RESEARCH PAPERS OF WROCŁAW UNIVERSITY OF ECONOMICS
No. 85 — 2009
Advanced Information Technologies for Management – AITM 2009

Andrzej Niesler

Wrocław University of Economics, Wrocław, Poland
e-mail: andrzej.niesler@ue.wroc.pl

EFFICIENT XML INTERCHANGE: BINARY ENCODING IN DATA PROCESSING AND INTEGRATION

Abstract: Extensible Markup Language (XML) is the foundation of many modern IT architectures. It is an extremely versatile integration tool and has been adopted in almost every business domain. The variety of applications imposes now the need for thorough analysis on how does the XML-orientation affect the enterprise data processing and integration scenarios. The paper discusses the problem of XML processing efficiency on various stages of the data interchange process. The main focus is on the information representation issue, with a special emphasis put on the binary encoding. As there are many different approaches to serialization of the textual XML, some common integration scenarios are subjected to the comparative analysis, in order to identify the crucial efficiency factors and implementation constraints. Based on the determined set of criteria, selected technologies are evaluated from the perspective of effective XML implementation strategy.

1. Introduction

The primary goal when introducing the Extensible Markup Language (XML) was to deliver a general-purpose, human-readable data interchange standard that would replace many proprietary binary formats in use. Due to its very high flexibility and real platform independence, it has become extremely popular as a common technology for exchanging structured and semi-structured data among various heterogeneous information systems and business applications, especially on the Internet. As it has been almost a decade since it was introduced on the market, there is a suitable moment for the preliminary assumptions to be revised. Also the question of efficiency analysis appears expected and legitimate.

XML provides a unique set of properties that most of the other data formats lack or implement only partially. It offers a single, flexible information model which can be applied to represent almost all the data used in many different data processing scenarios. The ability to expand the initial set of tags according to the changing needs makes it also an extremely versatile integration tool. Providing the foundation for the standardized and reliable integration platform, XML still remains a human-friendly

technology. Its self-descriptiveness and textual nature facilitate the readability and maintenance of the data which, therefore, can be viewed or modified using only a simple plain-text editor, available on every modern operating system. Regardless of the variety of advanced, sophisticated tools and solutions being offered currently on the market, it is extremely important, from the integration point of view, that the use of them is by no means mandatory and the whole XML processing model is based on open, easy adaptable international standards.

The introduction of XML in data representation, however, leads inevitably toward a substantial increase in verbosity, which is becoming a significant problem as the scope of deployment areas increases steadily and encompasses more and more business domains. That concerns not only the low-level data representation tier, but also a whole group of still emerging higher-level communication protocols, that make use of core XML syntax as the solid foundation for their own messaging structure. As a result, the indicated verbosity problem affects a great number of various aspects of the data interchange process and has a profound impact on the overall performance of XML-based IT solutions.

This paper discusses the problem of XML processing efficiency on various stages of the data interchange process. Its main focus is on the information representation issues, with a special emphasis put on the binary encoding solutions. As there are many different approaches to serialization of the textual XML, some common integration scenarios are subjected to the comparative analysis, in order to identify and verify the crucial efficiency factors and implementation constraints. The main goal is to analyze the advantages and disadvantages of various XML efficiency--increasing solutions. The paper is organized as follows: the next section provides an insight on the XML processing model, introduces the XML information set definition, characterizes standard programming approaches to XML document processing, XML communication protocols, and analyses some generic performance drawbacks. Section 3 presents the idea of binary encoded XML, lossless data compression methods, binary serialization and integration of such solutions with existing XML applications. Section 4 discusses selected use cases of XML in business integration scenarios and presents the results of a brief performance analysis. Finally, in section 5 some conclusions are drawn.

2. XML processing model

This section introduces the essential aspects of XML processing: XML information set, object-oriented and sequential processing approaches, and XML-based communication protocols. At the end, it discusses key performance-related XML processing drawbacks.

2.1. XML information set

From the structural point of view, XML imposes a hierarchical document structure, composed of entities, attributes, and values, denoted using the mechanism of textual tags. Therefore, a typical XML document consists of two major kinds of data: the markup and the actual content. The standard defines only a basic set of core syntax rules, which must be strictly satisfied for the document to reach the lowest level of compatibility, i.e., the so-called well-formed status. It is possible to make references between elements located inside the same document or to any other document or resource available on the Internet. The specification of the abstract data model of an XML document, i.e., the set of all valid information items, is called XML Information Set [*W3C XML*…].

In the real-world data computing scenarios, it is usually required to define the complete set of syntactical rules, e.g., to indicate precisely which elements are obligatory, which attributes are available and apply to them, in what order should be declared, etc. This kind of information is defined by the data schema and required in the validation process. As it is usually stored in a separate file, a valid XML document, besides having its structure organized accordingly, has to contain a direct reference to a proper schema file.

2.2. DOM and SAX

There are two common approaches for processing XML documents. The first one is the Document Object Model (DOM) which, according to its authors, is a multi-platform, language-independent interface standard allowing for dynamic representation, access, and update of document's content, style, and structure [*W3C Document*…]. The main idea in this case is based on the object-oriented representation of the whole data structure which can be easily accessed in a standardized way on many different programming platforms. The second approach is called Simple API for XML (SAX) and implements serial access based on event-driven processing paradigm. The programmer defines a list of methods and corresponding events. If a defined event occurs during the parsing process, the execution of required method is triggered with corresponding initial parameters.

In the DOM approach, a special processing programme called DOM parser performs a comprehensive analysis of the document and creates a representation of its hierarchical data structure in memory. The programme that originally requested access to the XML data operates on this intermediary structure, and after the last operation has been finished, the result of all the performed actions is incorporated back to the physical document. This approach is especially effective when the processing procedure consists of many repeating operations performed on a single XML document, that have to be executed inconsecutively or in random order. However, in the opposite situation, when there is only one-pass and/or sequential processing, the

SAX approach is much more effective as it omits the time and memory consuming procedure of building the hierarchical intermediary structure for object-oriented data representation in memory. Therefore, both approaches are valid and prove useful in different processing scenarios. The difference in performance is the result of correctness of selection and adjustment to the circumstances, rather than implementation of sequential or tree-based data processing algorithms.

2.3. XML-based communication protocols

As it was already stated, XML is not only used for structuralizing documents, but also as a foundation for many modern communication protocols, standards, languages, and platforms, e.g., Web Services Architecture [*W3C Web*…]. Besides the standard actors in the data interchange scenario, i.e., sender and receiver, there may be also many various additional processing points on the path between the indicated two. This includes common middleware solutions with the message-level routing functionality, which must analyze the message to find its recipient, or any other processing unit that has to examine in any way part of the transmitted data before passing it on. With the constantly growing popularity of web-based applications and systems, the amount of network traffic generated by the XML-based message interchange has significantly increased in recent years and, in consequence, affects bandwidth utilization and the overall networking performance.

Applying XML on the communication protocol level is alluring and convenient for more and more companies, as it fits perfectly in the single information model paradigm. However, the textual verbosity and processing requirements may in many cases be the cause of additional spending on CPU power and/or network infrastructure, which can be substantial and not lead directly to any instant business advantages or benefits. Moreover, unconsidered XML adoption may often result in occurrence of unexpected performance problems and limited scalability in the future. This may concern many various integration scenarios and platforms. Particularly vulnerable are those based on wireless networks and mobile devices, with limited processing power, storage capacity, operating memory, and, last but not least, battery life constraints [Kangasharju et al. 2007]. The situation becomes even more critical when considering the usage of any additional processing functionalities, like advanced authentication schemas, digitally signed message envelopes, or other important security features. Although many of these solutions are now under development, none of them has actually gained as high a level of acceptance as the XML standard itself.

2.4. XML performance drawbacks

To summarize the discussed characteristics, several XML performance drawbacks can be pointed out. One can assign different priorities to them depending on the particular circumstances, but they all are valid and should be taken into conside-

ration in almost every data processing and integration scenario. The first one is related to the textual verbosity of XML documents, which results in substantial increase of data that have to be stored, transmitted, and processed. It affects storage capacity, bandwidth, and operating memory, and can be problematic in wireless computing environment, as well as in any other, as the size of XML document can nowadays easily reach a few dozens of gigabytes. Besides the verbosity, the textual form itself is in many cases a serious processing drawback. Many applications operate on numerical data types or binary large objects, which have to be encoded and decoded to or from the textual representation every time the data are processed. This can yield an unnecessary latency in CPU-intensive computational tasks.

Another group of performance drawbacks is related to the XML processing model. In the traditional approach, the entire document has to be read before it can be processed. There are several phases that have to be completed in order to verify whether the document is well-formed in terms of core XML syntax or structurally valid according to a schema. That involves a lot of textual parsing which substantially slows down any data processing and is one of the most common bottleneck and scalability problems [Leventhal 2004]. Bearing in mind the previously mentioned increasing verbosity issue, this can lead to even bigger reduction in performance, as in many cases it is impossible to achieve processing goals in a single run due to the document size and restrictions in operating memory usage.

3. Binary representation

The addressed processing problems have been subjected to investigation in many recent scientific studies. Some of the proposed solutions even succeeded commercially. Almost all of them are based on two main concepts: lossless data compression and binary serialization. All these efforts lead towards a common standard for binary representation of XML data. This section discusses the major approaches and comments on the advantages and disadvantages from the perspective of XML processing efficiency.

3.1. Lossless data compression

The usual approach to solving the verbosity and limited bandwidth problem is to compress the data while transmitting on the network. The corresponding field of research is lossless data compression, based on C. Shannon's information theory and D. Huffman's statistical modelling for data encoding algorithms. Due to the textual representation of XML documents, many dedicated text compression methods can be applied. However, in most cases it is sufficient to use one of the available general-purpose compression tools, e.g., BZIP2 [Seward 2007], GZIP [Deutsch 1996], or ZLIB [Deutsch, Gailly 1996]. The interchange process starts with compression of the document, then the compressed data are transmitted over the network and de-

compressed by the receiver to the original, i.e. textual, form. This approach solves the bandwidth overload problem, unfortunately at the cost of additional processing time spent on compression and decompression. Moreover, during the transmission phase the data cannot be easily accessed or updated without executing the full process of decompression. This may cause additional problems to the systems that deliver content-based processing functionality on higher layers of the Open System Intercommunication Reference Model (OSI), e.g., LSI Tarrari Content Processors.

A much more effective approach is based on separating structure compression from content compression. In this case a better compression ratio can be achieved by taking advantage of the schema and data type information [Sundaresan, Moussa 2002]. This kind of XML-related compression methods has been introduced in such research projects as, e.g., XMill [Liefke, Suciu 2000] and Millau [Girardot, Sundaresan 2000]. The Millau architecture, depicted on Figure 1, is based on two compressed streams of data: one for encoded structure and the other for compressed content. The input can be either a pure XML data stream or a hierarchical DOM structure.

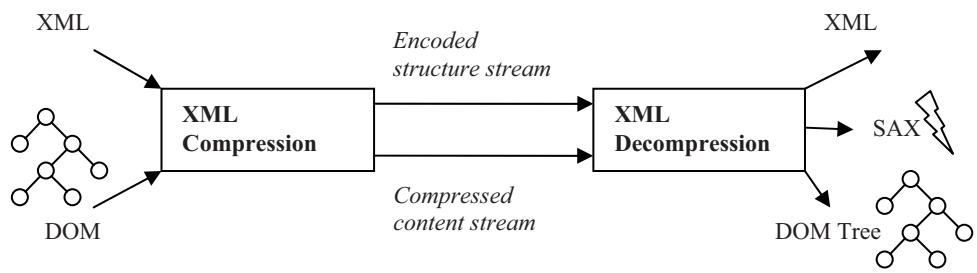

Figure 1. Architecture of the Millau Compression-Decompression System

Source: [Sundaresan, Moussa 2002, p. 683].

The compactness and size reduction of encoded XML data is achieved without any loss of functionality or semantic information because the Millau format is an extension of the WAP Binary XML format [*WAP Binary...* 1999]. However, some of the meta-information is removed in the encoding process, hence it is not always possible to generate the exact copy of the original input document on the output of the Millau system.

3.2. Infoset binary serialization

The experience with the Millau project shows the right direction on the path toward efficient XML processing. The best results can be achieved through a proper combination of content-aware data compression and compact encoding of the document structure. From the technical point of view, this approach amounts to the quest for a better serialization of XML, i.e., a more efficient infoset binary encoding that

would replace the original text-based representation. An example of such specification is ITU-T Rec. X.891 [*ITU-T...*]. Contrary to the generic compression methods, it optimizes both: the document size and the processing speed. The transformation is fully reversible, as there is no information loss during the conversion process conducted by a dedicated parser. The reduction in document size is significant, but not as good as the one delivered by the "true" compression methods based on statistical probability of the symbol's occurrence. However, considering the fact that there is no additional processing power required, this is a very reasonable trade-off.

As Fast Infoset is not the one and only available approach for binary serialization of XML information set, its implementation on a large scale is not as easy as in the case of the standard XML. To solve the emerging problem of compatibility and lay the foundation for future XML development, the Worldwide Web Consortium (W3C) started in 2005 a research to determine the set of key properties that such a standard should possess. As a result, AgileDelta's Efficient XML Interchange [*Efficient XML...*] specification was selected as a best solution and recommended for an open international standard.

Table 1. Minimum binary XML requirements

Property	
Must support	Must not prevent
Directly readable and writable	Processing efficiency
Transport independence	Small footprint
Compactness	Widespread adoption
Human language neutral	Space efficiency
Platform neutrality	Implementation cost
Integratable into XML stack	Forward compatibility
Royalty free	
Fragmentable	
Streamable	
Roundtrip support	
Generality	
Schema extensions and derivations	
Format version identifier	
Content type management	
Self contained	

Source: http://www.w3.org/TR/xbc-characterization.

In comparison to Fast Infoset, EXI delivers a better compactness, because it offers a stronger support for schemas and take advantage of the document vocabulary analysis. There is also an improvement in compression ratio due to applied bit align-

ment instead of byte alignment implemented in Fast Infoset format. Table 1 presents the minimum set of requirements established by the W3C Working Group for XML binary standard.

There are two main categories: “must support” and “must not prevent”. Some of the properties concern more the processor implementation than the standard itself. The second category contains some elective properties which can be traded off for any other improvements, but the standard cannot prevent those requirements.

3.3. Integration with existing XML applications

Binary serialization of XML information set increases processing performance and reduces the document size. However, the benefits are usually achieved at the expense of losing human readability and returning to the tight-coupled programming interfaces. The ability to read the transmitted data without any special tools or drivers is still essential from the programming or debugging points of view. Maintaining the property of loose coupling on the communication level is also very important in terms of supporting the information architecture flexibility. Finally, there is an issue of integrating new processing scheme with the existing XML applications. If it is not possible to implement the new binary processing functionality in a legacy solution, the encoded document has to be transformed back to the textual XML form. In today’s heterogeneous, distributed computing environment, it is very common to provide an automatic backward compatibility in order to provide a maximum openness and integration readiness.

To address these issues, some solutions offer a hybrid approach that combines the standard textual XML document and a binary extension to increase the processing speed. A good example of such an approach is XimpleWare’s Virtual Token Descriptor XML (VTD-XML) format. It introduces a new XML processing model based on non-extractive tokenization, where XML tokens representation is done by binary encoding with 64-bit integer VTD records that encode the token length, starting offset, type, and nesting depth [Zhang 2006]. It is a 100% processing performance-oriented solution and the total size of transmitted data is always larger than the size of the original document, due to the enclosed binary representation of pre-processed data structure.

4. Integration scenarios and performance analysis

This section presents selected use cases of XML in business integration scenarios. It discusses the possibilities of increasing processing effectiveness through application of the described data compression and binary encoding techniques. The section ends with a brief performance analysis, comparing the discussed XML interchange technologies.

4.1. Business warehouse data aggregation and loading

One of the common real-world business scenarios with XML data interchange and processing is the data aggregation and loading process for a corporate warehouse. In the use case published by SAP AG, this kind of aggregation is performed by a Point-Of-Sale (POS) Server which receives data from individual POS Interface applications or other intermediary POS Servers. For large retailers with hundreds of stores, the number of receipts per day can easily reach a level of about several millions. The local POS Server aggregates all sales items of a store and sends it in batch mode during closing hours to the central server. Using the standard XML 1.0 data encoding, this results in several gigabytes of data which have to be transmitted to the POS Server in a relatively short time.

In this scenario applying generic compression methods does not solve the problem. On the contrary, the overall processing time increases, as the gain achieved from shorter transmission time is marginal and negligible. The binary encoded XML infoset is a better solution, however, it imposes the upgrade of software in the parsing layer in all the data collection points. All tools above this layer in the processing stack can remain unchanged.

4.2. Enterprise web services

A very common integration scenario concerns the situation, when there is a certain number of enterprise systems that communicate with each other using one or more binary protocols, such as Remote Procedure Call (RPC), Remote Method Invocation (RMI), Distributed Component Object Model (DCOM), or Common Object Request Broker Architecture (CORBA). In the age of ubiquitous integration that exceeds the boundaries of a single enterprise, many companies are trying to implement the web services architecture, in order to reach a higher level of flexibility and interoperability with present and potential business partners. However, as the web services technology stack is based on XML, the transition from binary protocol to text-based SOAP and WSDL can lead to a significant degradation in performance. Therefore, in this integration scenario, the reasonable option is full binary encoded XML infoset for enterprise inner messaging infrastructure and a standard generic textual serialization for the outbound messaging interoperability.

Standard XML representation and processing model confines the performance and scalability of web services. Binary encoding can significantly increase the execution speed and throughput of web services architecture. It should also concern other performance-related issues, like, e.g., efficiency of data-binding and mapping of XML data to methods and methods parameters, or rapid mechanisms for ensuring message confidentiality, authentication, and integrity [Leventhal 2004, p. 12]. The standard XML interface can be established for outbound connectivity in order to provide less effective, but truly dynamic, loosely-coupled web services architecture.

4.3. Performance analysis

There are many various factors that determine performance of a particular XML processing approach. The difference can be significant even between two implementations of the same binary format and parsing algorithm. The thorough performance analysis of every major XML processing technology, especially involving consideration of various document sizes and levels of complexity, is beyond the scope of this paper. However, for the purpose of general comparison of the discussed solutions, some relative benchmarking tests will be provided. The aim is to juxtapose the performance of discussed four major technologies, i.e., standard text-based XML, XML compressed with GZIP, Fast Infoset, and Efficient XML (EXI). That will allow ranking the results, assessing the differences between them and their order of magnitude.

The two common performance measures are processing speed and compression ratio. Figure 2 presents the results of processing a set of real-world large web services SOAP messages. The fastest solution in the group is Efficient XML which operates up to 10 times faster than XML in case of small size messages, up to 15 times with medium size, and up to 35 times with large messages. Fast Infoset is on the second place and at least 2 times slower than Efficient XML. The smaller message size, the bigger advantage of EXI processing speed. GZIP compression does not have, as one would obviously expect, any noticeable impact on the processing speed, because the output after decompression is exactly the same as in case of standard XML.

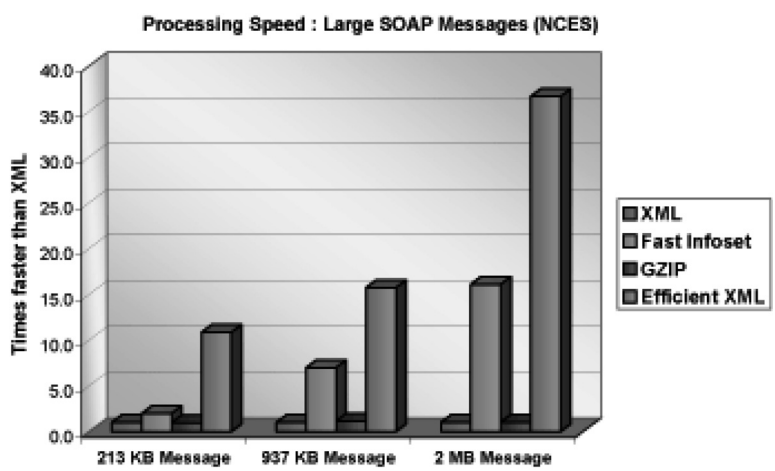

Figure 2. Processing speed of large SOAP messages

Source: www.agiledelta.com.

Figure 3 depicts the differences in compression ratios of the same large SOAP messages. In this case taller bars represent smaller messages. As one can notice, Ef-

ficient XML yields up to 90 times smaller output than standard XML for small size messages, up to 118 times smaller for medium size, and up to 120 times smaller for large messages. On the second place is GZIP, which produces several times smaller output than uncompressed XML, but is still 10 to 14 times bigger than Efficient XML. Fast Infoset is located on the third place with up to 4 times smaller output than standard XML. The huge advantage of Efficient XML over Fast Infoset in processing speed is only to a certain extent related to the smaller output message size. According to the AgileData, it can operate at over twice the speed when optimized for speed over size. This can be extremely useful if processing speed is the primary implementation goal.

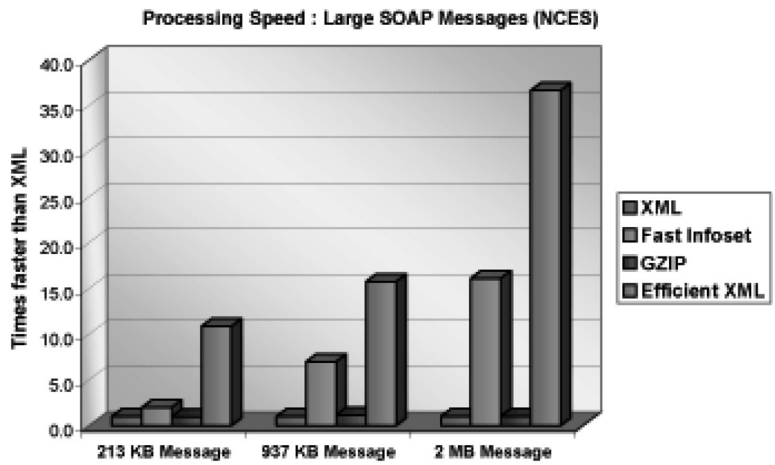

Figure 3. Compression ratios of large SOAP messages

Source: www.agiledelta.com.

The test results may vary in different computing circumstances, e.g., an optimized Fast Infoset driver could probably improve a little bit its processing speed. However, the order and the differences seem to be reliable. The Worldwide Web Consortium conducted extensive benchmarking of many binary XML technologies and in every considered test group and every use case, the Effective XML achieved the best result in compression ratio, and was among the fastest solutions in terms of processing speed.

5. Conclusion

XML specification was designed with the main focus on flexibility, easiness of use, and human readability. Nowadays, it is used in many different data interchange scenarios, where there is less need for human interaction and a very strong need for scalability and performance. Application of generic compression methods can solve

the limited bandwidth and communication latency problems. However, in most cases it is achieved at the cost of degraded performance and functionality. There is a need for an efficient way to exchange and process XML documents, and this efficiency can be achieved through the binary serialization of XML information set. The research conducted by the Worldwide Web Consortium resulted in standardization of XML binary encoding formats, which now can be freely implemented and utilized in business and non-business solutions. The analysis of XML use cases in common integration scenarios demonstrates the importance of binary XML, as a key technology to provide the new level of efficiency. Performance analysis of such solutions as Efficient XML indicates a great economic potential in this approach.

References

Deutsch P. (1996), *GZIP file format specification version 4.3, RFC 1952*, http://www.gzip.org/zlib/rfc-gzip.html.

Deutsch P., Gailly J. (1996), *ZLIB compressed data format specification version 3.3, RFC 1950*, http://www.gzip.org/zlib/rfc-zlib.html.

Efficient XML Interchange Working Group, http://www.w3.org/XML/EXI/.

Girardot M., Sundaresan N. (2000), Millau: An encoding format for efficient representation and exchange of XML over the Web. Proceedings of the 9th WWW Conference, Amsterdam, Netherlands, *Computer Networks*, Vol. 33, Issues 1-6, June, pp. 747-765.

ITU-T Rec. X.891, http://www.itu.int/rec/T-REC-X.891-200505-I/en/.

Kangasharju, J. et al. (2007), XML messaging for mobile devices: From requirements to implementation, *Computer Networks*, Vol. 51, No. 16, pp. 4634-4654.

Leventhal, M. (2004), Is now the time for binary XML? Report on current W3C activity, [in:] *XML 2004 International Conference and Exhibition*, Washington D.C., http://www.gca.org/xmlusa/2004/.

Liefke H., Suciu D. (2000), XMILL: An efficient compressor for XML data, [in:] *ACM SIGMOD Record*, Vol. 29, Issue 2 (June 2000), Eds. W. Chen, J. Naughton, Ph. A. Bernstein, ACM New York (http://portal.acm.org/citation.cfm?id=335191&coll=GUIDE&dl=GUIDE&type=issue&idx=J689&part=newsletter&WantType=Newsletters&title=ACM%20SIGMOD%20Record&CFID=76448438&CFTOKEN=9808248), pp. 153-164.

Seward J. (2007), *BZIP2 and libbzip2, version 1.0.5: A program and library for data compression*, http://www.bzip2.org/.

Sundaresan N., Moussa R. (2002), Algorithms and programming models for efficient representation of XML for Internet applications, *Computer Networks*, Vol. 39, No. 5, (http://www.informatik.unitrier.de/~ley/db/journals/cn/cn39.html#SundaresanM02), pp. 681-697.

W3C Document Object Model, http://www.w3.org/DOM/.

W3C Web Services Architecture, http://www.w3.org/TR/ws-arch/.

W3C XML Information Set, http://www.w3.org/TR/xml-infoset/.

WAP Binary XML Content Format, W3C Note 24 June 1999, http://www.w3.org/TR/wbxml/.

Zhang J. (2006), Simplify XML processing with VTD-XML. A new option that overcomes the problems of DOM and SAX, *JavaWorld*, March.

Websites

http://www.w3.org/TR/xbc-characterization.

www.agiledelta.com.