EARLY APPLICATION DEVELOPMENT/TUNING AND APPLICATION CHARACTERIZATION/ SEGMENTATION

Laxmikant Kale

PARALLEL PROGRAMMING LABORATORY, SIEBEL CENTER FOR COMPUTER SCIENCE, UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN, USA
(KALE@ILLINOIS.EDU)

Abstract

This position paper makes two separate, only tangentially related points:

1. A research program is necessary to produce tools and techniques that allow application developers to develop and tune applications for future exascale machines, well before the machines are deployed.
2. In order to cope with, or facilitate, segmentation of exascale architectures, it is necessary to carry out an extensive study characterizing the needs and behaviors of applications that are expected to run on exascale machines.

Key words: emulation, exascale machines, application characterization, early application development, simulation

The International Journal of High Performance Computing Applications,
Volume 23, No. 4, Winter 2009, pp. 411–412
DOI: 10.1177/1094342009347703

1 Supporting Early Application Development and Tuning

Applications running on exascale machines are likely to exhibit structural complexity. For example, one will certainly not refine the physical domain uniformly, but rather use adaptive and dynamic refinements whenever possible; multi-module and multi-physics codes will predominate, with computations from multiple modules interleaving on overlapping sets of processors. Communication and computation phases would not be neatly separated, but rather adaptively overlapped to mitigate impacts of communication latencies and to distribute communication over time.

The machines in the exascale timeframe are also likely to be more complex in certain ways. Nodes may consist of a mix of multicore and accelerator chips and individual chips may contain heterogeneous cores. Interconnection topologies and topology-aware mapping of objects to processors may become increasingly important for performance as the diameters of networks increase.

These factors make developing and tuning an application for an exascale machine time-consuming. At the same time, since the exascale machine is an expensive resource, one would like strategically important applications to start running efficiently on a new exascale machine as soon as it is deployed.

Therefore, we need an ability to develop and tune an application for a particular future machine well before the machine is deployed. The software community should explore multiple approaches towards developing such a capability. A promising approach is to develop a full-scale virtual environment on which a future exascale application can be developed, tested and tuned. Using such an environment, application developers can, for example, identify scalability bugs in their data structures, as well as examine detailed performance profiles of their application runs at scale. Such an environment may take advantage of the fact that many Computer Science and Engineering (CSE) applications are repetitive in nature and simulating a few timesteps or iterations (at slower speeds) may be adequate for testing and tuning purposes.

One possibility is to use emulation based on virtualization and/or overdecomposition to allow at-scale development and testing using a machine, say, 10 times smaller than the target exascale machine. For true emulation, techniques for handling the mismatch between the emulating machine and the target machine need to be developed. For performance tuning, one can explore a variety of techniques, including detailed simulation that utilizes the information gathered during the emulation, as well as the machine and network characteristics to predict performance or, at least, identify potential performance bottlenecks.

2 Potential Segmentation of Exascale Architectures

It is possible that the class of exascale machines will be segmented into multiple categories. A likely segmentation is based on two dimensions: memory-per-core and bisection bandwidth. Of the four possibilities these two dimensions lead to, it is likely that at least two extremes would be populated. For example, one can construct an inexpensive machine with little memory per core and a near-neighbor network. It can compete effectively with machines that have full bisection bandwidth and large dynamic random access memory (DRAM) per core, consistent with today's balance criteria, IF there are significant numbers of applications that can exploit such machines. Many arguments can be made as to why each of the four segments is useful (see below). However, to settle this issue with the level of certainty that the vendors and funding agencies feel comfortable in designing and deploying architectures in each quadrant, a careful and comprehensive study of potential exascale applications is necessary.

One argument suggesting that several applications will require low memory per core is as follows (some applications, such as biomolecular simulations, are demonstrably in this category as the number of atoms involved in protein-DNA-membrane assemblies is relatively small). For continuously modeled spatial domains, the Courant limit or the non-linear increase number of iterations for linear system solvers implies that, as resolution is increased, the memory needed increases as $O(n^3)$ while the computation time increases at a higher rate, as large as $O(n^4)$ in many cases. Assuming the time for running a simulation is a constant (because that is related to the life-time constants of humans – such as a 3-hour run for a weather forecast, an overnight run for an engineering design or a "hero" science run lasting a few months), this implies that memory capacity needs to increase more slowly than processing speeds. This argues for the utility of low-memory-per-core machines. It so happens that memory costs are a significant component of supercomputer costs and interesting low-memory designs that effectively utilize the pin-bandwidth of individual chips are feasible. So, such a class is worth exploring.

However, of course, rather than relying on abstract arguments, we should study specific applications to see if they can run within such restricted regimes. This exercise will require us to identify at least an initial set of applications that can run at the exascale and have a specific societal (or pure science) benefit. It will force the community to do at-scale studies of such applications, possibly using the "early development and analysis" methodologies mentioned earlier. Explicit consideration of costs of architectures needed while designing next-generation applications is necessary to counteract the current, non-sustainable, trend of over-provisioning machines for all classes of applications. The exercise in characterizing applications will also identify bottlenecks that all applications may face, underscoring needs for architectural innovation in specific areas. Finally, it will identify classes of architectures that should be explored and deployed at the exascale.

Author Biography

Laxmikant Kale is a Professor of Computer Science at the University of Illinois at Urbana-Champaign, where he has been a faculty member since 1985. He received a Ph.D. in computer science from State University of New York, Stony Brook, in 1985. His research has involved various aspects of parallel computing, with a focus on enhancing performance and productivity via adaptive runtime systems, and designing programming abstractions based on use-cases from multiple applications. He led the development of Charm++ and AMPI programming systems that embody an adaptive runtime system. He has collaboratively developed well-known parallel applications in areas including biophysics, astronomy and quantum chemistry. He was co-recipient of a Gordon Bell award in 2002.