Research plan
XML processing using GPGPU

Vincent Jordan
vjordan@utbm.fr
December 2010

During the last decade, the World Wide Web has undergone many changes: mainly from monolithic and static HTML pages to aggregated and dynamic XML documents. XML is now a widely used semi-structured data format for representing and exchanging data in a textual form. XML is found everywhere on the public web, but this format also gains adoption in the field of inter-organization data exchange. The widespread use of XML on the web, together with increasing quantities of XML data in the enterprise world, made it crucial to have an efficient processing capability.

XPath is the core retrieval language for XSLT, XPointer, and especially XQuery. XQuery gathered significant attention as the language for querying and transforming XML documents. It is the SQL-equivalent for XML databases and XML repositories. Since XQuery is a superset of XPath, any improvement to the latter would benefit to the first.

The main purpose of this proposal is the evaluation of a new execution environment for XPath queries computation. In this document, I will introduce a direct research extension of my 6-month internship spent at Kitagawa Data Engineering laboratory (KDE lab., Tsukuba University).

Performance issue of XPath queries over large XML documents

Large XML documents are better processed by native XML databases. MonetDB/XQuery is a such database presented in [1]. This database can handle documents of several gigabytes, but implies a data preprocess, which go beyond XML parse. At this scale, queries can last several seconds (see project's benchmark).

Previous research[3] carried out at the KDE laboratory showed that parallel execution of XPath queries can yield significant speed up. Cluster and multicore execution environments were addressed, but this hardware was not designed for parallel processing from the beginning. I propose a more disruptive technology.

Why XML and GPGPU?

There are several approaches that successfully use specialized parallel architectures for XML processing. XML parsing has been accelerated using Cell Broadband Engine multi-processor[4]. XML query matching has been accelerated using FPGA[5]. To the best of my knowledge, the GPGPU solution was never evaluated.

GPGPU is a recent creation, which gathers research attention because of its large diffusion and cheap price. It is a processor designed from the start for massive and efficient parallel processing. Unlike FPGA, GPU cannot be configured by user and has a fixed design. On the other hand, the spread of GPU benefits from computer video game market, thus a GPGPU board is much cheaper than an FPGA board.

If GPGPU is found to be suitable for XML processing, a dream hardcore gamer machine, could become an interesting webserver with no hardware changes. This solution would be scalable and affordable. Webserver hardware configuration could start as a conventional CPU-based solution. While querying load would grow, one can add up to four GPU as coprocessors.

This solution is challenging because XML hierarchic (tree-based) nature does not fit GPGPU design for stream processing (vector-based). For my master thesis, I studied the nVIDIA GPU architecture and I faced many limitations that prevent TwigStack[2] algorithm from being efficiently implemented through GPGPU. Even so, I found a way to overcome these limitations and it drove me to this research proposal.

Purpose of this research and proposition

By making use of the partitioning methods proposed by Machdi et al. (2009), I will develop an XML querying engine on GPGPU taking advantage of its highly parallel architecture.

The two main goals of this research are the followings:

— Explore non-uniform parallelism on GPGPU through the direct parallel implementation of TwigStack algorithm made during my masters education.
— Design an algorithm for XML query processing that better fits GPGPU architecture and demonstrates its efficiency using mass-market hardware.

My proposition extends and generalizes my previous work at KDE laboratory. I would use computer hardware that already belongs to the KDE laboratory (especially GPGPU workstation).

① Since I already developed a prototype of TwigStack algorithm using CUDA toolkit, my plan is to convert it in OpenCL. OpenCL is supported by major parallel processor vendors, such as nVIDIA (GPU), ATI (CPU+GPU), Intel (CPU) and IBM (CPU). OpenCL toolkits also exist for DSP found in some handheld devices (i.e., OpenCL Embedded Profile on Nokia N810 for example[6]). I would like to publish a research paper on that topic.
② The knowledge of several parallel architectures would allow me to design a new algorithm for XPath queries. This work would be published in a research paper as well.
③ According to the remaining time and the reception of my previous works, I can add querying features to the previous algorithm. XML document update will also be an issue since documents are stored in device (e.g., GPU) memory instead of host (i.e., CPU) memory.
④ Finally, my Ph. D. thesis will contain a consistent solution to non-uniform parallelism on GPGPU and the description of an algorithm for XPath query processing. In the case I would not succeed in creating the algorithm, I could address other problems that require non-uniform parallelism for XML, such as XML parser.

Related research

Recent advance at KDE laboratory involved XML-OLAP[7]. It focuses on topological rollup operation (aka. structure-based grouping), which is specific to XML, using a tuned algorithm based on structural joins. Since structural join algorithm shares the same representation of XML document as TwigStack, a successful GPGPU implementation of the latter could apply to this implementation of the OLAP operation as well. Furthermore, [7] already addresses parallelization issue for multicore processors.

I believe that an efficient XML framework for OpenCL containing XML partitioning methods and XML representation could be the base of several XML-related studies on a wide range of parallel coprocessors.

Please find references on the other side.

References

[1] Peter Boncz, Torsten Grust, Maurice van Keulen, Stefan Manegold, Jan Rittinger, Jens Teubner. ***MonetDB/XQuery: a fast XQuery processor powered by a relational engine***. SIGMOD '06 Proceedings of the 2006 ACM SIGMOD international conference on Management of data.
Available at: *http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.777&rep=rep1&type=pdf*
Project homepage: *http://monetdb.cwi.nl/*

[2] Nicolas Bruno, Nick Koudas, Divesh Srivastava. ***Holistic Twig Joins: Optimal XML Pattern Matching***. SIGMOD 2002
Available at: *http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.6883&rep=rep1&type=pdf*

[3] Imam Machdi, Toshiyuki Amagasa, Hiroyuki Kitagawa. ***Executing parallel TwigStack algorithm on a multi-core system***. International Journal of Web Information Systems (IJWIS), Vol. 6, No. 2, pp. 149-177, 2010.

[4] S. Letz, M. Zedler, T. Thierer, M. Schutz, J. Roth, R. Seiffert. ***XML offload and acceleration with Cell broadband engine***. XTech: Building Web 2.0 (2006).

[5] Roger Moussalli, Mariam Salloum, Walid A. Najjar, Vassilis J. Tsotras. ***Accelerating XML Query Matching through Custom Stack Generation on FPGAs***. Lecture Notes in Computer Science 5952 Springer 2010
Presented at High Performance Embedded Architectures and Compilers, 5th International Conference, HiPEAC 2010
Available at: *http://www.cs.ucr.edu/~tsotras/asterix/xml-fpga.PDF*
Project homepage: *http://www.cs.ucr.edu/~tsotras/asterix/*

[6] Kari Pulli, Jyrki Leskelä. ***OpenCL Embedded Profile***. Presentation for Multicore Expo, 16 March 2009.
Available at: *http://people.csail.mit.edu/kapu/papers/OpenCLEmbeddedProfileForMulticoreExpo.ppt* (slides)

[7] Chantola Kit, Toshiyuki Amagasa, Hiroyuki Kitagawa. ***Algorithms for Efficient Structure-based Grouping in XML-OLAP***. Proceedings of the 10th International Conference on Information Integration and Web-based Applications and Services (iiWAS 2008), Linz, Austria, Nov 24 - 26, 2008.
Available at: *http://dx.doi.org/10.1108/17440080910968436*