Research Article
HR Deshmukh, IJPRET, 2014; Volume 2 (6): 83-92

ISSN: 2319-507X
IJPRET

INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

OPTIMAL XML DATABASE COMPRESSION TECHNIQUE USING DYNAMIC HUFFMAN CODING ALGORITHM

Dr. H.R. DESHMUKH[1], Dr. G. R. BAMNOTE[2]

1. Professor & Head Dept. Of CSE , IBSS College Of Engineering Amravati.
2. Professor & Head, PRMIT & R, Badnera.

Accepted Date: 13/01/2014 ; Published Date: 01/02/2014

Abstract: The Extensible Markup Language (XML) is one of the most important formats for data interchange on the Internet. XML documents are used for data exchange and to store large amount of data over the web. These documents are extremely verbose and require specific compression for efficient transformation. In this proposed work we are enhancing the existing compressors which uses Adaptive Huffman coding. It is based on the principle of extracting data from the document, and grouping it based on semantics. The document is encoded as a sequence of integers, while the data grouping is based on XML tags/attributes/comments. The main disadvantage of using XML documents is their large sizes caused by highly repetitive (sub) structures of those documents and often long tag and attribute names. Therefore, a need to compress XML, both efficiently and conveniently to use. The re-organized data is now compressed by adaptive Huffman coding. The special feature of adaptive Huffman coding algorithm is that, it has extremely accurate compression as well as it eliminates the repetition of dictionary based words in xml database. Using Adaptive Huffman algorithm, we derived probabilities which dynamically changed with the incoming data, through Binary tree construction.

Keywords: Compression, decompression, , Efficient XML compression and decompression, Adaptive Huffman coding

PAPER-QR CODE

Corresponding Author: Dr. H. R. DESHMUKH

Access Online On:

www.ijpret.com

How to Cite This Article:

HR Deshmukh, IJPRET, 2014; Volume 2 (6): 83-92

INTRODUCTION

The Extensible Markup Language (XML) is one of the most important formats for data interchange on the Internet. XML documents are used for data exchange and to store large amount of data over the web. These documents are extremely verbose and require specific compression for efficient transformation. In this proposed work we are enhancing the existing compressors which uses Adaptive Huffman coding. It is based on the principle of extracting data from the document, and grouping it based on semantics[1].The document is encoded as a sequence of integers, while the data grouping is based on XML tags/attributes/comments. The main disadvantage of using XML documents is their large sizes caused by highly repetitive (sub) structures of those documents and often long tag and attribute names. Therefore, a need to compress

XML, both efficiently and conveniently to use. The design goal of Effective compression of XML database by using Adaptive Huffman coding is to provide extremely efficient accurate compression of XML documents while supporting "online" usage. In this context, "online" usage means: (a) only one pass through the document is required to compress it, (b) compressed data is sent to the output stream incrementally as the document is read, and (c) decompression can begin as soon as compressed data is available to the decompressor. Thus transmission of a document over a heterogeneous systems can begin as soon as the compressor produces its first output, and, consequently, the decompress or can start decompression shortly thereafter, resulting in a compression scheme that is well suited for transmission of XML documents over a wide-area network.

Why there is need to Compress XML?

XML representations are very large and can be up to ten-times as large as equivalent binary representations. Meaningful tags make an XML document self-describing, but they also increase the document's size. In most cases, an element contains at most one data item. In other words a data item usually comes with a start-tag and an end tags. XML is the lingua franca(verbose) of web services, thus necessitating large volumes of XML data to be sent over networks. Reducing data size helps conserve network bandwidth. Consider the invoice record in Example 1.3.

```
<?xml version="1.0" encoding="utf-8" ?>

<Invoice ID="ST87302" date="2003-03-17" paid="true" >

<Customer ID="2376">
```

<Name>

<FirstName>Wally</FirstName>

<LastName>Robertson</LastName>

</Name>

<BillingAddress>

<Street>200 University Ave.</Street>

<City>Waterloo</City>

<Province>ON</Province>

</BillingAddress>

</Customer>

<ProductList>

<Product ID="HI0941" >

<name>MP3 Player</name>

<Price>139.99</Price>

<Units>2</Units>

</Product>

<Product ID="XP6015" >

<name>CD Writer</name>

<Price>79.99</Price>

<Units>4</Units>

</Product>

</ProductList>

</Invoice>

Example 1.1: An Invoice Record In XML Format

This document contains 19 data items and its length is 688 bytes. The data itself, stored in plain text as in Example 1.2, has only 144 bytes. This value still can be further reduced if we use binary format [7]. The repeating tag characteristic makes the XML version almost 5 times longer than the plain text version.

ST87302 2003-03-17 true 2376

Wally Robertson

200 University Ave. Waterloo ON

HI0941

MP3 Player

139.99 2

XP6015

CD Writer

79.99 4

Example 1.2: An Invoice Record In Plain Text.

XML Compression:

XML compression is an lossless compression technique. A compression approach is lossless only if it is possible to exactly reconstruct the original data from the compressed version. There is no loss of any information during the compression process. Lossless compression is called reversible compression since the original data may be recovered perfectly by decompression [16]. XML is a text representation for a tree structured data. Hence, a straightforward logical approach for compressing XML documents is to use the traditional general purpose text compression tools. In compression process XML data symbols, tags and frequencies of its symbols are collected and maintained dynamically according to the source file.

XML Decompression

The decompression process is obvious or can be easily derived from the compression process. Any compression algorithm will not work unless a means of decompression is also provided due

to the nature of data compression.XML decompression is exactly opposite to the xml compression[16].

Compressed XML data is sent to the output stream incrementally as the document is read, and decompression can begin as soon as compressed data is available to the decompressor. Thus transmission of a document over a heterogeneous systems can begin as soon as the compressor produces its first output, and, consequently, the decompress or can start decompression shortly thereafter, resulting in a compression scheme that is well suited for transmission of XML documents over a wide-area network.

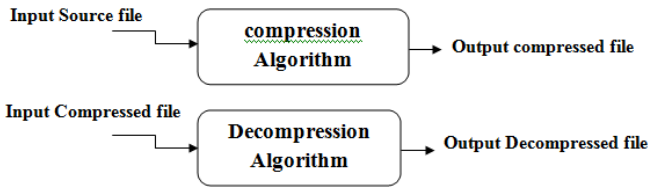


Fig 1.3: compression & Decompression.

II RELATED WORK

Various sophisticated algorithms have been proposed for lossless text compression A very promising development in the field of lossless data compression is the Burrows-Wheeler Compression Algorithm (BWCA), introduced in 1994 by Michael Burrows and David Wheeler. The algorithm received considerable attention since of its Lempel-Ziv like execution speed and its compression performance close to state-of-the-art PPM algorithms. A preprocessing method is performed on the source text before applying an existing compression algorithm. The transformation is designed to make it easier to compress the source file. The star encoding is generally used for this type of pre processing transformation of the source text.

XGrind: several proposals and references there in make use of the observation that the pioneering work in this domain was XGrind which was based on static Huffman coding. XGrind was the first XML-conscious compression scheme to support querying without full decompression [7].

XPRESS: Like XGRIND, features a homomorphic compression process requiring two passes over the document. XPRESS also supports querying of compressed data and claims to achieve better compression than XGrind [8]. However, it uses a semi-adaptive form of arithmetic coding which also necessitates two passes Over the original XML document.

III. IMPLEMENTATION

A.) Compression Techniques for XML Database:

Lossless Compression:

Lossless compression techniques provide exact recovery of the original data from their compressed version. Any information contained in an original cannot be lost during compression and reconstruction. These techniques are used widely in applications to save storage space and network bandwidth.

Adaptive Huffman coding for xml database compression

XML simplifies data exchange among heterogeneous computers, but it is notoriously verbose and has spawned the development of many XML-specific compressors and binary formats. We presented an XML test and a combined efficiency metric integrating compression ratio and execution speed. With the help of adaptive Huffman compression technique. The Adaptive Huffman compression is more efficient than static Huffman compression it is an important dimension for lossless data compression. In computer science and information theory Huffman coding is an entropy encoding algorithm used for lossless data compression[12].

In the adaptive Huffman coding, an alphabet and frequencies of its symbols are collected and maintained dynamically according to the source file on each iteration. The Huffman tree is also updated based on the alphabet and frequencies dynamically. When the encoder and decoder are at different locations, both maintain an identical Huffman tree for each step independently. Therefore, there is no need transferring the Huffman tree. The Adaptive Huffman algorithm provides effective compression by just transmitting the node position in the tree without transmitting the entire code[12]. Unlike static Huffman algorithm the statistics of the sensor data need not be known for encoding the data[12]. That is why adaptive Huffman is extremely accurate with respect to the compression.

Adaptive algorithm processed by following manner:

- Adaptive Huffman coding provide extremely efficient and highly accurate compression of XML documents while supporting "online" usage. In this context, "online" usage means: (a) only one pass through the document is required to compress it.
- Adaptive Huffman determines the mapping to code words using a running estimate of the source symbols probabilities.

- Compressed data is sent to the output stream incrementally as the document is read.
- Decompression can begin as soon as compressed data is available to the decompressor.
- Adaptive Huffman coding encodes an alphabet with fixed codes. That allows us to directly search keywords in the compressed data[6]. Since the data volumes reduced, such compressing of xml data may be even faster than the original data, it performs transformation of xml database to be fully reversible and decompress able so that the decompressed document is a mirror image of the original.

Compression & Decompression of XML database By Adaptive Huffman Codin

We proposed here an efficient way of compressing XML documents by using adaptive Huffman coding. High compression ratio and speed is equally important. We also require the transformation of xml database to be fully reversible and decompress able so that the decompressed document is a mirror image of the original. The main idea is for the compressor and the decompressor to start with an empty Huffman tree and to modify it as symbols are being read and processed

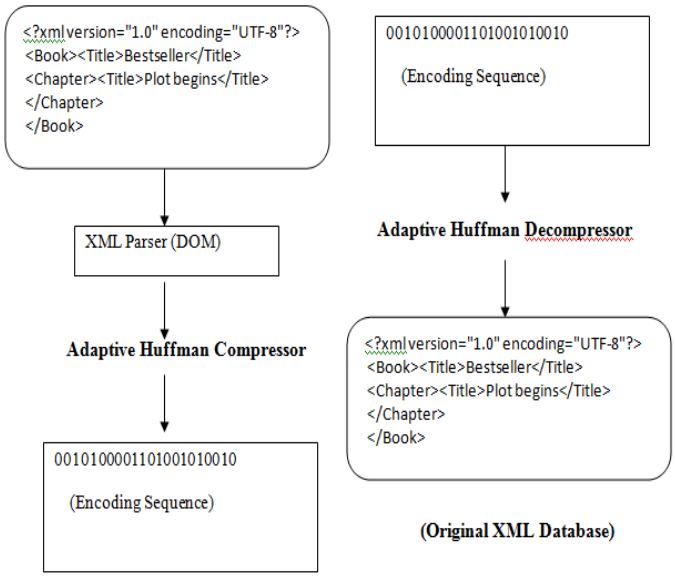


Fig: XML Compression & Decompression Process

XML Parsers

Parser is a program or module that checks a well-formed syntax and provides a capability to manipulate XML data elements. In this project work we used DOM(Document Object Model) PARSING[18].

DOM (Document Object Model):

Is a standardized platform and language independent object oriented interface defining a number of object with which document (in particular HTML or XML) can be described as a hierarchical tree structure. The standardized objects and methods are used to easily manipulate documents and produces uniform, reusable programs. Structure DOM describes a document as a tree structure of nodes. In tree structure XML data consists of nodes and edges that represent elements attributes, text. Parsing of XML data is done using DOM parser, in which all nodes of data are parsed according to node type[12].

Creating the hierarchy:

DOM, in essence, is a collection of nodes. With different types of information potentially contained in a document, there are several different types of nodes defined. In creating a hierarchy for an XML file, it's natural to produce something conceptually like the structure below. While it is an accurate depiction of the included data, it is not an accurate description of the data as represented by the DOM. This is because it represents the elements, but not the nodes[12].

Experimental Evaluation:

Performance metrics

We measure and compare the performance of the XML compression tools using the following metrics.

Compression Ratio

Represents the ratio between the sizes of compressed and uncompressed XML documents as given by:

Compression Ratio =

(Compressed Size)/(Uncompressed Size).

Compression Time

Represents the elapsed time during the compression process, i.e. the period of time between the start of program execution on a document until all the data are written to disk.

5.1.3. Decompression Time:

Represents the elapsed time during the decompression process i.e. the period of time between the start of program execution on reading the decompressed format of the XML document until delivering the original document.

Experimental Results

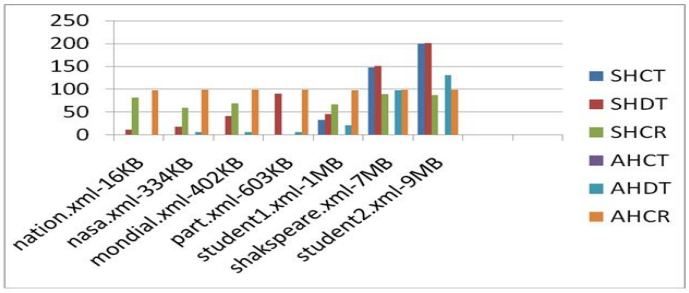
5.3.1 Comparative analysis

Here experimental results are there for some standard datasets of xml. The documents are selected to cover a wide range of sizes where the smallest document is 16 KB and the biggest document is 9MB.

XML files	SHCT (sec)	SHDT (sec)	SHCR	AHCT (sec)	AHDT (sec)	AHCR
nation.xml-16KB	0.178	11.2	82%	0.006	0.542	98%
nasa.xml-334KB	0.1835	18.413	59%	0.0069	5.803	99%
mondial.xml-402KB	0.2289	41.166	69%	0.0056	5.76	99%
part.xml-603KB	1.222	90.42	69%	0.007	5.87	99%
student1.xml-1MB	277.889	45.2322	67%	69.22	20.92	98%
shakspeare.xml-7MB	1473.77	150.88	89%	1230.56	120.12	99%
student2.xml-9MB	1987.34	250.88	87%	1565.33	190.777	99

Table 5.1: Comparative analysis

Performance Analysis



CONCLUSION

Adaptive Huffman coding encodes an alphabet with ¯fixed codes. That allows us to directly search keywords in the compressed data[6]. Since the data volumes reduced, such compressing of xml data may be even faster than the original data. The resulting output of proposed work will be that xml document compresses with the help of adaptive Huffman algorithm and that compressed data will be decompressed as well and provide original xml document over an heterogeneous systems

REFERENCES

1. Bray, et al. "Extensible Markup Language (XML) 1.0", October 2000, http://www.w3.org/TR/REC-xml.

2. G. Girardot and N. Sundaresam,:an encoding format for efficient representation and exchange of XML over the Web",*http://www9.org/w9cdrom/154/154.html*

3. D. Huffman, "A Method for Construction ofMinimum-Redundancy Codes", *Proc. of IRE*, September 1952.

4. H. Liefke and D. Suciu. XMill: An Efficient Compressor for XML Data. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 153–164, May 2000.

5. P. G. Howard and J. S. Vitter. Analysis of Arithmetic Coding for Data Compression. In *Proceedings of the IEEE Data Compression Conference*, pages 3–12,April 1991.

6. Tolani P.M. and Haritsa J.R., "XGRIND: a query-friendly XML compressor," in Proc. 2002 Int'l Conf. on DatabaseEng., pp. 225-34

7. World Wide Web Consortium. Document Object Model (DOM) Level 1Speci_cation Version 1.0, W3C Recommendation 1 October, 1998 edition.