ISSN: 2321-7782 (Online)
Volume 2, Issue 1, January 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Paper
Available online at: www.ijarcsms.com

Design and Development of XML Parser Using Tree-Branch Symbiosis Algorithm-Review

Ruchita. A. Kale[1]
Department of Computer Science and Engineering,
Prof Ram Meghe institute of Technology & Research,
Badnera, Amravati University
India

V. M. Deshmukh[2]
Prof.
Prof Ram Meghe institute of Technology & Research,
Badnera, Amravati University
India

Abstract: The extensible markup language XML has become the de facto standard for information representation and interchange on the Internet. As XML becomes widespread it is critical for application developers to understand the operational and performance characteristics of XML processing. The processing of XML documents has been regarded as the performance bottleneck in most systems and applications. XML parsing is a core operation performed on an XML document for it to be accessed and manipulated. Using the tree branch symbiosis algorithms a XML document are parsed and its elements are stored in a single table of database. The nodes need not be read according to their hierarchical structure. In this paper, we proposed the hash function when applied on the database would speed up the access time hence improve the XML processing performance.

Keywords: XML (extensible markup language), Tree-Branch symbiosis, DOM, SAX.

I. INTRODUCTION

XML stands for eXtensible Markup Language. XML is a meta-language derived from Standard Generalized Markup Language (SGMI) and is used to store and exchange structured information. Xml has become a de facto standard for data representation and exchange, XML data processing becomes more and more important for server workloads like Web Servers and Database Servers and also in messaging database and document processing.

XML was designed to provide flexible information identification in web documents. The important role of XML is the representation and exchanging the any kind of structured document because it is platform-independent, human readable and extensible and also its own defined well data format. XML data processing has technologies- DOM and SAX. DOM (Document Object Model) is a platform and language-neutral interface to represent XML document as an object oriented model. DOM usually represented by tree. In the DOM, all data is in memory that's why it is less efficient in the term of storage and time. SAX (Simple API for XML) is an event-driven, serial access mechanism for accessing XML document. A SAX parser reads an XML document as a stream and invokes call back functions provided by the application.

A XML document essentially can be represented as tree structure data model, the XML document thus can be regarded as the serialization of tree model in a depth-first search, left to right traveling order. DOM object is much more popular format for representation of tree structure. A DOM parser is much more complex and much slower. Because DOM parser store entire data in memory, and at the time of data accessing, every time data accessing or reading from the memory and degrading the speed of processing.

If we increase the speed of reading or accessing the data means automatically improve the efficiency. All data is stored in single database rather than multiple databases, the searching or reading data from the database is so fast as compared to the data stored in different database. Data is stored in different database; it required primary key and foreign key for each table for identifying the unique entry as well for the relating two tables.

XML processing occurs in four stages: parsing, access, modification, and serialization. Although parsing is the most expensive operation, there are no detailed studies that compare the processing steps and associated overhead costs of different parsing models, tradeoffs in accessing and modifying parsed data, and XML- based applications' access and modification requirements. Figure-1 explains the three-step parsing process. The first two steps, character conversion and lexical analysis, are usually invariant among different parsing models, while the third step, syntactic analysis, creates data representations based on the parsing model used.

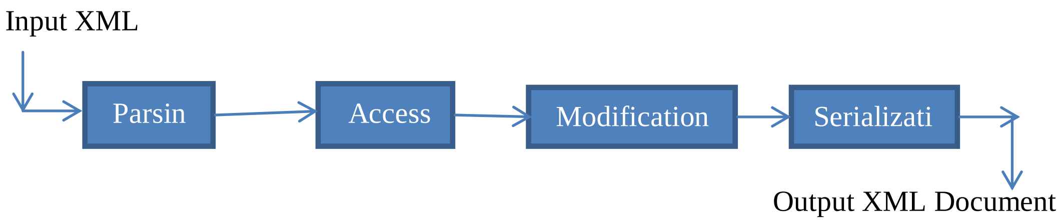

Fig-1 XML processing stages and parsing steps

Storing XML document in a relational database that maintains its nodes and their relationships is a popular way in a business application, which needs high precision of data and their relationship. The Web technology M&S, a series of high performance model for XML processing are needed be created to meet the characteristics if magnanimous data exchange and processing.

The storage of a XML document in a relational database with its original structure in general business application, the several important benefits are-

- A node and its path can be queried easily and fast due to its perspicuity structure.
- The data can be managed by DBMS easily.
- It is useful for data transfer.

II. Related Work

In design and development of XML processing by tree-branch symbiosis algorithm, XML (extensible markup language) language is used for data representation, exchange and document processing. XML parsing is the process of reading an XML document and providing an interface to the user application for accessing the document. Many proposals and models addressed quality of service (QoS) among mobile nodes of the wireless networks and considered the link quality in their designs and architectures.

In 2002, Srikanth Karre et. al explains that the parsing of XML documents can be done using two approaches, Event Based Parsing and Tree Based Parsing. In Event Based Parsing, the XML data is parsed sequentially, one component at a time, and the parsing of events such as the start of a document, or the end of a document are reported directly to the application. SAX (Simple API for XML) is the standard API for event-driven parsing. In Tree Based Parsing, the XML document is compiled into an internal tree structure and stored in main memory [1]. Applications can then use this tree structure for navigation and data extraction. For example, the Document Object Model (DOM) uses tree based parsing, providing a standard set of objects for representing HTML and XML documents, a standard model of how these objects can be combined, and a standard interface for accessing manipulating them.

In 2003, Kai Ning et. al. - Design and Implementation of DTD-based XML parser, the XML-based network management interface testing system, DTD (Document Type Dec1aration) based XML documents should be one of the most typical forms for the management information. XML provides a mechanism to impose validity constraints on the storage layout and logical structure of documents, which is known as the DTD. The purpose of a DTD is to define the document structure with a list of legal predefined entities such as elements or attributes. A DTD can be declared either inline in the XML document or as an external reference [2]. A well-formed XML document is valid if the document has an associated DTD declared inline in the XML document or as an external reference and that it complies with the constraints expressed in DTD. So, both well-formedness checking and validity verifying should be the two necessary features of the DTD-based XML parser [2]. Finally, DTD only supports a limited number of data types, which results in the fact that DTD-based XML parsers cannot check the semantics of data. But the use of the DTD-based XML parser can effectively improve XML documents reliability.

In 2003, Nicola, M. and John, J- XML Parsing: a Threat to Database Performance, explains- study of XML parsing performance across a selection of commercial database applications has shown that XML parsing is the major bottleneck [4]. They found that parsing even small XML documents can increase the computation cost of a database transaction by 2 to 3 times [3].

In 2006, Tong, T. et al, O. studied the XML parser. They conclude the work of parser can read the XML document components via Application Programming Interfaces (APIs) in two approaches. For stream-based approach (also known as event-based parser), it reads through the document and signal the application every time a new component appears. As for tree-based approach, it reads the entire document into a memory resident collection of object as a representation of original document in tree structure [5]. As a result, tree-based approach is not suitable for large-scale XML data because it can easily run out of memory.

In 2007, Wei Lu et. al. states that there are a number of ways to improve XML parsing performance. One of the approaches would be to use pipelining. In this approach, XML parsing could be divided into a number of stages. Each stage would be executed by a different thread. This approach may provide speedup, but software pipelining is often hard to implement well, due to synchronization, load-balance and memory access costs. More promising is a data-parallel approach. Here, the XML document would be divided into some number of chunks, and each thread would work on the chunks independently. As the chunks are parsed, the results are merged [7]. To divide the XML document into chunks, we could simply treat it as a sequence of characters, and then divide the document into equal-sized chunks, assigning one chunk to each thread. This requires that each thread begin parsing from an arbitrary point in the XML document.

In 2007, Yinfei Pan et. al. -A Static Load-Balancing Scheme for Parallel XML Parsing on Multicore CPUs, The advent of multicore machines provides a unique opportunity to improve XML parsing performance. Load-balancing then becomes the key to a scalable parallel algorithm. As far as general parallel algorithms are concerned, static partitioning schemes and dynamic schemes have their own advantages and disadvantages [8]. For parallel XML parsing, the static partitioning scheme introduced is more scalable and efficient than the dynamic scheme on multicore machines, this is due to the fact that most large XML documents usually contain array-like structures and their document structures tend to be shallow. Static load-balancing scheme leverages these characteristics and is designed to quickly locate and partition the array structures.

In 2009, Lan Xiaoji et. al. explains VTD-XML is a new open-source XML parsing model. It centers on a non-extractive XML parsing technique called Virtual Token Descriptor (VTD). It features random access capability, high performance, and low-memory consumption. VTD-XML provides a new parsing way for XML. Because VTD-XML is the open source software for XML, it cannot parse and identify the GML components. Based on this new XML processing model, and combined with the semantic analysis of GML, one GML parsing solution has been designed and implemented via the VTD-XML open-source interface [10].

In 2009, M. Van Cappellen, Z. H. Lui, J. Melton, and Maxim Orgiyan analyses the modification operations are essential to XML files manipulation once they are affected by any increasing amount of data, by the complexity of those operations, and by shorter periods of time needed to process them. Coupled with this data growing, XML documents can reach large number of megabytes (or even gigabytes), limiting and conditioning the technology used for development of applications appealing for XML data processing. Also coupled with the concept of portability, Java programming language provides a set of interfaces allowing for the manipulation of structured documents according to the XML format. Due to their portability, Java and XML are commonly used in application development and in native XML databases for data manipulation [12].

In 2009, Shu Yuan-zhong explains that the default XML parsing Technology is the Document Object Model(DOM), it is a series of mature standards published by W3C. It describes the logical structure, operation and accessing mode of document in detail. The advantage of DOM paring technology is that, because the document is all structured in the memory, the application can make the document navigation and document modification constantly and can access any part of document randomly [13]. DOM also meets some problems which are the document is all structured in the memory, the cost is very high when the document and the tree structure are loaded in it, especially when document is very large. The magnitude of typical DOM tree document is larger than the usual document and will cost much memory. The XML document has to be parsed entirely and cannot be parsed partly. When a subsection of the XML document is needed only, the rest part of creation is wasted. They will never be used. DOM has to read the whole document before application codes take over, which can cause a serious delay to the large document.

In 2010, Gong Li et. al.- Present a XML processing model on data exchange between XML document and relational database, model parsing a XML document to a DOM object, kernel of the model was the tree-branch symbiosis algorithm, by which, the efficiency of DOM building will be promoted significantly. All data was stored in relational database. Using tree branch symbiosis algorithms for XML processing, they shows the comparison between old algorithms and tree branch symbiosis algorithm [14].

In 2013, V.M. Deshmukh et. al. presented the performance study of XML data parsing by evaluating the parsers using time as a parameter. They mainly focused on different data structures which are linear in nature like stack, array , queue and linked list .Data structure based parser works in main memory and uses various data structure for parsing. In the implementation, the proposed parser removes the elements from document and serially checks if the document is well formed or not using Linked list, Queue, Stack and Array simultaneously, which increases its performance over SAX and DOM parser. Proposed the conclusion by observed analysis and graphical results that the data structure based parser is efficient than SAX and DOM parsers. [16]

In 2013, Jie Tang et. al. focused on acceleration of XML parsing through prefetching, by identifying memory access as one of the performance bottlenecks. Also proposed that to make acceleration for XML parsing from memory side by improving its data loading performance. They show that memory-side accelerators deliver considerable effectiveness across existing parsing models. They are able to reduce cache misses by up to 80 percent, which translates into up to 20 percent of performance improvement [17]. They verified the feasibility by checking the implementation impact on bandwidth, energy consumption, and hardware cost. The results show that memory-side acceleration can produce up to 12.77 percent of energy saving when implemented in 32-nm technology.XML parsing performance is hurt by the latency but not by the throughput of the memory subsystem, thus confirming that memory-side acceleration will not likely result in resource contention of memory bus. In conclusion, memory-side acceleration of XML parsing is not only feasible but also effective and efficient.

Li Zhao et. al. Studied Performance Evaluation and Acceleration for XML Data Parsing, they focusing on XML parsing compared the performance of two parsing implementations, namely SAX and DOM in terms of execution time and resource requirements. They found that compared to DOM parsing, SAX parsing is much faster and consumes less memory, but lacks many features like random access and update to the documents. XML data parsing requires a large memory mainly due to three data types: (1) the memory buffer that stores the input data, (2) the tree structure that consists of various element nodes, and (3) the dictionary that stores special strings for the XML document. However, most of these data shows no temporal locality [23]. Therefore a large data cache does not help. One the other hand, instruction cache has a great impact on the performance because the XML data parsing has a very large code size. Therefore, given a total cache size, more area should be devoted to I-Cache than to D-Cache.

III. Parser Overview

In this paper, are providing a comprehensive review of XML parsing for accessing the fast document from database in the term of time.

XML processing by tree branch symbiosis algorithm mainly works on XML file. The important role of XML is the representation and exchanging the any kind of structured document because it is platform-independent, human readable and extensible and also its own defined well data format. XML data processing has technologies- DOM and SAX. But proposed system specially works on DOM parser.

DOM (Document Object Model) is a platform and language-neutral interface to represent XML document as an object oriented model. DOM usually represented by tree. In the DOM, all data is in memory that's why it is less efficient in the term of storage and time.

The important role of XML is the representation and exchanging the any kind of structured document because it is platform-independent, human readable and extensible and also its own defined well data format. An XML file contains a list of elements. The XML parser mostly used DOM techniques for processing. DOM object is much more popular format for representation of tree structure. The DOM parser converts an XML document into a tree structure so that the hierarchical structure of the data can be exposed. For all these procedure firstly, you must create an DocumentBuilderFactory class to get a DocumentBuilder object instance, and use that to produce a document that conforms to the DOM specification.

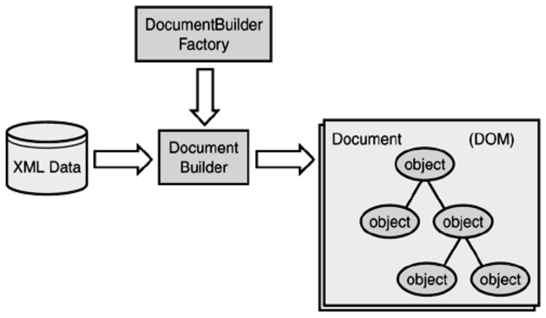

Fig 2. DOM parsing architecture.

Su Cheng Haw et. al. -A Comparative Study and Benchmarking on XML Parsers, they studied different XML parsers and determine the strength and weaknesses of the product. Various studies have been conducted which compare on conformance to standards, speed, memory usage and so on [6].

DOM-

Advantages: Easy navigation- Entire tree loaded into memory, Random access to XML document, Rich set of APIs.

Disadvantages: XML document must be parsed at one time, It is expensive to load entire tree into memory.

SAX-

Advantages: Entire document not loaded into memory which resulting in low memory consumption, Allows registration of multiple Content Handlers.

Disadvantages: No built-in document navigation support, No random access to XML document, No support for modifying XML in place, No support for namespace Scoping.

StAX/Pull-

Advantages: Contains two parsing models, for ease or performance, Application controls parsing, easily supporting multiple inputs, Powerful filtering capabilities provide efficient data retrieval.

Disadvantages: No built-in document navigation support, No random access to XML document, No support for modifying XML in place and is still in an immature state.

IV. CONCLUSION

In Tree-Branch symbiosis XML processing model the nodes of a XML document are stored in a list structure in relational database for the purpose of saving the time cost by the SQL sentences. Although there is a complication of tree building process, this work is done in memory that the cost is almost the same with traditional methods in the precondition of high performance hardware. The Tree-Branch symbiosis mechanism the DOM tree building can have the significant performance improvement. This improvement was only in aspect of total processing time because tree branch symbiosis algorithm for XML processing, arrange the data in the form of AVL tree using hashing technique and all data is stored in single database instead of multiple.

References

1. Srikanth Karre and Sebastian Elbaum," An Empirical Assessment of XML Parsers",2002.
2. Kai Ning, Luoming Meng,"Design and Implementation of DTD-based XML parser", proceedings of ICCT2003.
3. Nicola, M. and John, J., "XML Parsing: a Threat to Database Performance" International Conference on Information and Knowledge Management, 2003, pp. 175-178.
4. Robert A. van Engelen, "Constructing Finite State Automata for High-Performance XMLWeb Services", in the proceedings ofInternational Symposium onWeb Services and Applications (ISWS) 2004.
5. Tong, T. et al, "Rules about XML in XML", Expert Systems with Applications, Vol. 30, No.2, 2006, pp. 397-411.
6. Su Cheng Haw ,G. S. V. Radha Krishna Rao," A Comparative Study and Benchmarking on XML Parsers", Advanced Communication Technology, The 9th International Conference (Volume:1) ISSN :1738-9445 , 2-14 Feb. 2007 pp. 321 – 32.
7. Wei Lu, Dennis Gannon, "Parallel XML Processing by Work Stealing", SOCP'07, June 26, 2007, Monterey, California, USA.
8. Yinfei Pan, Wei Lu, Ying Zhang, Kenneth Chiu,"A Static Load-Balancing Scheme for Parallel XML Parsing on Multicore CPU"s, Seventh IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07) 0-7695-2833-3/07 $20.00 © 2007.
9. Yusof Mohd Kamir, Mat Amin Mat Atar, "High Performance of DOM Technique in XML for Data Retrieval", 2009 International Conference on Information and Multimedia Technology.
10. Lan Xiaoji Su Jianqiang Cai Jinbao, "VTD-XML-based Design and Implementation of GML Parsing Project", IEEE Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on 19 dec 2009 , pp.1 – 5.
11. Xiaosong Li, Hao Wang, Taoying Liu, Wei Li," Key Elements Tracing Method for Parallel XML Parsing in Multi-core System", 2009 International Conference on Parallel and Distributed Computing, Applications and Technologies, 978-0-7695-3914-0/09 $26.00 © 2009 IEEE DOI 10.1109/PDCAT.2009.64
12. M. Van Cappellen, Z. H. Lui, J. Melton, and Maxim Orgiyan, "XQJ - XQuery Java API is Completed", SIMOD Record, vol. 38, no. 4, 2009.
13. Shu Yuan-zhong,"Research of optimizing device description technology based on XML in EPA" 2009 Second International Symposium on Electronic Commerce and Security.
14. Gong Li and Liu Gao-Feng, Liu Zhong and An Ru-Kui, "XML Processing by Tree-Branch symbiosis algorithm", 2010 2nd International Conference on Future Computer and Communication, Volume 1.
15. V.M. Deshmukh, G.R. Bamnote, "DESIGN AND DEVELOPMENT OF AN EFFICIENT XML PARSING ALGORITHM: A REVIEW", International Journal of Applied Science and Advance Technologyz, January-June 2012, Vol. 1, No. 1, pp. 5-8.
16. Ms. V.M.Deshmukh, Dr. G.R.Bamnote, "An Empirical Study: XML Parsing using Various Data Structures", International Journal of Computer Science and Applications, Vol. 6, No.2, Apr 2013.
17. Jie Tang, Shaoshan Liu, Chen Liu, Zhimin Gu, and Jean-Luc Gaudiot," Acceleration of XML Parsing through Prefetching", IEEE TRANSACTIONS ON COMPUTERS, VOL. 62, NO. 8, AUGUST 2013.
18. Mohammad Khabbaz, Dirar Assi,Reda Alhaj, Moustafa Hammad," Parse Tree Based Approach for Processing XML Streams", IEEE IRI 2013, August 14-16, 2013, San Francisco, California, USA 978-1-4799-1050-2/13/$31.00 ©2013 IEEE

19. Bruno Oliveira1,Vasco Santos1 and Orlando Belo2," Processing XML with Java – A Performance Benchmark", International Journal of New Computer Architectures and their Applications (IJNCAA) 3(1): 72-85 The Society of Digital Information and Wireless Communications (SDIWC) 2013 (ISSN: 2220-9085) ,pp. 72-85.
20. Michael R. Head† Madhusudhan Govindaraju, "Parallel Processing of Large-Scale XML-Based Application Documents on Multi-core Architectures with PiXiMaL", Fourth IEEE International Conference on eScience.
21. Wei Lu , Kenneth Chiu, Yinfei Pan, "A Parallel Approach to XML Parsing".
22. Chengkai Li,"XML Parsing, SAX/DOM".
23. Li Zhao , Laxmi Bhuyan," Performance Evaluation and Acceleration for XML Data Parsing".
24. W3C, "Extensible Markup Language (XML)". [Online]. Available: http://www.w3.org/XML.
25. AVL Binary Search Tree. [online]. Available: en.wikipedia.org/wiki/AVL_tree.
26. Available:http://www.webopedia.com/TERM/H/hashing.html
27. Hashing techniques [Online]. Available:https://www.cs.tcd.ie/Owen.Conlan/4d2/4D2 5&6_Hashing_Techniques_v1.02.pdf

AUTHOR(S) PROFILE

Ms. R. A. Kale received the B.E. degrees in Computer Technology from Yeshwantrao Chavan College of Engineering, Wanadongri, Nagpur in 2011.Now I am pursing ME(CSE) from Prof. Ram Meghe Institute Of Technology & Research, Banera, Amravati.

Prof. Ms. V.M.Deshmukh completed her B.E (CSE) from College of Engineering, Badnera in 1990 and M.E (CSE) from College of Engineering, Badnera in 2007. Pursuing Ph.D in Computer Science & Engineering (Design and development of XML parser).