Communication Overhead for Space Science Applications on the Beowulf Parallel Workstation

Thomas Sterling
Center of Excellence in Space Data
and Information Sciences
Code 930.5 NASA Goddard Space Flight Center
Greenbelt, MD 20771
tron@cesdis.gsfc.nasa.gov

Daniel Savarese
Department of Computer Science
University of Maryland
College Park, MD 20742
dfs@cs.umd.edu

Donald J. Becker
Center of Excellence in Space Data
and Information Sciences
Code 930.5 NASA Goddard
Greenbelt, MD 20771
becker@cesdis.gsfc.nasa.gov

Bruce Fryxell Kevin Olson
Institute for Computational Science
and Informatics
George Mason University
{fryxell@neutrino, olson@jeans}.gsfc.nasa.gov

Abstract

The Beowulf parallel workstation combines 16 PC-compatible processing subsystems and disk drives using dual Ethernet networks to provide a single-user environment with 1 Gops peak performance, half a Gbyte of disk storage, and up to 8 times the disk I/O bandwidth of conventional workstations. The Beowulf architecture establishes a new operating point in price-performance for single-user environments requiring high disk capacity and bandwidth. The Beowulf research project is investigating the feasibility of exploiting mass market commodity computing elements in support of Earth and space science requirements for large data-set browsing and visualization, simulation of natural physical processes, and assimilation of remote sensing data. This paper reports the findings from a series of experiments for characterizing the Beowulf dual channel communication overhead. It is shown that dual networks can sustain 70% greater throughput than a single network alone but that bandwidth achieved is more highly sensitive to message size than to the number of messages at peak demand. While overhead is shown to be high for global synchronization, its overall impact on scalability of real world applications for computational fluid dynamics and N-body gravitational simulation is shown to be modest.

1 Introduction

The Beowulf parallel workstation defines a new operating point in price-performance for single-user computing systems. Beowulf couples the low cost, moderate performance of commodity personal-computing subsystems with the emergence of de facto standards in message passing hardware and software to realize a 1 Gops workstation with exceptional local file storage capacity and bandwidth. This experimental system is motivated by requirements of NASA Earth and space science applications including data assimilation, data set browsing and visualization, and simulation of natural physical systems. It exploits parallelism in processor, disk, and internal communication, all derived from mass market commodity elements. Thus enabling large temporary data sets to be buffered on the workstation in order to reduce demand on shared central file servers and networks while greatly improving user response time. This paper presents results of experiments to characterize the communication overhead of the Beowulf parallel workstation and establish the regime of effective operation.

While most distributed computing systems provide general purpose multiuser environments, the Beowulf distributed computing system is specifically designed for single user workloads typical of high end scientific workstation environments. The Princeton Shrimp [2] project is also targeted to parallel workstation systems comprising multiple personal-computer processors. This Pentium based distributed computer employs a custom communication unit to support a distributed shared memory model. Beowulf, by contrast, incorporates no special purpose parts, depending instead on parallel ethernet communication channels to achieve adequate sustained interprocessor message transfer rates. This has required some software enhancements at the operating system kernel level but has been achieved with commercial off-the-shelf hardware elements, specifically low cost Ethernet cards.

Much of the workstation operational demand is very coarse grained job stream parallelism. But, as with all workstations, some of the required workload is computationally intensive. Thus, there is a need to exploit parallelism within a single application. Ironically, where the solving of the parallel processing problem would ordinarily prove a challenge, in the computational sciences community, many active applications have already been crafted in the communicating sequential processes parallel programming style in order to run effectively on larger distributed computers such as the Intel Paragon [7], the TMC CM5 [14], or the CRI T3D [3]. Beowulf benefits from this parallel programming investment within the community it is intended to serve and provides an equivalent programming and compilation environment at the parallel workstation level. A number of parallel applications have been successfully, sometimes even easily, ported to Beowulf in this manner.

Beowulf inter-processor communications is provided by standard 10 Mbps Ethernet using dual channels with each channel connecting all 16 processing elements. These channels are equally accessible to all processors and the operating system kernel (based on Linux [8]) has been modified to dynamically distribute message packet traffic to load balance across both networks. Interprocessor communications performance may be characterized in several ways and this paper presents experimental results reflecting these aspects of communication on execution performance. Basic network capacity is characterized in terms of throughput, both as byte transfer rate and number of messages passed per unit time. These measurements are presented as functions of message size, message demand, and number of channels employed (one or two). Finally, the impact of parallel interprocessor communication is explored through two real-world application programs from the Earth and space sciences community. One problem is a computational fluid dynamics application employing a regular static data structure well suited to a system of Beowulf's architecture. The second is an N-body gravitational simulation with irregular dynamic global data that challenges the capabilities of Beowulf's communication. Both the scaling properties of these two applications and the communication overhead encountered will be presented.

2 Beowulf architecture

The Beowulf parallel workstation architecture comprises 16 PC processor subsystems, each with a half GByte disk and controller. The Beowulf prototype incorporates the Intel DX4 processor with a 100MHz clock rate and 256 KBytes of secondary cache. The DX4 delivers greater computational power than other members of the 486 family not only from its higher clock speed, but also from its 16 KBtye primary cache (twice the size of other 486 primary caches) [6]. Each processing subsystem has installed 16 MBytes of DRAM for a total system main memory of 256 MBytes. The processing elements are interconnected by two parallel Ethernet networks with peak capacity of 10 Mbps per network. For purposes of experimentation, one network is twisted pair using a small, inexpensive hub while the other is multidrop thin-net. Two processor subsystems include separate ethernet interfaces to external LAN's for remote access and data transfer. Two additional processor subsystems include high resolution video controller/drivers, one of these also providing user keyboard and mouse interface for conventional single-user workstation access. The Beowulf prototype subsystems are being packaged in a single floor-standing half-height utility cabinet including secondary storage, four 3.5" floppy drives, two CD-ROM drives, and fans and power supplies.

The Beowulf parallel architecture, while in many respects a conventional distributed structure, was developed to meet specific critical requirements dictated by the Earth and space science community. The NASA HPCC program and in particular the ESS project (involving the NASA Goddard Space Flight Center and CalTech JPL) identified the need for very high performance workstations at the level of a Gops for large science applications. The class of use of the required system was diverse and included simulation of physical processes and data browsing and navigation as well as visualization of large data sets. Therefore, beyond processor performance, large disk capacity and high disk I/O transfer rates were determined to be essential. Out of necessity and because of the experimental nature of the project and architecture, a base of system software had to be available that would provide robust and sophisticated operating system support while having available full source code and documentation. Perhaps more importantly, NASA's commitment to technology transfer with industrial and academic sectors required that all aspects of the Beowulf parallel workstation results be made easily available to other centers of computer and physical sciences that could benefit from the NASA results. Finally, the very nature of Beowulf as a single-user workstation demanded that the marginal replacement cost of the system be consistent with pricing of contemporary high end workstations, although of lower performance than anticipated by Beowulf.

Beowulf is an applied research project devised to explore an operating point for workstations not yet addressed by the commercial marketplace and of particular interest to the high performance computational science community. In light of the requirements above, a set of architecture choices were made that distinguishes Beowulf from conventional high end workstations and from many other distributed computing systems. The most apparent are the high degree of parallelism exploited and the use of personal-computing mass-market subsystems rather than the somewhat higher performance workstation oriented microprocessors. This scale of structure and devices were chosen based on careful analysis of the requirements and technology opportunities. In particular, the class of workload in a workstation is often different from that demanded of a typical distributed high performance computing system. A workstation workload may consist of a number of decoupled or at most loosely coupled tasks to manage the environment and resources. While it is expected that a shared memory environment is necessary for an effective workstation capability, the Beowulf project tests the viability of serving this role with a distributed memory logically related only through the process name space and message passing mechanisms. This distinguishes Beowulf from the Shrimp [2] project which will provide distributed shared memory through custom logic.

The use of PC-derived processors was determined not so much from the relatively low cost of the processors themselves, but from the low cost of the full processing subsystems (motherboards) due to true mass market scale of fabrication and distribution at the consumer level. This market is orders of magnitude larger than the workstation market and the cost benefits resulting from this economy of scale provided a new operating point largely unexplored by the scientific community. The full COTS subsystems and available networks were anticipated to impose a bottleneck. Two approaches were identified to alleviate this problem and achieve necessary interprocessor communication rates. The first, being tested by the Beowulf prototype and explored in this paper, is the use of parallel networks. If message packet traffic demand could be load balanced across multiple Ethernet networks, bandwidth scaling should be achievable. The second approach is the exploitation of the emerging 100 Mbps Ethernet networks. While not integrated into the Beowulf prototype, this exciting technology is being tested separately. In a previous paper [12], interprocessor communication was shown to constrain file transfer rates under certain circumstances but that the higher level of bandwidth possible from this new technology would largely alleviate this problem resulting in a balanced architecture.

The emergence of de facto standards in message passing hardware and software mechanisms along with the already considerable investment in scientific message passing application software development permits at least some important parallel computations to be performed on Beowulf without the use of a shared memory context. Some exceptions will exist and hinder Beowulf as a universal scientific workstation. But, as is shown in section 4, important problems in the Earth and space sciences, even with complex and dynamic resource demands can be effectively accomplished by the Beowulf architecture.

The Beowulf scalable communications is implemented by duplicating the hardware address of a primary network adaptor to the secondary interfaces, and marking all packets received on the internal networks as coming from a single pseudo-interface. This scheme constrains each internal network to connect to each node. With these constraints the Ethernet packet contents are independent of the actual interface used and we avoid the software routing overhead of handling more general interconnect topologies. The only additional computation over a using single network interface is the computationally simple task of distributing the packets over the available device transmit queues. The current method used is alternating packets among the available network interfaces.

3 Communication channel capacity

The degree to which multiple processors can be combined to perform a single application is ultimately limited by the capacity of the global interprocessor communication network. A series of experiments was conducted to characterize the capability of the multiple Ethernet communication channels provided for data transfer between processors and global synchronization. This is complicated by the many degrees of freedom that determine the domain over which communication can transpire. Also for communication capability, there is more than one metric of performance that should be considered.

In this section, communication capability is investigated using a synthetic program, the purpose of which is to generate message traffic. The test program exchanges a token (message) between a pair of processors. One processor generates a token which will comprise one or more packets and sends it to the second processor. This second processor then generates the same message and returns it to the first. The minimum transfer time is determined by hardware and software considerations and accurately reflects all factors contributing to sustained communication performance. The demand rate is increased by adding processor pairs ping-ponging tokens back and forth. While there is no contention for processors (no processor is involved with more than one token) the global communication channels are resources subject to possible contention. With 16 processors and 8 processor pairs, the range of active tokens is from 1 to 8. As will be seen shortly, this is enough to reveal the domain of network performance behavior.

A second parameter of importance to network performance is the size of the tokens measured in bytes. For these experiments, the token size is varied between four bytes (plus the header information) to 8K bytes. Ethernet packets can contain to about 1K bytes so the large tokens are implemented by the system using multiple packets. The third parameter is the number of channels active at one time. Here results for one and two channel configurations are reported, although some data has been collected on three channel organizations of 8 processors. The two performance metrics examined are sustained bandwidth or information throughput measured in MBytes per second, and message throughput measured in tokens per second.

The data bandwidth empirical results are presented in Figure 1 with the throughput in MBytes/sec plotted with respect to the size of the tokens measured in bytes and itself presented as log base 2. This chart shows 16 separate curves divided in two groups; solid lines represent runs using only one Ethernet and double-dashed lines show measurements of experiments using tow Ethernet networks simultaneously. For each of these two groups, there are 8 curves distinguished by the level of token demand; between 1 and 8 active ping-ponging tokens. The legend in the figure relates the plot-point symbols to the token demand.

The empirical results clearly demonstrate that the

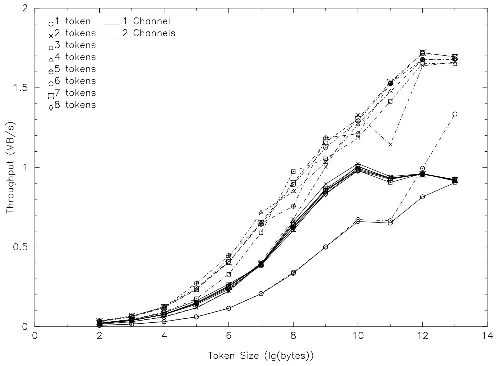

Figure 1: Beowulf Data Bandwidth

data rate has a strong relationship to token size and is insensitive to the token demand. Except for a token demand of 1, all higher demands are tightly clustered exhibiting about the same throughput for the same token size. This is a product of contention for the network that largely serializes the communication. A demand of 1 does not stress the network resources, but a demand of two or more tokens does, resulting from the contention arbitration strategy.

Where two networks are used, a similar clustering is demonstrated but at a significantly higher overall bandwidth for a given token size. The spread of behavior is broader for two networks than one because of the addition of the packet distribution algorithm included in the Beowulf Linux kernel. The packet dispatch algorithm employed by each processor is simply to use one network and then the other, blindly. While this does not exploit possibly useful information about traffic patterns, we see that generally, the dual network approach using this simple strategy returns 1.7 times the sustained bandwidth of the single network at the highest levels achieved. This is good scaling for such a trivial dispatch strategy. The effectiveness of two networks is even demonstrated with a demand of only one token. When the token size exceeds that of a single packet, multiple packets can take advantage of the availability of dual channels showing a 50% improvement over the single channel case for 8K byte tokens.

Beyond a certain packet size, throughput is dominated by network bandwidth. For the single channel case this occurs at token size of 1K bytes yielding a sustained throughput of about 1 MByte per second. This is good when compared to an absolute peak of 1.25 MBytes per second. For dual channels, saturation is not reached until a token size 4K bytes. It is

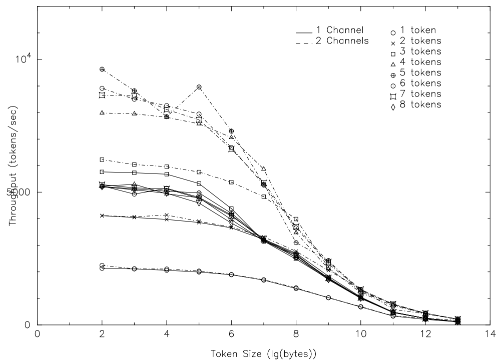

Figure 2: Beowulf Message Throughput

observed that even for a 1 token demand using a single channel, the saturation level is reached with tokens of 8k bytes.

The general behavior of this networking scheme is better illustrated in Figure 2 where the rate of token transfer is presented with respect to the token size. Again, single and dual network measurements are plotted and distinguished by token demand. The number of tokens transferred per unit time is insensitive to token size for small tokens but as the size significantly exceeds the packet send overhead, the token transfer rate declines until network saturation forces a direct tradeoff between token size and number of tokens transferred per unit time. As previously seen, generally the dual network configuration significantly out performed the single network system. But an interesting anomaly is exposed for the two token demand case. The dual network system does little better in terms of token transfer rate than the single network system until token size exceeds 256 bytes. At the time of submission of this research paper, this phenomenon is not understood and requires further study. Especially for the dual network configuration, token transfer rate shows marked sensitivity to token demand at least up to 5 active tokens.

Two key conclusions are drawn from these finding. The first is that multiple channels do scale in terms of both data bandwidth and token transfer rate. The second is that token size is more important than token demand for deriving best performance from available communication resources. The first conclusion motivates the use of parallel communication channels. The second conclusion requires application programmers to package their data in large aggregates rather than sending many smaller packets.

4 Applications scaling and communication overhead

We have selected two applications for this study which stress the computer architecture in very different ways. The first solves the equations of compressible gas dynamics on a structured grid. This algorithm is computationally intensive and requires only a small amount of local communication. As a result, the parallelization overhead is relatively small. The second code solves the gravitational N-body problem using a tree algorithm. Since the tree is unstructured, the code makes frequent use of indirect addressing and requires global communication.

4.1 Compressible Gas Dynamics

The first code to be discussed solves Euler's equations for compressible gas dynamics on a structured, logically rectangular grid. The code, named PROMETHEUS [5], has been used primarily for computational astrophysics simulations, such as supernova explosions non-spherical accretion flows, and nova outbursts. Euler's equations are solved using the Piecewise-Parabolic Method (PPM) for hydrodynamics [4]. This is a very high-resolution finite volume technique, which is particularly well-suited to calculation of flows which contain discontinuities such as shock fronts and material interfaces. Systems can be studied in either one, two, or three spatial dimensions using rectangular, cylindrical, or spherical coordinates. The results discussed below are for calculations performed on a two-dimensional rectangular grid.

This code was parallelized using a domain decomposition technique, in which the grid was divided into a number of rectangular tiles. This approach has been used successfully on a large number of parallel computers, including the Cray C-90, MasPar MP-1 and MP-2, IBM SP-1 and SP-2, Cray T3D, and Intel Paragon. For the case of Beowulf, each processor is assigned one or more tiles to calculate. Each tile is surrounded by a frame of *ghost* points which are used for specifying boundary conditions. Since the formulation of the PPM algorithm used in PROMETHEUS is a nine-point scheme, the *ghost* points need to be updated only once per time step if frame is four grid points wide. The only communication required using this approach is that four rows of values must be exchanged between adjacent tiles once per time step. Since a few thousand floating point operations are needed to update each zone for a single time step, the ratio between communication costs and computational costs is quite low.

4.2 The Gravitational N-body Problem

Tree codes are a collection of algorithms which find an approximate solution to the equations of force of a system of gravitationally interacting particles [1]. In these algorithms the particles are sorted into a spatial hierarchy which forms a tree data structure. Each node in the tree then represents a grouping of particles. Data which represents average quantities of these particles (e.g. total mass, center of mass, and high order moments of the mass distribution) are computed and stored at the nodes of the tree. The forces are then computed by having each particle search the tree and pruning subtrees from the search when the average data stored at that node can be used to compute a force on the searching particle below a user supplied accuracy limit. For a fixed level of accuracy this algorithm scales as $Nlog(N)$ although $O(N)$ algorithms are also possible.

Since the tree search for any one particle is not known *a priori* and the tree is unstructured, frequent use is made of indirect addressing. This presents problems for distributed memory, parallel implementations of this algorithm since one wishes to minimize any off processor accesses of data. One the other hand, the problem does possess a highly parallel component: each particle searches the tree structure completely independently of all other particles in the system.

4.3 Performance Scaling

The scaling characteristics of these two application codes were tested were evaluated on the Beowulf parallel workstation. The results are shown in Figure 3. For the N-body code, the problem size was kept fixed for all processor numbers. Two separate curves are shown for the PPM gas dynamics code. For one case, the problem size was kept fixed, while for the second curve, the problem size was scaled with the number of processors so that the amount of work assigned to each processor remained fixed. Both codes were run on up to 16 processors. For the PPM code, the tile size was kept fixed at 30×30 (38×38 including the *ghost* points).

For fixed problem sizes, both codes showed significant performance degradation when run on 8 or more processors. The performance increased only about 10% when the number of processors was increased from 8 to 16. When 16 processors are used, there is too

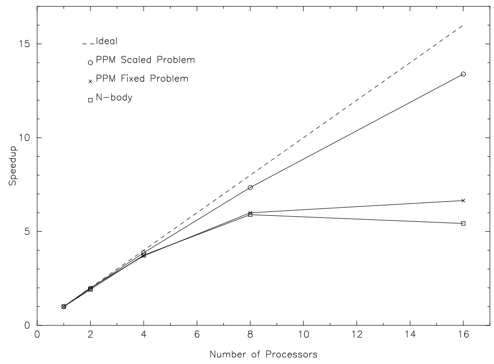

Figure 3: ESS Code Scaling

little work left per processor to make up for the parallelization overhead. Single processor performance for the N-body code was 3.5 MFLOPS, while the PPM code achieved 4.5 MFLOPS on a single processor. For the full 16 processor Beowulf, the speed attained was 19 MFLOPS for the N-body code and 31 MFLOPS for the PPM code. For comparison, a shared memory version of the N-body code on the Convex SPP-1000 [10] with 8 processors yields a performance of 216 MFLOPS [11].

Much better characteristics were obtained for the PPM code when the problem size was scaled with the number of processors. At 16 processors, the performance decreased by only 16% with respect to ideal. Single processor performance for the code was 4.5 MFLOPS. The full 16 processor Beowulf delivered 62 MFLOPS. This compares favorably with the performance of a comparably sized TMC CM-5 (without the vector chips) and the Intel Paragon. The CRI T3D achieved less than a factor of 2.5 better than Beowulf for the same number of processors. Whether the scaling performance of the N-body code will also improve if the problem size is scaled with the number of processors remains to be determined.

The parallelization overhead for both codes is plotted in Figure 4. As expected, the overhead for the N-body code is quite high due to the large amount of global communication required. The overhead actually exceeds 50% for 16 processors, meaning that more time is spent in overhead than in communication. Surprisingly, for the PPM code with fixed problem size, the situation is even worse. For 16 processors, the overhead is greater than 60%. In this case, however, the overhead is not due to communication time. The total time required for the communication routines is only 1% of the wall clock time. The ma-

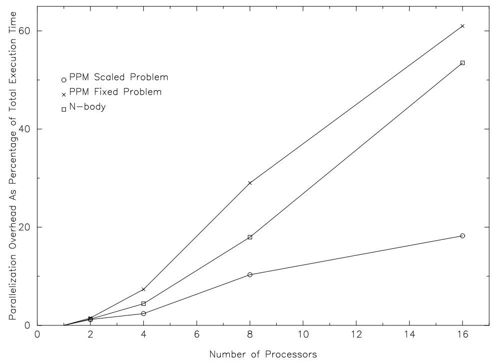

Figure 4: ESS Code Communication Overhead

jority of the overhead time appears to be taken up by synchronization of the processors at the end of each time step. This could probably be improved considerably by modifying the algorithm to allow for dynamic load balancing, but this experiment has not yet been attempted. It also remains to be determined if it can be improved by increasing the tile size.

An additional experiment was performed to determine whether the use of multiple networks will increase performance. For the PPM code (both cases) the effect was negligible, as expected, since the communication overhead never exceeds 1%. For the N-body code, the overall improvement obtained by using two networks was only a few percent. However, when the dynamic load balancing portion of the algorithm was tested, it was found to be network bandwidth constrained. Using two networks provided a 50% improvement for this portion of the code.

These scaling experiments with real problems showed surprisingly good performance, considering that PC microprocessors were being used. In the case of the PPM code, Beowulf provided competitive performance in what should have been its weakest area, namely in a floating point intensive application. It is likely that the performance of this code can be improved even more, as discussed above, allowing even larger problems to be executed. The N-body code also attained very creditable performance. When cost is considered, Beowulf delivered a far better price-performance ratio than general MPP's. It is true that Beowulf is limited in the scope of its scaling. Its configuration is only intended to exploit the first order of magnitude in parallelism. But within that regime, Beowulf appears to provide satisfactory scaling performance, at least for the classes of problems represented by the codes discussed in this section.

5 Discussion and conclusions

A key observation from the Beowulf project is that a 1 Gops peak performance can be implemented at costs well within the price range of high end workstations (the total cost of the Beowulf prototype was approximately $40,000). The use of low cost, moderate performance, moderate disk I/O bandwidth elements in a distribute configuration of 16 elements demonstrates that greater parallelism combined with mass market subsystems can provide a superior price-performance capability for certain types of operation. This also demonstrated the utility of employing the Linux operating system as a basis of research in distributed systems and as a platform for operational use. Linux proved robust, efficient, and ready to use. The availability of source code and no licensing constraints permitted modification of the kernel to support multi-channel Ethernet communications, transparent to the user application. It also provides an ideal vehicle for technology transfer of new system software tools developed in support of distributed computing.

The empirical results presented in this paper clearly demonstrate the viability as well as ease of employing multi-channel Ethernet communications. Although the message packet dispatching strategy was very simple, and the Ethernet interface/drivers were COTS elements, at saturation a dual system sustainable bandwidth exceeded a single channel configuration by 70% where the single unit sustained bandwidth achieved as at least 80% of absolute peak bandwidth of the hardware. Other experiments not discussed in this paper extended these results to three channels for a smaller configuration system showing good continued scaling [12]. This opens the way to the use of multiple 100 Mbps Ethernet technology that is just now emerging. Analysis shows that this technology combined in parallel will exceed the worst case communications requirements of Beowulf in terms of bandwidth although latency considerations have yet to be addressed. While overall behavior of the multi-channel approach proved very good, some anomalous behaviors were observed that are not fully understood but are probably artifacts of the message scheduling policy adopted. Future studies will have to be conducted to determine the precise causes.

One severe test of a parallel workstation such as Beowulf is when performing real world applications in parallel. This paper presented findings from extensive tests conducted on two space science applications. Under different but realistic conditions, both application codes demonstrated good scaling across 8 processors and useful additional performance achieved using all 16 processors. In fact, when compared to an Intel Paragon of comparable number of processing nodes, Beowulf did equally well. Also, when run on the CRI T3D again of a comparable number of processing elements, Beowulf delivered about half the performance seen on the T3D. In both cases, it is possible that future optimizations will change these performance ratios either way. But overall, the sustained performance compared favorably in light of the relative costs of the respective systems. However, communication overhead again consumed a considerable amount of computing time and therefore will be a target of future enhancement.

A second generation Beowulf parallel workstation is in production. While the overall configuration (and software systems) is unchanged, principal differences include replacing the Intel DX4 with the Intel Pentium, replacing the 10 Mbps Ethernet networks with 100 Mbps technology, and doubling the disk capacity. These improvements are expected to more than double the sustained floating point performance, and largely eliminate the inter-processor interconnects as a source of performance constraint. In particular, disk file transfer rates between separate pairs of disks should be significantly improved and parallel application scaling characteristics should be enhanced as well.

Future studies will resolve some of the unanswered conditions about network behavior and test more efficient methods of achieving global operations. A broader base of parallel applications is being developed including data intensive visualization applications with real-time requirements. An implementation of MPI is to be ported shortly and will be evaluated against the performance achieved with PVM. More advanced task scheduling and parallel disk management techniques developed at other research centers will be integrated into the current system software framework and tested against the needs of the workstation workload. An important outcome of this work is to provide a baseline of measurements by which other projects can determine the relative benefits of incorporating custom hardware mechanisms; because if you can't do better than Beowulf, you might as well use Beowulf.

References

[1] J. E. Barnes and P. Hut, *Nature, 324*, 1986, p. 446.

[2] M. Blumrich, K. Li, R. Alpert, C. Dubnicki, E. Felten, and J. Sandberg, "Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer," *Proceedings of the Twenty-First International Symposium on Computer Architecture (ISCA),* Chicago, April 1994, pp. 142-153.

[3] Cray Research, Inc., "CRAY T3D System Architecture Overview," Eagan, Minnesota.

[4] P. Colella and P. R. Woodward, "The Piecewise-Parabolic Method for Hydrodynamics," *Journal of Computational Physics, 54,* 1984, p. 174.

[5] B. Fryxell, E. Müller, and D. Arnett, "Hydrodynamics and Nuclear Burning," *Max-Planck-Institut für Astrophysik, Preprint 449.*

[6] Intel Corporation, "DX4 Processor Data Book," 1993.

[7] Intel Corporation, "Paragon User's Guide," Beaverton, Oregon 1993.

[8] Linux Documentation Project, Accessible on the Internet at World Wide Web URL *http://sunsite.unc.edu/mdw/linux.html*.

[9] K. Olson and J. Dorband, "An Implementation of a Tree Code on a SIMD Parallel Computer," *Astrophysical Journal Supplement Series,* September 1994.

[10] T. Sterling, D. Savarese, P. Merkey, J. Gardner, "An Initial Evaluation of the Convex SPP-1000 for Earth and Space Science Applications," *Proceedings of the First International Symposium on High Performance Computing Architecture,* January 1995.

[11] T. Sterling, D. Savarese, P. Merkey, K. Olson, "An Empirical Evaluation of the Convex SPP-1000 Hierarchical Shared Memory System," To appear in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniquess*, 1995.

[12] T. Sterling, D. Becker, D. Savarese, et al. "BEOWULF: A Parallel Workstation for Scientific Computation," To appear in *Proceedings of the International Conference on Parallel Processing*, 1995.

[13] V. Sunderam, "PVM: A Framework for Parallel Distributed Computing," *Concurrency: Practice and Experience,* December 1990, pp. 315-339.

[14] Thinking Machines Corporation, "Connection Machine CM-5 Technical Summary," Cambridge, MA, 1992.