Scalability and Performance of CTH on the Computational Plant

Ron Brightwell, H. Eliot Fang, and Lee Ward

Abstract—
In this paper, we present performance results from CTH, a family of codes developed at Sandia National Laboratories for modeling complex multi-dimensional, multi-material problems characterized by large deformations and/or strong shocks. We compare results from the 400-node Computational Plant (Cplant™) Linux cluster to performance obtained on traditional supercomputing platforms: Intel TeraFLOPS, IBM SP-2, and a cluster of Digital Equipment Corporation 8400 servers. We provide an overview of the Cplant™ platform and system software environment, and discuss the characteristics of the machine that influence performance and scalability of the application.

I. Introduction

Parallel computing platforms composed of commodity personal computers (PC's) have become an attractive alternative to traditional supercomputing platforms. Recently, procurements for small- and medium-sized parallel computers have been bid on, and in some cases have been awarded to, commodity-based PC clusters. PC clusters with 32 and 64 nodes have become commonplace additions to many scientific computing sites. For many applications, these small-scale PC clusters are able to compete and even surpass the performance of large traditional commodity parallel machines.

In spite of this success, few results on large-scale clusters with hundreds of nodes have been published. Traditional large parallel computing platforms have benefited from many years of research, development, and experience dedicated to improving their scalability and performance. PC clusters are only now starting to receive this kind of attention. In order for clusters of PC's to compete with traditional large-scale platforms, this type of knowledge and experience will be crucial as larger clusters are built.

The goal of the Computational Plant (Cplant™) [1] project at Sandia National Laboratories is to provide a large-scale, massively parallel computing resource composed of commodity-based PC's that not only meets the level of compute performance required by Sandia's key applications, but that also meets the levels of usability and reliability of past traditional large-scale parallel machines. Cplant™ is a continuation of our research into system software for massively parallel computing on distributed-memory message-passing machines. We are transitioning our scalable system software architecture that was developed for large-scale massively parallel processing machines to clusters of PC's. In this paper, we compare the performance and scalability of one of Sandia's key applications on Cplant™ to other traditional parallel computing platforms.

The following section describes the different platforms utilized in this study. In section III, we describe the application suite used for comparison. Section IV presents a comparison of the performance of the application suite on the platforms and an analysis of the factors that influenced performance and scalability. We conclude in Section V with a summary of results, and describe our plans for future work in Section VI.

R. Brightwell, and L. Ward are with the Computational Sciences, Computer Sciences, and Mathematics Center, Sandia National Laboratories, PO Box 5800 MS 1110, Albuquerque, NM, 87185-1110,(505)845-7432,(505)845-7442 FAX, {rbbrigh,lward}@sandia.gov.

H. E. Fang is with the Materials and Process Science Center, Sandia National Laboratories, PO Box 5800, Albuquerque, NM, 87185-1411, hefang@sandia.gov.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

II. Platforms

The following section describes the hardware and software components of the four different platforms involved in this study.

A. Computational Plant

The software and hardware architecture of Cplant™ is modeled after the Intel TeraFLOPS (TFLOPS) machine, described below. The compute nodes of TFLOPS run a lightweight kernel designed and developed by Sandia and the University of New Mexico [2]. Our experience with the poor performance and scalability of full-featured UNIX kernels on MPP's, such as OSF on the Intel Paragon, motivated much of the research that led to this lightweight kernel. Fundamental to the Cplant™ project is the ability to acquire the latest commodity hardware that occupies the "sweet spot" of the price/performance curve, and make it available to users. The time required to port and maintain a lightweight kernel on successive generations of hardware, BIOS's, and PCI chipsets makes this impossible. With Linux, we hope to leverage its portability and open source model. Linux allows us to have an operating system that runs well on the very latest commodity hardware, and the source code availability allows us to manipulate the standard kernel. We hope to be able to create a Linux-based kernel that exhibits the characteristics of past lightweight kernels and overcomes the scalability and performance limitations of previous full-featured UNIX operating systems.

The Cplant™ runtime environment is based on a message passing layer called Portals, which is an important component of the TFLOPS lightweight kernel. All of the system software in the runtime environment uses Portals for message passing. See [1] for a description of Portals and the components of the scalable runtime environment. By implementing Portals in Linux, we were able to re-use much of the code that had been developed for the Paragon and TFLOPS. The MPI implementation for Cplant™ uses Portals as the underlying transport layer.

There are several methods of doing disk I/O available on Cplant™. A parallel application is launched from the service partition into the compute partition using a command called *yod*. The yod process works with an allocator process and a compute node manager processes to distribute the user's executable and environment to all of the nodes in the parallel job. Once the application has started on all of the nodes, the yod process acts as a proxy for all standard I/O functions, including file I/O. The standard I/O library is replaced by a library that turns I/O calls into a remote procedure call to the yod process. This strategy allows for all printouts from the compute node processes to be displayed by the yod process, and makes any file system visible to the yod process visible to the compute node processes. However, this many-to-one strategy also creates a bottleneck that significantly reduces disk I/O performance.

Cplant™ also supports parallel independent disk I/O operations, through a daemon process called an *fyod*, or file yod. Each node in the Cplant™ disk I/O partition runs an fyod server that writes to a local RAID. A compute node process that opens a file in the fyod namespace will communicate directly with a specific fyod for disk I/O operations. fyod processes essentially perform the same disk I/O operations as a yod process, only they are persistent daemons and may service more than one application simultaneously. Compute node processes are assigned fyod servers in a round-robin fashion. This mechanism allows a many-to-many mapping of compute node processes to disk I/O servers. However, each process must manipulate a single independent file. This method for parallel I/O was used on Sandia's large Intel Paragon and is also used with Intel's Parallel File System on TFLOPS.

Cplant™ also supports a global parallel file system for disk I/O operations through an ENFS file system. As with fyod, a small set of service nodes are usurped by a special daemon. The collection of I/O nodes are symmetric in nature, and each compute node binds to an ENFS I/O node at boot. The binding is done in a round-robin fashion in to attempt to spread the load evenly. Each client may use only one server, but many cooperating clients will, typically, use all available I/O nodes when attempting parallel operations. The ENFS-based file system does not maintain coherency, nor does it support locking. Instead, the protocol has been extended inside the system to support application control of data caches. Externally, only normal NFS calls are used. The performance gain comes from the ability to have a very large number of transactions simultaneously in flight against the external server. Each compute node may open the same file and coordinate its own access, or separate files may be utilized, as CTH does, or some combination of the two. Measured performance remains the same, as available bandwidth is a global resource in ENFS.

In the Fall of 1998, Digital Equipment Corporation (DEC), currently Compaq Computer Corporation, installed a 400-node cluster at Sandia. Each compute node in this cluster is composed of a 500 MHz Alpha 21164 processor, a 2 MB level-3 cache, and 192 MB of main memory. In addition, each compute node has a 32-bit 33 MHz Myrinet [3] LANai 4 network interface card. These machines are connected via a 16-port Myrinet SAN/LAN switches in a modified hypercube topology.

This machine has a theoretical compute performance peak of 400 GigaFLOPS (GFLOPS). It achieved 125.2 GFLOPS on the LINPACK [4] benchmark running on 350 nodes, which would place it at number 71 on the November 1999 list of the Top 500 fastest computers in the world [5] had the results been submitted.

B. TFLOPS

TFLOPS [6] is the Department of Energy's (DOE) Accelerated Strategic Computing Initiative (ASCI) Option Red machine. Installed at Sandia in the Spring of 1997, it is the culmination of more than ten years of research and development in massively parallel distributed-memory computing by both Intel and Sandia.

TFLOPS is composed of more than nine thousand 300 MHz Pentium II Xeon processors connected by a network capable of delivering 400 MB/s unidirectional communication bandwidth. Each compute node contains two processors and 256 MB of main memory. The the nodes are arranged in a 38x32x2 mesh topology providing 51.2 GB/s of bisection bandwidth. Each compute node has a network interface chip that resides on the memory bus, allowing for low-latency access to all of physical memory. Parallel I/O on TFLOPS is provided via Intel's Parallel File System.

The theoretical peak compute performance of this machine is 3.2 TFLOPS. It has achieved 2.37 TFLOPS on the LINPACK benchmark, and has held the number one spot on the Top 500 list since June of 1997.

C. IBM SP-2

The IBM SP-2 used in this comparison is part of the DOE's ASCI Blue/Pacific machine installed at Lawrence Livermore National Laboratory. The particular machine we used is composed of 320 4-processor compute nodes, each with a 332 MHz PowerPC 60-4e, for a total of 1280 CPU's. Each node has 1.5 GB of main memory, and is connected to a network capable of delivering 150 MB/s to other nodes. Parallel I/O on this machine is provided via IBM's General Parallel File System.

The entire ASCI Blue/Pacific machine has a theoretical compute performance of 3.8 TFLOPS. It has achieved 2.1 TFLOPS on the LINPACK benchmark, and is in second place on the November 1999 Top 500 list.

D. DEC 8400 Cluster

The final machine used in this study is a cluster of 7 DEC model 8400 servers. Each server contains 12 622 MHz Alpha 21164 processors, 4 GB of main memory, and 2 GB of local disk. The servers are networked using a 32-bit 33 MHz Memory Channel II interface connected to 4 Memory Channel II hubs. In additional to local disk, access to other file systems is provided via NFS. For runs of CTH on this platform, the local disk was used. The theoretical peak compute performance of this machine is 104.5 GFLOPS, and it has achieved 30.9 GFLOPS on the LINPACK benchmark.

III. CTH

The CTH [7] family of codes developed at Sandia models complex multi-dimensional, multi-material problems characterized by large deformations and/or strong shocks. It uses a two-step, second-order accurate finite-difference Eulerian solution algorithm to solve the mass, momentum, and energy conservation equations. CTH has material models for equations of state, strength, fracture, porosity, and high explosives. The classes of problems that can be analyzed with CTH include impact, penetration and perforation, shock compression, and high explosive initiation and detonation phenomena. The production CTH software family runs on a variety of computing platforms, from low-end PC's to high-end massively parallel supercomputers. It is used extensively within both the Department of Defense and the Department of Energy laboratory complexes for studying armor/anti-armor interactions, warhead design, high explosive initiation physics, and weapons safety issues. CTH is a sophisticated analysis tool that can supplement testing in some situations and complement it in all situations.

The message passing version of CTH was developed based on the single program multiple data (SPMD) computing technique. The same executable image is running on each computational node, but each executable is working with a data set unique to the local node. It uses domain decomposition where the entire problem is divided up into subdomains that reside on the individual computational nodes. Communication between nodes is handled by the employment of *ghost cells* and explicit messages that are passed between nodes. The ghost cells allow the finite-difference equations to be independent of edges and corners. For an external boundary, the ghost cell data is based on the selected boundary condition approximation. For an internal boundary, the ghost cells contain real data acquired via messages from neighboring nodes.

Two codes from the CTH family were used for this study, CTHGEN and CTH. These two codes are both required for all simulations. A setup run of CTHGEN must precede the CTH calculation. CTHGEN is used to build a time-zero representation of the problem specifications, map the problem space onto each node, assign material properties, and create the initial data set, called a restart file, on a persistent storage device for CTH to start with later. Each node involved in a CTHGEN run should have a restart file written when the run completes. In a CTHGEN run, data broadcasting is the main type of message passing and there is very little computing involved.

The work performed by CTH includes reading the initial restart files from CTHGEN, simulating the shock wave physics, and writing calculated results to restart, history and/or visualization files. During a CTH run, many nearest-neighbor communications and a few global minimization routines are called at each time step, or computational cycle. However, most of the time taken by CTH is spent on solving finite difference algorithms of partial differential equations. Due to the large amount of data generated during the calculation that must be stored in restart and/or visualization files, efficient disk I/O is critical to good performance.

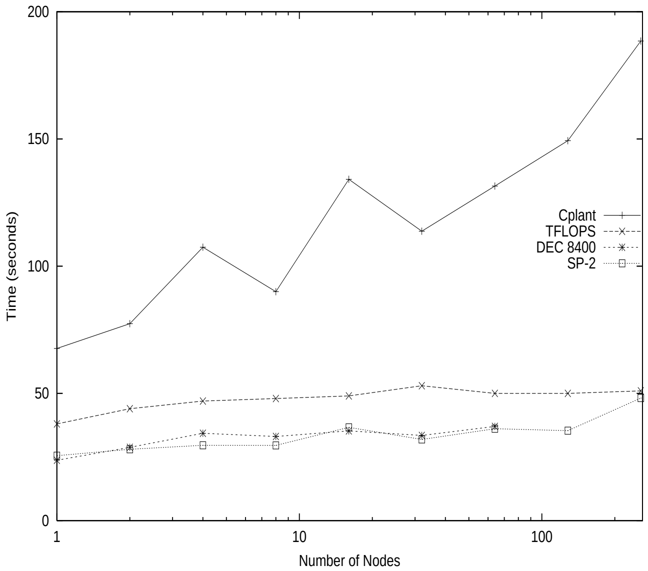

Fig. 1. CTHGEN Time

To assess the I/O performance and CTH scalability on Cplant™, a three-dimensional simulated problem was designed. This problem has moving materials filling the entire computational mesh throughout the simulation time, so load balancing problems are not evident. A series of CTH calculations with different problem sizes was performed on different number of nodes, ranging from 1 to 256 nodes. For each calculation, the problem size was scaled such that the total number of computational cells allocated on each node and the resolution of the problem always remained the same, independent of the number of nodes. Each calculation was repeated a few times so that the average performance index was as objective as possible.

IV. Results

Figure 1 shows the computation time, including message-passing, taken for a run of CTHGEN on the different platforms. This measurement does not include time taken to perform the writing of restart files. CTHGEN on 256 Cplant™ nodes takes 188.5 seconds, while the two ASCI platforms take approximately 50 seconds. Since CTHGEN is mainly broadcasting data, the slower performance of Cplant™ can be attributed to its slower network. TFLOPS and the SP-2 both have networks that offer an order of magnitude greater bandwidth than the Myrinet network in

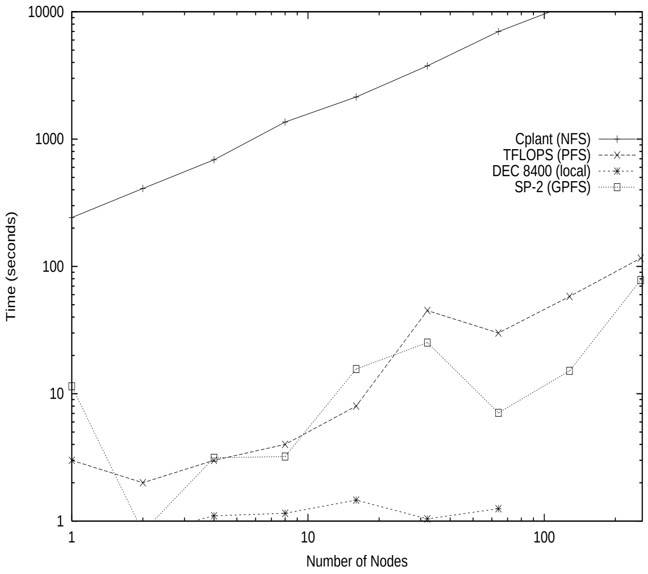

Fig. 2. CTHGEN Restart File Write Time

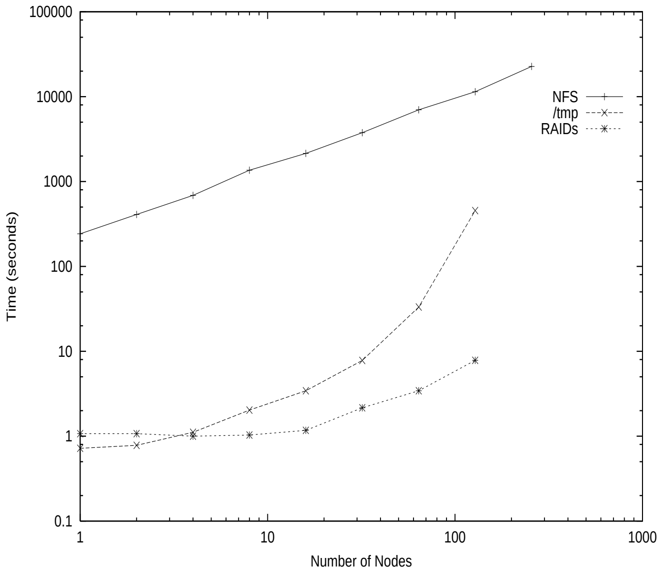

Fig. 3. CTHGEN Restart File Write Times for Cplant™

Cplant™. The 32-bit 33 MHz PCI bus on Cplant™nodes limits the achievable bandwidth to approximately 70 MB/s, and MPI communication benchmarks have typically measured bandwidth at 62 MB/s.

Figure 2 shows the time taken for CTHGEN to write the restart files on each platform. Each restart files is approximately 2 MB. The time taken by Cplant™to write the restart files on only 64 nodes is nearly two hours, compared to only 1.25 seconds on the DEC 8400 cluster. This was a clear indication of a significant problem with writing to the NFS-mounted file system on Cplant™. After further investigation, we discovered a problem with the NFS protocol between Cplant™ service nodes running Linux and the NFS fileserver, a DEC AlphaServer 4100 running Tru64 UNIX. Compaq employs delayed writes in order to bundle many NFS transactions per physical I/O request. Unfortunately, the interaction between the Compaq implementation and current Linux (2.2.x) kernels introduces huge latencies because the I/O request is not posted until a timer expires. In the presence of read-ahead and write-behind clients, the I/O request would be fully utilized and the request queued prior to timer expiration. Unfortunately with Linux, which supports only version 2 NFS (with small, 8KB payloads), multiple transactions are issued only in, what for Cplant™amounts to a, special case. Even then, the transaction payload does not appear large enough to avoid waiting for the timer to expire. The latency introduced per request slows observed I/O rates by one to two orders of magnitude, depending on a tuning parameter in Tru64 UNIX.

Figure 3 shows the time to write the restart files for the other methods of performing disk I/O on Cplant™. Writing to /tmp on the service node avoids using NFS, but all I/O operations are serialized through a single yod process. This method results in slightly better times. For 128 nodes, writing to NFS took a little more than 3 hours, while writing to /tmp took only 7.5 minutes. Using the parallel independent I/O capabilities of the fyod servers reduced the time needed to write the restart files on 128 nodes to only 7.83 seconds. Clearly this is the preferred method of doing disk I/O on Cplant™.

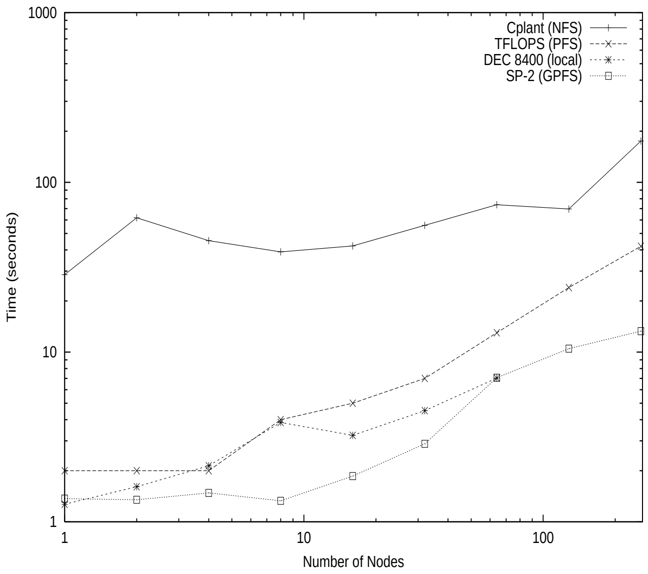

Fig. 4. CTHGEN Restart File Read Time

Figure 4 shows the time needed by CTH to read the restart files over NFS on the various platforms. Reading via NFS on Cplant™ does not exhibit the protocol problem that writing does, but it is still significantly more than the other platforms. At 256 nodes, the SP-2 takes 13.2 seconds, TFLOPS takes 42 seconds, and Cplant™ takes nearly 3 minutes. While we do not present data for the other methods of disk I/O on Cplant™ for reading the restart files in CTH, the performance improvement is similar to that shown in writing the restart files.

Figure 5 shows the compute performance of CTH on the

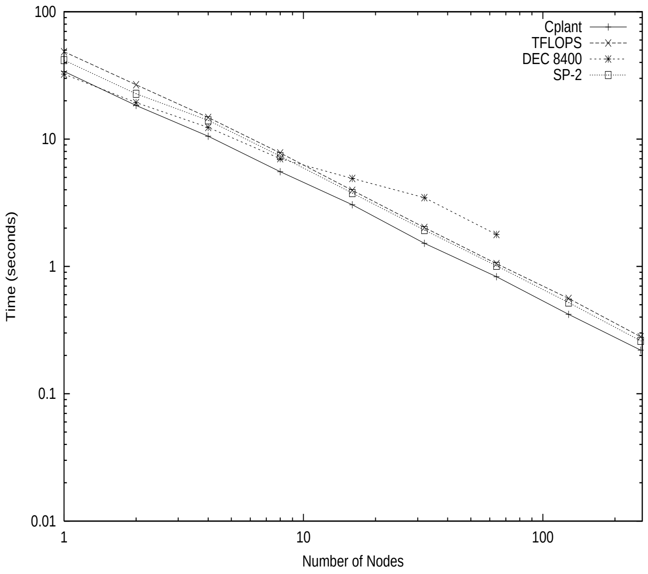

Fig. 5. CTH Grind Times

different platforms. The results are presented in *grind time*, or the amount of CPU time necessary to complete all calculations on a single cell for a single time step. The calculations of grind time do not include CTH startup overhead. Cplant™ outperforms all of the other platforms. This level of performance can be attributed mostly to the superior floating-point performance of the Alpha processor on Cplant™ nodes.

While the compute performance is greater, CTH does not scale as well on Cplant™ as the other platforms. The scaling efficiency on 256 nodes is 88% for the SP-2, 87% for TFLOPS, and 79% for Cplant™. However, given the architecture, network performance, and cost of the various platforms, the ability of Cplant™ to compete with, and in some cases outperform, the other platforms is encouraging.

V. Summary

Through exercising CTH calculations on Cplant™ and comparing the results with other large platforms, some I/O features and performance bottlenecks of the cluster were revealed. For example, writing data from compute nodes to service nodes through NFS was found to be much, much slower than reading, due to a protocol problem. The compute performance of Cplant™ is greater than the other platforms, even out to 256 processors. We expect that improvements in processor performance and network bandwidth of the latest commodity components will improve the scaling performance of CTH and allow for greater utilization on more than 256 processors. We also expect that improvements in the I/O architecture of Cplant™ will greatly improve I/O performance and significantly reduce the turnaround time for jobs that manipulate large numbers of files.

VI. Future Work

Although we described the ENFS file system of Cplant™, we have yet to measure the performance of CTH using this capability. Early production tests with the ENFS and the CTH I/O sub-system showed 38 MB/s delivered on an older configuration of I/O hardware. System benchmarks at the time delivered 38 MB/s as well. Recent testing with a new hardware configuration of the I/O nodes has shown 113MB/s, when writing. We hope to be able to gather CTH results for this new I/O configuration.

In October of 1999, Compaq Computer Corporation installed a 592-node Cplant™ cluster at Sandia. Each compute node in this new cluster is composed of a 500 MHz Alpha 21264 processor, 2 MB of level-3 cache, 256 MB of main memory, and a 64-bit 33 MHz Myrinet LANai 7 network interface card. This machine achieved 247.6 GFLOPS on the LINPACK benchmark, which would place it at number 40 on the November 1999 list of the Top 500 fastest computers in the world had the results been submitted. Previous results that were submitted placed it at number 44 in the list.

Compared to the 400-node Alpha 21164 cluster, this new machine has increased compute performance, increased peak memory bandwidth, increased network link bandwidth, a different topology, and more processors. Our initial experience with these machines has shown a significant increase in computation performance over the Cplant™cluster discussed here. We expect to gather further results for the CTH on this platform when the machine is deployed into the Sandia production computing environment in the Spring of 2000.

VII. Acknowledgments

The authors would like to acknowledge the contributions of the Cplant™ system software development and operations teams. Gratitude is also expressed to Sue Goudy who collected the data for the various platforms.

References

[1] R. B. Brightwell, L. A. Fisk, D. S. Greenberg, T. B. Hudson, M. J. Levenhagen, A. B. Maccabe, and R. E. Riesen, "Massively Parallel Computing Using Commodity Components," *Parallel Computing*, vol. 26, no. 2-3, pp. 243–266, February 2000.
[2] P. L. Shuler, C. Jong, R. E. Riesen, D. van Dresser, A. B. Maccabe, L. A. Fisk, and T. M. Stallcup, "The Puma operating system for massively parallel computers," in *Proceedings of the 1995 Intel Supercomputer User's Group Conference*. Intel Supercomputer User's Group, 1995.
[3] Nannet Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, Charles L. Seitz, Jakov N. Seizovic, and Wen Su, "Myrinet- a gigabit-per-second local-area network," *IEEE Micro*, vol. 15, no. 1, pp. 29–36, February 1995.
[4] J. J. Dongarra, "Performance of Various Computers Using Standard Linear Equations Software," Tech. Rep. CS-89-85, Department of Computer Science, University of Tennessee, 1994.
[5] Netlib, *Top 500 Supercomputers*, 1998, http://www.top500.org.
[6] Sandia National Laboratories, *ASCI Red*, 1996, http://www.sandia.gov/ASCI/TFLOP/ Home_Page.html.
[7] E.S. Hertel, Jr, R.L. Bell, M.G. Elrick, A.V. Farnsworth, G.I. Kerley, J.M. McGlaun, S.V. Petney, S.A. Silling, P.A. Taylor, and L. Yarrington, "CTH: A Software Family for Multi-Dimensional Shock Physics Analysis," in *Proceedings of the 19th International Symposium on Shock Waves, held at Marseille, France*, July 1993, pp. 377–382.