Invited Panel

Language Innovations for HPCS

Katherine Yelick, *Chair & Moderator*
University of California at Berkeley and Lawrence Berkeley National Laboratory
yelick@eecs.berkeley.edu

As part the DARPA-sponsored High Productivity Computing Systems (HPCS) program, three new languages are being designed with the goal of improving programmer productivity at high performance computing. The languages are targeting very large-scale parallelism and may take advantage of features of the systems that are also under design by each of the three companies involved in this project. Panelists will give an overview of these new languages and some ideas about how the language features will be used in parallel applications. The panelists will be:

Brad Chamberlain from Cray, Inc. on the Chapel Language

Chapel supports a multithreaded parallel programming model at a high level by supporting abstractions for data parallelism, task parallelism, and nested parallelism. It supports optimization for the locality of data and computation in the program via abstractions for data distribution and data-driven placement of subcomputations. It supports code reuse and generality via object-oriented concepts and generic programming features. While Chapel borrows concepts from many preceding languages, its parallel concepts are most closely based on ideas from High-Performance Fortran (HPF), ZPL, and the Cray MTA's extensions to Fortran/C.

Vijay Saraswat from IBM on the X10 Language

X10 is intended for high-performance, high-productivity programming of high-end computer systems with hierarchical heterogeneous levels of parallelism (e.g., cluster, SMP, co-processors, and SMT levels) that exhibit large nonuniformities in data-access latency and bandwidth. Compared to Java, X10 removes threads, lock-based synchronization, built-in primitive classes and arrays, and adds places (as a locus for activities operating on local data), user-definable value types, atomic sections, asynchronous activities and futures, multidimensional arrays (over regions and distributions) and clocks for repeated quiescence detection of dynamically varying, data-dependent sets of activities. X10 is a strongly typed language. The X10 type system supports generic type-abstraction (over value and reference types), is place- and clock-sensitive and guarantees the absence of deadlock (for programs without conditional atomic sections), even in the presence of multiple clocks.

David Chase from Sun Microsystems on the Fortress Language

The Fortress^{TM} Programming Language is a general-purpose, statically checked, programming language designed for creating and maintaining robust high-performance software with high programmer productivity. To allow growth into new and unanticipated uses, Fortress provides modular and extensible parsing, and provides interfaces ("traits") for its own implementation in its library to allow user extension and replacement. To aid program reuse, Fortress has a component and contract system that allows separate program components to be independently developed, deployed, and linked in a modular and robust fashion. To simplify concurrent programming, Fortress provides transactional memory. For performance, Fortress promotes both scale-independent and data-parallel programming, and relies on just-in-time compilation to allow specialization and speculative optimizations.

PPoPP'05, June 15–17, 2005, Chicago, Illinois, USA.
ACM 1-59593-080-9/05/0006.