IMPORTANT ASPECT FOR DESIGNING AN EFFICIENT XML PARSING TECHNIQUE: A REVIEW

Swati Gupta[1], Prof. Y.P. Singh[2]
School of Engineering, University of Technology, Jaipur

Abstract: Internet has become the main aspect for searching and transfer of information. For the same reason, it requires a language to represent and transfer information. XML (eXtensible Markup Language) acts as the standard for representing and interchanging data and information on web. The core operation performed on XML document is parsing. Parsing is used to access and manipulate XML data. But this parsing process is also one of the biggest hindrance in the development of XML. This paper is a review of different parsing techniques which are used to perform the core operation on XML document.

Keywords: XML Parsing, DOM, SAX, Data Structure Parsing, Database Parsing

I. INTRODUCTION

XML stands for eXtensible Markup Language, is derived from SGML (Standard Generalized Markup Language) and is a meta-language. It is broadly used for interchange and representation of data on web. XML is playing important role in web services and commercial workloads. Because of this it is important for data servers and web servers to process XML document. Processing of XML file completes in four steps namely Parsing, Access, Modification and Serialization.

Input XML Document
Parsing
Access
Modification
Serialization
Output XML Documet

Figure 1. XML Processing Stages

Parsing is the core operation performed on XML document. Though XML is easy to understand, its parsing causes various bottlenecks in the real-world environment [9]. Small files can be parsed easily but for large file size, it is difficult to parse them without degrading the performance of the parser. By keeping this in mind, parallel processing is used to boost the performance of XML parsing of large files [7,8]. There are different models for XML parsing. Document Object Model (DOM) uses tree structure for storing parsed data, Virtual Token Descriptor (VTD) uses integer array, Simple API for XML (SAX) and Streaming API for XML (StAX) create sequence of events for parsing. VTD did not create any object as it works via integer array while DOM, SAX, StAX creates objects for the representation of data.

II. LITERATURE REVIEW

Parsing is the main operation performed on XML document before navigation, query and manipulation. Also, high performance XML parsing is used to carry the operation [8].

In 2003, Nicola, M. and John, J. explains bottleneck of XML parsing across few commercial database applications. For a database transaction, the computational cost is increased twice or trice by parsing even a small XML document [9]. In 2006, Tong, T. et al, gave their reviews about the XML parser. According to their study, tree-based approach is not feasible for large size XML documents because it requires memory to store the entire document and for bigger files, it can easily run out of memory [4]. In 2009, Lan Xiaoji et. al. explains a new parsing model named VTD-XML. It is an open source model, based on the technique called Virtual Token Descriptor (VTD). Major characteristics of this model is random access capability, high performance and low memory usage [11].

In 2010, Gong Li et. al. suggests a model to perform data exchange between XML document and relational database. With the help of tree-branch symbiosis algorithm, the efficiency of DOM building will be promoted significantly [12].

In 2012, Ma Jianliang et. al. proposed a design of a parallel speculative DOM-based XML parser (PSDXP) which is implemented with the help of Field Programmable Gate Array (FPGA). This design is tested for both two threads parallelism and four threads parallelism [13].

In 2015, V.M.Deshmukh and G.R. Bamnote perform an empirical analysis of different parsers (DOM,SAX, PULL Parser, VTD etc.) for an android based application. Also, they proposed a new model SRDOM which is based on structure recurrence and suggested that SRDOM is 9 times faster as compared to the DOM parser in presence of redundant structure [14].

In 2016, Rashmi P.Sonar and M.S. Ali presents detail analysis of XML parser for embedded systems. They proposed iXML model for embedded systems which uses inline method for parsing. It identifies the structure of the document and then the attributes of the tokens are identified. Further, attributes are checked for the well formedness of the document. Significant improvement in terms of time is measured using this novel approach [15].

III. PARSER OVERVIEW

Various Parsing Techniques

Mainly two technologies are used for XML parsing of data. First one is SAX (Simple API for XML) [1] and the other is DOM (Document Object Model) [2]. SAX is an event-based technique which uses serial-access mechanism for parsing the XML document. It considers XML data as stream and callback functions are invoked by the application for processing. On the other hand, DOM is an object-oriented model. Data structure is used to represent parsing of XML document. It uses tree structure to represent parsed data and once it is created, further manipulations are done dynamically on the data structure. If both the mechanisms are compared, DOM is much complex and hence slower than SAX.To increase the performance time in DOM parser, ache is used as a temporary file [5]. Data is stored in cache file after fetched from the main database. This approach is very efficient as compared to the initial model as it reduces the fetching time of data, hence increasing the overall performance.

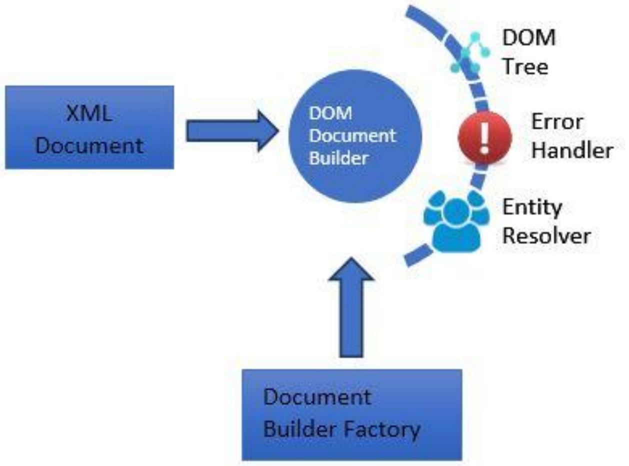

Figure2. DOM Parsing Architecture

SAX is event based and allows to process data while reading, which avoids the waiting time till the whole document is stored in the memory. But events are generated by fetching data from secondary memory and unfortunately, secondary memory is slow.

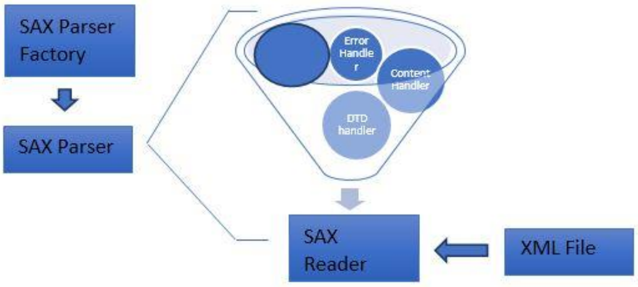

Figure 3. SAX Parsing Architecture

Parallel XML Parsing

There are various techniques to improve parsing performance of XML document. One of them is pipelining [3]. In this, parsing is divided into different stages and for each stage one thread has been assigned which execute that stage. This will minimize the time required for parsing the document but at the same time it is quite difficult to implement pipelining in software because of synchronization, high costing for memory and load-balancing. To overcome this, data parallel approach is used. In this, the document is divided into small chunks and each chunk is implemented by different thread. Before dividing the document into chunks, the XML document is passed from a pre-parsing phase to determine the document structure. After that it is passed by full parser [8,17]. The result corresponding to the parsing of chunks are merged. But it is difficult to merge it into a single tree because the chunks are created corresponding to arbitrary parts of the tree. Parallel parsing focuses on DOM parsing. In DOM, tree data structure is used to store the document in the memory.

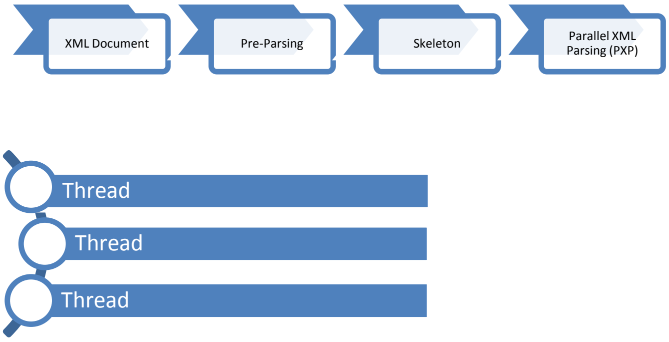

Figure 4. The PXP Architecture

XML Database Parsing

With respect to transactional database processing, XML parsing is having poor characteristic graph of performance. Yet its poor impact is not taken into consideration. For XML deployed database applications, XML parsing proves as a major hindrance. Processing model is available to transfer data between XML and relational database [6,9]. In this model, single table is used to store parsed data. Also, for storing and enforcing data definition, XML can be used. It provides a synergetic framework for Knowledge Management (KM) tools [16].

XML Parsing using Data Structure

The main task of XML parser is to check the document for its well form. It read the document, check the syntax and generate error report. The syntactically correct document is termed as well-formed XML document. It requires a root element, a start and an end tag, other tags should be properly nested and attribute values in quotes. XML DOM and SAX parser access the content of the document programmatically using language. DOM parser generates a tree structure to store parsed data in the memory. SAX parser is based on events for parsing the document. An XML parser reads a document from starting and extract tokens like start and end tag. The document can be parsed with the help of data structure. For example, DOM tree is built by extracting start and end token of the XML document. Further stack (S) can be used to store information of all the ancestors (in DOM tree) of current element. While reading the document, if a start tag appears, its corresponding DOM node is created, and attribute value associated with the element is parsed and stored in the memory. When end tag is read, it is matched with the top of the stack. If the element is same, popping is done, and parsing continues. Otherwise the process is aborted as the XML document is not formed well. After finishing the reading of the entire document, the stack S should be empty for the document to be well formed. Various data structures like queue, linked list, array and stack are used for parsing XML document. [10] shows that data structure-based parser works in main memory and hence increases its performance as compared to SAX and DOM parser.

IV. CONCLUSION AND FUTURE WORK

This paper reviews problems related to different XML parsers. It also discusses various aspects of developing and designing an efficient XML parsing algorithm. It also analyzes the performance of XML parsing using different data structures. In future work, we can carry out comparative study of XML parser using multiple data structures. We can also perform empirical analysis of different XML parsers to check their performance for multiple data structures.

REFERENCES

[1] W3C, “Extensible Markup Language (XML)”. [Online]. Available: http://www.w3.org/XML.

[2] W3C, “Document Object Model (DOM) Level 2 Core Specification”. [Online]. Available: http://www.w3.org/TR/ DOM-Level-2-Core.

[3] Wei Lu, Dennis Gannon, ―Parallel XML Processing by Work Stealing”, SOCP'07, June 26, 2007, Monterey, California, USA.

[4] Tong, T. et al, “Rules about XML in XML”, Expert Systems with Applications, Vol. 30, No.2, 2006, pp. 397-411.

[5] Yusof Mohd Kamir, Mat Amin Mat Atar, “High Performance of DOM Technique in XML for Data Retrieval”, in International Conference on Information and Multimedia Technology IEEE, 2009.

[6] Li Gong, Liu Gao-Feng, Liu Zhong, Ru-Kui. “XML Processing by Tree-Branch Symbiosis Algorithm”, in 2nd International Conference on Future Computer and Communication. IEEE 2010.

[7] Lu W., Dennis Gannon “Parallel XML Processing by Work Stealing”, in High Performance Distributed Computing Archive Proceedings of the 2007 Workshop on ServiceOriented Computing Performance. 2008. Monterey Clifornia USA pp. 31-38.

[8] Lu W., Y. Pan, and K. Chiu, “A Parallel Approach to XML Parsing”, in the 7th International Conference on Grid Computing, IEEE/ACM 2006.

[9] Nicola M. and J. John, “XML Parsing: A Threat to Database Performance”, in Proc. of the12th International Conference on Information and Knowledge Management, pp. 175-178, 2003.

[10] V.M.Deshmukh, G.R. Bamnote, “An Empirical Study: XML parsing using Various Data Structures”, in International Journal of Computer Science and Applications, 6(2), 2013, pp.400-405.

[11] 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.

[12] 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.

[13] Ma Jianliang, Shaobin Zhang, Tongsen Hu, “Parallel Speculative Dom-based XML Parser”, in 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), June 2012.

[14] V.M.Deshmukh, G.R. Bamnote, “An Empirical Study of XML Parsers across Applications”, in 2015 International Conference onComputing Communication Control and Automation (ICCUBEA), Feb 2015.

[15] Rashmi P.Sonar and M.S.Ali, “ iXML – generation of efficient XML parser for embedded system”, 2016 International conference on Computing Communication Control and Automation (ICCUBEA), Aug 2016.

[16] James R. Otto, James H. Cook and Q.B. Chung, “Extensible Markup Language and Knowledge Management”, in Journal of Knowledge Management, 5(3), 2001, pp. 278- 284, MCB University Press.

[17] Y. Pan, W. Lu, Y. Zhang, and K. Chiu. A Static Load-Balancing Scheme for Parallel XML Parsing on Multicore CPUs. In Proc. of the 7th International Symposium on Cluster Computing and the Grid (CCGRID), pages 351–362, Washington D.C, May 2007.