On XML Optimization

Mark Nottingham and David Orchard, BEA Systems
mark.nottingham@bea.com, david.orchard@bea.com

Ever since it was conceived, XML has suffered the perception – justified or not – that it presents serious performance problems compared to other formats. As a result, there have been a number of proposals made over time, each claiming its place as The Solution To XML Performance.

Unfortunately, there has been very little explanation of the tradeoffs of various techniques, the qualities of and reasons for their performance advantages (or disadvantages), and above all use cases and requirements. As the W3C considers the possibility of work in this area, we are very concerned that due consideration will not be given to these matters, resulting in a failure to address real user needs.

As a result, BEA Systems believes that the community needs solid criteria for making a decision and the benefit of real experience, rather than more choices. This position paper captures our thinking on the requirements that are driving use cases, the fundamental techniques and their associated tradeoffs, and issues that should be considered. We intend to evolve it over time, and we believe that the result can be used as the basis of evaluation for any proposal.

Why Optimize XML?

A number of use cases motivate alternate encodings of XML. Here, we attempt to identify their underlying issues. Often, a use case will involve a number of underlying requirements (e.g., a high-volume message processing system will be storage, bandwidth and processor-constrained, relative to its expected load).

1.) Storage/Bandwidth Constrained - XML Is Too Big.
The cost of storing and transporting XML markup is often difficult to justify. For example, applications with extremely limited bandwidth (or where a usage charge is associated with network traffic) may avoid XML because of its verbose nature. Likewise, applications that need to send large numbers of messages across a larger link may make the same decision, because they are bandwidth-constrained in a different dimension. The same arguments apply to applications that have either limited amounts of storage available, or very large amounts of data to store.

2.) Processor Constrained - XML Is Too Expensive To Parse and Generate.
The overhead of generating and parsing a complex, dynamic and text-based data format often limits the capacity of systems with limited available processing power. Often, this deficiency is seen in terms of latency and throughput; small and other limited devices may not have the power to parse and validate XML in a way that maintains the user experience, while high-volume message processing systems may not scale to meet expectations.

3.) Embedded Binary Data – XML Doesn't Accommodate Other Formats.
XML is a textual markup language. As a result, binary formats are not easily accommodated by XML, even though they are widely deployed and will be for the foreseeable future.

XML does allow for encoding such content into base64 and hex encodings but these techniques cause problems #1 and #2 to arise. When payloads are large, such as embedding a base64Binary-encoded image in a document, this becomes inconvenient at best, and in many cases limits system performance. For example, systems that act as intermediaries may only wish to process a small part of such a message. As such, this issue could be seen as a subcase of problems #1 and #2 above, but it is so large in some people's minds that it deserves consideration on its own. Additionally, the problem of including binary data and the problem of reducing markup size and processing requirements are treated orthogonally in some solutions.

4.) Random Access - XML is Too Monolithic.
XML requires that the entire document be available before it is considered complete, if one wishes to know that it is well-formed and/or valid. This means that XML also has relatively high memory requirements because of the need to buffer arbitrary chunks of structured data. Additionally, the use of syntactic mechanisms like XML Base and XML Namespaces means that portions of an XML document that would otherwise be self-contained must be evaluated in the context of the greater whole. All of this disadvantages applications that wish to quickly locate and/or update information in an XML document.

Potential Technical Tradeoffs

Candidates for improved encodings of XML must necessarily strike a balance between the benefits that the XML 1.0 format brings and the desired performance improvements. If no perceptible benefit is lost by a particular scheme, we argue that such techniques be incorporated into XML 1.0, rather than into an alternate encoding.

The properties and constraints of each solution that we believe are the most important are:

Self-Describing – It is possible to understand an XML 1.0 document without external context. Encodings that require external context will not be able to be understood by any consumer not foreseen and accommodated for by the original author (such as an intermediary, or alternate use of data).

Evolvable and Extensible – XML formats can be designed to evolve gracefully over time, whilst retaining backwards and forwards compatibility. Encodings that do not allow for evolution lose this key benefit of XML.

Isomorphic to XML – Lossy encodings are unable to reconstruct (or "round-trip") the XML Infoset and may require separate APIs and toolsets to accommodate, losing the benefits of XML's wide deployment and support.

Streamable – XML can be emitted as it is produced, reducing memory requirements considerably.

Human-Readable – XML, as a text format, is easy for people to read and understand with common, non-proprietary tools (i.e., a text editor), and is simple to modify with those same tools. This encourages the adoption of XML-based formats and aids greatly in debugging and ad hoc uses of them. For example, XML 1.0 provides for encoding characters in an easy-to-remember entity notation; many proposals for alternatives do not provide this facility.

Simplicity – XML's simplicity, with few encodings, has helped lead to an enormous variety of implementations. The rigorous application of the simplicity constraint – "what is the minimum necessary to declare victory" – has led to the property of widespread interoperability.

Available Techniques

As noted, there have been a number of proposals for alternate encodings of XML, ranging from Gzip-based compression to ASN.1 encoding. The XML Protocol Working Group is also active in this area with the MTOM Working Group Draft.

BEA has been tracking these candidates closely (as doubtless other vendors have). We have also been quite active in this area, including our joint effort in MTOM's precursor, PASWA, and our part in bringing that work to the W3C. We have also developed an alternate encoding of XML that we call TokenStream. The result of a multi-year effort, it has matured to something we feel may be appropriate in this space, once the community's requirements and direction are clear.

In our experience, candidate solutions can be characterized by the underlying techniques used to improve different performance characteristics by making a variety of tradeoffs. Below, we provide a brief evaluation of a number of these techniques. Once again, a particular solution may combine several techniques, and this is not a complete enumeration.

Text Compression – Traditional compression techniques reduce the size of a document, thereby reducing storage requirements, bandwidth usage on the network, and improving network latency. However, compression often involves significant processor costs for both writing (sending) and reading (receiving). In a network scenario, this often dwarfs the benefits of compression, especially for small documents.

Typically, compression also makes XML even more monolithic; in most schemes the document is opaque until the entire document is decompressed. Some schemes offer selective text compression to improve the ability to access arbitrary parts of the document.

Length Encoding – Encoding the length of elements, attributes and other structures allows implementations to efficiently access arbitrary parts of the document because they do not have to touch every byte during parsing. This efficiency comes at the cost of the ability for humans to easily modify documents and the potential errors arising from duplication of information

Tag Dictionaries – Replacing XML QNames with more homogenous, fixed-length identifiers (e.g., integers) improves processor efficiency and reduces the size of documents. However, it reduces human-readability and impairs random access.

Selective Reordering – Flexibility in the serialization order of the document's components allows some applications to achieve considerable efficiency gains if they only need to process parts of the document. For example, if an element contains a large chunk of encoded binary data (e.g., an mp3 or JPEG), it could be serialized at the end of the physical representation, allowing applications the ability to access the Infoset without reading those bytes.

This technique is particularly useful in networked scenarios. Depending on its implementation, it may trade off human readability and editability, and may affect streamability.

Selective Recoding – Some data types common in XML, particularly base64Binary and hexBinary, are particularly inefficient in both size and processor overhead. By opportunistically recoding sections of the document that match the canonical form of these types, benefits can be realized in both of these dimensions. This technique comes at the price of human editability (as many text editors can't display binary).

Type-Aware Encoding – When type information is known, it can be used to re-encode a document with those types in canonical forms that are often fixed-length (e.g., integers), efficient (e.g., base64Binary and hexBinary) and easy for implementations to coerce into native data structures. This enables very low processor overhead and usually results in size reduction as well, at the cost of human readability and editability, and may lose information (because the canonical representation is not always that in the Infoset).

Schema-Aware Encoding – Schema-specific formats can be generated based on the particular structures therein, requiring the same version of the schema to generate and consume a particular document. This technique results in size savings as well as considerable gains in processor efficiency, but is not self-describing, poses problems when evolving the format, and generally is not human-friendly.

How To Measure Candidate Solutions

Candidates for an alternate encoding of XML will be measured by many criteria not captured above. For example, if they require substantial changes to existing XML APIs – or entirely new APIs – it's unlikely that they'll see much adoption. Flexible solutions that accommodate a variety of use cases will be more successful than rigid ones, and simple techniques will be more likely to produce results for broad ranges of applications. We trust the market and the standards process will filter out prospective solutions that are unsuitable on these grounds.

However, we do believe that the selection of candidates needs guidance in determining both their applicability and relative merits. In particular, the aforementioned tradeoffs must be made in the light of the use cases that they enable; it does not make sense to lose many of the benefits of XML to accommodate a small number of specialized (and perhaps imaginary) use cases. Conversely, if there are significant use cases that would otherwise be unable to leverage XML, considerable tradeoffs may be justified. Additionally, while the W3C represents many interests and organizations, the selection of use cases must also consider interests external to it, such as those of the IETF in its use of XML.

There is also a need to objectively quantify the relative merits of candidates, which to date has not been done adequately or on even footing. We believe that a solid benchmarking effort, with a diverse workload and a number of measurements, is required to provide unbiased data for a reasoned comparison of candidates.

At a minimum, the workload should encompass a range of document sizes, documents dominated by content and by markup, with homogenous and heterogeneous markup, and with a variety of encoded data therein. Each test should consider document size, processor requirements for processing and generation, memory requirements for the same, and partial and random access characteristics.

We also observe that extensibility and self-description are two of the key constraints in the Web architecture and REST architectural style. The loss of either of these two constraints would have significant ramifications, and therefore the benefits of doing so must be correspondingly high. Without proven benefits, we predict a lack of community endorsement.

Recommendations

BEA believes that a standard in this area would be beneficial, and that the W3C – with its history of technical strength and well-understood IPR policy – is the appropriate forum for such work. However, due care needs to be taken in order to assure a successful outcome; in particular, we do not feel that a Workshop is sufficient to gather appropriate use cases and requirements, nor is it adequate to accept measurements at their face value.

We propose the following process for a Working Group:

1. Collect real-world use cases
2. Determine which use cases should be / must be met
3. Gather candidate solutions
4. Evaluate candidates against:
    a. Use cases covered (more complex than simple yes/no)
    b. Tradeoff of properties
    c. Measurable benefit (properties) from benchmarks
    d. Integration with the Web architecture and its constraints
5. Adopt a candidate as the basis for a Recommendation

In particular, the use cases, requirements and benchmark suite should be Working Group deliverables, in addition to the selected format.

We would also expect the Working Group to pay special attention to architectural questions. In particular, it must be possible to easily identify formats using the new format, and it must be possible to interchange them with those encoded using XML 1.0. Additionally, due consideration should be given to what abstraction the serialization chooses to model (e.g., Infoset, PSVI or XML Query Data Model), as this may affect existing (e.g., XML Digital Signatures) as well as future Web specifications considerably.

Finally, although there are a number of candidates available, we particularly urge that the W3C should only develop one such alternate format that addresses the most important, but not necessarily all, use cases. Recommending or even enabling too many encodings of XML violates the simplicity constraint and will surely rob the community of much of XML’s derived benefits, because gaining the important property of interoperability will become considerably more difficult.