JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 1, March 2004

High Performance Computing: Current Issues and Future Challenges

Paul E. Plassmann*
The Pennsylvania State University, University Park, Pennsylvania 16802

High Performance Computing (HPC), the capability of people to achieve computational objectives on existing computers, has made fantastic progress over the past decades. The situation today is excellent in many respects—high performance processors are relatively inexpensive, we understand how to build intermediately-sized parallel computers with scalable performance and cost, and more academic and industry groups than ever before have direct access to HPC resources. For example, such HPC systems are widely used to solve complex application problems in biology, chemistry, physics, mathematics, almost every engineering discipline, and even business and finance.

However, current HPC systems have a number of deficiencies that limit their application. For example, the programming environment for parallel computers has improved significantly but still represents a daunting challenge for many potential users. The capability to store and manipulate the large-scale data sets generated by many simulations on today's HPC machines is often prohibitively expensive. And finally, software reuse and interoperability is almost nonexistent in many communities, which forces substantial investments in software development and severely limits the potential benefits of HPC resources. Given this current situation, I would like to briefly address, from the perspective of the JACIC community, what I think the current issues and future challenges are in High Performance Computing.

Architectures for High Performance Computing

High Performance Computing was defined in the 1993 NSF Blue Ribbon Panel on High Performance Computing[1] as "a computation and communications capability that allows individuals and groups to extend their ability to solve research, design, and modeling problems substantially beyond that available to them before." This definition is still a good one, but much has changed in the field over the past decade—perhaps the most dramatic changes have been in the hardware available to the HPC community. One of the most remarkable changes has been the convergence in processor performance. A decade ago one could trade-off price for processor performance; today, however, almost all processors are relatively high performance. For example, using the SPEC CPU2000 benchmark as a performance metric, one can purchase a refurbished computer with a 2.8GHz Pentium IV processor for less than $400 whose performance is within a factor of two of the best performing processor available (an Itanium 2). Never have high performance processors been so affordable.

With the advent of Beowulf, or Commodity Off The Shelf (COTS) based architectures, high performance parallel computing is being employed by a wider community than ever before.[2] By leveraging the affordability of high performance processors and the availability of high bandwidth, low latency networks (e.g., Myrinet or Dolphin), it is now widely understood how to build distributed memory parallel computers with hundreds of processors that exhibit scalable performance on standard benchmarks such Linpack. The price to performance ratio for these machines is staggering to those of us who fondly remember supercomputer centers—a current 200-processor Beowulf can achieve a performance on the Linpack benchmark of more than half a teraflop for a cost well under a dollar per megaflop. Just seven years ago such a machine would have easily been the fastest computer in the world, yet today many moderate-sized groups have built and operate such parallel computers. For these groups such machines represent an unprecedented computational resource that is enabling significant advances in a wide variety of fields in computational science and engineering.

The Earth Simulator in Japan, currently first on the top 500 list of supercomputer sites can be said to represent the apex of the "cost is no object" approach to HPC. A significant difference between this architecture and the COTS approach (in addition to scale) is the use of the NEC vector processors. A distinct advantage of this architecture is

*Associate Professor, Department of Computer Science and Engineering, 343J Information Science and Technology Building. AIAA Member.

the prospect of better performance of legacy codes that have been optimized for vector compilers; however, the price-performance of COTS-based systems makes these architectures infeasible for all but the best-funded organizations. In addition, there are distinct efficiencies in a group directly managing a smaller cluster rather than time-sharing a state-of-the-art large machine. These efficiencies include the optimization of the architecture for a particular application, proximity to simulation data, direct control over scheduling policies, and security. In fact, I believe that it is increasing the case that the majority of the progress in applications requiring HPC will occur on such systems.

Programming Environments for High Performance Computing

The problem of developing a parallel simulation code that achieves a good percentage of this raw performance on a parallel computer can be a significant challenge for user groups. For these computers, the message-passing programming model based on a standard language, such as C, C++, or Fortran, plus the industry standard message-passing programming interface MPI has been widely adopted. This programming environment is available on most any parallel computer, and simulation codes written under this model are to a large extent ensured portability between different parallel computers.

The wide acceptance of this parallel programming environment has made possible the education of a large user community with a common background. However, developing software with this programming model represents a significant challenge relative to programming for a sequential computer or using commercial programming environment. To address this issue, many universities have developed HPC educational programs (usually a certificate, option, or minor) based on backgrounds in programming, algorithms and data structures, parallel programming with MPI (or openMP for the shared memory programming model), and a detailed introduction to one or more application areas. These programs are highly popular with students from a wide variety of Science and Engineering disciplines. The continued support by universities of these programs helps to ensure a growing community of educated HPC users with the capability to make significant advances in their respective fields through HPC.

Perhaps the central challenge for the future of software development for HPC applications is not specific simulation codes, but rather the interoperability of these software systems. An analogy might be made to the development of widely available software libraries on sequential computers for the solution of common tasks such as the solution of linear equations or ordinary differential equations, fast Fourier transforms, or standard visualization methods. The availability of robust and efficient implementations through software libraries of these core algorithms is what makes highly sophisticated simulations possible. To a large extent, this core functionality still does not exist for parallel HPC systems.

One exciting approach toward providing this interoperability is work being done to provide a Common Component Architecture (CCA) that provides a mechanism for encapsulating functionality and the high-level specification of component interfaces.[3] Such new approaches are promising; however, in the interim it is probably more important that individual groups make it a priority to document, test, and distribute software that they develop to a wider community. As we have seen with the development of HPC software for sequential computers, it has been the leveraging of innovations in algorithms and their software implementations that has made possible many of the significant advances in computational science and engineering.

Although the popularity of COTS-based systems has made a HPC level of performance available to a wide audience, this architecture does exhibit a number of limiting characteristics. Perhaps the most limiting, as anyone using such systems for large-scale simulations has found out, is their inability to store and manipulate large-scale datasets in a scalable way. For example, the state of large-scale fluid simulation on a medium scale parallel machine can be on the order of hundreds of gigabytes. A time-accurate simulation can easily generate terabytes of data per hour, or petabytes of data over the course of the entire simulation. In addition, many simulation codes are still written based on a networked file system where the simulation state is periodically written out to a large, shared file for postprocessing or checkpointing. Efforts have been made to develop a standardized interface for parallel input/output; however, the underlying architectural limitations of these COTS-based systems make their scalable implementation impossible. Ultimately, a better approach has to be developed to archive such simulation data. Potential avenues of research that may help to address this problem include the development of new data compression algorithms, feature extraction and dimensional reduction methods, or new database technologies specific for these large-scale simulation data sets.

Summary

The potential for HPC over the next decade is extraordinary. With the continued improvements in processor performance and affordability, the availability of significant computational resources to individual research groups should continue to improve. In fact, Gordon Bell has coined a nice term for this democratization of computational resources—he calls it "community centric computing."[4] For the JACIC community in particular, these advances in High Performance Computing represent unprecedented opportunities. As stated in the journal's scope, I look forward to seeing significant contributions ``to the applied science and engineering of aerospace computing, information, and communication" enabled by these opportunities in high performance computing.

References

[1]NSF Blue Ribbon Panel on High Performance Computing, Lewis Branscomb (chairman), "From Desktop to Teraflop: Exploiting the U.S. Lead in High Performance Computing," Technical Report NSB 93-205, National Science Foundation, Aug. 1993. Report text available online at http://www.nsf.gov/pubs/stis1993/nsb93205/nsb93205.txt.

[2]Sterling, T., Salmon, J., Becker, D. J., and Savarese, D. F., *How to Build a Beowulf*, The MIT Press, Cambridge, MA, 1999.

[3]Armstrong, R., Gannon, D., Geist, A., Keahey, K., Kohn, S., McInnes, L., Parker, S., and Smolinski, B., "Toward a Common Component Architecture for High-Performance Scientific Computing," *Proceedings of the 8th IEEE International Symposium on High-Performance Distributed Computing*, IEEE, Piscataway, NJ, Aug. 1999. Also available as Argonne National Laboratory preprint ANL/MCS-P759-0699 at ftp://info.mcs.anl.gov/pub/tech reports/reports/P759.ps.Z.

[4]Bell, G., "Progress, Prizes, and Community-Centric Computing," *International Conference on Computational Science (ICCS2003)*, June 2003. Also available online at http://research.microsoft.com/users/GBell/Supers/ICCS2003 Melbourne 030602.ppt.