

## **Performance Evaluation of XML Parsing for Tree-Branch Symbiosis Algorithm**

Ms. Ruchita A Kale<sup>1</sup>, Prof. Ms. V. M. Deshmukh<sup>2</sup>

<sup>1</sup>*Department of Computer Science & Engineering, SGBAU, India, kaleruchita11@gmail.com*

<sup>2</sup>*Department of Computer Science & Engineering, SGBAU, India, msvmdeshmukh@rediffmail.com*

---

**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 (SGML) 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.

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.

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**

### **A. XML Parser Overview**

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 2006, Tong, T. et al, 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.

Chengkai Li- XML Parsing, SAX/DOM, explains that in every application that takes XML document to process, parsing is the first important step to be done. Also explains that DOM previously was used for modeling HTML after that the different levels of DOM were proposed and that has been used to parse the XML document which models the XML document as a tree of node where the application can read, write and update the contents of the nodes whereas SAX (Simple API for XML) is a event driven interface which reads the XML document, generates the events and triggers the corresponding XML handlers [24].

### **B. Parallel XML Parsing**

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, and 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.

Since an XML document is the serialization of a tree-structured data model (called XML Infoset) traversed in left-to-right, depth-first order, such a division will create chunks corresponding to arbitrary parts of the tree, and thus the parsing results will be difficult to merge back into a single tree. Correctly reconstructing namespace scopes [7].The results of parsing XML can vary from a

DOM-style, data structure representing the XML document, to a sequence of events manifest as callbacks, as in SAX-style parsing. The parallel approach focuses on DOM-style parsing, where a tree data structure is created in memory that represents the document. Their implementation is based on the production quality libXML2 parser, which shows that their work applies to real-world parsers, not just research implementation.

### C. Evaluating performance Using Different Techniques Or Algorithm-

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. [18].

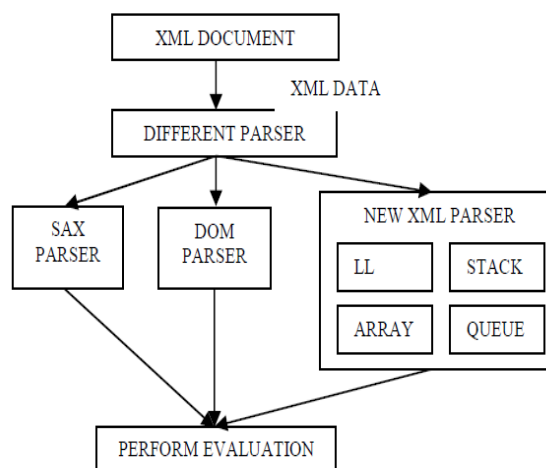


Fig-2.1 Architecture View for parser using data structure

They compare the performance of different parsing like DOM, SAX and also the different data structure. Using the different data structure for accessing or reading the data from the database required less time as compared to others method. Also they explain the design and development of an efficient XML parsing algorithm, Parsing is a core operation performed before an XML document can be navigated, queried, or manipulated. Recently, high performance XML parsing has become a topic of considerable interest.

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. The processing of XML document s has been regarded as the performance bottleneck in most systems and applications. A number of techniques have been developed and improve the performance of XML processing, ranging from the schema-specific model to the streaming-based model to the hardware acceleration. These methods only address parsing and scheduling the XML document in memory. Although there are a few of works have discussed the efficiency of the data read- write between XML and Relational Database, they constructed the DOM and reading relational database synchronously and neglected the differences of pace between DOM ( a general format of XML document in memory) building and relational database reading, which will reduce the performance of the entire system.



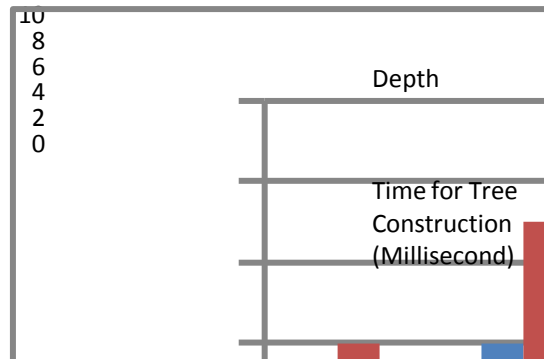


Fig-3.3 Performances of Tree-Branch Symbiosis Algorithm

#### 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 XML Web Services", in the proceedings of International Symposium on Web 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. Sebastian Graf, Marc Kramis, and Marcel Waldvogel, "Distributing XML with Focus on Parallel Evaluation", Databases, Information systems, and Peer-to-Peer computing, Sixth International Workshops, DBISP2P 2008, Auckland, New Zealand, August 23, 2008.
10. B.Naga malleswara Rao, N.Samba Siva Rao, V. Khanaa, "Exploiting XML Dom for Restricted Access of Information", International Journal of Recent Trends in Engineering, Vol 2, No. 4, November 2009.
11. 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.



12. 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.
13. 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
14. M. Van Cappellen, Z. H. Lui, J. Melton, and Maxim Orgiyan, “XQJ - XQuery Java API is Completed”, SIMOD Record, vol. 38, no. 4, 2009.
15. ] Shu Yuan-zhong, ”Research of optimizing device description technology based on XML in EPA” 2009 Second International Symposium on Electronic Commerce and Security.
16. 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.
17. 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. 58.
18. 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.
19. 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
20. 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
21. Bruno Oliveira<sup>1</sup>, Vasco Santos<sup>1</sup> and Orlando Belo<sup>2</sup>, ” *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.
22. 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.
23. Wei Lu , Kenneth Chiu, Yinfei Pan, “A Parallel Approach to XML Parsing”.
24. Chengkai Li, ”XML Parsing, SAX/DOM”.
25. Li Zhao , Laxmi Bhuyan, ” *Performance Evaluation and Acceleration for XML Data Parsing*”.
26. W3C, “Extensible Markup Language (XML)”. [Online].  
Available: <http://www.w3.org/XML>.
27. AVL Binary Search Tree. [online].  
Available: [en.wikipedia.org/wiki/AVL\\_tree](http://en.wikipedia.org/wiki/AVL_tree).
28. Available: <http://www.webopedia.com/TERM/H/hashng.html>
29. Hashing techniques [Online]. Available: [https://www.cs.tcd.ie/Owen.Conlan/4d2/4D2\\_5&6\\_Hashing\\_Techniques\\_v1.02.pdf](https://www.cs.tcd.ie/Owen.Conlan/4d2/4D2_5&6_Hashing_Techniques_v1.02.pdf)
30. XML Pull Parser, <http://www.extreme.indiana.edu/xgws/xsoap/xpp/>

**Author's Profile**

	<p>Ms. R. A. Kale received the B.E.degrees in Computer Technology from YeshwantraoChavan College of Engineering ,Wanadongri, Nagpur in 2011.Now I am pursuing ME(CSE) from Prof. Ram Meghe Institute Of Technology &amp; Research ,Badnera, A mravati.</p>
	<p>Prof. Ms. V.M.Deshmukh mpleted 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 &amp; Engineering (Design and development of XML parser).</p>