The Challenge of Exascale (Extended Abstract)

Jayadev Misra
The University of Texas at Austin

July 18, 2011

The programming problem for exascale machines is similar to that faced by petascale or even terascale programming. Problems of climate modeling, computational biology, code breaking, machine learning, simulations, etc. will be solved at a much larger scale. Additionally, certain problems that are beyond our reach, due to limitations of computational resources, can possibly be addressed.

The magnitudes of the problems that will be attempted on exascale machines will be so much larger that adhoc solutions, that may have sufficed in an earlier time, are no longer admissible. We will be dealing with massive amounts of concurrency. Implicit creations and manipulations of millions of threads will be the norm. Lock-based management of threads, or even present-day notions of transactional memories, seem inadequate for this challenge.

Structured management of concurrency will be the main theme of this talk. Fast Fourier Transform or hypercubic sort algorithms admit well-structured concurrent computations, and relatively simple algorithms for mapping computations to resources. At the other extreme, we will have to address irregularly structured problems, such as discrete-event simulations and graph manipulations, for which concurrency is dynamically discovered, and resources (and/or computations) migrate as needed.

The talk will illustrate two different models of parallelism: (1) *Powerlist*, a model for synchronous data parallel programming [3, 4], and (2) *Orc*, a model for highly asynchronous light-weight thread creation and manipulation [1, 2].

The powerlist model of synchronous computations is based on a novel data structure and its associated operations. Using traditional control structures along with recursion, we show how to solve important synchronous parallel algorithms, such as Fast Fourier Transform, routing and permutation, Batcher sorting schemes, solutions of tridiagonal linear systems by odd-even reduction, and prefix-sum algorithms. The inherent concurrency is completely exposed by the underlying powerlist data structure.

Asynchronous computations have to go far beyond just a novel data structure. They include sequential computing as a special case, and, hence, all algorithmic problems. Today's programming models are inadequate to efficiently and explicitly: manage threads, assign threads to resources, specify data migration, or integrate concurrent and sequential computing. I will present an algebraic approach, *Orc*, that embodies structured concurrency including hierarchy and recursion, implicit thread creation and manipulation, and integration of heterogeneous software modules.

References

[1] William Cook and Jayadev Misra. Computation orchestration: A basis for wide-area computing. *Journal of Software and Systems Modeling*, May 2006.

[2] David Kitchin, Adrian Quark, and Jayadev Misra. Quicksort: Combining concurrency, recursion, and mutable data structures. In A. W. Roscoe, Cliff B. Jones, and Ken Wood, editors, *Reflections on the Work of C.A.R. Hoare*, History of Computing. Springer, 2010. Written in honor of Sir Tony Hoare's 75th birthday.

[3] Jayadev Misra. Powerlist: A structure for parallel recursion. *ACM Transactions on Programming Languages and Systems*, 16(6):1737–1767, Nov 1994.

[4] Jayadev Misra. Generating–functions of interconnection networks. In *Millennial Perspectives in Computer Science: the proceedings of the 1999 Oxford–Microsoft Symposium in honour of Sir Tony Hoare*, St. Catherine's College, Oxford, Sep 1999.

[5] Adrian Quark, David Kitchin, John Thywissen, William R. Cook, and Jayadev Misra. Orc language project website. http://orc.csres.utexas.edu.