Advanced Programming and Execution Models for Future Multi-Core Systems

Dr. Hans P. Zima
Jet Propulsion Laboratory
California Institute of Technology
Pasadena,CA
and
Institute of Scientific Computing, University of Vienna, Austria

CMOS manufacturing technology has reached a state where physical limits of semiconductor-based microelectronics lead to serious heat dissipation and data synchronization problems. As a result, microprocessor clock speeds and straight-line instruction throughput have not significantly risen over the past few years. This has led to a revolutionary change in chip design characterized by multi-core architectures. In the not-too-distant future, a single chip may contain hundreds or thousands of processors arranged in homogeneous or heterogeneous collections. At present no generally accepted strategies for the programming and execution models of such systems, and the associated programming environments have emerged.

This presentation will discuss some key issues in this context based on the objective of finding a viable compromise between the goals of providing an API at a high level of abstraction and meeting the challenges related to target code performance, power consumption, and fault tolerance. We will particularly address the question to which degree recent experiences in language design for peta-scale computing systems, such as those developed in the High-Productivity-Computing- Systems (HPCS) program, can contribute to the problem of programming multi-core systems in a productive, efficient, and reliable way.

The final part of the talk will deal with a new approach for exploiting the massive parallelism provided by the abundance of threads in future multi-core systems. Besides their conventional use for fine-grain parallelism in application programs, threads can be used to support introspection realized by a hierarchical arrangement of asynchronous mobile agents with internal state. Introspection enables a software system to become self-aware by monitoring its execution behaviour, reasoning about its internal state, and making decisions about appropriate changes of the system or system state when necessary. In addition to supporting graceful degradation in the case of faults, introspection can be also applied to areas such as intrusion detection, behaviour analysis, and performance tuning.