2009 International Conference on Computer Engineering and Applications
IPCSIT vol.2 (2011) © (2011) IACSIT Press, Singapore

A Proposed Petri Nets Model for Updating XML Documents on a Multicore Architecture

Alaa M. Wadi [1] , Sai Peck Lee [2]

[1] Faculty of Computer Science & Information Technology, University of Malaya[1]

[2] Faculty of Computer Science & Information Technology, University of Malaya

Abstract. Updating XML documents is getting more and more attention from both academic and commercial fields. The time needed to update XML documents is a key factor for the success of any application especially for those related with the e-commerce. In this article, we suggest using Petri Nets as a tool for designing a concurrent model that is able to achieve XML update operations. Also, we suggest using the same tool for validating these operations. Finally, we propose to apply the concurrent model by using multithreading on a Multicore Architecture. We plan to prove that these models can minimize the time spent in updating and validating XML documents

Keywords: XML Update, Petri Nets, Multicore.

1. Introduction

XML, Extensible Mark-up Language, has become the standard format for storing and exchanging information on the Internet. The importance of updating information has increased due to the increased usage of XML as a standard format of data. In the past, XML update was considered as unnecessary operation. Replacing an old XML document with a new one was the solution for updating; this solution is no longer acceptable. Many techniques have been developed to carry out these operations on XML documents [1].

On the hardware side, the world has witnessed the emergence of Multicore software. Due to physical limitations, the clock speed of microprocessors didn't show any notable improvement within the last six years. This has led vendors to change their interest to multiprocessing industry with multi core per chip instead of improving the clock speed of a single processor. The new tendency is to double the number of cores with each generation. This has its influence on software industry since Multicore programming differs from traditional linear programming. As a result, software developers have to adapt their software to the new architecture [2].

Combining our problem "XML update" with the new architecture imposes using a modelling tool that abstracts away inessential properties and let us focus on what we are about more precisely which is a concurrent model for updating and validating XML documents. In this study, we suggest using Petri Nets as a modelling tool.

Petri Nets can be defined as a graphical and mathematical tool for modelling many systems. They are widely used to model asynchronous concurrent systems that have parallelism, synchronization, and resource sharing. Petri Nets also can model both the sequential and concurrent events for real time systems besides allowing us to discover the weak points in any system especially deadlock points [3] [4] [5].

To the best of our knowledge, no work to date has investigated the modelling of updating XML documents using Petri Nets.

[1] Alaa M. Wadi, Email: alaa@perdana.um.edu.my.
Sai Peck Lee, Tel: +60379676305, Email: saipeck@um.edu.my.

2. Problem Statements & Motivation

The importance of XML update has increased since XML has moved from a publishing language to a data format. Many researches have been done regarding XML storage, indexing and querying processing. However, XML updating remains the weakness in many XML databases. The increase in the size of XML documents along with the non-convincible performance of the single processor architecture have led to a bottleneck stage in most systems and applications. This has a bad impact to the end users because they have to spend more time especially when they access web sites that are frequently change, as a consequence, governments, companies, institutes…etc, will suffer because their electronic transactions will take a long time.

On the hardware front, the Multicore processor quickly becomes the mainstream on desktop-computing machines with dual-core, quad-core producing now and 16 core system within two or three years. It seems that the time when the programmer depends on the faster clock to speed up the execution has ended. On the software front, the rising of Multicore technology creates a great impact on the software industry. Soon or later the design of all the software will change to be compatible with the Multicore architecture. Some software companies have adapted their software to be compatible with new technology by exploiting concurrency and parallelism so they can utilize the new architecture otherwise their products will not find their way in the software market. XML update as any other technology has its share in this race. Unfortunately, the current approaches in XML updating didn't show any cooperation with the Multicore architecture besides that the majority of these researches deal with XML documents by using serial techniques [6]. We conclude that there should be a study for devising a new technique that uses concurrency to update XML documents on the Multicore architecture.

The motivation behind this study is to design and apply a concurrent model that can minimize the time in updating XML documents on a Multicore architecture without breaking the validity of these documents.

3. Related Works

XML update can be categorized as follows [7]:

A- XML Updates in Database Management Systems (DBMS):

The process of storing XML documents in a DBMS can be straight forward. We can make use of all the tools to manage the documents, updating is one of them, once the documents are stored in the DBMS. It's obvious that the structure of the XML document will be mapped into tables and classes. This makes it possible to use the same language both for managing database and the XML documents. [8], [9] and [10] discussed storing XML documents into RDBMS, OODBMS and ORDBMS respectively.

Mapping XML documents into a DBMS shows some weaknesses. Firstly, the mapping itself may be not correct, as a consequence, the updating may not be correct. Secondly, highly complicated structure of XML documents makes the process of mapping so expensive which leads to altering the semantics of the documents. Finally, even if the process of mapping is done very well, using SQL as a language for querying XML documents couldn't totally succeed because SQL has been designed to deal with specific data format like tables.

The problem of mismatching between XML nature and SQL has been investigated by [11] [12] [13] [14] [15]. They suggested translating XML query language into a language similar to SQL. However, they didn't cover all the update operations, besides that, they couldn't define a standard XML query language.

B- XML Updates in Native XML Database (NXD):

The process of updating XML documents in NXD is still under investigation [16]. [15] suggested including simple update operation inside XQuery. Many NXD products use their own properties for updating these documents inside servers. XQuery embedded the update facilities (insert, delete, replace and rename nodes) into the language [17]. A group of other products like eXist [18] and dbXML [19] used a language called XUpdate which was designed to be independent of any implementation. In order to retrieve information, XUpdate has been designed to be used with another language.

Another technique is to retrieve the XML documents from the database and then update them using API XML after that the documents are returned back to the database. This technique is a popular one but it suffers from the highly cost of API [20].

C- XML Updates in XML Enabled Database (XEnDB):

Most of the work in this group covers only the primitive update operations without considering the constraints of the data because XEnDB products have different data models. Oracle 10g presents only a limited number of update operations which still has to utilize other functions to the latest SQL/XML standard [21]. DB2 [22] declares that its product can store XML documents as columns and collections. It also claims that the product is able to process updating but unfortunately there is no clear information about the product synchronization or whether it really maintains the full constraints of XML documents. [7] transformed XML constraints into a schema then they proposed a generic update methodology that utilizes the proposed schema.

Another group of researchers investigated different issues related to XML update. [23] Investigated the detection of conflicts during updating valid XML documents when many users work concurrently on these documents. They proposed an algorithm that is able to detect conflicts in valid XML documents. [24] Extended their previous work [25] by adding XML update. Originally, their work represented XML as a graph. They have developed a graphical query language for semi-structured data.

[26] Suggested different architectures for dealing with XML updating. Some of these architectures accept any kind of updating without checking the validity. Other architectures rejected any kind of updating that violates the DTD (Document Type Definition) or Schema and the final type of architectures accepts updating by altering the content of DTD or schema.

[27] Investigated access control for updating XML documents. They proposed a language called XACU for specifying access control on XML data in the presence of update operations. The access control which consists of a group of rules written in a natural language is supported by a formal model that determines the update rights. However, their rules lack consistency in certain cases.

Some researchers have directed their efforts to improve the labeling techniques of XML nodes. Theses techniques help in reaching these nodes in an efficient way. [28] Proposed a binary encoding method QED for avoiding nodes relabeling. They extended their work in [29] by presenting an encoding algorithm called CDBS (Compact Dynamic Binary String) which is a recursive procedure that generates a new binary string inserted between two consecutive CDBSs. [30] devised a new labeling schema based on ordinal bit string. Their approach provides a reasonable querying performance besides avoiding of relabeling. Comparing their work with [29], the encoding schema of the former doesn't need extra spaces besides that encoding and the labeling for nodes can be done at the same time. Consequently, they need less time in labeling.

During the last two years, a new direction of interest has been established. Although it's not related directly to XML update, it plays in the same field. Developing parallel and concurrent software to be compatible with Multicore architecture can be seen as a new movement in software industry. An application can be parallelized under different levels of granularity. According to [31] a stream of independent incoming service request under a naturally concurrent workload can be concurrently assigned each one to a separate core. They devised a data-parallel approach to parallel XML parsing. The results show a significant speedup however they faced a cost problem in applying their technique. [32] investigated a parallel XML model on a Multicore computer. The basic idea of the model is a stealing-based dynamic load-balancing mechanism. In this model, multiple threads are able to process the disjointed parts of the XML document in parallel with balanced load distribution. The dynamic workload redistribution, however, incurs more sophisticated interaction among the threads.

4. Proposed Methodology

We aim to carry out this research in three different phases as follows:

1-The first phase will focus on the study of XML update principles. The researcher will conduct a complete comparative study of the existing techniques, which includes:

- A study for the available XML documents updating techniques.

- Evaluating and criticizing the current techniques.
- Specifying the requirements for the new model; this includes the tool that we are going to use for the designing model, the hardware and the operating system.

2-The second phase will deal with the design of the updating model. This phase can be described as follows:

- The researcher will design a concurrent model, using Petri Nets as a tool, which achieve XML updating. This model will be shown as a bipartite graph.
- The document will be represented as a tree. The updating operations on this tree includes: Inserting a new node (Before, after, certain location), deleting a node, renaming a node.
- The model shouldn't violate the validity of the XML documents. A checking for validity will be done with each operation to ensure the consistency of the documents after updating.
- The model should be optimized in order to get best results. The optimization will focus on the full utilization of the available resources; i.e. the cores and memory. We will try not to let any core in an ideal status. Finally, after optimizing, we will finalize the model to be adapted with multithreading programming environment.

3- Phase three is dedicated for implementation and verification. This phase includes:

- We will use multithreading technique to program our model since multithreading allows one program to do multiple tasks concurrently. We are planning to assign each active task (thread) to an individual core. For example; one core will do insert a new node in the XML tree and at the same time another core will rename an existing node, etc. The number of tasks (threads) that can run at the same time is determined by the number of cores that is available in the machine.
- Regarding work environment; we are going to run our program on a quad-core machine. The language we are going to use is C/C++ programming language since this language has a built-in thread library besides that, the operating system that we are going to use is Microsoft Vista which supports multithreading.
- Each updating operation will be tested against DTD (Document Type Definition) to ensure its validity. Any kind of operation that tries to violate the validity of the XML documents will be rejected. Therefore, we can guarantee the consistency of the whole documents.
- To test our program, we are going to feed the programmed model with several samples of XML documents (trees) along with their DTDs besides a set of updating operations. The documents will be varied in their sizes and types of data.
- The program will be designed to register the time needed to achieve all the updating operations for each XML document. Also, the program will register the activity of each core at run time.
- Finally, all the registered data will be drawn on a graph which will represent the activity of each core versus time. This graph will clearly prove the importance of using concurrent model in updating XML documents on the Multicore architecture versus updating the same documents on the same machine without using the concurrent model nor multithreading.

5. Summary and Future Work

In this paper, we present a research proposal for designing a model for updating XML documents on a Multicore architecture. We proposed using Petri Nets as a modeling tool to present a concurrent model for updating XML documents on Multicore machines. The future work of this study is to put our proposal in work. Designing the model and then apply it in a multithreaded environment will be our next contribution. The Multicore technology will enforce many software developers to re-evaluate their products to cope with the new technology. Multithreading will be the master key in developing any kind of software in the future. Thus, the door is open to convert existing techniques, algorithms, models, etc, to take the advantages of the new technology. This also will open the door for designing new languages that are basically designed for the new architecture.

6. References

[1] D. X. Le and E. Pardede, "Towards Performance Efficiency in Safe XML Update," in *The 8th International Conference on Web Information Systems Engineering (WISE 2007)*, Nancy, France, 2007, pp. 563-572.

[2] C. Leiserson and L. Mirman, *How to Survive the Multicore Revolution (or at Least Survive the Hype)*.: Cilk Arts, 2008.

[3] T. Murata, "Petri Nets: Properties, Analysis and Applications," *IEEE*, vol. 77, no. 4, 1989.

[4] G. Chiola and A. Ferscha, "Distributed Simulation of Petri Nets," *IEEE*, 1993.

[5] A. S. Al-Eraqi, A. Tajudin, and A. Mustaffa, "Model of Arrival / Departure and Charge / Discharge Management in Port Zone Using Petri Net," in *The 2nd International Conference on Research and Education in Mathematics ICREM2*, 2005.

[6] W. Lu and D. Gannon, "ParaXML: A Parallel XML Processing Model on the Multicore CPUs," 2008.

[7] E. Pardede, W. Rahayu, and D. Taniar, "XML data update management in XML-enabled database," *Journal of Computer and System Sciences*, vol. 74, no. 2, pp. 170-195, 2008.

[8] M. Scardina, B. Chang, and J. Wang, *Oracle Database 10g XML & SQL: Design, Build & Manage XML Applications in Java , C, C++, & PL/SQL*. Osborne: McGraw-Hill, 2004.

[9] W. Han, K. Lee, and B. Lee, "An XML storage system for object-oriened/object relational DBMSs," *Journal of Object Technology*, vol. 2, no. 1, pp. 113-126, 2003.

[10] W3C and D. Chamberlin. (2005, June) XQuery Update Facility Requirements, W3C Working Draft. [Online]. HYPERLINK http://www.w3.org/TR/xquery-update-requirements/.

[11] P. Amornsinlaphachai, M. Ali, and B. Rossiter, "Updating XML using object-relational database," *Lecture Notes in Computer Science*, vol. 3567, pp. 155-160, 2005.

[12] D. DeHaan, D. Toman, M. Consens, and M. Ozsu, *A Comperhensive XQuery to SQL translation using dynamic interval coding, IN sigmod.*: ACM, 2003.

[13] F. Du, S. Amer-Yahia, and J. Freire, "ShreX: Managing XMl documents in relational database," in *VLDB*, 2004, pp. 1297-1300.

[14] L. Khan and Y. Rao, "A performance evaluation of storing XML documents in relational database management system," in *WIDM*, 2001, pp. 31-38.

[15] I. Tatarinov, Z. Ives, A. Halevy, and D. Weld, "Updating XML," in *SIGMOD*, 2001, pp. 413-424.

[16] D. Xuan and E. Pardede, "Towrads Performance Efficiency in Safe XML Update," pp. 563-572, 2007.

[17] X. Franc. (2008) XQuery Update for the impatient. [Online]. HYPERLINK http://www.xmlmind.com/_tutorials/XQueryUpdate/index.html.

[18] W. Meier, "eXist Native XML Database," in *XML Data Management: Native XML and XMLEnabled Database System*, 2003, pp. 43-68.

[19] dbXML-Group. (2008) dbXML Group: dbXML (native XML database).HYPERLINK http://www.dbxml.com/product.html.

[20] H. Jagadish et al., "TIMBER: A native XML database," *VLDB*, vol. 11, no. 4, pp. 279-291, 2002.

[21] A. Eisenberg and J. Melton, "SQL/XML is making good progress," in *SIGMOD*, vol. 2, 2002, p. 101–108.

[22] (2009) DB2 Product Family. [Online]. HYPERLINK http://www-01.ibm.com/software/data/db2/.

[23] C. Byun, I. Yun, and S. Park, "An Efficient Detection of Conflicting Updates in Valid XML," in *seventh International Conference on Computer and Information Technology*, 2007.

[24] W. Ni and T. W. Ling, "Update XML Data by Using Graphical Languages," in *Twenty-Sixth International Conference on Concept Modeling - ER 2007*, 2007.

[25] W. Ni and T. W. Ling, "GLASS: A Graphical Query Language for Semi-Structured Data," in *DASFAA2003*, 2003.

[26] M. Klettke, H. Meyer, and B. Hansel, "Evolution-The Other Side of the XML Update Coin," in *Proceeding of the 21st International Conference on Data Engineering ICDE '05*, 2005.

[27] I. Fundulaki and S. Mneth, "Formalizing XML Access Control for Update Operations," in *Symposium on Access Control Models and Technologies*, Soohia Antipolis, France, 2007.

[28] C. Li and T. Ling, "QED: A Novel Quaternary Encoding to Completly Avoid Relabling in XML Updates," in *ACM CIKM 2005*, 2005, pp. 501-508.

[29] C. Li, T. Ling, and M. Hu, "Efficient Updates in Dynamic XML Data: from Binary String to Quaternary String," *Springer*, 2006.

[30] J. Min, J. Lee, and C. Chung, "An Efficient XML Encoding and Labeling Method for Query Processing and Updating on Dynamic XML Data," *The Journal of System and Software*, 2008.

[31] Y. Pan, Y. Zhang, K. Chiu, and W. Lu, "Parallel XML Parsing Using Meta-DFAs," in *The Third IEEE International Conference on e-Science and Grid Computing*, 2007.

[32] W. Lu and D. Gannon, "Parallel XML processing by work stealing," in *Proceedings of the 2007 Workshop on Service-Oriented Computing Performance: Aspects, Issues, and Approaches*, Monterey, california, 2007, pp. 31-38.