Early application development/tuning and application characterization/segmentation

Laxmikant (Sanjay) Kale

This report makes two separate, only tangentially related points:

1. A research program is necessary to produce tools/techniques that allows application developers to develop and tune applications for future exascale machines.

2. In order to cope with, or facilitate, segmentation of exascale architectures, it is necessary to carry out an extensive study characterizing needs and behaviors of applications that are expected to run on exascale machines.

Supporting Early Application Development and Tuning

Applications running on exascale machines are likely to exhibit structural complexity. E.g. 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 diameters of networks increase.

These factors make developing and tuning an application for 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.

We therefore 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 ability. 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 performance tuning, one can explore a variety of techniques, including simulation, for utilizing the information gathered during the emulation, as well as the machine and network characteristics to predict performance art, at least, identify potential performance bottlenecks.

Potential segmentation of exascale architectures:

It is possible that the class of exascale machines will be segmented in 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, 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. This can compete effectively with machines with full bisection bandwidth and large DRAM memory 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, Courant limit or the non-linear increase number of iterations for linear system solver 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 life-time constants of humans – such as a 3-hour run for weather forecast, an overnight run, a “hero” run lasting a few months), this implies that memory capacity needs to increase slower 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.

But 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 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. It will identify bottlenecks that all applications many face, underscoring needs for architectural innovation in specific areas. Finally, it will identify classes of architectures that should be explored and deployed at exascale.