Productivity in HPC Clusters

Bob Kuhn

Intel Corp.
Email: bob.kuhn@intel.com

Abstract. This presentation discusses HPC productivity in terms of: (1) effective architectures, (2) parallel programming models, and (3) applications development tools. The demands placed on HPC by owners and users of systems ranging from public research laboratories to private scientific and engineering companies enrich the topic with many competing technologies and approaches. Rather than expecting to eliminate each other in the short run, these HPC competitors should be learning from one another in order to stay in the race. Here we examine how these competing forces form the engine of improvement for overall HPC cost/effectiveness. First, what will the effective architectures be? Moore's law is likely to still hold at the processor level over the next few years. Those words are, of course, typical from a semiconductor manufacturer. More important for this conference, our roadmap projects that it will accelerate over the next couple of years due to Chip Multi Processors, CMPs. It has also been observed that cluster size has been growing at the same rate. Few people really know how successful the Grid and Utility Computing will be, but virtual organizations may add another level of parallelism to the problem solving process. Second, on parallel programming models, hybrid parallelism, i.e. parallelism at multiple levels with multiple programming models, will be used in many applications. Hybrid parallelism may emerge because application speedup at each level can be multiplied by future architectures. But, these applications can also adapt best to the wide variety of data and problems. Robustness of this type is needed to avoid high software costs of converting or incrementally tuning existing program. This leads to OpenMP, MPI, and Grid programming model investments. Third, application tools are needed for programmer productivity. Frankly, integrated programming environments have not made much headway in HPC. Tools for debugging and performance analysis still define the basic needs. The term debugging is used advised because there are limits to the scalability of debuggers in the amount of code and number of processors even today. How can we breakthrough? Maybe through automated tools for finding bugs at the threading and process level? Performance analysis capability similarly will be exceeded by the growth of hardware parallelism, unless progress is made.