International Journal of Applied Science and Advance Technology
January-June 2012, Vol. 1, No. 1, pp. 5-8

DESIGN AND DEVELOPMENT OF AN EFFICIENT XML PARSING ALGORITHM: A REVIEW

V.M. Deshmukh[1] and G.R. Bamnote[2]

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. XML processing occurs in four stages: parsing, access, modification, and serialization. Parsing is an expensive operation that can degrade XML processing performance. This paper is a literature review of the various issues involved in XML parsing. The aim of this paper is to review the various design issues involved in XML parsing with an emphasis on the data structures used.

Keywords: XML parsing, DOM, SAX.

1. INTRODUCTION

XML (Extensible Markup Language) [1] was designed to provide flexible information identification in web documents. However it has come to play an increasingly important role in representation and exchange of any kind of structured document because it is platform-independent, human readable and extensible. As commercial workloads and web services rely more and more on XML for data storage and communication, XML data processing becomes an important workload for web servers, database servers, etc.

Recent years have witnessed a multitude of applications and systems that use XML such as web services and service oriented architectures (SOAs) [16], grid computing, RSS feeds, ecommerce sites, and most recently the Office Open XML document standard (OOXML) [12]. Parsing is a core operation performed before an XML document can be navigated, queried, or manipulated. Though XML is simple to read and process by software, XML parsing is often reported to cause performance bottlenecks for real-world applications [9]. For example, in a SOA using web services technology, services are discovered, described, and invoked using XML messages [17]. These messages can reach up to several megabytes in size, and thus parsing can cause severe scalability problems. With the emergence of large-scale throughput oriented multi-core processors parallelism is a natural way to boost the performance of XML parsing. Leveraging multi-core processors can offer a cost-effective way to overcome the scalability problems, given that future multi-core processors will support hundreds of cores, and thus, offer a high degree of parallelism in hardware [7, 8].

XML parsers use different models to create data representations. DOM (Document Object Model) creates a tree object, VTD (Virtual Token Descriptor) creates integer arrays, and SAX (Simple API for XML) and StAX (Streaming API for XML) create a sequence of events. Both DOM and VTD maintain long-lived structural data for sophisticated operations in the access and modification stages, while SAX and StAX do not. DOM as well as SAX and StAX create objects for their data representations, while VTD eliminates the object-creation overhead via integer arrays.

2. MOTIVATION

A number of techniques have been developed to improve the performance of XML processing, ranging from the schema-specific model [13,15] to the streaming-based model [14] to the hardware acceleration [8]. These methods only address parsing and scheduling the XML document in memory. XML documents when parsed are represented as a hierarchical tree structure in memory. DOM is a tree based model where the document's data resides in memory. The parallel processing model uses queues as a data structure for work stealing. The schema specific parsing method uses a two stack PDA (Push Down Automaton).This motivates to think of using various data structures to evaluate the performance of XML document parsing.

3. LITERATURE REVIEW

Parsing is a core operation performed before an XML document can be navigated, queried, or manipulated.

[1] Department of Computer Science and Engineering, Prof. Ram Meghe Institute of Technology and Research Badnera (Rly), Amravati, *E-mail: msvmdeshmukh@rediffmail.com*

[2] Department of Computer Science and Engineering, Prof. Ram Meghe Institute of Technology and Research Badnera (Rly), Amravati, *E-mail: grbamnote@rediffmail.com*

Recently, high performance XML parsing has become a topic of considerable interest [8].

3.1 Overview of XML Parsing Technologies

There are two technologies for XML data parsing. One is SAX (Simple API for XML) [1] and the other is DOM (Document Object Model) [2]. SAX is an event-driven, serial-access mechanism for accessing XML documents. A SAX parser reads an XML document as a stream and invokes callback functions provided by the application. On the other hand, DOM is a platform and language-neutral interface to represent XML document as an object-oriented model, which is usually a tree. Once the tree is built up, it allows applications to dynamically access and update its content as well as its structure. Compared to a SAX parser, a DOM parser is much more complex, and hence, much slower. In DOM the data retrieval time can be improved by using cache as a temporary file [5]. After retrieving data from database, the data is saved at cache. This model is very efficient compared to old model and gives better performance for data retrieval. A different approach described SEDOM [4], based on a new compression approach and a set of manipulation algorithms, which enable many DOM operations to be performed when the data are in the compressed format, and allow individual parts of a document to be compressed, decompressed and manipulated. It can be used to efficiently manipulate very large XML documents. The SAX like validating XML parser [13] uses a schema-specific approach. The schema compiler first transforms the schema into an intermediate representation called generalized automata which abstracts the computations required to parse XML documents.

3.2 Parallel XML Parsing

Parallel XML parsing (PXP) leverages the growing prevalence of multicore architectures in all sectors of the computer market and yields significant performance improvements. The parallel XML parser consists of an initial preparsing phase to determine the structure of an XML document followed by a full parallel parse [8] as in Figure 1.

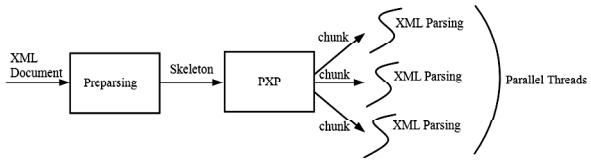

Figure 1: The PXP Architecture

The results of preparsing are then used to help partition the XML document for data parallel processing. 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. This parallel approach focuses on DOM-style parsing where a tree data structure is created in memory that represents the document.

The same research is being extended further to design a general model. The kernel of the model is a stealing-based dynamic load balancing mechanism by which multiple threads are able to process disjointed parts of the XML document in parallel with balanced load distribution [7]. The model also provides a novel mechanism to trace the stealing actions, and the equivalent sequential results are obtained by gluing the multiple parallel-running results together. The basic idea of stealing based scheme is that every thread works on its own local task queue and whenever it runs out of the task it steals the task from other thread's task queue. This model uses queue as a data structure as shown in Figure 2.

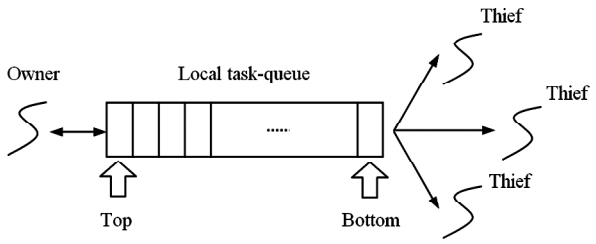

Figure 2: The Deque Structure as the Local Task-queue in the ThreadCrew

The Key Element Parse Tracing (KEPT) method [18] of parallel XML parsing parallelizes the pre-parsing and parsing at element level. It remolds the pre-parsing as a key element extracting process and schedules the processing of key elements in the framework of KEPT. Then parsing process is parallelized as a whole as shown in Figure 3.

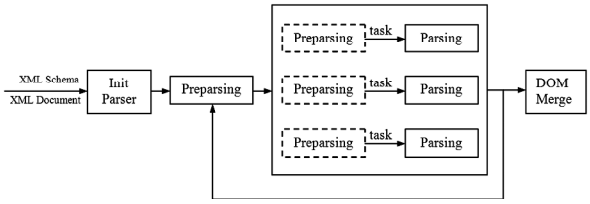

Figure 3: KEPT Architecture

3.3 XML Parsing in Databases

XML parsing is generally known to have poor performance characteristics relative to transactional database processing. Yet, its potentially fatal impact on overall database performance is being underestimated. XML parsing performance is a key obstacle to a successful XML deployment for real-world database applications. There is a considerable share of XML database applications which are prone to fail at an early and simple road block: XML parsing [9]. Processing model for storing and building XML

document in data transfer between XML and relational database is available [6]. In this model, a XML document is parsed and its elements are stored in a single table of database instead, it is not necessary to read the nodes according to their hierarchical structure, thus leveraging the workload of DOM building to memory by the algorithm called Tree-Branch inter growth. XML can be used to both store and enforce organizational data definitions, thus providing a synergetic framework for leveraging the potential of knowledge management (KM) tools [19]. XML provides a flexible markup standard for representing data models. KM provides IT processes for capturing, maintaining, and using information.

3.4 XML Web Services for Real-time Applications

Service-Oriented Architecture (SOA) is the most widely used methodologies for building and integrating different types of software applications. This because the extreme benefits that it offers to their adopters including agility, dynamicity, and loose-coupling. XML Web Services is the most used technology for realizing SOA because it is easy to use [12]. However XML Web Services suffers from a number of drawbacks such as low performance, bad utilization of hardware resources, and network latency. . These pitfalls may prevent some users from utilizing SOA in some large systems. Therefore these issues need to be addressed first and resolved before leveraging it into real time systems.

Over the past few years, business-to-business (B2B) e-commerce has grown exponentially and for this companies need a common language. XML is the most natural evolution of Web technologies to support modern applications such as e-commerce [3]. Hence it is extremely necessary to review the trends in B2B e-commerce and understand the critical needs to support constraints and the integrity of data. A new XML parsing method based on deterministic finite state automata (DFA) [15] is also proposed. A DFA generator is described that automatically translates XML Schemas to DFAs for efficient parsing of XML documents and SOAP/XML messages.

4. CONCLUSION AND FUTURE WORK

This paper reviews issues involved in XML parsing .It examines several techniques of XML parsing. It also discusses various aspects in design and development of an efficient XML parsing algorithm. In future work we plan to improve and analyse the performance of XML parsing techniques employing various data structures. Also further design and develop an efficient XML parser using multiple data structures. We will also try to carry out comparative study of various data structures used in XML parsing across applications.

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] Billy B.L. Lim, H. Joseph Wen, "The Impact of Next Generation XML", *in Journal of Information Management and Computer Security*, **10(1)**, 2002, pp. 33-40, MCB University Press.

[4] Fangju Wang, Jing Li, Hooman Homayounfar, "A Space Efficient XML DOM Parser", *in Journal of Data and Knowledge Engineering*, **60(1)**, January 2007, pp. 185-207.

[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 Service-Oriented 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] Power James F., Brian A. Malloy. "Program Annotation in XML: A Parse-tree Based Approach", *in Proceedings of the Ninth Working Conference on Reverse Engineering (WCRE'02)* 2002.

[11] Seung Min Kim, Suk I. Yoo, "DOM Tree Browsing of a Very Large XML Document: Design and Implementation", *in Journal of Systems and Software*, **82(11)**, November 2009, pp. 1843-1858.

[12] Hazem M., El-Bakry and Nikos Mastorakis, "Performance Evaluation of XML Web Services for Real-Time Applications", *in International Journal of Communications*, **3(2)**, 2009.

[13] Gao Z., Y. Pan, Y. Zhang, and K. Chiu. "A High Performance Schema-Specific XML Parser", *IEEE Intl. Conf. on e-Science and Grid Computing*, pp. 245-252, Dec. 2007.

[14] Lu W., K. Chiu, A. Slominski, and D. Gannon, "A Streaming Validation Model for SOAP Digital Signature", *in 14th IEEE International Symposium on High Performance Distributed Computing (HPDC-14)* July 2005.

[15] Engelen Robert A. van, "Constructing Finite State Automata for High Performance Web Services", *in Proceedings of the International Symposium on Web Services(ISWS)*, 2004.

[16] M. Huhns and M.P. Singh. "Service-Oriented Computing: Key Concepts and Principles". *IEEE Internet Computing*, **9(1)**, 75-81, Jan. 2005.

[17] R.A.V. Engelen. "A Framework for Service-oriented Computing with C and C++ Web service Components". *ACM Transactions on Internet Technology*, **8(3)**, pp. 1-25, 2008.

[18] Li Xiaosong, Hao Wang, Taoying Liu, Wei Li, "Key Elements Tracing Method for Parallel XML Parsing in Multi-core System", *in International Conference on Parallel and Distributed Computing, Applications and Technologies*, IEEE, 2009.

[19] 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.

[20] Fernando Farfán, Vagelis Hristidis, Raju Rangaswami, "2LP Double-lazy XML Parser" *in Journal of Information Systems*, **34(1)**, March 2009, pp. 145-163.

[21] Giorgio Busatto, Markus Lohrey, Sebastian Maneth, "Efficient Memory Representation of XML Document Trees", *in Journal of Information Systems*, **33(4-5)**, June-July 2008, pp. 456-474.

[22] Pan Yinfei, Ying Zhang, Kenneth Chiu, "Hybrid Parallelism for XML SAX Parsing", *in International Conference on Web Services IEEE*, 2008.