APPLICATIONS

David A. Bader
NEW MEXICO, USA

Robert Pennington
NCSA, USA

The International Journal of High Performance Computing Applications, Volume 15, No. 2, Summer 2001, pp. 181-185

1 Introduction

Cluster computing for applications scientists is changing dramatically with the advent of commodity high performance processors, low-latency/high-bandwidth networks, and software infrastructure and development tools to facilitate the use of the cluster. The performance of an individual processor used in a high-end personal workstation rivals that of a processor in a high-end supercomputer, such as an SGI Origin, and the performance of the commodity processors is improving rapidly. For example, MIMD Lattice Computation (MILC) (Bernard et al.) executes at

- 55 Mflop/s on an Intel 300 MHz Pentium II
- 105 Mflop/s on an Intel 550 MHz Pentium III Xeon
- 165 Mflop/s on a single 250 MHz R10000 processor in an SGI O2000

In the aggregate, the performance is even more impressive. The excellent performance of the interconnect network and related software is shown by comparing the same application running on a large cluster and a large SGI Origin. On 128 processors using Message-Passing Interface (MPI), MILC executes at

- 4.1 Gflop/s on 128 Intel 300 MHz Pentium Iis
- 8.4 Gflop/s on 128 Intel 550 MHz Pentium III Xeons
- 9.6 Gflop/s on a 128-processor SGI O2000 with 250 MHz R10000 processors

The successful standardization of system-level interfaces and supporting libraries has removed many of the issues associated with moving to a new computational platform. High performance clusters are one of the new platforms that are coming to the forefront in the computational arena. There are still a number of outstanding problems to be resolved for these systems to be as highly effective on applications as a supercomputer. For the current generation of clusters, the most significant areas for the applications scientists are performance of the I/O systems on a cluster system and compilation, debugging, and performance monitoring tools for parallel applications.

2 Application Development Environment

It is essential that the applications scientist be able to move the code to new systems with a minimum amount of effort and have the same or similar tools and environment

available to use on different systems. As recently as a few years ago, the lack of a canonical high performance architectural paradigm meant that migrating to a new computer system, even from the same vendor, typically required redesigning applications to efficiently use the high performance system. As a result, the application had to be redesigned with parallel algorithms and libraries that were optimized for each high performance system. This process of porting, redesigning, optimizing, debugging, and analyzing the application is generally prohibitively expensive and certainly frustrating when the process is completed just in time to greet the next-generation system.

"The importance of supporting portable code that uses accepted standards, such as MPI (Message Passing Interface Forum, 1995), on clusters cannot be overstressed. The current generation of clusters is becoming useful to researchers and engineers precisely because of its direct support of these standards and libraries."

The importance of supporting portable code that uses accepted standards, such as MPI (MPI Forum, 1995), on clusters cannot be overstressed. The current generation of clusters is becoming useful to researchers and engineers precisely because of its direct support of these standards and libraries. Research software messaging systems and interfaces from vendors such as the Virtual Interface Architecture (Compaq, Intel, and Microsoft, 1997) may provide the underlying protocol for a higher level message-passing interface such as MPI but are not directly useful to the vast majority of applications scientists.

The development environment for clusters must be able to provide the tools that are currently available on supercomputing systems. The current sets of applications running on supercomputing clusters are generally developed, debugged, and tested on supercomputing systems. For first-generation applications that may be developed for large-scale clusters, tools such as debuggers, profilers, and trace utilities that work identically or very similarly to those available on supercomputers need to be available to the applications developers. Common tools such as shell, make, tar, and scripting languages also need to be included in the development environment.

3 Application Performance

To illustrate the competitive performance of clusters, we show an application, MILC, which has been run on a large-scale cluster in the Alliance, the NT Supercluster (Chien et al., 1999), at the National Center for Supercomputing Applications (NCSA). This application is also being used on the Roadrunner Supercluster (Bader et al., 1999) at the University of New Mexico. In addition, one of the major performance issues confronting clusters is exemplified in the data for the ARPI 3D weather mode code, which shows timings for computations, message passing, initialization I/O, and I/O at the completion of the application.

3.1 MILC

Kostas Orginos and Doug Toussaint from the University of Arizona undertook these benchmarks. They are for the conjugate gradient calculation of quark propagators in quantum chromodynamics with Kogut-Susskind quarks. The conjugate gradient typically takes 90% of the time in full QCD calculations, so it is the reasonable thing to measure. These are with the simplest version of the action, which is fairly standard. In each case, they have chosen the lattice size so that there are 4096 lattice sites per processor. This is in the range typically used in production runs. For the simplest variants of the code, this works out to about 4 MB of data per process.

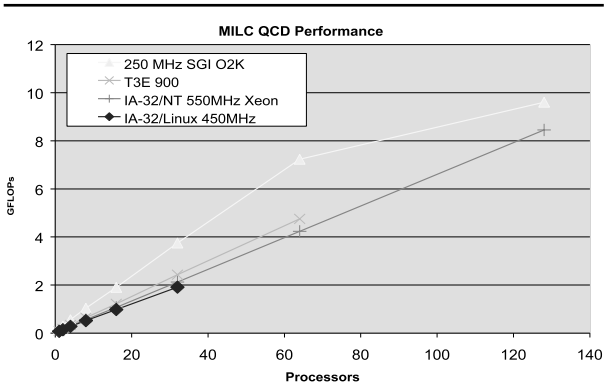

Fig. 1 MILC QCD performance

3.2 ARPI 3D

These graphs show the performance of the ARPI 3D weather research model on two large-scale clusters: the NCSA NT Supercluster and the University of New Mexico Roadrunner Linux Supercluster. These particular tests use a 3D numerical weather prediction model to simulate the rise of a moist, warm bubble in a standard atmosphere. The simulations were run to 50 seconds with a $35 \times 35 \times 39$ grid, in which time each processor wrote three 2.107 MB/s files to a centralized file system. The data were provided by Dan Weber at the University of Oklahoma from his runs on the two systems.

The NT Supercluster data were taken in two different modes. In the first mode, only one processor in the dual-processor systems was being used by the application, and the second processor was idle. In the second mode, both processors in the dual-processor systems were being used by the application. The dual-processor runs show approximately a 20% performance degradation compared with runs using the same number of processors in separate systems. The Roadrunner data are available for only the one processor per dual-processor system case.

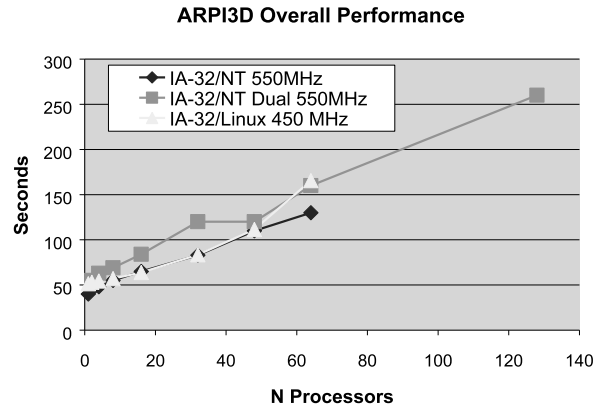

Fig. 2 ARPI 3D overall performance

This is a very well instrumented code, and it shows where the time is spent on the system to determine where the performance bottlenecks are located in the systems. The instrumentation provides times for the

- Overall runtime of the application
- Amount of time spent performing initialization with an input data read from a common file system
- Amount of time spent doing the computations
- Amount of time spent performing message passing
- Amount of time spent writing out the resultant data files to a common file system

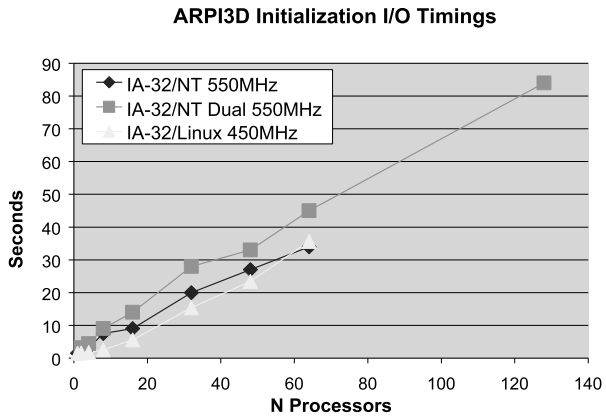

Fig. 3 ARPI 3D initialization I/O timings

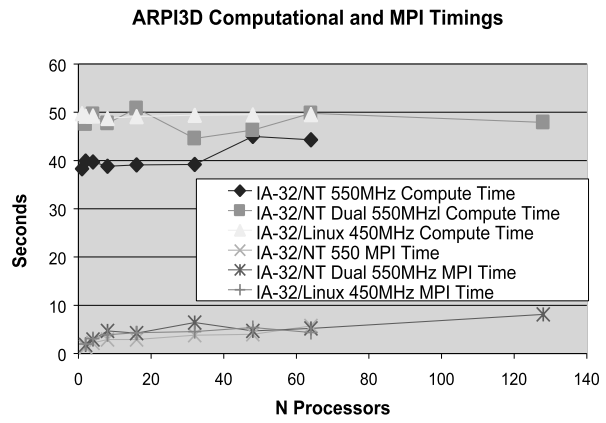

Fig. 4 ARPI 3D computational and Message-Passing Interface (MPI) timings

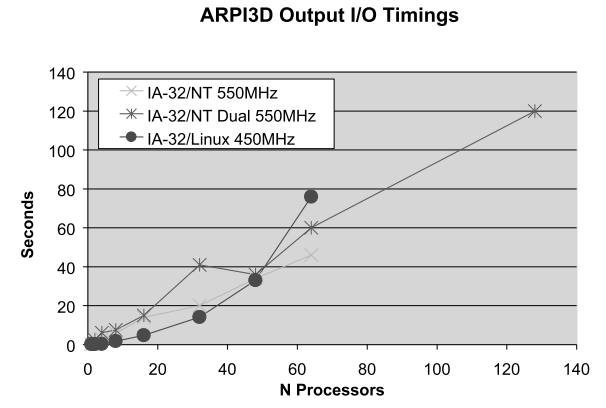

Fig. 5 ARPI 3D output I/O timings

Figure 3 shows the overall performance of the application on the two systems. The amount of time necessary for the runs increases roughly linearly as the number of processors increases.

The size of the computational task per processor remains the same, and the time taken for the computations is clearly shown to be nearly constant as a function of the number of processors in Figure 3—roughly 40 to 50 seconds, depending on the processor speed. Also shown in Figure 3 is the time spent by the application performing MPI calls, and this is insensitive to the number of processors at 5 to 10 seconds for all of the runs, illustrating excellent scalability.

Figures 4 and 5, respectively, show the time spent reading in an initialization file and writing out an output file. The input data file that is read by all of the processors from the common file system is approximately 8 KB, and each processor writes three output data files of 2.107 MB/s. The total volume of data moved is proportional to the number of processors in use, up to 800 MB/s of output for the 128-processor NT run. Both superclusters have a similar type of configuration with a common file system available from a file server to the compute nodes across the commodity Ethernet, differing primarily in the speed of the link from the file server to the switch and the file-sharing software.

The IA-32/NT Supercluster file-serving configuration:

- Fast Ethernet from the file server to the switch
- Compute nodes are all on individual Fast Ethernet nodes on the switch
- SMB to all nodes

The IA-32/Linux Roadrunner file-serving configuration:

- Gigabit Ethernet from the file server to the switch compute nodes are all on individual Fast Ethernet ports on the switch
- NFS to all nodes

At smaller numbers of processors, the increased bandwidth of the server to the switch for the Roadrunner Linux system is apparent. At larger numbers of processors, the performance of the NT file server is clearly better. The NT Supercluster I/O times for input and output both increase linearly with the number of processors. In neither case is the I/O performance of the system sufficient for what is actually needed by the application. The overall increase in execution time as a function of number of processors is due

almost entirely to the increased time necessary for file I/O as the number of machines increases.

4 Conclusions

Applications can perform very well on current-generation clusters with the hardware and software that is now available. There are a number of areas where major improvements can be made such as the programming tools to generate new applications on these systems and the I/O systems that are available for clustered systems.

BIOGRAPHIES

David A. Bader is an assistant professor of electrical and computer engineering at the University of New Mexico and received a Ph.D. in electrical engineering in 1996 from the University of Maryland. His recent research experiences highlight his ability to bridge the gap between application and computer science—for example, working closely with Earth scientists at NASA/GSFC to produce a terascale computing system for land cover dynamics while a National Science Foundation (NSF) CISE postdoctoral research associate in experimental computer science. He is also coinvestigating NSF biological applications in computational biology, genomics, and landscape ecology. His recent research has produced a new, preliminary methodology for programming UMA shared-memory machines and clusters of symmetric multiprocessor nodes, supported in part by an NSF Information Technology Research (ITR) award in advanced computational research.

Rob Pennington currently heads the Computing and Communications Division at the National Center for Supercomputing Applications (NCSA). In this role, he is responsible for the major systems and supporting infrastructure at the NCSA, which include a large SGI Origin cluster, Linux and Windows clusters, the mass storage system, and networking. These resources include production systems that are available to the national community as well as research and development systems, with a strong emphasis on clusters. Prior to this, he was the head of the NT Cluster Group at the NCSA, which has responsibility for the NT Supercluster. The group is focused on supporting high performance computing on a large NT cluster and making it a production system for NCSA users. In addition to experience with clusters, he has worked on mass storage system performance analysis and usage patterns as part of the High Performance Data Management Group at the NCSA and oversaw the transfer across the vBNS of more than 2 TB of user's data to the NCSA from the Pittsburgh Supercomputing Center for the PACI program. Prior to the NCSA, he led the Advanced Systems Group at the PSC and was working with software interfaces and performance analysis for mass storage systems. He received a Ph.D. in astronomy (1985) from Rice University.

REFERENCES

Bader, D. A., Maccabe, A. B., Mastaler, J. R., McIver, J. K., III, and Kovatch, P. A. 1999. Design and analysis of the alliance/University of New Mexico Roadrunner Linux SMP SuperCluster. *Proceedings of the IEEE International Workshop on Cluster Computing*, Melbourne, Australia.

Bernard, C., Blum, T., De, A., DeGrand, T., DeTar, C., Gottlieb, S., Krasnitz, A., Karkkainen, L., Labrenz, J., Sugar, R. L., and Toussaint, D. *Recent Progress on Lattice QCD with MIMD Parallel Computers*. MIMD Lattice Computation (MILC) Collaboration, a Grand Challenge Application Group.

Chien, A., Lauria, M., Pennington, R., Showerman, M., Iannello, G., Buchanan, M., Connelly, K., Giannini, L., Koenig, G., Krishnmurthy, S., Liu, Q., Pakin, S., and Sampemane, G. 1999. Design and analysis of an HPVM-based Windows supercomputer. *International Journal of High Performance Computing Applications* 13:201-219.

Compaq, Intel, and Microsoft. 1997. *Virtual Interface Architecture Specification, Version 1.0* [Online]. Available: http://www.viarch.org.

MPI Forum. 1995. MPI: A message-passing interface standard. Technical report, University of Tennessee, Knoxville.