XBrevity: XML Data Compression using Brevity Encoding

Praphan Lakhasophon and **Kanda Runapongsa**

Department of Computer Engineering, Faculty of Engineering, Khon Kaen University
Khon Kaen 40002, Thailand.
{superjing@gmail.com , krunapon@kku.ac.th}

ABSTRACT

XML becomes the standard language for data representation and exchange on the Internet because it is simple, flexible, and platform neutral. However, XML data is often large and verbose since it consists of many repeated tags which are used to self-describe data. The large size of data results in an excessive amount of space required for storing data on the disk space and that of time required for transmitting data over the network. Thus, it is necessary to find an effective compression technique for XML data. In this work, we propose XBREVITY, an XML compressor which supports compressing and uncompressing XML data. XBREVITY adopts a novel encoding method that has the compressed XML data in the XML format. Thus, the compressed XML data still preserves the advantage features of XML but the XML document has the smaller size.

Keywords: XML, Compression, Brevity

1. INTRODUCTION

Currently, XML [10] becomes the standard language for data exchange because it is self-described and flexible. XML uses tag names to describe data so that a created document is easy to understand. When XML tags are used to describe data, an XML document usually has many repeated tags. Therefore, the large size of data results in an excessive amount of space required for storing data on the disk space and that of time required for transmitting data over the network.

The remainder of the paper is organized as follows: In section 2 we present related work. Section 3 presents features of XBrevity. Section 4 presents compression techniques in XBrevity. The experiments of these techniques are studied in section 5. Finally, conclusions are given in section 6.

2. RELATED WORK

File compression often uses gzip [2] because it is freeware and does not need the information of document structure. The disadvantage of using gzip in XML is that it cannot check continued repeated elements since it is not designed for XML compression.

Important or recent related work of XML compression includes XMill, XGrind, XPRESS and XPACK [3]. These methods reduce the size of an XML document using different techniques as described in the following subsections.

2.1 XMILL

XMill [4] achieves better compression rate compared to gzip (by a factor of 2, for data-like XML documents) without sacrificing speed. This owes to the fact that it separates structure from content. This makes it a clear winner for applications like data archiving since these applications require lesser disk space. At the same time, it reduces network bandwidth. XMill is moderately faster than gzip in XML publishing. However, relative advantage of XMill depends on the application it is used. However, compressed output of XMill is not queryable except the document has to be decompressed.

If the size of the input document is less than 20KB, XMill will not exhibit any significant advantage over gzip. Here the compression is targeted for applications, such as data exchanging, data archiving, but not for deriving a meaningful view of the input document as is the case of compressing images or video sequences. To apply specialized compressors to containers, human intervention is required to specify the required container. Path processor is configured by user commands to map values. XMill precludes incremental processing of compressed documents; it actually hinders compressors other than gzip, and requires user assistance to achieve the best compression [7].

2.2 XGRIND

XGrind [9] has a considerable improvement in query response time and disk bandwidth. Its effective compression is processed through increased information density so that memory hit buffer ratio increases. In addition, XGrind compresses at the granularity of element/attribute value using context-free compression scheme. Furthermore, for range and partial match queries, on the fly decompression is required for only those elements that feature in the query predicates.

However, XGrind does not support several operations such as non-equality selections. In addition, it cannot perform any join, aggregation, and nested queries or construct operations. XGrind depend on one-time statistics of an XML document before the compression, but the statistics can change due to updates made to the compressed XML document. Lastly, XGrind uses a fixed root-to-leaf navigation strategy, which is insufficient to provide alternative evaluation strategies.

2.3 XPRESS

Like XGrind, XPRESS [6] also allows queries on compressed data. XPRESS works only on XML trees. It cannot handle ID/IDREF tags. It creates bisimilar partitions of elements in the XML document. Then, it encodes partitions by assigning disjoint intervals and allows query evaluation by operations on these intervals. It uses an encoding method known as the reverse arithmetic encoding. It is a combination of differential and binary encoding methods. This is an efficient path encoding method, which yields fewer overheads of partial decompression and quicker path evaluation. XPRESS provides high compression ratio.

2.4 XPACK

XPACK [5] is an XML compressor that uses grammar for compressing and decompressing XML documents. XPACK is composed of the following three parts: the Grammar Generator, the Compressor, and the Decompressor. The Grammar Generator creates grammar. The Compressor compresses XML document and the Decompressor decompresses compressed XML document by using the old structure of the document.

The disadvantage of XPACK is that it cannot compress XML documents that have mixed-content elements (elements that have both elements and character data). Moreover, users cannot query data from compressed XML documents.

3. FEATURES OF XBREVITY

The important features of XBrevity are as follows:

- Being able to compress and decompress an XML document
- Being able to compress an XML document with mixed-content elements
- Being able to query compressed XML documents
- Being able to compress a folder of XML documents

Following paragraphs describe one of features of XBrevity which is that it can compress an XML document that has mixed-content elements. The example of the XML document with mixed-content elements is shown in Fig.1. This document consists of "movie" element which is the root element, then the "mtitle" element which has "mname" element. Here, it can be seen that "mtitle" element is a mixed-content element since it contains both sub-elements ("mtitle") and the character data ("the prisoner of"). "actor" element is also a mixed-content element since it contains both character data ("Tim Robbins") and other elements ("actor") which are nested. The "mtitle" element has one attribute which its name is "year" and its value is "1994", and two sections of character data that has value "the prisoner of" and "redemption".

```
<?xml version="1.0"?>
<movie>
   <mtitle year="1994">the prisonner of
     <mname>shawshank</mname>redemption</mtitle>
  <actor>Tim Robbins
    <actor>Morgan Freeman <actor>Bob Gunton
      </actor>William Sadler<actor>Clancy Brown
      </actor>Gil Bellows</actor>
  </actor>
</movie>
```

Fig.1: *An Original Sample XML File*

```
<?xml version=="1.0"?>

<c>

<d>
   <e1 a2="1994" v="the prisonner of"
          e3v="shawshank"/>
   <xe1 v="redemption"/>
   <e4 v="Tim Robbins" e4v="Morgan Freeman"
          e4v2=" Bob Gunton" />
   <xe4e4 v="William Sadler" e4v="Clancy Brown"/>
   <xe4e4 v="Gil Bellows" />
</d>
<m f="movie mtitle year mname actor"
          b="0 1 a2 3 4 "/>
</c>
```

Fig.2: *The Compressed Sample XML File*

When an XML document has mixed-content elements, XBrevity uses the following rules in compressing the document. If a close tag occurs in a mixed-content element, XBrevity separates the results into multiple parts. For example, in the "mtitle" element, there is a close tag which is </mname>. The first part of the result is <e1 a2="1994" v="the prisoner of" e3v="shawshank"/>. Then, "redemption" is included in the second part of the result which is a new element that has "x" as a prefix to indicate that this element is the part of the previous element which is the "mtitle" element.

The actor tags are nested in three levels and are mapped into "e4" with three attributes which are "v" (which is corresponding to the content of the first actor), "e4v" (which is corresponding to the content of the second actor) , and "e4v2" (which is corresponding to the content of the third actor). When the same element is nested more than three levels, XBrevity adds level number starting from the number "2". Fig. 2 shows the compressed XML version of the sample XML document with mixed-content elements, which the document is shown in Fig. 1.

4. COMPRESSION TECHNIQUES IN XBREVITY

The architecture of XBrevity consists of Compressor which is used to encode the original XML document to the compressed XML document and Decompressor which is used to decode the compressed XML document back to the original XML document. XBrevity has the XML File Filter to distinguish the program input that is a single XML document and the program input that is a folder containing several XML documents. SAX Parser is used to parse and analyze documents for encoding and decoding the documents. The architecture of XBrevity is shown in Fig.3.

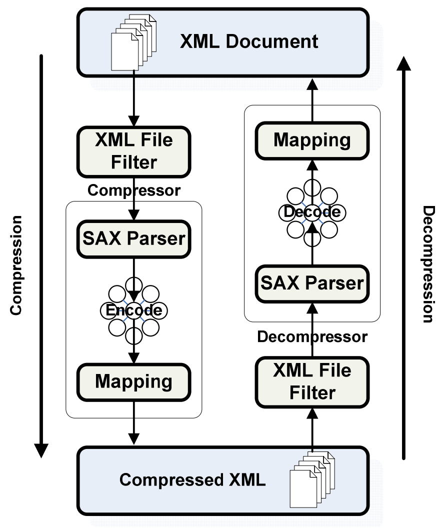

Fig.3: *The Architecture of XBrevity*

4.1 COMPRESSOR

The function of Compressor is to compress an XML document which uses SAX Parser for reading and analyzing the input XML document. While SAX Parser analyses the document, it encodes the data by using the mapping method that produces the result which is then saved in a new well-formed XML file.

The new XML document consists of the "c" element which is the root element. "c" stands for "compressed" data. The "c" element has two child elements: the "d" element and the "m" element. "d" stands for data and "m" stands for "metadata" Thus, the information between <d> and </d> is the compressed XML data and the information inside <m…/> is the metadata of the compressed XML data. The new compressed XML document can be stored, exchanged over the network, and is easy to understand since it is in an XML format and has metadata.

The next subsection will describe how the Decompressor component of XBrevity decodes the compressed XML document back to the original document.

4.2 DECOMPRESSOR

The Decompressor of XBrevity also uses SAX Parser to read and analyze the compressed XML document by mapping information inside <m…/> tag then using the mapped information to construct the data inside that is in between <d> and </d> in a new XML file. As a result, it will generate an XML document that is the same as the original XML document.

5. EXPERIMENTAL EVALUATION

In this section, we present the results from an experimental evaluation of XBrevity, and compare it with current compression techniques.

5.1 EXPERIMENTAL SETUP

XBrevity that we implemented is a stand-alone Java application. It uses the Apache Xerces Java version 2.8 [1] as SAX Parser APIs. The source code can be downloaded at

http://gear.kku.ac.th/~krunapon/research/xbrevity/

All our experiments operate on Intel Pentium 4 1.8 GHz, RAM 256 MB, Windows XP Professional with Service Pack 2 OS.

We use XMark [8] to randomly generate XML documents with different sizes. The properties of XML files are shown in Table 1. After generating XML documents, we compare the performance of XBrevity and other current techniques which include gzip and XMill. We cannot test XGrind, XPRESS, and XPACK because the available XGrind program is available only on the platform that is different from the platform that we use. XPRESS and XPACK do not have the programs available for downloading.

Table 1: *XML Data Set*

XML Files	Size (bytes)	Depth	Elem	Attr	XMark factor
a.xml	27,233	8	58	3	0.0001
b.xml	118,274	9	72	4	0.001
c.xml	1,182,547	10	72	5	0.01
d.xml	11,875,066	10	72	5	0.1
e.xml	118,552,713	10	72	5	1

5.2 EXPERIMENTAL RESULTS

In this section, we first present the compression ratio of each compressor. The compression ratio is defined as follows:

$$\text{Compression ratio} = \frac{\text{Size of Compressed XML}}{\text{Size of Original XML}}$$

Table 2 shows that compressed XML documents have smaller sizes than that of original documents.

Table 2: *Compression Size after Performing gzip*

XML Files	Compressed Size (bytes)		
	Original	XBrevity	XBrevity +gzip
a.xml	27,233	26,587	10,813
b.xml	118,274	116,051	41,535
c.xml	1,182,547	1,158,109	383,921
d.xml	11,875,066	11,631,650	3,850,626
e.xml	118,552,713	115,701,249	38,860,417

XMILL and gzip are binary compression but XBrevity compresses only tag. When we combine using XBrevity with gzip, the result from applying different techniques are shown in Table 3.

Table 3: *Compression Ratio after Performing gzip*

XML Files	**Compression Ratio (%)**		
	gzip	**XMill**	**XBrevity+gzip**
a.xml	61.47	61.83	60.30
b.xml	65.55	67.18	64.88
c.xml	67.83	70.72	67.53
d.xml	67.82	71.37	67.57
e.xml	67.73	71.50	67.22
Average	**66.08**	**68.52**	**65.50**

We also plot the graph to compare the performance of gzip, XMill, and XBrevity+gzip as shown in Fig. 4

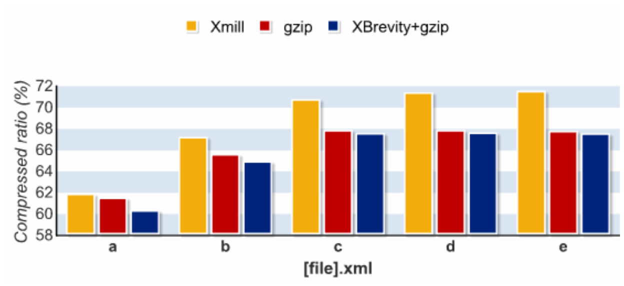

Fig.4: *Compression Ratios of Different Techniques*

6. CONCLUSIONS

XML compression is important because XML language has many advantages but its disadvantage is its large size since an XML document usually has repeated tags. Large size of data results in an excessive amount of time in transmitting XML data since it requires a large network bandwidth and also results in a large disk space. Several researchers have developed many compression techniques that can reduce XML data size but some methods need XML schema [11,12,13] in order to be able to compress data and some methods have to decode document to original document before applying queries [14].

In this paper, we propose the technique for compressing and decompressing XML data, which is called XBrevity. It can reduce the document size of a single XML file as well as a folder of several XML files. XBrevity has several advantages, such as it does not require the schema information of XML document and the user can query compressed document which saves decoded time.

In the future, we plan to enhance XBrevity to be able to compress the character data section in an XML document which will result in greater reduced size of the document.

7. REFERENCES

[1] Apache Software Foundation. “Xerces2 Java Parser 2.8.0 Release”. Available at http://xerces.apache.org/xerces2-j/

[2] J.L. Gailly and M. Adler, “gzip : The compressor data”, Available at http://www.gzip.org/

[3] P. Lakhasophon and K. Runapongsa. “A Survey of XML Data Compression”. *Proceeding of the 1st Northeastern Computer Science and Engineering Conference (NECSEC2005)*, 2005 March 31–April 1, Khon Kaen, Thailand.

[4] H. Liefke and D. Suciu. “XMill: an Efficient Compressor for XML Data.”, *Proceeding of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 153-164, May 2000.

[5] K. Mairiang, and C. Pluempitiwiriyawej. “XPACK: A Grammar-based XML Document Compression”, In *Proceeding of NCSEC2003 the7th National Computer Science and Engineering Conference*, Oct 28-30, 2003.

[6] J.-K. Min, M.-J. Park, and C.-W. Chung. “XPRESS: A Queriable Compression for XML Data.” *Proceeding of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 122-133, June 9-12, 2003.

[7] K. Runapongsa, J.M. Patel. “Storing and Querying XML Data in Object-Relational DBMSs”. *Proceeding of the EDBT Workshops 2002. Conference on Extending Database Technology*. 2002 March 24-28, Prague, Czech Republic, 2002. p. 266-285.

[8] A. R. Schmidt, F. Waas, M. L. Kersten, M. J. Carey, I. Manolescu, and R. Busse. “XMark: A Benchmark for XML Data Management”, *Proceedings of the International Conference on Very Large Data Bases (VLDB’02)*, pp. 974-985, Hong Kong, China, August 2002, Available at http://monetdb.cwi.nl/xml /index.html

[9] P. M. Tolani and J. R. Haritsa. “XGRIND: A Query-friendly XML Compressor.” *Proceedings of the 18th International Conference on Databases Engineering (ICDE’02)*, Feb 2002.

[10] W3C. “Extensible Markup Language (XML) 1.0 (Third Edition)”, Feb 4, 2004, Available at http://www.w3.org/TR/2004/REC-xml-20040204/.

[11] W3C. “XML Schema Part 0 : Primer ” , May 2, 2001, Available at http://www.w3.org/TR /xmlschema-0/.

[12] W3C. “XML Schema Part 1 : Structures”, May 2., 2001, Available at http://www.w3.org/TR /xmlschema-1/.

[13] W3C. “XML Schema Part 2 : Datatypes ”, May 2, 2001, Available at http://www.w3.org/TR/xmlschema-2/

[14] W3C. “XQuery : An XML Query Language”, Available at http://www.w3.org/XML/Query