A Quantitative Comparison of Binary XML Encodings

Wolfgang Hoschek

Lawrence Berkeley National Laboratory
whoschek@lbl.gov

GridWorld / Global Grid Forum 15, Boston, Oct 2005
Outline

• XML established for interop. data exchange
• Grid and Web services
  – Large numbers of XML messages at high frequency
  – XML serialization & deserialization bottlenecks
    • Markup verbose in size
    • Expensive to produce and parse
      ➢ Restricts XML adoption
  – Alternative: Binary XML?
• Novel binary XML encoding (*bnux*)
• Quantitative evaluation
  – Production-quality XML and Binary XML toolkits
  – Tree and streaming deserialization, serialization, compression
Bnx Properties

- Faithful to XML
  - General purpose
  - Preserves all information without loss or change
  - Preserves W3C XML InfoSet and W3C Canonical XML
- Self-contained
  - No external resources required (e.g. no schema)
- Tree and streaming deserialization mode
  - For end user applications and filter pipelines
- Tunable for either performance or size
  - High vs. low bandwidth networks
  - Moderate compression via simple FAST means (tokenization)
  - Additional strong GZIP (ZLIB) compression (optional)
- Production quality implementation
  - Serialization of XOM XML object model (~DOM)
  - http://dsd.lbl.gov/nux
- **Serialization**
  - Extract unique symbols (strings) via hash table
  - Sort symbols by frequency (top N)
  - Encode symbol table as zero terminated UTF8
  - Encode each XML node as binary token, with compact pointers into symbol table (Vint)

- **Deserialization**
  - Decode UTF-8 symbol table
  - Decode tokens, hand info to app handler (e.g. type, prefix, name, URI)
• **Workloads:** 60 distinct test document flavours
  – Wide range of real-world documents
  – File size
    • Small (0.2 - 1 KB), medium (1KB - 4 MB), large (4 - 100 MB)
  – Documents
    • Messaging-oriented, record-oriented (database), narrative text
    • E.g. WSDL, SOAP, RSS, ATOM, DB, Shakespeare, P2PIO, …
  – With and without namespaces, attributes, whitespace, repetitions, nesting depth, …

• **Memory-to-memory tests (no I/O perturbation)**

• **Setup**
  – Sun Java 1.5.0_04, server VM, PentiumIV Xeon@2.8 Ghz, 2GB memory, Linux 2.4.20
  – xom-1.1, nux-1.4, saxonb-8.5.1, java.net-fastinfoset-CVS (ISO/ITU), xerces-2.7.1 for SAX and DOM, woodstox-2.0.2 for STAX
# XML Models Studied

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xom-NV</td>
<td>XOM via SAX/Xerces with XML verification performed by Xerces. <strong>Comparison baseline</strong> for tree speedup.</td>
</tr>
<tr>
<td>xom-V</td>
<td>Same as xom-NV except with XML verification performed by XOM instead of Xerces. More expensive than xom-NV</td>
</tr>
<tr>
<td>saxon</td>
<td>Saxon tinytree model with shared namepool (via SAX/Xerces)</td>
</tr>
<tr>
<td>dom</td>
<td>Xerces Document Object Model without “deferred node expansion”</td>
</tr>
<tr>
<td>bnux0</td>
<td>Bnux binary XML with XML verification; no GZIP compression</td>
</tr>
<tr>
<td>bnux0-NV</td>
<td>Same as bnux0, except that PCDATA verification is omitted</td>
</tr>
<tr>
<td>bnux1</td>
<td>Same as bnux0, plus weak GZIP compression at level 1</td>
</tr>
<tr>
<td>bnux9</td>
<td>Same as bnux0, plus strong GZIP compression at level 9</td>
</tr>
<tr>
<td>fi0</td>
<td>FastInfoset binary XML with default indexing (via SAX)</td>
</tr>
<tr>
<td>fi1</td>
<td>FastInfoset binary XML with “full indexing” feature (via SAX)</td>
</tr>
<tr>
<td>xom-NNF</td>
<td><strong>Streaming</strong> XOM via SAX/Xerces with NullNodeFactory handler, throwing away all data, building an empty tree instead. <strong>Comparison baseline</strong> for streaming speedup.</td>
</tr>
<tr>
<td>bnux0-NNF</td>
<td>Same as xom-NNF except that bnux0 is used; no verification</td>
</tr>
<tr>
<td>fi0-NNF</td>
<td>Same as xom-NNF except that fastinfoset is used</td>
</tr>
</tbody>
</table>
bnux0 Speedup vs. Compression Factor

- bnux0-NNF (stream deser)
- bnux0-NV (tree deser)
- bnux0-NV (tree ser)
Tree Deserialization Speedup vs. Compression Factor
CRD

Streaming Deserialization Throughput

Streaming Deserialization Throughput by File

- xom-NNF
- bnux0-NNF
- fi1-NNF
- fi1-NV
- xom-NV
- bnux0-NV

Normalized Throughput [MB/s]
Conclusions

- **Standard textual XML satisfactory for many commodity use cases**
  - But requires non-intuitive configuration wizardry
- **Binary XML significantly faster for demanding data-intensive use cases**
  - Small to medium sized messages: 5-20x
  - Large variance stemming from document flavour
- **Moderate compression via tokenization (1-5x) is fast**
- **Strong compression via GZIP (5-50x) is too slow**
- **Trading efficiency for standardization?**
- **Input for potential W3C standardization?**
### Backup slides

Accumulated Frequencies of Top 16 symbols

<table>
<thead>
<tr>
<th>File</th>
<th>Compression Factor</th>
<th>Streaming Deser Speedup</th>
<th>XML File Size [KB]</th>
<th>Unique Symbols [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>p2pio.xml</td>
<td>1</td>
<td>21.4</td>
<td>0.2</td>
<td>92.6</td>
</tr>
<tr>
<td>map-spain.svg.xml</td>
<td>1.2</td>
<td>3.2</td>
<td>258.4</td>
<td>81.5</td>
</tr>
<tr>
<td>i18n-german.xml</td>
<td>1.2</td>
<td>3.8</td>
<td>6.1</td>
<td>64.1</td>
</tr>
<tr>
<td>ns-soap-req.xml</td>
<td>1.3</td>
<td>18.9</td>
<td>0.5</td>
<td>80</td>
</tr>
<tr>
<td>ns-rss-news.xml</td>
<td>1.5</td>
<td>7.4</td>
<td>10.6</td>
<td>57.3</td>
</tr>
<tr>
<td>romeo.xml</td>
<td>1.5</td>
<td>5.2</td>
<td>228.2</td>
<td>97</td>
</tr>
<tr>
<td>itunes.xml</td>
<td>4.5</td>
<td>12.4</td>
<td>1882</td>
<td>83.7</td>
</tr>
<tr>
<td>romeo10.xml</td>
<td>6</td>
<td>17</td>
<td>2470</td>
<td>0.1</td>
</tr>
</tbody>
</table>

**Acc. Frequency [%]**

![Graph showing accumulated frequencies of top 16 symbols](image)
Streaming Deserialization Speedup by File

- xom-NNF
- bnux0-NNF
- fi1-NNF
- xom-NV
- bnux0-NV

File

- p2pio-noindent.xml
- p2pio.xml
- disclaimer.xml
- xinclude.xml
- articles.xml
- ns-soap-req.xml
- ns-order.xml
- ns-soap-spec.xml
- ns-invoice1a.xml
- i18n-chinese.xml
- README.xml
- i18n-korean.xml
- i18n-japanese.xml
- i18n-russian.xml
- i18n-thai.xml
- books.xml
- ns-docbook-title.xsl
- ns-xforms-spec.xml
- ns-ubl-noindent.xml
- ns-ubl.xml
- ns-rdf.xml
- ns-atom.xml
- wsdl-counter.xml
- ns-invoice1.xml
- i18n-german.xml
- wsdl-ogsi-bind.xml
- taglib.xml
- ns-mathml.xml
- blog-changes.xml
- ns-rddl.xml
- wsdl-ogsi.xml
- ns-invoice10.xml
- ns-invoice100.xml
- db10.xml
- db100.xml
- db1000.xml
- db10000.xml
- romeo.xml
- map-spain.svg.xml
- ns-openoffice.xml
- ns-docbook.xml
- xpdf-xml1.1.xml
- teams-noindent.xml
- teams.xml
- auction-0.01.xml
- auction-0.1.xml
- auction-0.5.xml
- auction-1.0.xml
- wurfl.xml
- weblog.xml
- factbook.xml
- xpdf-css2.xml
- auction-0.5.xml
Tree Serialization Speedup by File

- xom-V
- bnux0-NV
- bnux1
- bnux9
- fi1
- fi0
- saxon
- dom
Tree Deserialization Throughput

Tree Deserialization Throughput by File

Normalized Throughput [MB/s]

File

- xom-NV
- xom-V
- bnux0-NV
- bnux0
- bnux1
- bnux9
- fi1
- fi0
- saxon
- saxon
- dom
- dom
Tree Serialization Throughput by File

- xom-V
- bnux0-NV
- bnux1
- bnux9
- fi1
- fi0
- saxon
- dom

Normalized Throughput [MB/s]
Tree Serialization Speedup vs. Compression Factor

- **bnux0**
- **bnux1**
- **bnux9**
- **fi1**
Backup

**Streaming Deserialization Speedup vs. XML File Size**

- **xom-NNF**
- **bnux0-NNF**
- **fi1-NNF**
- **xom-NV**
- **bnux0-NV**

**Tree Deserialization Speedup vs. XML File Size**

- **bnux0-NV**
- **bnux1**
- **bnux9**
- **fi1**
- **fi0**
- **saxon**
- **dom**

**Tree Serialization Speedup vs. XML File Size**

- **bnux0-NV**
- **bnux1**
- **bnux9**
- **fi1**
- **fi0**
- **saxon**
- **dom**
STAX vs. SAX Streaming Deserialization Throughput for FastInfoset & XOM

Normalized Throughput [MB/s]

File

- fi0-NNF
- fi0-NNF-stax
- fi1-NNF
- fi1-NNF-stax
- xom-NNF-stax-wood
- xom-NNF-stax-sun
- xom-NNF

Graph showing normalized throughput for various files with different serialization methods.