BEOWULF: A PARALLEL WORKSTATION FOR SCIENTIFIC COMPUTATION

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

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

John E. Dorband
NASA Goddard Space Flight Center

Udaya A. Ranawake Charles V. Packer
Hughes STX Corp.

Abstract – *Network-of-Workstations technology is applied to the challenge of implementing very high performance workstations for Earth and space science applications. The Beowulf parallel workstation employs 16 PC-based processing modules integrated with multiple Ethernet networks. Large disk capacity and high disk to memory bandwidth is achieved through the use of a hard disk and controller for each processing module supporting up to 16 way concurrent accesses. The paper presents results from a series of experiments that measure the scaling characteristics of Beowulf in terms of communication bandwidth, file transfer rates, and processing performance. The evaluation includes a computational fluid dynamics code and an N-body gravitational simulation program. It is shown that the Beowulf architecture provides a new operating point in performance to cost for high performance workstations, especially for file transfers under favorable conditions.*

1 INTRODUCTION

Networks Of Workstations, or NOW [?] technology, is emerging as a powerful resource capable of replacing conventional supercomputers for certain classes of applications requiring high performance computers, and at substantially lower cost. Another, less frequently considered, domain is the realization of the high performance workstations themselves from ensembles of less powerful microprocessors. While workstations incorporating between 2 and 4 high performance microprocessors are commercially available, the use of larger numbers (up to 16 processors) of lower cost commodity subsystems within a single workstation remains largely unexplored. The potential benefits in performance to cost are derived through the exploitation of commodity components while the performance gains are achieved through the concurrent application of multiple processors. The MIT Alewife project [?] seeks to provide a fully cache coherent multiprocessor workstation through modifications of the SPARC processor. The Princeton SHRIMP project [?] employs standard low cost Intel Pentium microprocessors in a distributed shared memory context through the addition of a custom communication chip. While both projects make heavy use of available VLSI components, they require some special purpose elements, extending development time and incurring increased cost. An alternative approach, adopted by the Beowulf parallel workstation project, recognizes the particular requirements of workstation oriented computation workloads and avoids the use of any custom components, choosing instead to leverage the performance to cost benefits not only of mass market chips but of manufactured subsystems as well. The resulting system structure yields a new operating point in performance to cost of multiple-processor workstations.

2 BEOWULF ARCHITECTURE

The Beowulf parallel workstation project is driven by a set of requirements for high performance scientific workstations in the Earth and space sciences community and the opportunity of low cost computing made available through the PC related mass market of commodity subsystems. This opportunity is also facilitated by the availability of the Linux operating system [?], a robust Unix-like system environment with source code that is targeted for the x86 family of microprocessors including the Intel Pentium. Rather than a single fixed system of devices, Beowulf represents a family of systems that tracks the evolution of commodity hardware as well as new ports of Linux to additional microprocessor architectures.

The Beowulf parallel workstation is a single user multiple computer with direct access keyboard and monitors. Beowulf comprises:

- 16 motherboards with Intel x86 processors or equivalent
- 256 Mbytes of DRAM, 16 MByte per processor board
- 16 hard disk drives and controllers, one per processor board
- 2 Ethernets (10baseT or 10base2) and controllers, 2 per processor
- 2 high resolution monitors with video controllers and 1 keyboard

The Beowulf prototype employs 100 MHz Intel DX4 microprocessors and a 500 MByte disk drive per processor. The resulting 8 GBytes of secondary storage available locally to Beowulf applications can substantially reduce LAN traffic to remote file servers in certain important cases such as dataset browsing. 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 KByte primary cache (twice the size of other 486 primary caches) [?]. Each motherboard also contains a 256 KByte secondary cache. Two Ethernets running at peak bandwidths of 10 Mbits per second are used for internode communications, one a twisted pair 10baseT with hub and the other a multidrop 10Base2. Future Beowulf systems will employ more advanced versions of these component types but the basic configuration will remain the same. The Beowulf architecture has no custom components and is a fully COTS (Commodity Off The Shelf) configured system.

3 SCALING CHARACTERISTICS

3.1 Internode Communications

Communication between processors on Beowulf is achieved through standard Unix network protocols over Ethernet networks internal to Beowulf. Therefore the communication throughput of Beowulf is limited by the peformance characterisitics of the Ethernet and the system software managing message passing. However, Beowulf is capable of increasing communication bandwidth by routing packets over multiple Ethernets. This is made possible by a special device driver written by one of the authors which was facilitated by the free access to Linux kernel source code.

To evaluate the performance improvement derived from multiple networks, we measured the network throughput under a range of traffic demands using one, two, and three Ethernet networks. We assigned send/receive processes to pairs of processors which would exchange a fixed-sized token a particular number of times over the network. In this experiment, the load on the net was increased by exchanging larger tokens and also by increasing the number of tokens being exchanged. No processor was involved in the exchange of more than one token, i.e. each processor involved in the exchange of a token was assigned only one send/receive process. We used the BSD sockets interface and the User Datagram Protocol (UDP) to perform the token exchanges.

Figure **??** shows network throughput, measured in mega-bytes per second, as a function of the number of Ethernet channels available, token size, and number of tokens exchanged. At the time of the experiment, one of Beowulf's 16 processors was unavailable, allowing us to involve a maximum of only 7 pairs of processors in token exchanges. When performing the experiment using three channels, sufficient Ethernet cards were available to configure only 8 processors to use three channels. Hence a maximum of 4 tokens could be exchanged for that phase of the experiment.

It is evident that the small 64 byte tokens do not come anywhere near saturating the network for any number of channels. The 1024 byte tokens are able to saturate the 1 channel network with a throughput of about 1 MB/s, or 80% of the peak 1.25 MB/s possible on 10 Mbit/s Ethernet. Throughput for the 8192 byte tokens at 4 and 7 token exchanges is less than that for 1024 byte tokens

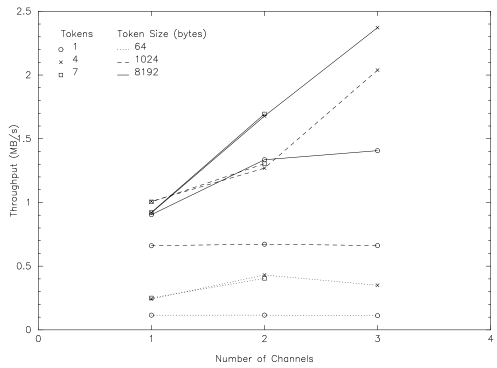

Figure 1: Beowulf Network Throughput

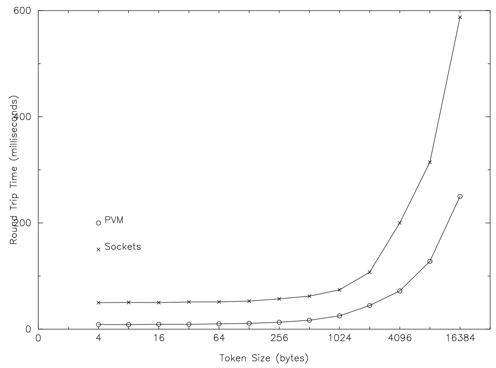

Figure 2: Beowulf Network Round Trip (1 channel, 16 processors)

because of additional network packet collisions. The minimum and maximum sizes of an Ethernet packet are 64 and 1536 bytes respectively [?]. Thus a 64 byte token and a 1024 byte token each require only one Ethernet packet for transmission. However, an 8192 byte token must be broken up into 6 Ethernet packets, increasing the likelihood of collisions on a 1 channel network. Figure **??** shows that multiple networks alleviate network contention, achieving throughputs of up to 1.7 MB/s (68% of peak) and 2.4 MB/s (64% of peak) respectively for 2 and 3 channel configurations.

System level applications like NFS are written making direct use of sockets, but most user level parallel programs use some higher level interface – usually PVM [?]. Figure **??** shows the overhead incurred by such high level message-passing interfaces. It shows the round trip time on one network channel across 16 processors for tokens of sizes ranging from 4 to 16384 bytes using PVM 3.3 versus BSD sockets and UDP. We define the round trip time as the time for a token to be sent from an initial processor, be received by a neighbor and be passed on to its neighbor, etc., visiting 15 intervening processors only once before finally returning to the initial processor. The PVM overhead is rather marked; the round trip time of a 256 byte token is 10 ms using sockets while the time us-

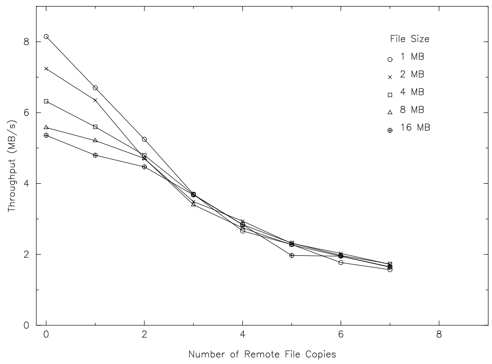

Figure 3: Beowulf File Transfers (2 channels, Total of 7 local and remote files)

ing PVM is 60 ms. The PVM experiment was run using the `PvmDataDefault` option for `pvm_initsend()` and the timing included the overhead of packing and unpacking a message.

3.2 Parallel Disk I/O

To ascertain the I/O performance of Beowulf, we measured the throughput of simultaneous file transfers across a mix of intraprocessor copies and interprocessor copies for a range of file sizes. No processor was involved in more than one file transfer, i.e. each processor involved in a file transfer was either performing a local disk file copy or was participating in a remote file transfer. Therefore there was no disk or processor contention caused by competing file transfers. File copies were performed using the Unix `read()` and `write()` system calls. In the case of remote transfers, BSD sockets and UDP were used to transmit a file between processors. At the time we performed this experiment, only 15 of Beowulf's processors were available to us. That meant we could only perform 7 simultaneous transfers instead of 8, because remote file transfers require the participation of 2 processors. Beowulf was configured to use 2 network channels for the experiment.

The empirical results of this key experiment are presented in Figure ??. The total number of file transfers is held constant while the ratio of local file transfers (those by a single processor to its local disk) versus remote file transfers (those between two disks and two processors over the network) is adjusted from 0.0 (all local) where there are 0 remote file copies to 1.0 (all remote) where there are 7 remote file copies. The data shows two dominant characteristics related to file transfer. Not surprisingly, file transfer throughput exceeds 8 MBytes per second (sustained) when all copies are local. It is seen that this value decreases for local copy only as the file size increases due to issues related to local buffering policies. As the number of remote file copies is increased to 1, overhead for the additional burden of moving over the network degrades overall throughput by about 15% in the worst case. Increasing number of remote copies beyond 1 adds an additional source of performance degradation, network contention. Two networks ameliorates the degradation at two remote copies but between two and three remote copies, the rate of throughput degradation is seen to accelerate. Ultimately where all file copies are remote, throughput is entirely constrained by the network bandwidth and is approximately equal to the maximum network throughput measured for two networks as discussed in the previous subsection.

4 BENCHMARK EXPERIMENTAL RESULTS

Performance scalability of a multiple processor workstation is best characterized through the use of complete real-world applications rather than synthetic test programs. To this end, two full applications from the Earth and space sciences community were selected to bracket the dimension of communication and load balancing demands.

A 2-dimensional compressible fluid dynamics code, called Prometheus [?], has been implemented on a number of high performance computers including vector, shared memory, distributed memory, and SIMD architectures. The code solves Euler's equations for gas dynamics on a logically rectangular mesh using the Piecewise Parabolic Method (PPM). The message passing version of this code previously used on the IBM SP-1 and Intel Paragon was easily ported the the Beowulf parallel workstation and its PVM message passing environment. Parallelization was accomplished using a domain decomposition technique for which the computational grid was divided into 128x128 tiles. Communication between neighboring tiles is necessary only twice per time step. Because of the large number of floating-point operations required to update each grid point communication costs are relatively small.

A tree code for performing gravitational N-body simulations has been developed to reduce a classically $O(n^2)$ computation to $O(n \log n)$ and has been applied to shared memory [?], distributed memory, and SIMD parallel architectures [?]. The code is being used to study the structure of gravitating, star forming, interstellar clouds as well as to model the fragmentation of comet Shoemaker-Levy 9 in its close encounter with Jupiter. A range of number of particles were used from 32K to 256K. It is estimated that Beowulf can support a 1 million particle simulation for in-core computation and much larger if appropriate disk accessing can be coordinated.

Scaling characteristics of these two codes were evaluated on the Beowulf parallel workstation. The results are shown in Figure ??. The CFD code was executed on up to 16 processing modules and the tree code was performed on up to 8 processing modules. The CFD application showed good scaling characteristics with total degradation at 16 processors approximately 16% with respect to ideal. Single processor performance for this code is 4.5 MFLOPS. The full Beowulf delivered a sustained performance of 60 MFLOPS. This compared favorably with the Paragon of equivalent size as well as the TMC CM-5 (without vector chips). The CRI T3D performed less than 2.5 times better than Beowulf for the same number of processors. An additional experiment was performed to demonstrate the impact of multiple networks versus a single network in Beowulf. The dual network showed

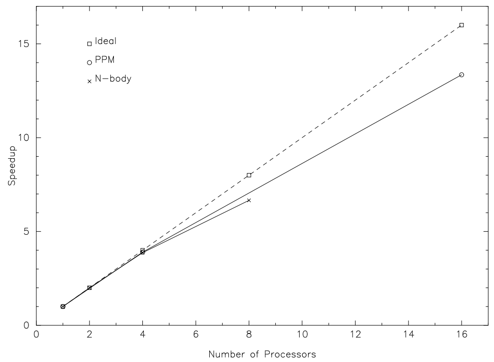

Figure 4: ESS Code Scaling

essentially no performance advantage over the single network case, indicating that communication bandwidth and contention is not an issue for this application.

The scaling characteristics for the tree code is somewhat poorer as should be expected. Much more communication per computation is involved with communication requirements being relatively global. Because the data structure is time varying, one part of the program does load balancing periodically (once every ten time steps) through a sorting strategy. Unfortunately, this problem has not been run on more than 8 processors. Performance degradation at 8 processors was observed to be 19% with respect to the ideal. Single processor performance was 1.9 MFLOPS with a total of 12.4 MFLOPS using 8 processing modules. The overall impact of multiple networks versus a single network was found to be only a few percent. But when the dynamic load balancing portion of the algorithm was tested, it was found to be network bandwidth constrained. Two networks provided a 50% improvement in performance for the sort algorithm versus a single network run.

5 DISCUSSION AND CONCLUSIONS

The Beowulf research project has been initiated to explore the opportunity of exploiting Network-of-Workstation concepts to provide high performance single user workstation performance at exceptional cost. The operating point targeted by the Beowulf approach is intended for scientific applications and users requiring repeated use of large data sets and large applications with easily delineated coarse grained parallelism.

Interprocessor communication proved to be the most interesting aspect of the Beowulf operation. Enhancements to the Linux kernel enabling multiple communications channels to be employed simultaneously showed excellent scaling factors. This new functionality alone will impact how Linux based PC's will be used in the future. But it was equally clear that the network, even in its dual configuration, is inadequate under certain loads. There was at least one instance where full application behavior was perturbed by network capacity. More importantly, parallel file transfers were seen to be limited by the network. It is clear from these results that higher bandwidth networks are required. Fortunately, 100 Mbps Ethernet-like networks are now coming on the commodity market. The Beowulf project is beginning evaluation of this new technology and it is anticipated that dual 100baseTX or 100VG type networks will be incorporated in the new Beowulf demonstration unit being assembled in 1995.

Future work will focus primarily on advanced software technology that will make better use of the parallel computing resources. These include load balancing, parallel file distribution, global synchronization, parallel debugging and optimization, and distributed shared memory environments. But the lesson of this initial work is that a relatively simple capability as that offered by the Beowulf prototype can be of immediate value to real users in the arena of scientific computation

REFERENCES

[1] A. Agarwal, D. Chaiken, K. Johnson, et al. "The MIT Alewife Machine: A Large-Scale Distributed-Memory Multiprocessor," *M. Dubois and S.S. Thakkar, editors, Scalable Shared Memory Multiprocessors,* Kluwer Academic Publishers, 1992, pp. 239-261.

[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] D. Boggs, J. Mogul, and C. Kent, "Measured Capacity of an Ethernet: Myths and Reality," *WRL Research Report 88/4,* Western Research Laboratory, September 1988.

[4] K. Castagnera, D. Cheng, R. Fatoohi, et al. "Clustered Workstations and their Potential Role as High Speed Compute Processors," *NAS Computational Services Technical Report RNS-94-003,* NAS Systems Division, NASA Ames Research Center, April 1994.

[5] B. Fryxell and R. Taam, "Numerical Simulations of Non-Axisymmetric Accretion Flow," *Astrophysical Journal, 335,* 1988, pp. 862-880.

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

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

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

[9] 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.

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