Topic 01
Support Tools and Environments

Barton P. Miller and Michael Gerndt

Topic Chairmen

Parallelism is difficult, yet parallel programs are crucial to the high-performance needs of scientific and commercial applications. Success stories are plentiful; when parallelism works, the results are impressive. We see incredible results in such fields are computational fluid dynamics, quantum chromo dynamics, real-time animation, ab initio molecular simulations, climate modeling, macro-economic forecasting, commodity analysis, and customer credit profiling.

But newcomers to parallel computing face a daunting task. Sequential thinking can often lead to unsatisfying parallel codes. Whether the task in to port an existing sequential code or to write a new code, There are the challenges of decomposing the problem in a suitable way, matching the structure of the computation to the architecture, and knowing the technical and stylistic tricks of a particular architecture needed to get good performance. Even experienced parallel programmers face a significant challenge when moving a program to a new architecture.

It is the job of the tool builder to somehow ease the task of designing, writing, debugging, tuning, and testing parallel programs. There have been notable successes in both the industrial and research world. But it is an continuing challenge. Our job, as tool builders, is made more difficult by a variety of factors:

1. Processors, architectures, and operating systems change faster than we can follow. Tool builders are constantly trying to improve their tools, but often are forced to spend too much time porting or adapting to new platforms.
2. Architectures are getting more complicated. The memory hierarchy continues to deepen, adding huge variability to access time. Processor designs now include aggressive out-of-order execution, providing the potential for great execution speed, but also penalizing poorly constructed code. As memories and processes get faster and more complicated, getting precise information about execution behaviors is more difficult.
3. Standards abound. There is the famous saying "The wonderful thing about standards is that there are so many of them!" It was only a short time ago that everyone was worried about various types of data-parallel Fortran; HPF was the magic language. PVM followed as a close second. Now, MPI is the leader, but does not standardized many of the important operations that tool-builders need to monitor. So, each vendor's MPI is a new challenge to support and the many independent MPI platforms (such as MPICH or LAM) present their own challenges. Add Open/MP to the mix, and life becomes much more interesting. And don't forget the still-popular Fortran dialects, such as Fortran 77 and Fortran 90.

A. Bode et al. (Eds.): Euro-Par 2000, LNCS 1900, pp. 45–46, 2000.

4. Given the wide variety of platforms, and fast rate of change, users want the same tools each time they move to a new platform. This movement increases the pressure to spend time porting tools (versus developing new ideas and tools). Heterogeneous and multi-mode parallelism only make the task more challenging.

The long-term ideal is for an environment and language where we can write our programs in a simple notation and have them automatically parallelized and tuned. Unfortunately, this is not likely to happen in the near future, so the demand for tool builders and their tools will remain high. This current instance of the Support Tools and Environments topic present new results in this area, hopefully bringing us closer to our eventual goal.