The Multicore Programming Challenge

Barbara Chapman

Houston University, USA
http://www2.cs.uh.edu/~chapman/

Abstract. Dual-core machines are now actively marketed for desktop and home computing. Systems with a larger number of cores exist, and more are planned. Some cores are capable of executing multiple threads. At the very high end, programmers need to design codes for execution by thousands of processes or threads and have begun to consider how to write programs that can scale to hundreds of thousands of threads. Clearly, the future is multi- and many-core, as well as many-threaded.

In the past, most application developers could rely on Moore's Law to provide them with steady performance improvements. But we have entered an era in which they may have to expend considerable effort if their codes are to exploit the processing power offered by next-generation platforms. At least in the medium term, a broad variety of parallel applications will need to be developed.

Existing shared memory parallel programming APIs were not necessarily designed for general-purpose computing or with many threads in mind. Distributed memory paradigms do not necessarily allow the expression of fine-grained parallelism or provide full exploitation of architectural features. The fact that threads share some resources in multicore systems makes it hard to reason about the impact of program modifications on performance and results may be surprising. Will programmers be able to use multicore platforms effectively?

In this presentation, we discuss the challenges posed by multicore technology, review recent work on programming languages that is potentially interesting for multicore platforms, and discuss on-going activities to extend compiler technology in ways that may help the multicore programmer.

M. Xu et al. (Eds.): APPT 2007, LNCS 4847, p. 3, 2007.