

Research Statement

Wei Lu (welu@cs.indiana.edu) -Indiana University

My research interests lie at the intersection of *parallel and concurrent programming*, *XML and the service oriented computing*, and *high performance computing*. As manufacturers have encountered difficulties to further exponentially increases in clock speeds, they are increasingly utilizing the march of Moore's law to provide multiple cores on a single chip. The multicore processor is rapidly becoming the mainstream on the desktop-computing machines with quad-core shipping now and 16 core system within two or three years. The architectural changes, however, benefit only those software with the concurrency in mind and therefore have little value for most existing softwares. Consequently, the good time when people merely rely on the faster CPU clock to speed up the software execution is over, the software must be able to exploit the the parallelism or concurrency so that it can take advantage of multicore resource. My current research strives to devise such approaches to help the softwares, particular the XML processing and service oriented computing, to utilize the multicore resource efficiently. In the remainder of this document, I will first describe my dissertation research, then describe research I have one outside of my dissertation, and finally, provide my future research agenda.

Dissertation Research Recent years have witnessed the prevalence of the XML and web-service based Service Oriented Computing (SOC). The XML processing and web services, however, is well known for their poor performance and scalability and there has been imminent needs for the effective solutions. My dissertation research tackles the challenge by starting from the two pillars of the SOC: XML processing and the service orchestration.

Parallel XML Processing The very characteristics of XML that have led to its success, however, such as its verbose and self-descriptive nature, can incur significant performance penalties. The processing of the XML documents has been recognized as the performance bottleneck in the distributed systems and database systems. Various techniques have been proposed to improve the performance of XML processing, ranging from the schema-specific ones to the streaming ones. Unfortunately, few of them is designed to be parallel, thus being able to benefit from the multicore resource. Hence I believe that a parallel XML processing model is a more general and cost-effective solution in this multicore era.

To exploit the parallelism for XML, I start from the most challenging task: XML parsing. Since a XML document is the serialization of a tree data model, the obstacle lies at how to partition the document into the parse-able partitions which can be processed in parallel. I devise a novel data-parallel XML parsing algorithm, called PXP [4], which adopts a two-pass-scanning based processing, namely a quick sequential scanning of the document to identify the structural information of the document, followed by a complete parallel parsing. Aided by the structural information the algorithm can partition the XML document into well-formed chunks and parse them in parallel. By our best knowledge, PXP is the first XML parser running in parallel and my research has shown that PXP can speedup the XML parsing well on the multicore machine. Based on the PXP, I further co-develop an optimal static partition strategy [8] for the large XML documents.

Since more and more services rely on the higher level XML tasks, such as XML security and XPath query, it is crucial that the parallel XML processing model can be generalized to cover the common pattern of most XML tasks. Motivated by that, I develop a general-purpose parallel XML processing model, ParaXML [7] which servers as the basic paradigm for the potential parallelizations of the higher level XML tasks. General speaking, ParaXML treats the XML document as a tree structure and the XML processing task as the extension of the parallel tree traversal algorithm of the discrete optimization. However my research has observed that the XML processing has quite distinct characteristics from the traditional discrete optimization problems, thus demanding the special fine-grained optimization. ParaXML internally adopts a lock-free work-stealing scheme to dynamically control the load balance; a novel stealing tracing approach

is introduced to facilitate the reducing of the parallel-running results. My research has demonstrated how ParaXML parallelizes various XML tasks, including XML traversal and searching, serialization and XML signature, parsing and DOM building. The empirical study shows that those parallel implementations present substantial speedup with good scalability on the multicore machine.

Concurrent Service Orchestration Runtime The essence of the Service Oriented Computing is programming the software in the service-as-component manner, and one of the main goals of SOC is enabling the service composition, namely building the complex service out from the existent simpler ones. WS-BPEL is defined as the stand way for the web-service composition and it adopts the orchestration based workflow paradigm, in which a central engine orchestrates the execution of the services. That implies the orchestration runtime, which need to handle thousands of current workflow instances, is the hot-spot and potential bottleneck of the entire system. Meanwhile WS-BPEL allows very complex and sophisticate concurrency and coordination logic in one workflow, thus even one workflow instance will present substantial concurrency during the execution. Therefore the challenge not only lied at the efficiency of the orchestration, but also at the programming productivity of the service-as-component software.

To implement a highly concurrent service orchestration runtime, the conventional thread/lock model is inappropriate either in term of the performance or the expressiveness. Instead, my research strives to investigate an alternative model, the event-driven programming model together with join pattern synchronization primitives, which can handle the complex concurrency and coordination logic with good system performance and scalability. My research has shown that the event makes the concurrent scheduling of the system extremely light weight, thus enabling the good performance; and most concurrency and coordination logic of WS-BPEL can be mapped to the join operators in a very elegant manner.

Meanwhile, in pursuit of the programming productivity of the service-as-component software for the high performance computing, I further investigates a concurrent service orchestration C# library which provides a higher-level and type-safe API for the service creation, deployment and orchestration. This library allows the services to be deployed either on the individual core or the remote machines. With the presence of the manycore CPU we can have the multiple services be deployed and be orchestrated on a single chip. Also the library provides a viable service-as-component solution for the distributed HPC programming for the Non-Uniform Cluster Computing system, a cluster with nodes that are built out of multicore chips.

Other Research

High performance XML/Web service Before the multicore solutions, my earlier research work strives to tackle the performance issue of the XML and web service from various aspects. First, I co-investigate how the higher level XML schema information can be utilized to accelerate the processing of the XML document [2]. This technology is called schema-specific parsing and its basic idea is the schema information can effectively decrease the search space of the XML parsing by generating a specific parser. The experiment results suggest the schema-specific parsing is a viable approach to the high-performance XML parsing. Secondly, I have been interested in the alternative encoding schema of the XML model. Although technologies have been proposed to address the performance issue, none of them meets the high-performance requirements of the scientific applications where the large amount of binary scientific data dominates. To make sure the binary data be the first class member in the XML and web service system, I have co-designed and developed a binary XML encoding schema for scientific applications [1] and based on that I developer a web service engine which supports both textual XML and binary XML in a generic manner [3]. Our experiments show that performance of the binary XML and its use in the web service can outperform to that of commonly used scientific data formats and transportation methods. I also have investigated the streaming processing model, especially for the XML message security processing [6] which is a well-known bottleneck in the web service protocol stack. My research has shown that in most cases the XML message signature validation actually can be embedded within XML parsing and both can be processed by an augmented automaton in a streaming way. This streaming model can not only provide high performance, but is also memory efficient

Parallel Components & Workflow for HPC Building large scale scientific applications needs the problem decomposition scheme into manageable size. Two problem decomposition schemes, component assembly and workflow orchestration, have been widely adopted. The two methodologies describe the problem decomposition from different perspectives with different concerns, thus each have their own advantages and disadvantages. To bring benefits from both methodologies I develop a hybrid problem decomposition scheme based on the CCA parallel component architecture and Kepler workflow system [5]. In this schema we uses Kepler to arrange the temporal ordering of computations as a workflow, but CCA to organize the functional units of the parallel computation. The hybridization hinges on the web services interface which we have provided the use of CCA components in Kepler transparently. This hybridization form a unified problem solving environment and be beneficial for most scientific applications.

Future Research Goals During my research on the multicore programming one frequently asked question is “*What differs the multicore programming from the traditional SMP programming?*”. There are various answers by different people from the different viewpoints and a large part of them emphasize on the shift of the hardware architecture. Multicore does have some difference at the hardware level, such as shared cache, possibility of fast intercore communication and synchronization, etc. It is not yet clear what implications those differences will have in the programming model. What is clear, however, is that the multicore brings the parallel programming from the high-end computation to the desktop computation. This means, for me, the major difference has its root in the programming environment rather than in the hardware environment. For example instead of the scientific applications by most SMP programs, the applications solved by multicore programming is much more diverse, ranging from document processing to graphics softwares. The programmer community has also been changed. The multicore programming is supposed to be easily mastered by most application programmers, while SMP programming usually requires skillful experts. Consequently, the programming productivity is an emerging concern for the multicore programming. I believe a set of novel programming methods which enables intuitive parallel/concurrent programming model, efficient synchronization primitives and high productivity, will be the right research direction for the multicore programming.

In the near term, I would like to build on my dissertation research.

- First, I will be very interested in applying the ParaXML model for the more ambitious XML processing tasks, such as XPath and XSLT. Both XPath and XSLT are too complicate to be fully parallelized by simply applying the ParaXML model. However it would be interesting to see how the ParaXML model is extended to embrace more parallel programming patterns to solve the challenges. Also for the super-large XML documents which don't fit in the memory it would be interesting to investigate the potential parallelization by integrating the ParaXML and the technologies borrowed from the database research area. Ideally, I would like to pursuit a programming-language supported parallel programming model for the XML data.
- It is predictable of the coming of the “manycore” era. I would like to explore the viable software architecture and high productivity programming models for building the large-scale service-as-component software, which can effectively exploit hundreds of cores. Extending the ordinary Object oriented language(e.g., Java and C#) to enable the service-as-component software architecture will be one of key issues, That may entail a refinement of the OO programming model as well as a new mechanism for the concurrency and synchronization.

In the long term I am interested in the appropriate programming methodology for the general applications on the multicore platform. I am especially interested in the novel languages, compilers and libraries, such as software transaction memory and the lock-free algorithms, which can greatly reduce the complexity and improve the productivity of developing parallel/concurrent software with the assurance of the high performance on the multicore platform.

References

- [1] K. Chiu, T. Devadithya, W. Lu, and A. Slominski. A binary xml for scientific applications,. In *e-science'05 (1st IEEE international conference for e-science and Grid Computing)*, Melbourne, Australia,

Dec 2005.

- [2] K. Chiu and W. Lu. A compiler-based approach to schema-specific xml parsing. In *The First International Workshop on High Performance XML Processing, Satellite workshop of WWW2004 International Conference*, 2004.
- [3] W. Lu, K. Chiu, and D. Gannon. Building a generic soap framework over binary xml,. In *HPDC'06 (The 15th IEEE International Symposium on High Performance Distributed Computing)*, Pairs, France, June 2006.
- [4] W. Lu, K. Chiu, and Y. Pan. A parallel approach to xml parsing. In *The 7th IEEE/ACM International Conference on Grid Computing*, Barcelona, September 2006.
- [5] W. Lu, K. Chiu, S. Shirasuna, and D. Gannon. A hybrid decomposition scheme for building scientific workflows. In *HPC'07 (Proceedings of High Performance Computing Symposium, SCS Spring Simulation Multiconference*, Norfolk, VA, March 2007.
- [6] W. Lu, 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.
- [7] W. Lu and D. Gannon. Parallel xml processing by work stealing. In *Workshop on Service Oriented Computing Performance In Conjunction with HPDC*, Monterey Bay California, June 2007.
- [8] Y. Pan, W. Lu, Y. Zhang, and K. Chiu. A static load-balancing scheme for parallel xml parsing on multicore cpus. In *CCGrid'07 (IEEE International Symposium on Cluster Computing and the Grid)*, Rio de Janeiro - Brazil, May 2007.