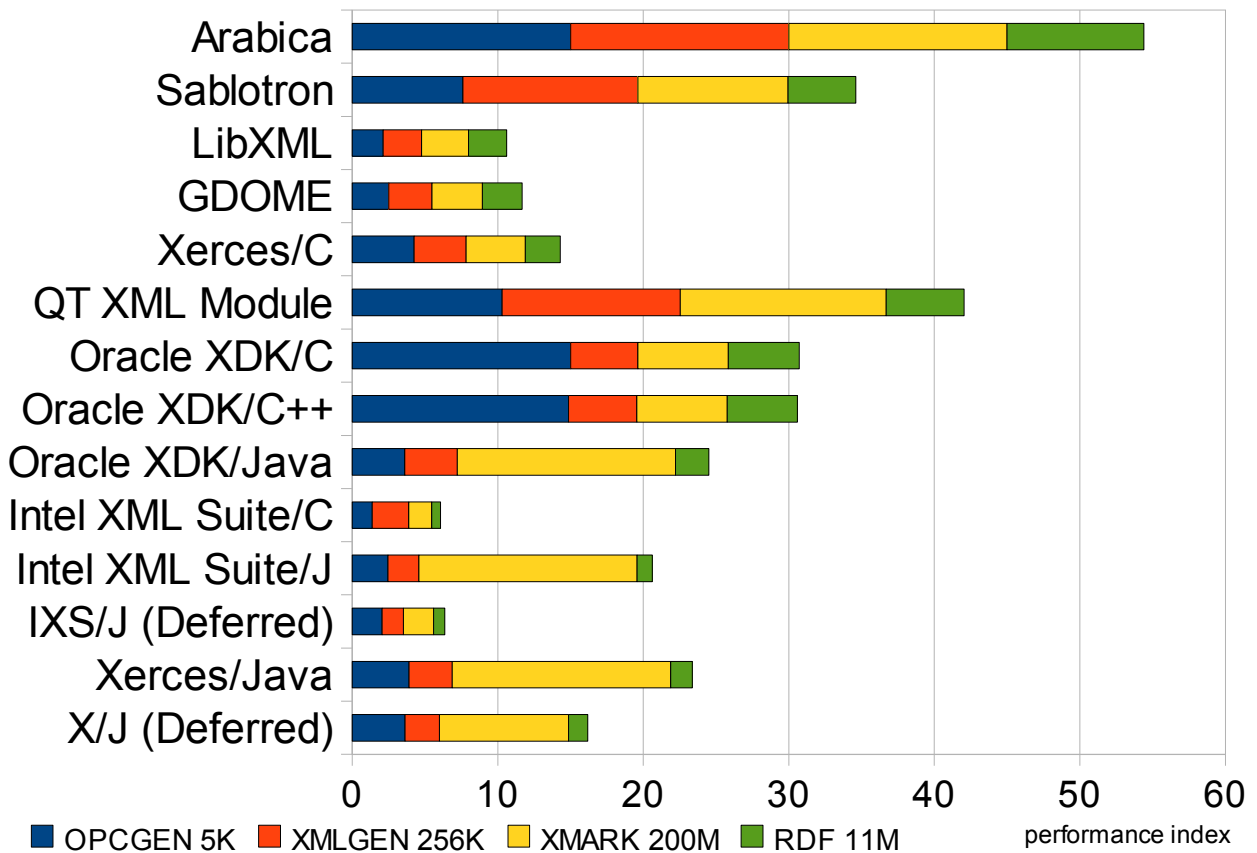
The time required to process data is measured in all tests. This time, then, is divided by the time required by a reference implementation to accomplish the same task on the same data. The resulting value is called performance index and shown on performance charts below. To prevent poisoning of overall result by a single failed test, the maximal value of the performance index for a single run is limited by 10 (and 15 for DOM parsing benchmark). The libraries from Gnome XML Toolkit (LibXML, LibXSLT, and XMLSec) are used as the reference implementations.

In all benchmarks lower value is better!

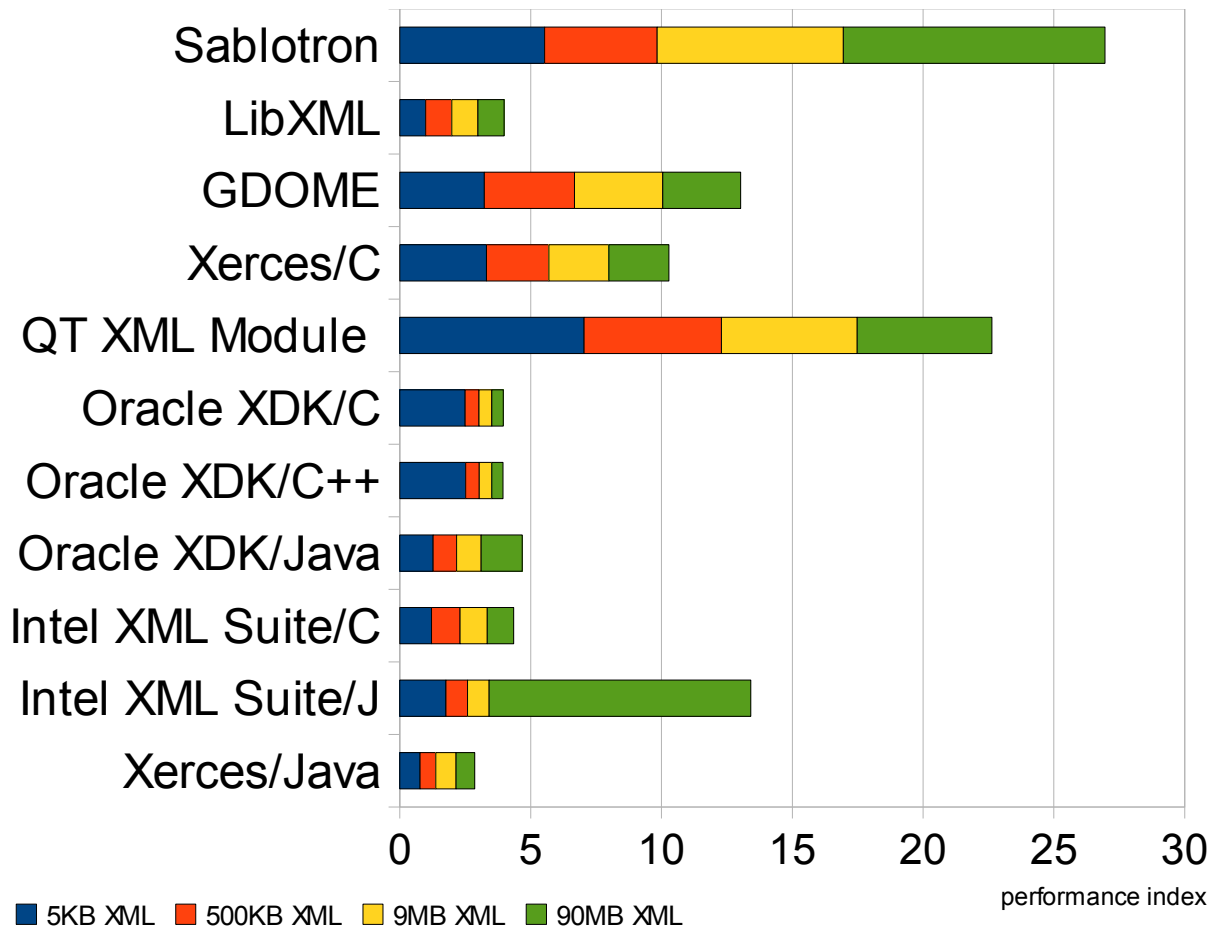
SAX Parsing Benchmark



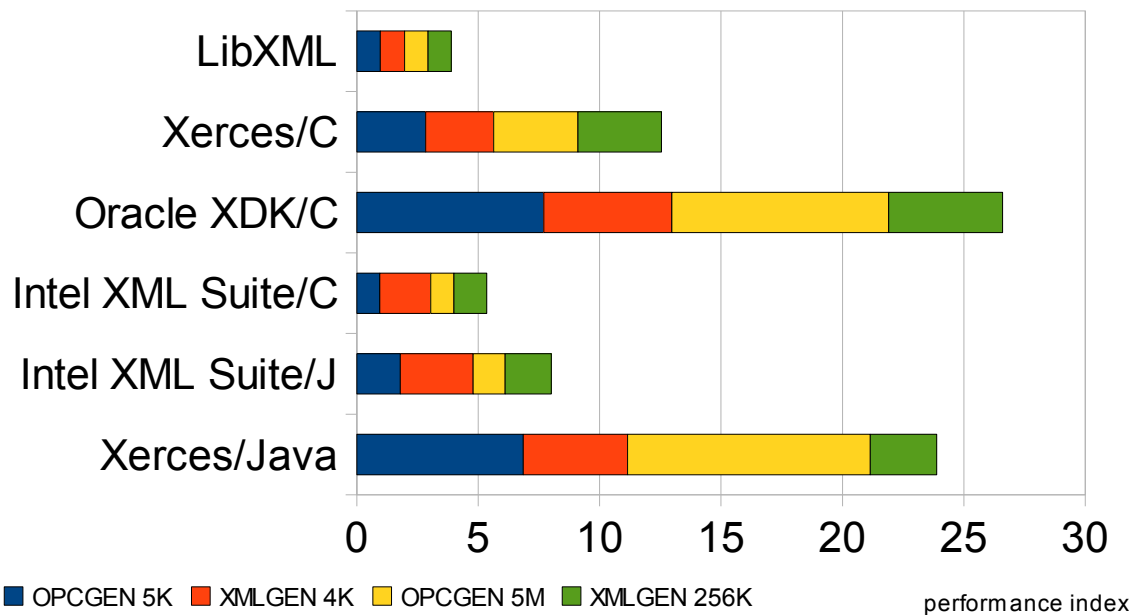
DOM Parsing Benchmark



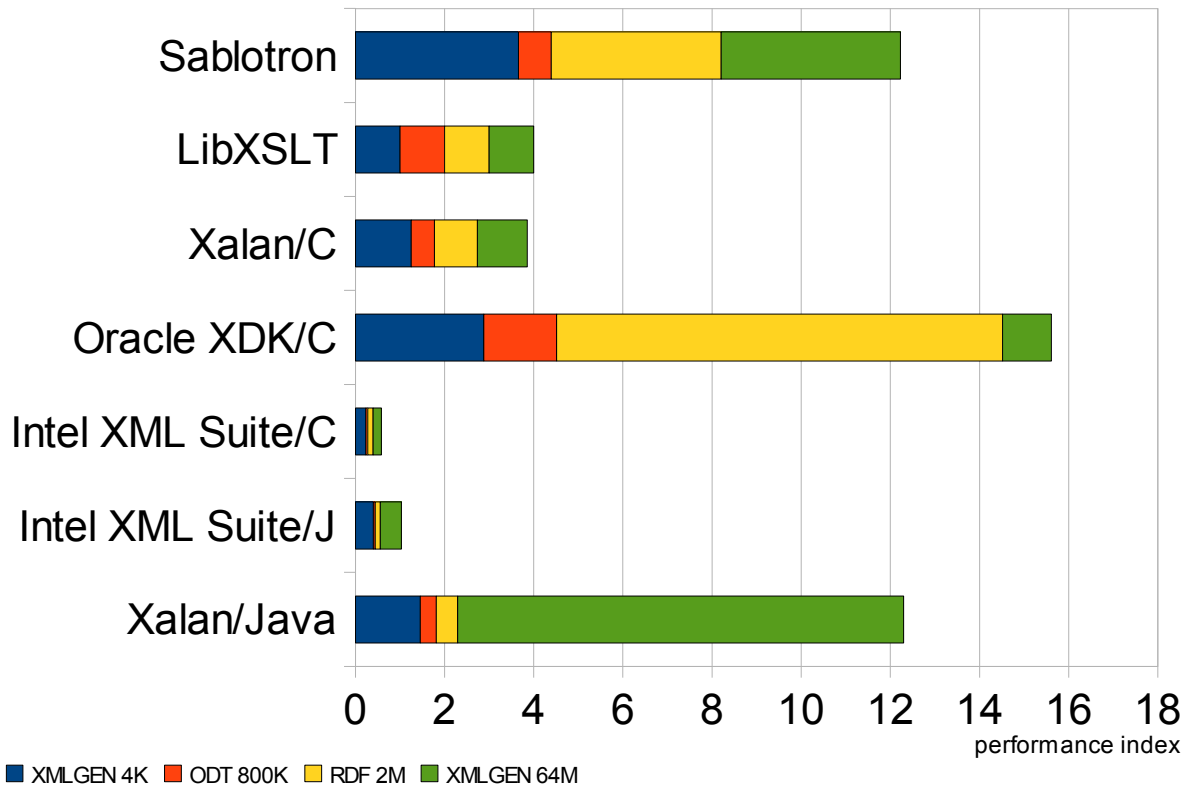
DOM Manipulations Benchmark



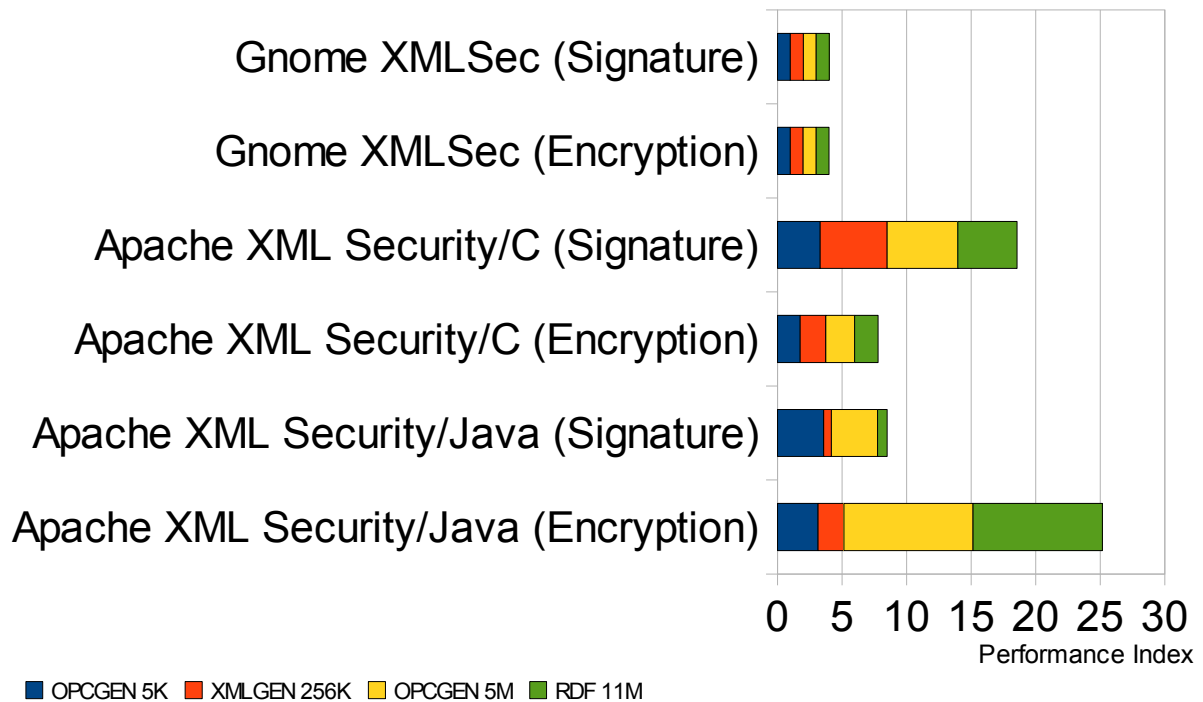
XSD Validation Benchmark



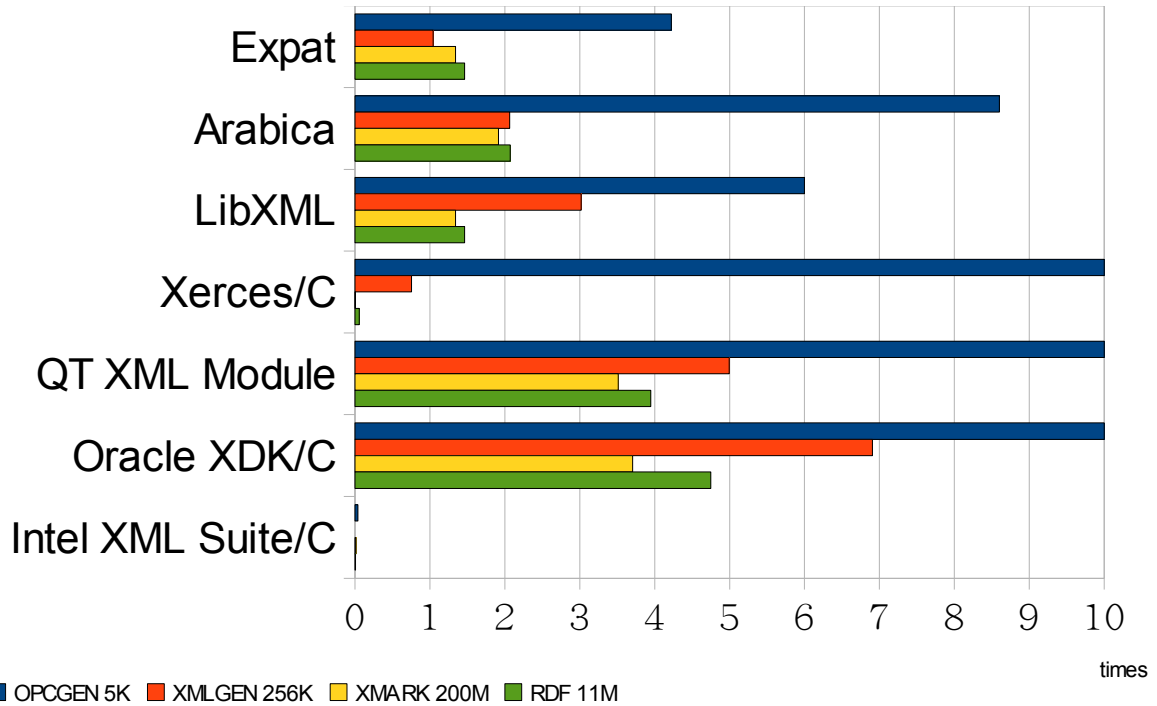
XSL Transformation Benchmark



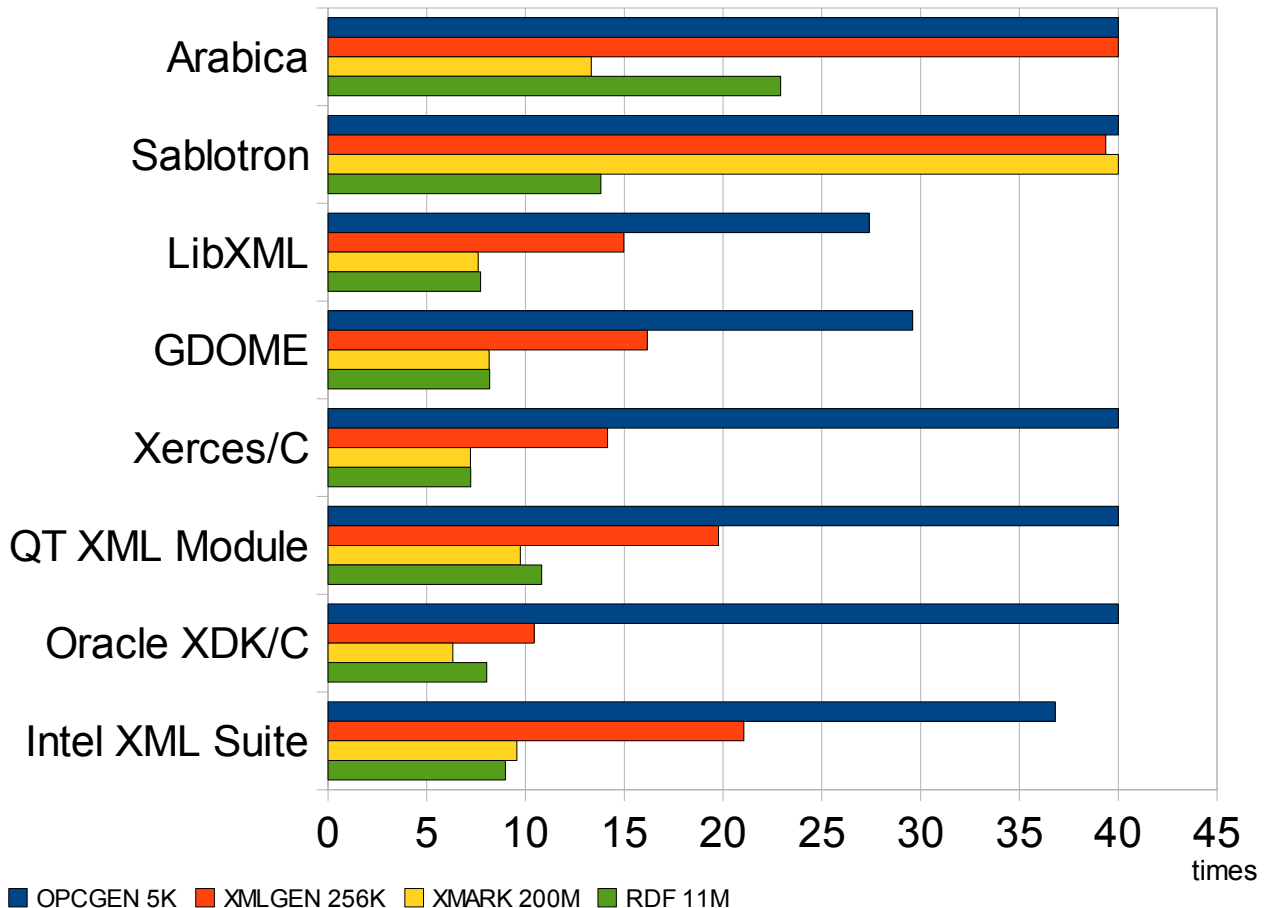
XML Security Benchmark



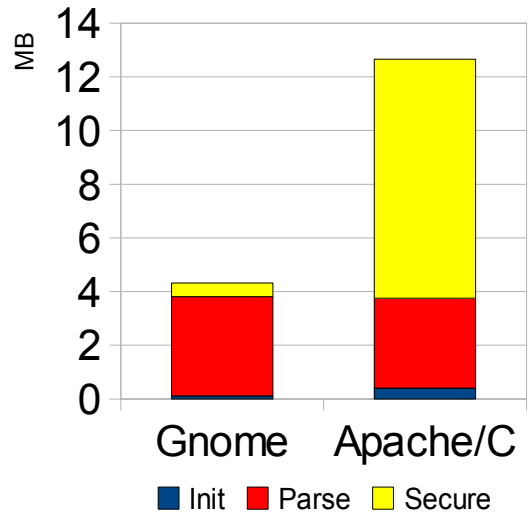
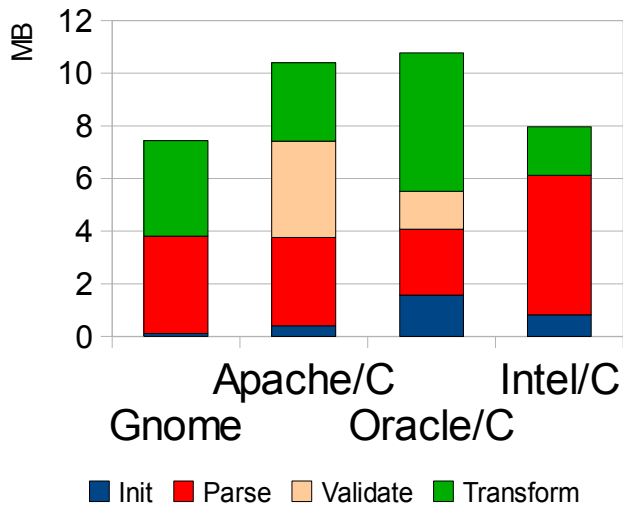
Memory Usage: SAX Parsing



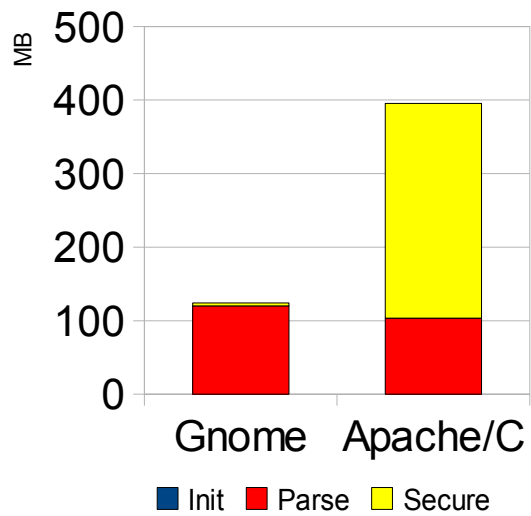
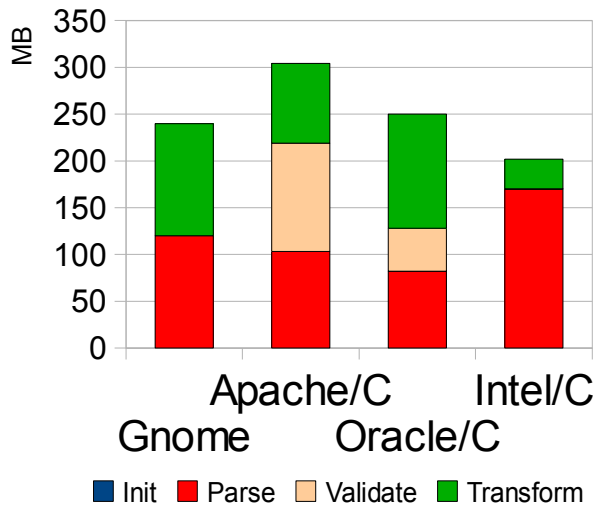
Memory Usage: DOM Parsing



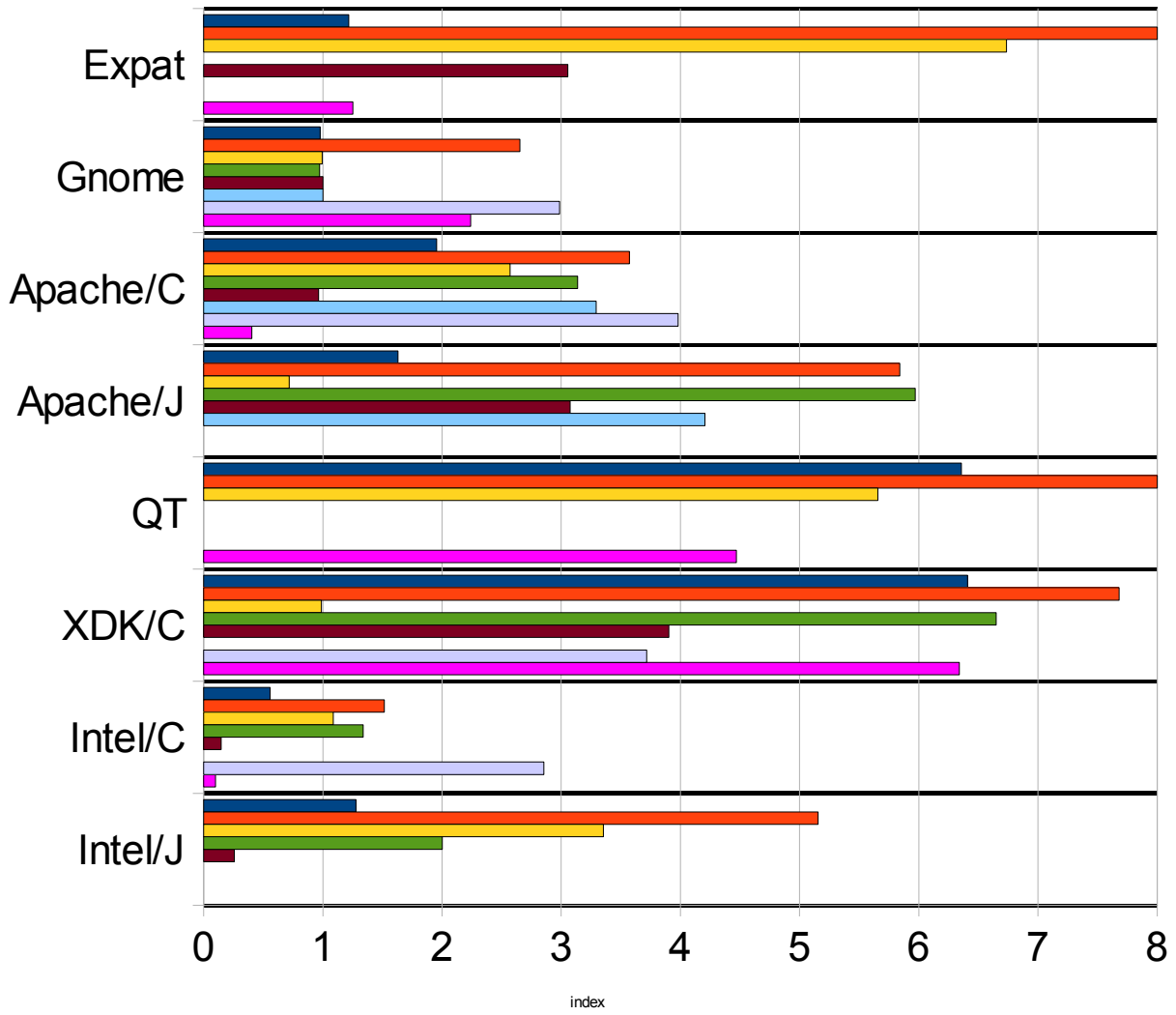
Memory Usage: Processing 256 KB File



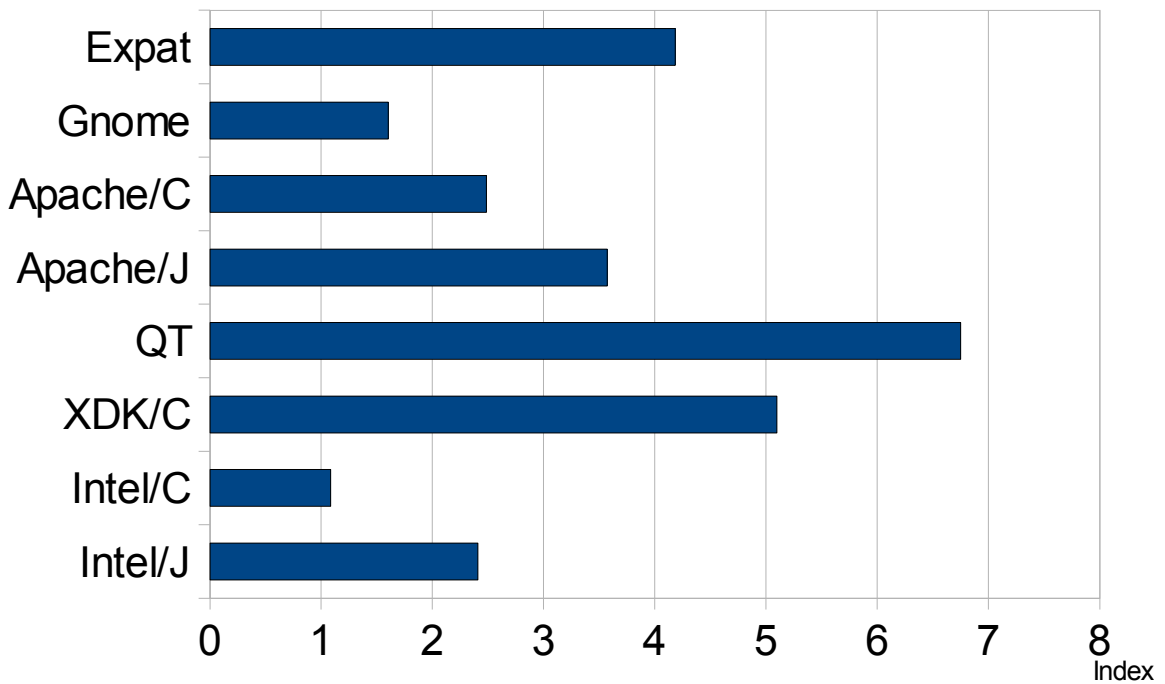
Memory Usage: Processing 8 MB File



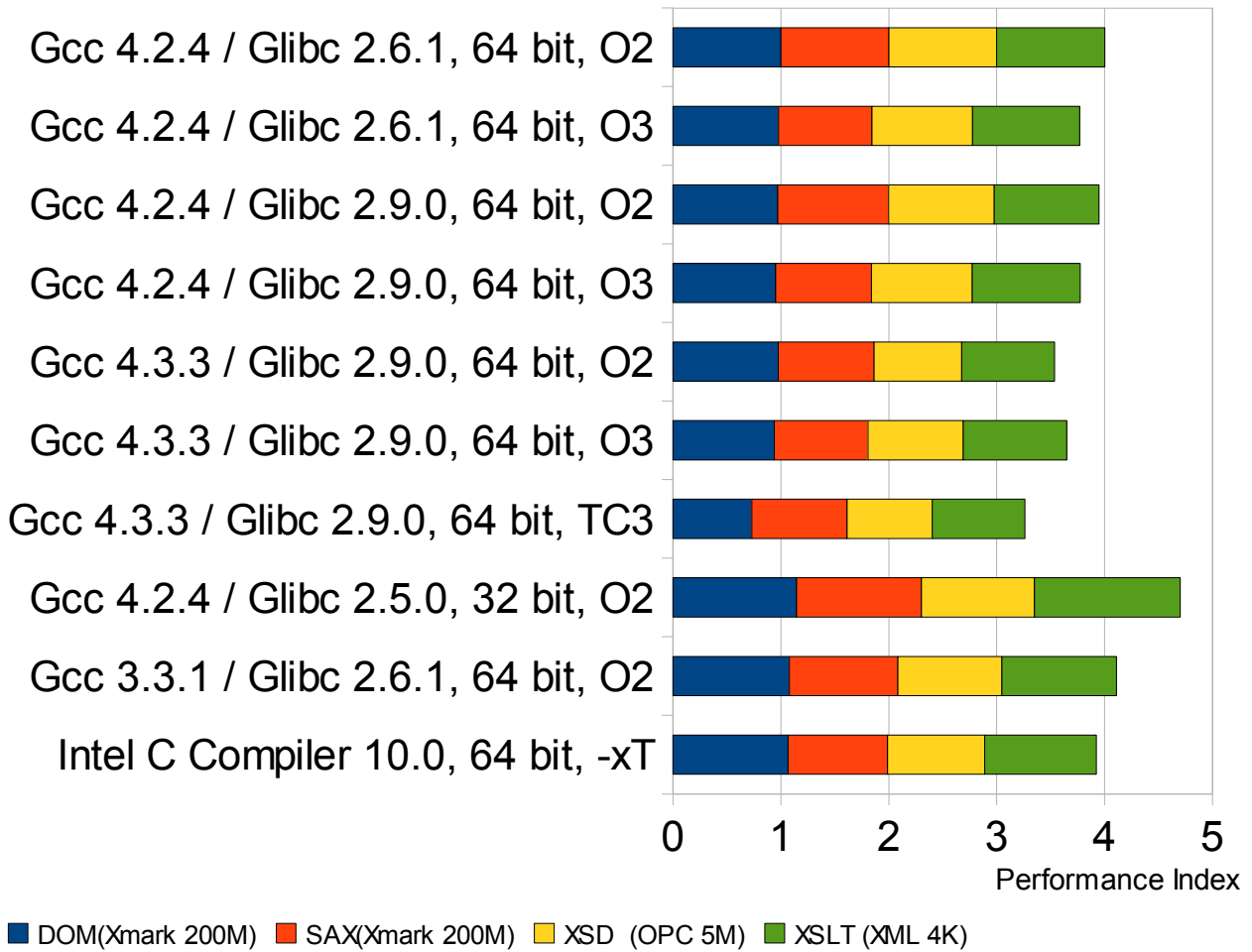
Overall Charts



- SAX Parser
- DOM Parser
- DOM Engine
- XSD Engine
- XSLT Engine
- XML Security
- Memory (General)
- Memory (SAX)

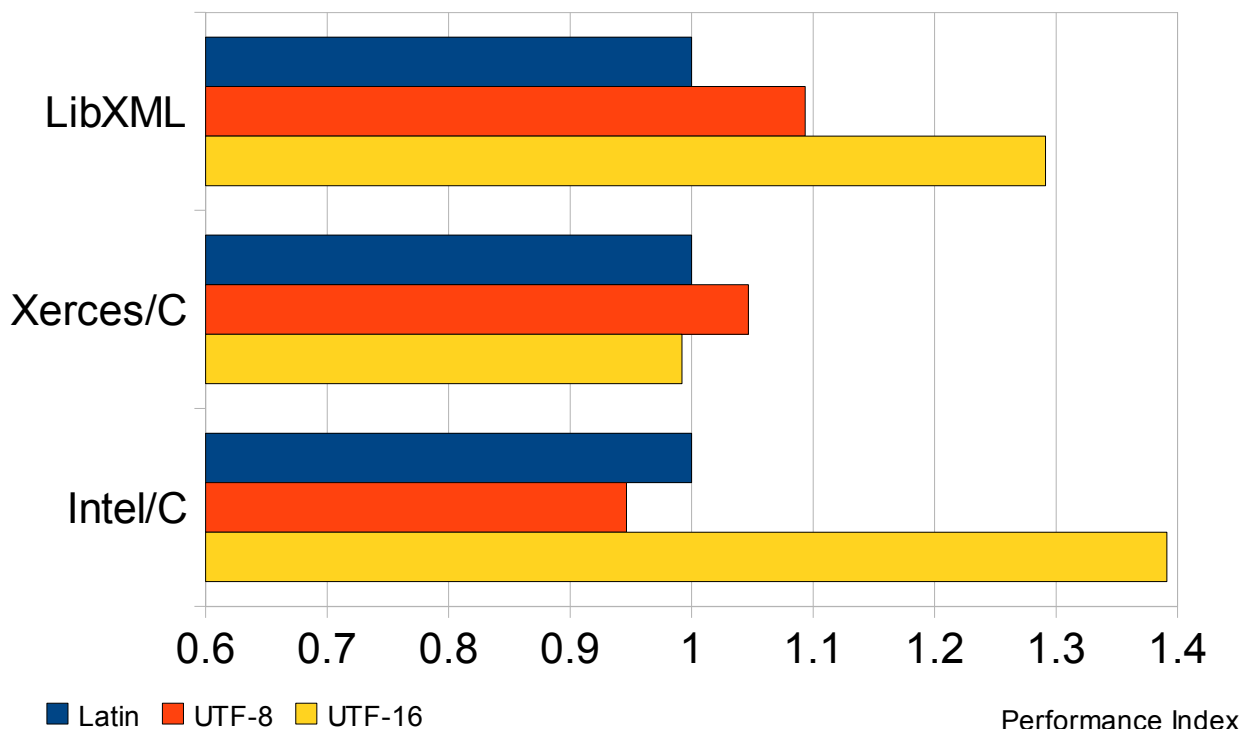


Compiler Benchmark



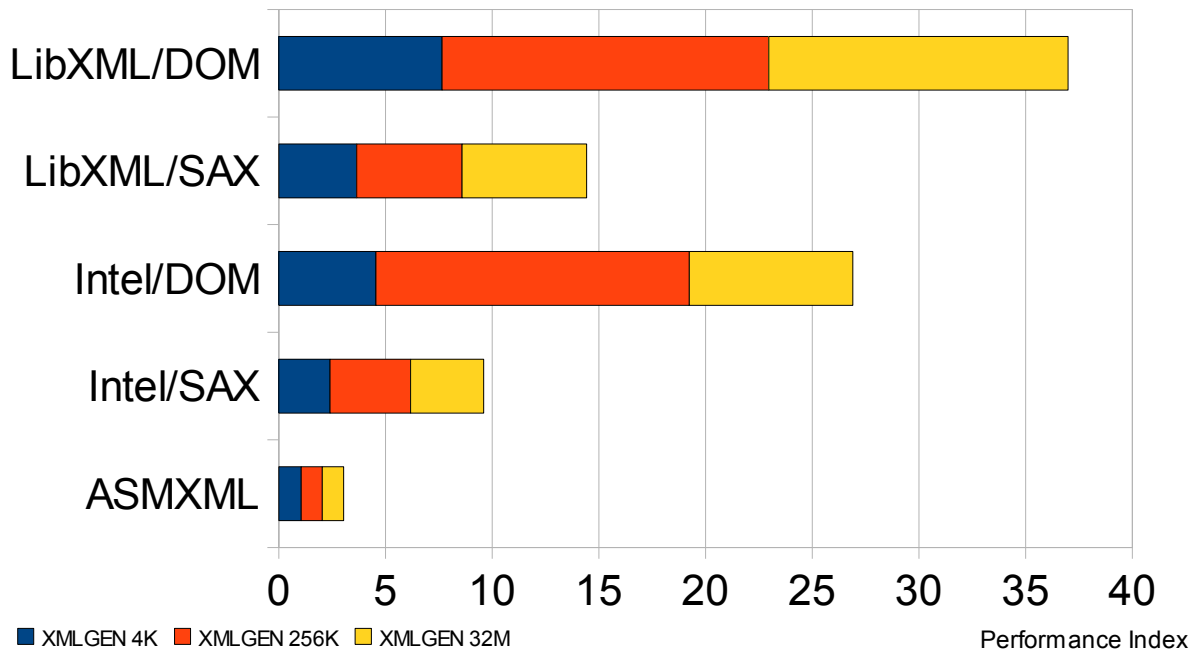
O2: -O2 -march=nocona **O3:** -O3 -unroll-loops -mfpmath=sse,387 -march=nocona
XT: -O2 + SSE3 **TC3:** O3 + TCMalloc memory allocator (by Google)

Encodings Benchmark



ASXML Performance

ASXML is a very basic parser implemented in pure assembler language.



Parsing in Deferred Mode

If Deferred Mode is used Xerces/J and Intel/J postpone creation of most DOM objects until they are requested using DOM API.

