

Benefits of Alternate XML Serialization Formats in Scientific Computing

Workshop on Service-Oriented Computing Performance 2007

Jaakko Kangasharju Sasu Tarkoma

Helsinki Institute for Information Technology

June 26, 2007

Blockers to XML Adoption

- XML widely adopted, but some application areas reluctant
- Size of XML documents issue where compact data representation needed
- Efficiency of text processing concern in high-performance systems
- Particular problem in data marshaling and unmarshaling
- These issues lead to binary formats based on some XML information model, “binary XML”
- XML Binary Characterization (XBC) Working Group (WG) charted binary XML space, defined desired properties

Improving Compactness

- XML highly redundant, XML documents very large
- Generic compression works well, but adds processing and destroys structure
- Binary formats use streaming serialization with **tokenization**, string compression at XML level
- Direct input and output between data model and bytes
- Schema usable for additional compactness

Improving Other Properties

- Processing Efficiency usually improved, state-of-the-art binary formats faster than state-of-the-art XML parsers
- Depending on processing bottleneck, Compactness may dictate Processing Efficiency
- Binary formats usually include *Embedding Support* and *Specialized Codecs* for more efficient data representation
- Binary format may include indexing for *Accelerated Sequential Access* or *Random Access*

Efficient XML Interchange

- W3C chartered Efficient XML Interchange (EXI) WG to standardize alternate format
- EXI WG solicited contributions, measured Compactness and Processing Efficiency
- Selected format called Efficient XML as basis, features from other formats will be considered
- Current measurements and first draft of format being published

- EXI based on **grammars** with associated **event codes** for productions
- Basic grammar derived from XML specification for arbitrary XML
- Basic grammar modified when new content discovered
- Schema can be used to derive grammar for more compact encoding of conforming documents
- Additional compression, based on splitting event codes into streams, available

EXI Grammar Example

StartTagContent:

EE	0.0
AT(*) StartTagContent	0.1
NS StartTagContent	0.2
SE(*) ElementContent	0.3
CH ElementContent	0.4

ElementContent:

EE	0
SE(*) ElementContent	1.0
CH ElementContent	1.1

All codes encoded in minimal number of bits

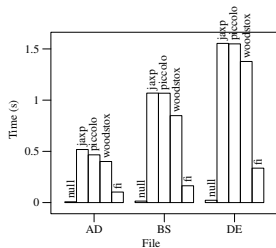
Satisfied Properties

- EXI satisfies properties identified by XBC
- Support for *Specialized Codecs* included, can be used for *Embedding Support*
- *Explicit Typing* not specially supported, need schema or `xsi:type` for type information
- *Random Access* possible with proposed **Indexed Elements** feature
- *Memory-Mappable* possible with proposed **Byte-Aligned Mode**

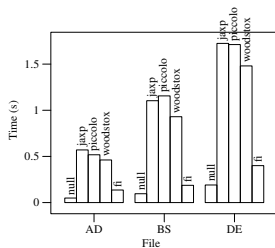
Experiment Description

- Two experiments, one on parsing, one on floating-point decoding
- Parsing experiment: Fast Infoset against high-performance XML parsers on MAGE-ML data
- Decoding experiment: Xebu with XAS type layer against default XML implementation on Seismic sensor data

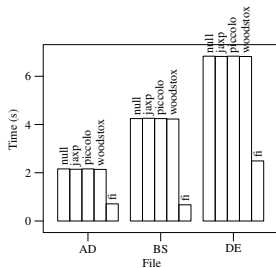
Parsing Experiment



Memory
(3 GB/s)

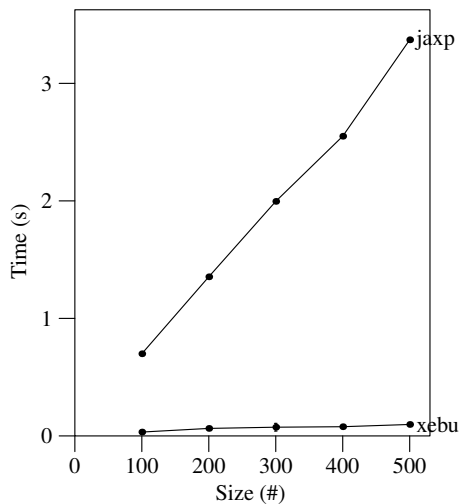


Loopback
(460 MB/s)



100 Mbps
(10 MB/s)

Decoding Experiment



Size	JAXP (ms)	Xebu (ms)
100	700	33
200	1350	64
300	2000	74
400	2550	78
500	3370	97

Conclusions

- Binary format not to replace XML but extend XML's applicability
- W3C as the keeper of XML in unique position to produce widely-acceptable format
- Main beneficiaries mobile and scientific communities
- Support for typed data crucial, EXI provides that

Thank You

- First draft of EXI format published soon, comments on public-exi@w3.org
- EXI home page <http://www.w3.org/XML/EXI/>

Questions?