Expanding the Web: Is XML Sufficient?

Sharmistha Chatterjee
Helsinki University of Technology
schatter@cc.hut.fi

Abstract

The first elements of the web infrastructure includes HTML tagging, simple hypertext linking and hardcoded representation. It had scalability problem when Internet became more popular and the number of applications provided by the web increased steadily. This scaling limitation gave the birth of XML to suffice to the huge demands of the web. This paper explores the cost benefit analysis behind the technologies used for XML processing. It also presents various optimization techniques that can be applied to normal schema-based XML document and other encoded versions of binary XML. In this context it finds the drawbacks of using only text or binary XML in the web. This helps the paper to focus on its main objective for justifying the reason for use of a combined form of XML in all applications for increasing efficiency.

KEYWORDS: XML, parser, binary, XQuery, processor.

1 Introduction

In today's world of Internet communication whether it is SOAP based system-to-system commmunication or REST model of distributed computing application or business-2-business application, all makes use of XML widely. This has been possible due to huge extensibility of XML. XML has provided many advantages to the web world due to its flexibility, expressiveness, and platform-neutrality. The time consuming parsing technologies of text XML fails to satisfy the performance requirements of XML-based applications and their computing infrastructure [11]. Moreover the documents when used in electronic format fails to embed binary formats of resources such as fonts, images, and video. Research on text based XML parsing succeeded in improving the performance of parsers to some extent by generating special bytecodes [11]. These bytecodes are designed to reduce the time of parsing and validation.

Standard textual XML applications use XQuery/XPath processing technique to decrease processing time. This is brought about by efficient Schema and XQuery processors. XQuery is a declarative language used for exchanging and querying XML data [8]. It contains prologue declarations, function, variable, XML element construction, and advanced iteration constructs depending on the type of XML application [16]. XQuery/XPath data model uses XPath, an expression and addressing language for selection of nodes, integers, strings, and booleans, and sequences from XML's tree type representational model. Inspite of application of suitable techniques to text XML it is unable to meet the performance expectations of all XML based applications.

1.1 Growth of Binary XML

The limitations of text XML in parsing huge XML data gave the foundation to the growth of binary XML. Binary XML has been primarly designed to improve parsing and processing capability of text XML. It is a format that does not follow XML specification or direct interoperability with text XML, but maintains a practical resemblence to text XML [20]. This resemblence helps to represent almost all types of documents of text XML in binary XML. The objective behind the proposal of binary XML is to represent XML documents in such a manner that it can be generated, serialized, transmitted and interpreted effectively from one end of application to the other end of application.

Binary XML also allows conversion of file formats from text XML to its equivalent binary representation and vice versa for easy storage and transmission. As binary XML has been developed as an improved version of text XML, it supports advanced properties, the most important among which are reduced size, navigability, presence of more amount of dynamic memory at run time, low redundancy and efficient update. With the idea of an alternate representation, the obvious question is that which format is more suitable not only in efficient processing but also in low cost implementation. The paper tries to explain that a possible combination of the two formats can yield best performance by comparing and contrasting benefits and drawbacks of each format.

The paper is organised as follows. Section 2 cites some applications of binary XML in different environments. Section 3 gives an outline of the basic properties of XML that any efficient representation of XML should support. Section 4 discusses about the most important techniques used by binary XML for improving application processing time. Section 5 presents some of the limitations of binary XML. Section 6 brings into picture the most popular measures used with text based XML documents like parallel bit-streaming, XQuery Processors and XML Screamers. It also tries to estimate the results if these processes are applied to binary XML. Section 7 concludes the topic by justifying that a combination of binary and text XML can be the most effective solution.

2 Applications of Binary XML

The growth of binary XML and its advantages over text XML in high speed parsing, compressed representation and updation efficiency has prompted its use in various applications. Binary XML then came to be introduced in various

XML applications of television (TV), XHTML, SVG (Scalable Vector Graphics) and XForms (XML data model).

XML in television and mobile XML applications in TV consumes huge bandwidth and puts a limitation on the number of XML using channels that can be broadcasted [20]. Also change in broadcast systems in TV require changes in XML schema that is costly due to time consuming parsing process of text XML. On the other hand binary XML is resilient to XML-schema changes. The drawbacks of text XML over binary XML propmted to use binary XML in TV.

Multimedia applications Binary XML reduces overhead significantly when it is used in representing multimedia XML documents [2]. Audio-video data are transmitted by network or file streaming. Their encoding in suitable binary format boosts transmission time. Geometric data contains large floating point numbers. Binary encoded multimedia XML representation (X3D) supports simpler versions of geometric data in compressed formats by means of Specialized Codecs. This helps in optimal encoding and decoding.

Seismic data Seismic and other huge datasets represented in XML consists of large floating point numbers [20]. Such representation is not efficient due to their large size and the time taken by XML parsers in converting them to character representations. One way to reduce the application processing time is to compress the data in codec format. The other way is to combine them in text and binary XML. Text XML embeds seismic data control information and binary XML represents floating point arrays in a attachment using XOP (XML-binary Optimized Packaging) [19], [20]. Applications can directly use the parsed data, or obtain its base64 binary character representation from the XOP package.

Business applications Binary XML in small, medium or large business processes is capable of enhancing random access, search and updation of any XML document [20].

Sensor Processing and Communication The application of XML for sensor processing languages demands small inexpensive encoded packets. It is needed for simple encoding-decoding purposes in safety-critical environment. Binary XML in sensor communication can process sensor commands and retrieve reports with enormous speed without using much power and battery life of sensors [20].

3 Properties

XML is a markup language that has some important properties which affects the format's utility in different applications. To make XML format universally acceptable in different devices and platforms with high efficiency the W3C working group took into consideration some properties of XML represenation [21]. The following section discusses about the the most important properties and the principal design issues involved in attaining those properties.

3.1 Processing Efficiency

Processing efficiency is determined by the total time required to generate the byte codes from the XML data model, the parsing speed and the time incurred in data binding [21]. Data binding creates the application data model in XML from the data contained in the format. For instance data binding during serialization is directed at retrieving the actual document from the bytecodes of the serialized data. Taking into account the principle objective of reducing the processing time of XML based applications, different schemes have been proposed and researched both on standard and other encoded versions of XML. While some of the techniques are successful in gaining the desired objective, research [1], [2], [7], [12], [13], [20] on different types of applications indicate that a trade off factor is associated between the technology, implementation scenario and cost [21].

The most popular optimization procedures for rapid processing of XML applications are serlialization and streaming using binary XML. Serialization is a technique by which an object's bits are converted to some format (e.g bytes) and that state is stored in a storage medium [4]. This procedure helps in direct parsing the bytes without having transformation in any intermediate form. The other useful technique of streaming allows transfer of XML data over the Internet so that it can be processed steadily in a continuous stream.

For standard textual XML documents efforts have been made in increasing the efficincy of the XQuery and XPath processing by using efficient Schema and XQuery processors. The development and application of XPath and XQuery techniques have been successful in decreasing the time in evaluating the XQuery. Comparative studies indicate after application of the respective processing techniques on both text and binary XML, binary XML have a higher edge over text based XML in terms of processing efficiency [21].

3.2 Small Memory footprint

Memory footprint is given by the size of the processor processing binary or text XML [21]. The different XML formats require different types of XML porcessors depending on the number, complexity of features and amount of data given to the processors. The platform, programming languages used together with the XML format serve as major factors in specifying the type and footprint of XML processor to be used.

Memory footprint plays an important role in database, data caching and messaging applications. These applications should have low memory footprint for efficient storage and retrieval [14]. High Memory footprint leaves a negative effect on CPU cache utilization for parsers like Xerces [12].

3.3 Space Efficiency

Space efficiency is determined by the amount of dynamic memory needed to decode, process, and encode a XML format [21]. Space efficiency is one of the most important criteria in XML processing as a format should be processable not only in desktop Pcs or servers, but also in small devices like mobile handsets where power and memory are limited. This requirement has further neccessitated the development of alternate processors and alternate processing mechanisms whose memory footprint are smaller and which are capable of providing the amount of dynamic memory required to run the application in the specific platform.

One sort of optimization technique applied on text based

XML documents for attaining space efficiency is parallel bit streaming methodology. [13]. However parallel processing incurs more cost as it is directly linked with CPU cycles of the processor and the number and type of processors used.

3.4 Compactness

Compactness is determined by the size of XML representation residing in memory or by the size of the stored XML format. This size is reduced by including very little extraneous information. The commonly used compression methods are lossy/loss-less, schema-based/non-schema-based and delta-based/non-delta-based compression [21]. While lossless compression helps to recover every single bit of data after uncompression, lossy compression eliminates certain information from the file to attain permanent compression. The other useful compression is the delta compression that represents an updated compressed version of a file than before.

An example of lossless compression of an XML document is schema-based encoding. But the degree of compactness is entirely dependent on prior information of the structure and content of the document. This places limitation on the presence of schema during compaction for successful reconstitution of the original document. The other useful technique is the Delta Compression used in Information Push Services [6]. Push services with XML relies on huge data exchange and communication resources due to verbosity of XML. One application of compressing XML data using Delta procedure is Jdelta compression. It improves the compression ratio most efficiently with XML documents partcularly in limited bandwidth and resource constraint scenarios.

The first and foremost advantage acquired by compaction is in storing large XML documents in a reduced space. It also helps to transmit huge XML documents in least time. It is favourable to compress binary XML as the compression means based on domain-specific knowledge are more powerful than generic compression. Other than obtaining compression with binary XML, significant compression ratio has been attained from hardware-based network compression (e.g. MNP-5) [14]. Its compression and performance outsets compression obtained through binary XML. Moreover binary compression suffers from additional time and CPU overhead required to generate the encoding [14].

4 Techniques on Binary XML

This section of the paper compares and contrasts the various optimization techniques applied on binary encoded versions of XML. It also analyses the factors that led to the development of the techniques, their efficiency and overhead. Further by analysing the mode of binary XML representation, this section tries to justify whether binary XML could be more suitable form of representation over text based XML.

4.1 Binary XML Serialization

When an XML document is serialized processing efficiency derived in transmission and reception of documents is increased compared to normal transmission of huge XML data files [13]. Serialization helps storing the data in a storage medium like file, or memory buffer or transmitted across a network connection link using HTTP.

Binary XML serialization could be used based on binary encoding to yield more compaction to the encoded version. This type of serilaized data are used for storing and using it on socket-based network streams [2]. Serialization uses a technique known as multiple stream serialization. In this technique related documents are grouped and compressed together. If the document has several redundant parts they are treated individually for compression.

Serialization is also used for serializing abstract dataset of XML for creating XOP packages inside an extensible packaging format (like MIME) [19]. This technique helps to optimize XML Infoset (Information Set) by extracting, decoding and placing base64-encoded binary data back into the package. By this it preserves one to one mapping between the XML Infoset and XOP Packages.

The benefits of serialization are that it achieves some degree of compaction and interoperability by using only bytes. The most important advantages gained by developers in using binary XML serialized data is that it requires minimal changes to existing application layers. This ensures minimal cost for implementation of binary serialization procedure.

But there are some limitations with XML serialized data [18]. Any malicious user attempting a denial of service attack (DOS) is capable of sending a continuous stream of XML data to a Web server that will processes the data till the computer falls short of resources. The problem of security attacks like DOS is prevalent with binary serialized data similar to serialization of text XML. Hence it cannot be considered as a secured format of replacement over text XML.

4.2 Binary XML Streaming

Among other means of providing efficiency in space and time one of them is XML streaming that uses XML stream processors to generate dynamic XML data. Streaming of binary XML was introduced to overcome the huge memory management overhead brought about by XML parsers using DOM (Document Object Model) methods. Parsing text XML by DOM parser slows down XML based applications highly as the parser reads full XML data into memory and converts it into an XML DOM object [15]. DOM treats the entire data as a tree structure which incurs heavy expenses while copying and moving subtrees of a DOM tree.

Binary XML Streaming also shows improved performance over event driven text parsers that rely on SAX events (event-based APIs for parsing elements, tags, etc) to construct XML document [15]. These kind of parsers are dependent on application for collection and storage of SAX events. XML documents processed on SAX events are error prone and results in generation of proprietary code. This limits the exchange of data structures between different applications.

With the limitations seen in text XML parsers streaming technology on binary XML came to be known widely due to its several advantages. Stream based processing is capable of generating optimized application specific XML documents through proper selection of portions of XML document. Streaming yields high compression ratio, compression speed, decompression speed, document updation efficiency,

efficiency in incremental processing and querying of documents from database [15].

XML streaming is used in X3D documents and other real time applications in web environments. It finds more usefulness where network bandwidth and storage conditions are limited. During streaming the compression algorithm uses tokenization of XML tags to achieve good compression [12].

4.3 Tokenization

XPath and XQuery uses tokenization for parsing XML documents. A token is a context dependent meanigful string (for example: declare namespace) that affects the speed of XQuery processing based on its type, length and meaning [16]. On binary encoded XML tokenization process eliminates repeated strings. It also achieves compression by representing strings through a single byte [12].

Tokenization can be treated as a powerful compression strategy for binary XML. The advantages include improving parsing time and application performance for applications running under limited network or disk throughput. Proper tokenizing strategies might improve XQuery performance on binary XML in future due to its high compressing power.

However tokenization has computation overhead and hence cannot offer significant performance improvement where the CPU performance is low. The compresion rate in tokenization is closely related to the number of repeating tags and the content of the text [12], [15]. XML documents having more repeated strings and tags attain more efficiency due to parsing than documents which do not have structural similarity. The style and structure of document together with the algorithm is dependent on how fast the document can be processed and streamed incrementally over the Internet [15]. This kind of processing strategy cannot yield efficiency on structurally different binary XML documents.

4.4 Random Access and Accelerated Sequential Access

Binary stream optimizations also support random access through pointers [12]. The pointers allow the parser to progress to any section of the document and retrieve those portions of document with considerable speed. However the process of embedding pointers into stream-based processing affects the speed of retrieval. Hence relevant information regarding the nodes should be properly encoded to avoid heavy computations. At the same time it should leave minimum changes on XML processing APIs. This type of random access apart from navigating and updating a document provides dynamic document management at an affordable cost.

Random access works on streamed binary XML data where pointers hold the value of distance in bytes from the present location to the position of interest in the document. Random access is also provided by XML stream procesing APIs, one of which is Random Access XML(RAX) programming. RAX developed by an organisation named Tarari allows random access to formatted messages and files without parsing [9]. Recent advancements on Tarari's hardware architecture using RAX 4 and Zlib compression (lossless data compression technique that works all hardware platforms and OS) has produced outstanding results with binary XML by reducing the parsing time at the receiver end.

Other than random access, Accelerated Sequential Access is used to retrieve data items [21]. Its functionality differs from random access in lookup time for accessing data items. In contrast to fixed lookup time in random access, sequential access uses search methods in streaming mode, where lookup time is independent of the data items in the document. This results from using an index with an offset value in a XML document that skips over the document until start tag of the next peer element is reached. This ensures that once the application is aware the queried data item is not present in the current element it skips over to the next element.

The performance benefit obtained is calculated from the time complexity used to construct and update the indexes and special strucutures. This sort of random access is absent in text XML. Even though binary XML derive certain benefits out of this this process is entirely dependent on the design of pointers and indices and their size. In some cases the pointer structure may be such that it may not yield significant performance results during search over text XML.

4.5 Fragmentation

The objective behind this technique is to enable processing of small parts of XML documets that are requested by the receiver instead of processing all the data up to the part being requested [17]. The processing technique dependent on processors is similar to streaming process with the only difference of treating smaller fragments independently in arbitary order [21]. The fragments contain the information of the context in scope specific namespaces (and also in xml:base, xml:space, and xml:lang). This context information helps the parser to detect from language/syntax or notation at which point it should start processing the fragment body [17]. The fragment body is often associated with the fragment context specification into a single XML-encoded object through proper encoding or mulipart MIME. The fragment body can also be linked with the specification separately by means of referencing and co-referencing.

Fragmentation facilitates transmission of prioritized independent parts of the document with improved error resilience and access times [21]. Proceesing cost is also minimized when one or more adjacent extracted fragments are stored together. The update operations have become much simpler and less costly due to fragment-level validation compared to previous document-level validation.

Fragmentation decreases application overhead of the receiver by receiving only the requested section of the document. This technique can be treated as very useful in binary XML as it does not impose restrictions to application of other efficient processing techniques. This characteristic feature offers fragmentation to be used during binary XML serialization and streaming to enhance application processing time even further at a reduced cost.

4.6 Schema based optimization

A significant amount of compression can be obtained from schema based optimization. A normal XML document is

driven by XML schema one of which is the DTD (Document Type Defination). It contains in detail the structure, content and semantics of XML documents. Schema-based binary solutions eliminates parts of the Infoset. This helps to exibit good peer-to-peer transmission when the sender and the receiver encode the documents on the basis of same schema [14]. Such encoding and decoding are common with the WSDL (Web Service Description Language). In future it can improve XQuery processing of binary XML by obtaining type information directly from the schema instead of determining at run-time from the incoming stream [12].

This type of optimization is more benefical in peer-2-peer systems that have no intermediaries [18]. As the intermediaries are not supplied with the complete information of the schema, it is not possible for an intermediary to update the decoder on any change in schema structure. This restricts the schema based solutions to a single-platform and gives rise to portability problem [10]. Moreover they are unsuitable for mobile applications with low-power and limited battery where decoding process consumes more power. In addition it creates a strong coupling between parsing and schema [12]. Hence this technique should not be used for large scale optimization of binary XML.

5 Limitations of binary XML

The above techniques when successfully applied on binary XML are able to increase the performance of XML processing [21]. But there are certain limitations.

The use of different representations of binary XML is oriented with the goal and function of the application [10]. As for example, for server based applications speed is of prime factor. Hence parsing/generation time of XML documents should be given prior importance. Web servers and database systems send out data in chunks that should be buffered during parsing. This sort of buffering degrades the scalability and should have proper compression techniques. Side by side with the server applications if the requirements of client and middle-tier applications using XML documents can be analysed they are found to be different. While client-side applications demand faster parsing speed to have minimal rendering time, middle-tier server (like web services) emphasizes on portability of data. The techniques applied on binary XML are directly linked with the purpose, role and platform of application.

6 Methods of optimizing text XML

This section discusses about the most popular efficient processing methods of text XML. It tries to analyse their respective overheads, performance levels, cost of implementation and possible outcomes on applying them to binary XML.

6.1 Parallel Bit Stream Processing

Parallel bit stream XML parsing is noted for rendering space efficiency to XML applications. It transforms byte oriented character data to a set of 8 parallel bit streams, and then validates, transcodes, and forms the lexical stream using parallel bitwise logic and shift operations [13]. The mapping of each character code of XML to a bit value not only reduces the processing amount of information but also reduces the storage cost of storing any intermediate results during parsing. CPU cycles are reduced by applying hash and regular expression matching on parallel bit streams. However the encoding operation may take few cycles depending on the input characteristics of the XML document, and contribute to some extent in performance degradation.

By carrying out the overall processing technique in Single-Instruction Multiple-Data processors, it has shown considerable performance improvement over single byte text processing. It also exibits processing improvements over standard XML parsers like Expat and Xerces. The intra-chip parallelism provided by multicore processors can successfully parse and validate the schema of XML documents parallely without the need of additional resources.

In many applications the space efficiency obtained from XML processors is inversely proportional to processing efficiency [21]. This again leads to a trade off between processing and space efficiency which should be carefully judged from the application perspective and applied suitably. For instance, XML in sensor networks deployed in military battlefield should be proceesed by advanced processors to have miminal processing time, while XML in mobile applications should use run-time memory more efficiently due to its limited amount of memory. Considering this tradeoff more advanced architecture should be designed that will remove this tradeoff factor and make XML processing combine the advantages of processing efficiency and space efficiency.

6.2 XQuery Processors

XQuery processors try to attain optimization in efficient path processing from XML processors and Schema processors [12]. The optimization procedure adds scalability to XML documents. The degree of processing efficiency obtained in XML document or binary encoded versions of it are dependent on type of data received from XML and Schema processors and the parsing algorithm applied to the received data. Parsers on binary encoded versions take into consideration the input document and what type of binary encoding has been applied to it, to generate stream of SAX events. These events identify and extract necessary portions of the document to the end user. Therefore, if the input document is compressed and the schema structure is simple, the binary encoding proceeds quickly and XQuery processors should give satisfactory performance. Other than having processor performance a new framework called self-tuning can be introduced to the overall query processing system architecture [1]. It continuously monitors the incoming queries and adjusts the system configuration accordingly. By this way it tries to maximize the query performance dynamically.

6.3 XPath Evaluation

The major step in optimizing query languages for XML is reducing the XPath complexity [5]. XQuery processing is dependent on XPath for extraction of data, joining of several documents and construction of new documents. The opti-

mization procedure of XPath has successfully evaluated mutiple queries parallely in a pipelined sequence in linear time. Through this approach it has become much easier to navigate elements and attributes in an XML document.

6.4 Semantic Query Optimization for XQuery

This type of optimization (SQO) technique uses the knowledge of XML schema to optimize queries [3]. It has produced extraordinary performance benefits in deductive, relational, object databases using XML documents in various search procedures including pattern retrieval, pattern filtering (e.g, join) and pattern restructuring (e.g, group-by). The first type of optimization technique applied to persistent and streaming XML generates a pruned query after query simplification that is efficient to evaluate. The second type of optimization involves rewritting the query. It is restricted to only persistent XML applications that can preprocess the data and build indices. Some SQO which are XML stream specific have drawbacks of addressing queries that have limited processing power. Another limitation is that actual physical implementations for optimizations grow complex for more powerful and complex queries.

XQuery Optimization Based on Rewriting have been implemented in data integration system called BizQuery [8]. BizQuery exibits remarkable performance in executing complex queries. It can also simplify cost estimation and query decomposition process. Further the XQuery Rewritter optimizer avoids redundant data scanning during processing queries. This is a type based optimization technique used to generate precise queries on the basis of XML schema.

Advantages of XQuery The wide usage, functionalities and predominant role of XQuery in various applications makes it evident that text based XML can be used in numerous areas of data integration and data services. The improved productivity and performance of XQuery proves that binary XML is not mandatory for increased performance. However the implementation of XQuery engine should be considered from application viewpoint as Query engine performs a sequence of complex operations. These operations can be costly for some lightweight applications. But inspite of its short comings, it is one of the cheapest processing technique as XQuery engine obtained from open source can be easily integrated with XML applications at no cost.

In future binary XML may become a potential way of increasing the performance of XQuery. This is due to its support for relational database operations. It also allows scalar quantities and huge data sets to be embedded within it. This way it could easily enhance query performance by allowing index, search and retrieval operations on queries.

6.5 XML Screamers

XML schema validation provides error checking for XML applications. But it has severe performance bottlenecks. To overcome the overhead and automate the validation of parsing XML documents, the XML Screamer has been designed. It is faster than most available processors. Its processing efficiency speeds up the entire application through single read operation and efficient generation of application data model [7]. It also exibits the capacity of integrating deserialization with scanning, parsing, validation and providing compiled optimizations specific to each and every XML API.

It is able to undertake schema validation and compiler-based optimizations at high speed by precomputing data from SAX events available from the XML Schema. Further the design architecture of XML Screamer allows it to optimize performance of SAX, business object APIs, and of other specialized APIs. An unique feature possessed by the Screamer is to generate efficient parsers in C and Java that optimizes the whole processing cycle. Web service applications using business object APIs have gained enhanced processing speed from the speed of individual XML parsers.

The range of API support currently available with the XML Screamer is limited, but it can be extended widely to different environments or applications. Though parser capability and business object generation functionality of the Screamer can be extended, the actual performance level is highly dependent on the XML APIs used. Some design of APIs may involve excessive string and buffer manipulations with a lot of overhead. Further the parsers generated by the Screamer must be compatible with the operating system, hardware, compiler, libraries present on the target system where the application is running.

7 Conclusion

The advantages of using binary XML in different applications are dependent on the OS architecture and application puropose [10]. Such conflicting requirements compel designers to decide on which application should memory footprint be given more importance than processing efficiency.

This restricts implementation of binary XML in all XML applications of the web. The same optimization scheme ranging from serialization, streaming, parallel bit streaming or processing XQuery cannot be applied to light-weight and heavy-weight applications in a similar fashion. This is evident from the difference in the encoding format of multimedia applications using real time streaming with the encoding format of applications using XML documents locally.

Difference in encoding styles prompt processors at the receiving end to have advanced architectural design to understand all types of binary encoding. This will help receivers to decode and retrieve the actual document. Also the application logic at the sender side should be strong enough to generate any binary format. Such factors of software application design lead to increased cost and complexity.

Considering the limitations of binary XML or text XML alone, any advanced architecture should combine binary and text XML to have an intermediate solution. This type of interleaving of the formats can yield the maximum efficiency. This can be seen in seismic data represenation. The logical operations in control information present in text XML can be optimized by XQuery. Control data processing by XQuery processors, schema validation by XML Screamers and binary XML represenation using XOP can yield remarkable performance results. Future research should focus on combining text XML with binary attchments for any XML data. Such research activities should aim at using XML Screamers on different otimized versions of encoded, compressed and

streamed fragments of binary XML to boost performance level of applications supporting any binary XML format.

This sort of combination of text and binary XML will preserve important aspects of XML that are lost with proprietary binary formats. It seen seen that Schema based binary optimization looses original Infoset and faces portability problem. Infoset change/loss limits one of the major characteristics of maintaining a well-formded XML document, present with text XML. The integration can preserve original content by efficiently serializing XML Infosets in a manner similar to XOP [19]. This will help to overcome the portability issue of binary format. If the range of support of the XML Screamer can be extended to binary XML, the combined format can serve a host of XML applications with high parsing speed.

References

[1] XML Query Optimization, 2006. http://ralyx.inria.fr/2006/Raweb/gemo/uid18.html.

[2] Don Brutzman, Don McGregor,Alan Hudson and Yumetech Inc. XML Binary Serialization using Cross-Format Schema Protocol (XFSP) and XML Compression Considerations for Extensible 3D (X3D) Graphics. Technical report, September 2003.

[3] Hong Su, Elke A. Rundensteiner, Rundensteiner and Murali Mani. Semantic Query Optimization for XQuery over XML Streams. In *31st international conference on Very large data bases*, pages 277 – 288, 2005.

[4] Jaakko Kangasharju and Sasu Tarkoma. Benefits of Alternate XML Serialization Formats in Scientific Computing. In *2007 Workshop on Service-oriented computing performance: aspects, issues, and approaches*, pages 23–30, 2007.

[5] Jason McHugh, Jennifer Widom. XQuery Optimization for XML. In *25th International Conference on Very Large Data Bases (VLDB)*, pages 315 – 326, 1999.

[6] Junze Wang; Yi Guo; Benxiong Huang; Jianhua Ma; Yijun Mo. Delta Compression for Information Push Services. In *22nd International Conference*, pages 247 – 252, March 2008.

[7] Margaret G. Kostoulas, Morris Matsa, Noah Mendelsohn, Eric Perkins, Abraham Heifets, Martha Mercaldi. XML Screamer: An Integrated Approach to High Performance XML Parsing, Validation and Deserialization. In *15th international conference on World Wide Web*, pages 93–102, 2006.

[8] Maxim Grinev, Sergey D. Kuznetsov. XQuery Optimization Based on Rewriting. In *6th East European Conference on Advances in Databases and Information Systems*, pages 340 – 345, 2002.

[9] Michael Leventhal. Random Access XML Programming Assisted with XML Hardware. In *XML Conference and Exposition 2004*, November 2004.

[10] Michael Rys, Shankar Pal, Jonathan Marsh and Andrew Layman. Standardize Binary Representation of XML?. http://www.w3.org/2003/08/binary-interchange-workshop/presentations-microsoft.pdf.

[11] Morris Matsa, Eric Perkins, Abraham Heifets, Margaret Gaitatzes Kostoulas, Daniel Silva, Noah Mendelsohn and Michelle Leger. A high-performance interpretive approach to schema-directed parsing. In *16th International Conference on World Wide Web*, pages 1093 – 1114, 2007.

[12] R. J. Bayardo and D. Gruhl and V. Josifovski and J. Myllymaki. An evaluation of binary xml encoding optimizations for fast stream based xml processing. In *13th International World Wide Web Conference*, pages 345 – 354, April 2004.

[13] Robert D. Cameron and Kenneth S. Herdy and Dan Lin. High performance XML parsing using parallel bit stream technology. In *IBM Centre for Advanced Studies Conference*, April 2008.

[14] World Wide Web Consortium. *A Case against Standardizing Binary Representation of XML*, Sept 2003. http://www.w3.org/2003/08/binary-interchange-workshop/29-MicrosoftPosition.htm.

[15] World Wide Web Consortium. *The W3C Workshop on Binary Interchange of XML Information Item Sets*, Sept. 2003. W3C Note. http://www.w3.org/2003/08/binary-interchange-workshop/31-oracle-BinaryXML_pos.htm.

[16] World Wide Web Consortium, Cambridge, Massachusetts, USA. *Building a Tokenizer for XPath or XQuery*, Apr. 2005. W3C Note. http://www.w3.org/TR/xquery-xpath-parsing.

[17] World Wide Web Consortium, Cambridge, Massachusetts, USA. *XML Fragment Interchange*. Feb. 2001. W3C Recommendation. http://www.w3.org/TR/2001/CR-xml-fragment-20010212.

[18] World Wide Web Consortium, Cambridge, Massachusetts, USA. *Web Service Architecture*, Feb. 2004. W3C Note. http://www.w3.org/TR/ws-arch/.

[19] World Wide Web Consortium, Cambridge, Massachusetts, USA. *XML-binary Optimized Packaging*, Jan. 2005. W3C Recommendation. http://www.w3.org/TR/xop10//.

[20] World Wide Web Consortium, Cambridge, Massachusetts, USA. *XML Binary Characterization*, Mar. 2005. W3C Note. http://www.w3.org/TR/xbc-characterization.

[21] World Wide Web Consortium, Cambridge, Massachusetts, USA. *XML Binary Characterization Properties*, Mar. 2005. W3C Note. http://www.w3.org/TR/xbc-properties.