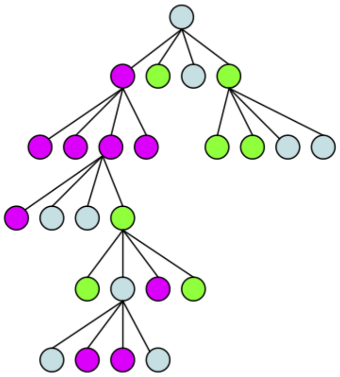
Scalable Dynamic Load Balancing for Task Parallel Programming Frameworks

Department of Computer Science **University of North Carolina at Chapel Hill** **January 2010**

The Challenge

Parallel computing is now the primary means to higher performance for all applications. This imperative has prompted the development of new *task parallel* programming languages and libraries: Intel Thread Building Blocks, Microsoft Task Parallel Library, OpenMP 3.0, and IBM X10. Largely based on the work stealing techniques developed as part of the experimental Cilk language in the 1990's, the behavior of these programming frameworks on modern platforms, such as multi-core, is not well understood. The high level of abstraction offered by the task parallel model makes programming easier for many applications. However performance is often hit-or-miss due to poor decisions or overhead costs in the run time systems that manage execution. Of particular concern is the scalability of applications with inherently irregular and unpredictable structure, which often require dynamic load balancing.



Snapshot showing parallel execution of a tree-structured computation by three threads. Each thread executes a portion of the tree.

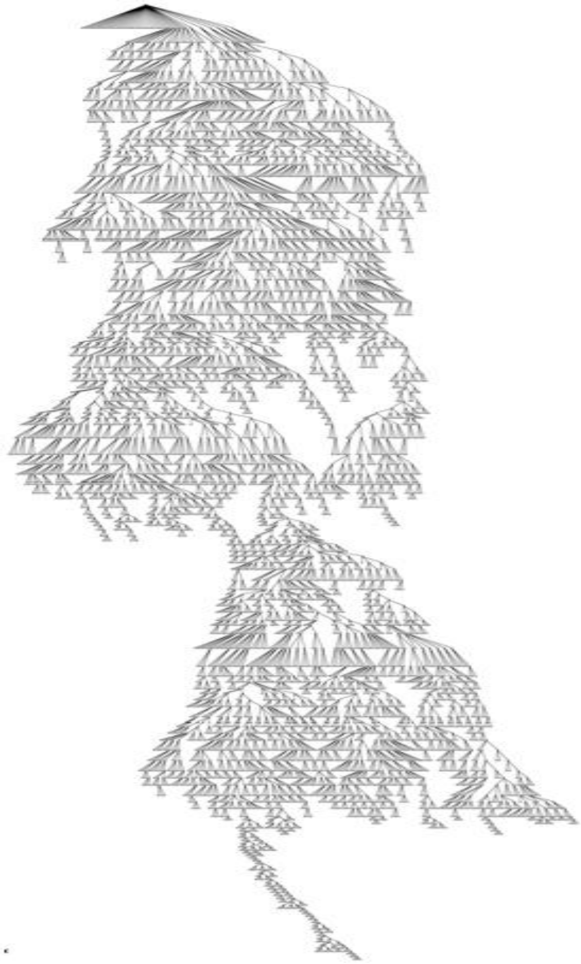
Our research group develops techniques to achieve better load balancing and lower overhead costs to enable efficient execution of applications programmed using the task parallel model.

The Approach

We created the Unbalanced Tree Search (UTS) benchmark, which presents a synthetic tree-structured search space that is highly imbalanced. Since parallel execution of the search requires continuous dynamic load balancing to keep all processors engaged in the search, UTS represents a challenging problem for parallel systems. The implementation and evaluation of the benchmark on different architectures and in different parallel programming languages has allowed us to uncover and resolve important issues: the tradeoff between load imbalance and overhead costs of load balancing operations, efficient discovery and transfer of work between processors, and scalable termination detection.

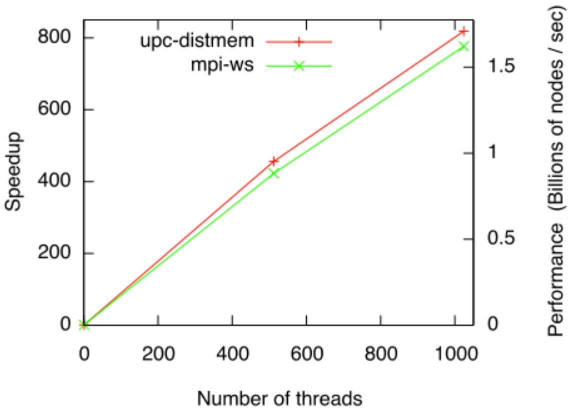
Highlights

- **We are investigating techniques for dynamic load balancing of parallel computations and their application to run time systems for task parallel programming.**
- **Our solutions push the envelope of scalability while allowing higher-level abstractions for the application programmer through parallel programming frameworks.**
- **We are also applying our techniques to enterprise applications such as business event processing in cloud computing environments.**



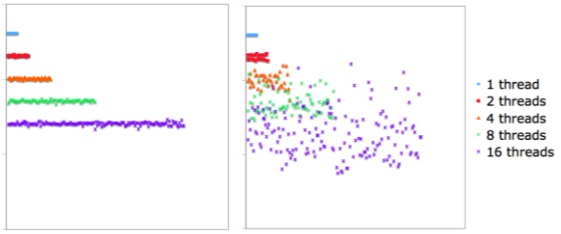
An example of tree-structured computation. Imbalance in the tree necessitates significant load balancing efforts to achieve parallel speedup in a multiprocessor traversal.

OpenMP Run Time Support for Tasks. Explicit task parallelism provided in OpenMP 3.0 enables easier expression of unbalanced applications. However, performance evaluation of UTS in OpenMP 3.0 reveals limited scalability using recent compiler and run time systems. We have identified high overhead costs and load imbalance in the run times as key contributors to poor performance, and are working to improve run time scheduler design and implementation to improve load balance and lower overhead costs. We are also experimenting with load balancing techniques that incorporate locality-based heuristics. Our work is not specific to OpenMP 3.0, but also extends to other task parallel languages and libraries, such as IBM X10 and Intel TBB. A particularly important research effort is the design of performance models and measurement frameworks to accurately predict and evaluate the impact of scheduling techniques and run time systems.



Parallel speedup of the Unbalanced Tree Search benchmark achieved using 1024 threads on a cluster. Over 85000 asynchronous dynamic load balancing operations per second were performed during execution.

Business Applications. In a related project with IBM, we are applying the techniques we have developed to problems in the enterprise. Business applications increasingly include components with complex structure and large-scale parallel computing requirements. For example, WebSphere Business Event Processing may operate on a large stream of many disparate event types with processing distributed over many processors in order to meet real-time performance guarantees. Classic business problems such as transportation logistics, supply-chain planning, scheduling, and data mining are solved using combinatorial optimization and enumeration techniques, requiring parallel search of a large state space of possibilities to obtain a timely answer.



Load imbalance during execution of Unbalanced Trees Search Benchmark using OpenMP 3.0 tasks using the Intel (left) and Sun (right) run time systems. Straight lines of data points indicate good load balance while scattering indicates poor load balance.

Current Project Members

Prof. Jan Prins
Stephen Olivier

Collaborators

Prof. P. Sadayappan (Ohio State Univ.)
Prof. Chau-Wen Tseng (UMD at College Park)
James Dinan (Ohio State Univ.)

Research Sponsors

US Department of Defense, IBM

Selected Publications

Stephen Olivier, Jan Prins. Evaluating OpenMP 3.0 Run Time Systems on Unbalanced Task Graphs.. *Proc. of 5th International Workshop on OpenMP (IWOMP 2009)*. LNCS 5568. Matthias S. Muller, Bronis R. de Supinski, Barbara M. Chapman (Eds.) Springer: Dresden, Germany, June 2009. pp. 63-78.

James Dinan, Stephen Olivier, Gerald Sabin, Jan Prins, P. Sadayappan, Chau-Wen Tseng. A Message Passing Benchmark for Unbalanced Applications. *Simulation Modelling Practice and Theory*, 16 (9), 1177-1189, October 2008.

Stephen Olivier, Jan Prins. Scalable Dynamic Load Balancing Using UPC. *Proc. of 37th International Conference on Parallel Processing (ICPP-08)*. Portland, OR, September 2008.

James Dinan, Stephen Olivier, Jan Prins, Gerald Sabin, P Sadayappan and Chau-Wen Tseng. Dynamic Load Balancing of Unbalanced Computations Using Message Passing. *Proc. of 6th Intl. Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS 2007)*. Long Beach, CA, March 2007.

Stephen Olivier, Jun Huan, Jinze Liu, Jan Prins, James Dinan, P Sadayappan and Chau-Wen Tseng. UTS: An Unbalanced Tree Search Benchmark. *Proc. of 19th Intl. Workshop on Languages and Compilers for Parallel Computing (LCPC 2006)*. LNCS 4382. George Almasi, Calin Cascaval, Peng Wu (Eds.) Springer: New Orleans, LA, November 2006. pp. 235-250.

Keywords

Load balancing; Task Parallelism; Run Time Systems; Parallel Programming Languages; Scalability; Search; Business Event Processing

For More Information

Jan Prins
Phone (919) 962-1913
E-mail: prins@cs.unc.edu
http://www.cs.unc.edu/~prins