

# Scalability and Performance of a Large Linux Cluster <sup>1</sup>

Ron Brightwell and Steve Plimpton

*Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM, 87185-1110,  
(505)845-7397, FAX (505)845-7442*

E-mail: bright@cs.sandia.gov,sjplimp@cs.sandia.gov

---

In this paper we present performance results from several parallel benchmarks and applications on a 400-node Linux cluster at Sandia National Laboratories. We compare the results on the Linux cluster to performance obtained on a traditional distributed-memory massively parallel processing machine, the Intel TeraFLOPS. We discuss the characteristics of these machines that influence the performance results and identify the key components of the system software that we feel are important to allow for scalability of commodity-based PC clusters to hundreds and possibly thousands of processors.

---

*Key Words:* massively parallel, workstation cluster, message passing

## 1. INTRODUCTION

The commodity-based personal computer (PC) cluster machine has become an attractive alternative to traditional supercomputing platforms. The performance of clusters of PC's is being compared to distributed-memory message-passing supercomputers, such as the Intel Paragon, SGI/Cray T3E, and IBM SP-2, as well as shared-memory supercomputers, such as the SGI/Cray Origin 2000, SUN E10000, and DEC 8400. For many applications, small-scale PC clusters are able to compete and even surpass the performance of these traditional supercomputing platforms. For example, in September of 1999, the Forecast Systems Laboratory division of the National Oceanographic and Atmospheric Administration awarded a five-year multi-million dollar contract for a machine with a peak performance of four trillion floating point operations per second (FLOPS) to a vendor that supplies Linux-based PC clusters. The vendor, High Performance Technologies, Inc., outbid several traditional supercomputing vendors and platforms by offering a PC cluster that provided comparable performance on a set of FSL-supplied benchmarks for significantly less cost.

<sup>1</sup>This work was supported by the United States Department of Energy under Contract DE-AC04-94AL85000.

The ability of a small-scale dedicated cluster of standard desktop PC's to compete with and sometimes outperform small-scale supercomputers has been demonstrated [29, 1]. However, very few performance results have been published on clusters with hundreds of processors. Thus, it is unknown whether PC clusters will be able to scale up to a comparable level of compute performance on several hundred or thousands of processors to compete with large-scale massively parallel processing (MPP) machines and clusters of large shared-memory processing (SMP) machines. Traditional massively parallel computing platforms have benefited from many years of research, development, and experience dedicated to improving their scalability and performance. The Computational Plant (Cplant) project at Sandia National Laboratories 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 from large-scale MPP machines to clusters of PC's. In this paper, we hope to show that large-scale PC clusters can compete with large-scale MPP machines, provided that proper attention is given to scalability in both hardware and system software.

The following section describes the hardware and software components of a traditional MPP machine, the Intel TeraFLOPS. In Section 3, we describe the hardware and software components of the 400-node Cplant cluster. Section 4 presents a comparison of the performance of several benchmarks on the two platforms, and Section 5 continues with a performance comparison of several applications. We conclude in Section 6 with a summary of relevant results and outline our plans for future work in Section 7.

## 2. SANDIA/INTEL TERAFLUPS MACHINE

The Sandia/Intel TeraFLOPS machine (TFLOPS) [23] is the Department of Energy's 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. The following describes the hardware and software components of TFLOPS.

### 2.1. Hardware

TFLOPS [23] is made up 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, and 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 (NIC) that resides on the memory bus, allowing for low-latency access to all of physical memory.

The theoretical peak compute performance of this machine is 3.2 TFLOPS. It achieved 2.37 TFLOPS on the LINPACK [9] benchmark, placing it at number one on the November 1999 list of the Top 500 [18] fastest computers in the world.

### 2.2. Software

The compute nodes of TFLOPS run a variant of a lightweight kernel, called Puma [26], that was designed and developed by Sandia and the University of New

Mexico. The design of this kernel evolved from earlier experiences in providing a high-performance operating system optimized for distributed-memory message-passing MPP's [15]. The Puma operating system was originally developed on a 1024-processor nCUBE-2 and later ported to an 1800-node Intel Paragon. Intel and Sandia worked together to port Puma to the x86 processor architecture for TFLOPS, at which point it was productized and renamed Cougar by Intel. Cougar consumes approximately one percent of the total main memory on a node.

A key component of the design of Puma is a high-performance data movement layer called Portals [25]. Portals are data structures in an application's address space that determine how the kernel should respond to message-passing events. Portals allow messages to be delivered directly to the application without any intervention by the application process. In particular, the application process need not be the currently running process or perform any message selection operations to process incoming messages. Commodity networking technology has recently begun to realize the benefits of such strategies, with emerging technologies such as the Virtual Interface Architecture [8], Scheduled Transfer [28], and the newly formed InfiniBand Trade Association. In addition to providing high-performance message passing, the runtime components of TFLOPS work together to provide a scalable job launch capability that can allocate processors and start processes on thousands of nodes in a matter of several seconds.

### 3. COMPUTATIONAL PLANT

The Computational Plant (Cplant) [24] project at Sandia is a commodity hardware based Linux cluster that combines the benefits of commodity cluster computing with our experience and expertise in designing, developing, using, and maintaining massively parallel distributed-memory machines. The goal is to provide a large-scale computing resource 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 machines such as TFLOPS. The following sections describe the initial approach, and the hardware and software components of Cplant.

#### 3.1. Approach

The focus of Cplant is on scalability – in every aspect of the machine. Scalability in terms of application performance is critical, but scalability in other areas is also critical. For example, just as TFLOPS does, application launch should happen in seconds on several thousand nodes. The machine should be able to distribute system software and boot in tens of minutes. The machine should remain stable and performance of the runtime environment should not degrade as the number of interactive users increases.

Our approach to Cplant was to leverage as much as possible from the design of TFLOPS. We designed a support and diagnostic infrastructure for Cplant analogous to the Reliability, Availability, and Supportability (RAS) system that Intel developed. We followed the partition model (service, compute, I/O, etc.) of resource provision [11] that Intel developed. We decided to leverage the code development that we had done for Puma to create a scalable runtime environment.

Our experience with the poor performance and scalability of full-featured UNIX kernels on MPP's, such as OSF on the Paragon, motivated much of the research that

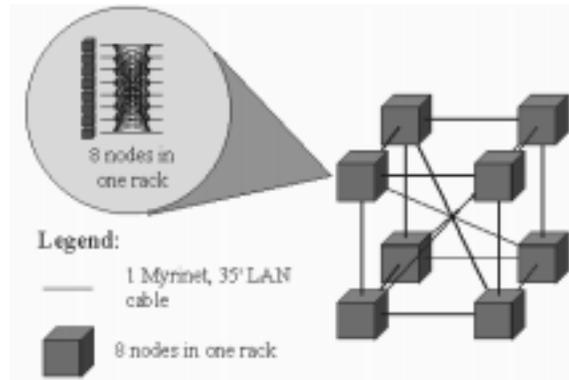


FIG. 1a. Cplant 64-node topology.

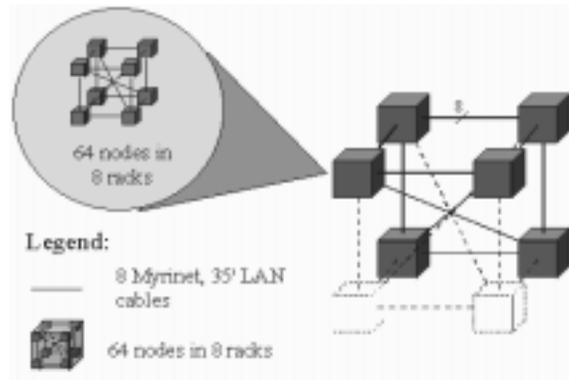


FIG. 1b. Cplant 384-node topology.

led to the lightweight kernel design and development. 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.

### 3.2. Hardware

In the Fall of 1998, Digital Equipment Corporation (now Compaq Computer Corporation) installed a 400-node cluster at Sandia National Laboratories. Each compute node in this cluster is composed of a 500 MHz Alpha 21164, 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 illustrated in Figures 1a and 1b. Figure 1a illustrates

the 64-node topology that is used to construct the 384-node topology illustrated in Figure 1b. We extend this topology to 400 nodes by partially filling one of the empty cubes.

The theoretical peak performance of the 32-bit 33 MHz PCI bus interface is the limiting factor in Myrinet bandwidth capability. The PCI bus limits the possible performance to 132 MB/s, but in practice most PCI chipset implementations can only deliver about 100 MB/s.

This machine has a theoretical compute performance peak of 400 gigaFLOPS (GFLOPS). It achieved 125.2 GFLOPS on the LINPACK [9] benchmark running on 350 nodes, which would place it at number 71 on the November 1999 list of the Top 500 [18] fastest computers in the world had the results been submitted.

### 3.3. Software

The Cplant runtime environment is based on the Portals data movement layer from the Puma operating system. All of the system software in the runtime environment uses Portals for message passing. See Brightwell [4] for a description of 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. For example, the high-performance implementation [6] of the Message Passing Interface (MPI) standard [16] that had been validated by Intel as a product for TFLOPS ported to Cplant with virtually no changes.

Portals in Linux are currently implemented via two kernel modules that work with a Sandia-developed Myrinet Control Program (MCP) that runs on the LANai processor on the Myrinet interface card. The Portals module is responsible for determining how incoming messages are processed. It reads the application process' memory and interprets the Portal data structures. The Portals module communicates information about message delivery to the RTS/CTS module, which is responsible for packetization and flow control. The RTS/CTS module communicates packet delivery information to the MCP, which is essentially a packet delivery device.

## 4. PERFORMANCE BENCHMARKS

The following sections present a comparison of the communication and computation performance of the Cplant cluster versus TFLOPS. We describe the network performance of the two machines with a standard ping-pong message-passing benchmark. Results of the NAS Parallel Benchmark suite version 2.3 Class B are also presented to characterize the compute and communication performance of the two platforms.

### 4.1. Message-Passing Benchmarks

Figure 2 shows the MPI one-way message-passing bandwidth for Portals in Linux and Portals in Puma. The asymptotic MPI bandwidth is 311 MB/s for Puma Portals and 54 MB/s for Portals in Linux. The high-performance network on TFLOPS significantly outperforms the Myrinet.

Figure 3 shows the MPI one-way latency performance. The zero-length MPI latency is  $13\mu\text{sec}$  for Puma Portals and  $106\mu\text{sec}$  for Linux Portals. Not surprisingly, the highly optimized MPP network with the tightly integrated interface outperforms

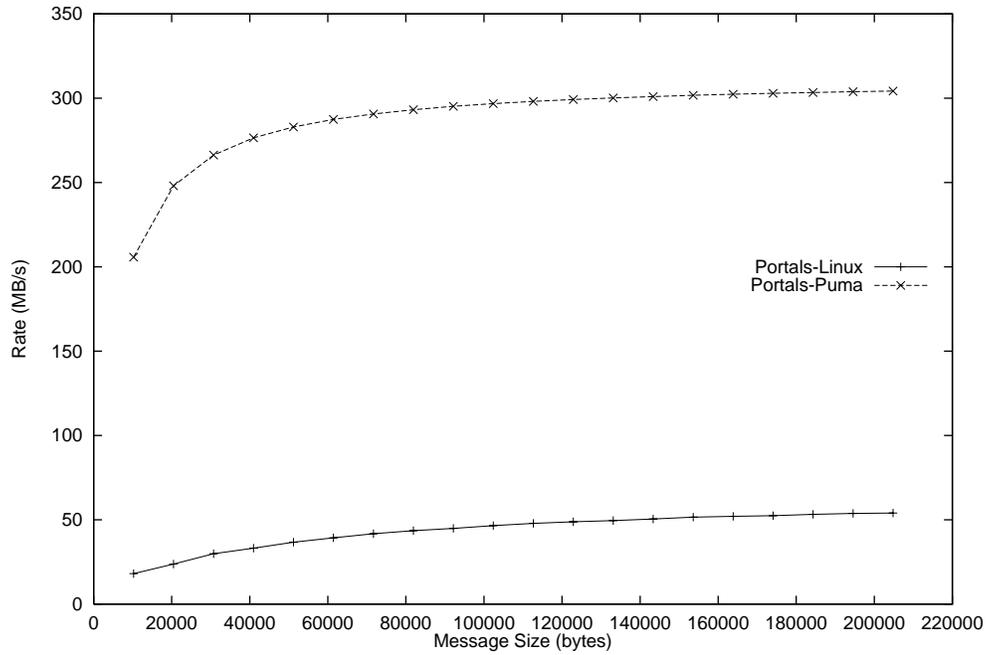


FIG. 2. MPI one-way bandwidth on TFLOPS and Cplant.

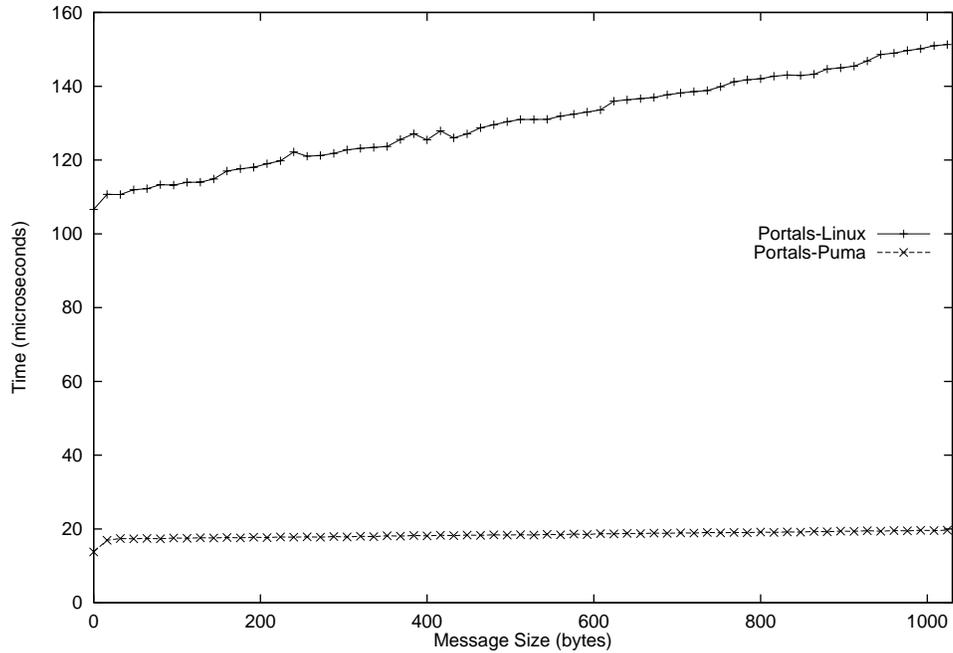


FIG. 3. MPI one-way latency on TFLOPS and Cplant.

the Myrinet network with the PCI bus interface. However, the latency performance of Portals in Linux is significantly less than what other Myrinet software implementations have been able to achieve [17, 14, 22]. The programmability of Myrinet has allowed for the development of MCP's and support libraries that can achieve less than  $10\mu\text{sec}$  for MPI one-way latency. The reasons for such high latency numbers for Portals can be attributed to several factors.

First, Portals was designed for platforms where the network interface resides on the memory bus and has access to all of the physical memory on a node. For example, the Portals implementation in Puma uses data structures in an application's address space to describe to the kernel how to respond to incoming messages. This lack of an explicit API does not allow important data structures to be placed anywhere but in user space. This placement severely limits the performance possible for the Portals implementation in Linux. Because the Linux kernel does not have access to all of physical memory, data structures in user space must be copied into kernel space to be inspected by the Portals module. After inspection, these data structures are manipulated and then copied back into user space. Crossing these protection boundaries and moving data back and forth between user and kernel space is expensive in Linux.

In addition to excessive memory copying, interrupts are used to process incoming messages. Rather than wasting host processor cycles polling the network or using a dedicated message co-processor, the MCP generates an interrupt to service the network. Some Myrinet implementations have even been able to achieve extremely low latencies by dedicating the host processor to polling the network. While low latency is desirable, we feel that a message passing implementation that maximizes the amount of compute cycles delivered to the application, rather than to the network, is more desirable.

The limitations with respect to latency of the Portals implementation in Linux were discovered early in the porting effort. However, we also knew from experience that a critical component of a scalable massively parallel distributed memory machine is bandwidth performance. The bandwidth performance of Portals is greater than many other Myrinet implementations, especially those optimized for short message performance.

Since the entire scalable runtime environment was being layered on Portals, we continued to develop our code on top of this interface while searching for a new message-passing layer. We developed a new message-passing interface, Portals 3.0 [5], designed specifically to support massively parallel clusters. We expect this new interface to allow us to overcome the latency performance limitations of the current generation of Portals in Linux.

## 4.2. Application Benchmarks

We now describe the performance of several of the NAS Parallel Benchmarks [2] version 2.3 Class B on Cplant and TFLOPS. The suite is widely used to compare the performance of all types of parallel computing platforms, since it contains computational kernels that are representative of several different algorithms used in many real-world applications. The individual codes in the suite are limited in the number of processors on which they can run, either by requiring a power of two or

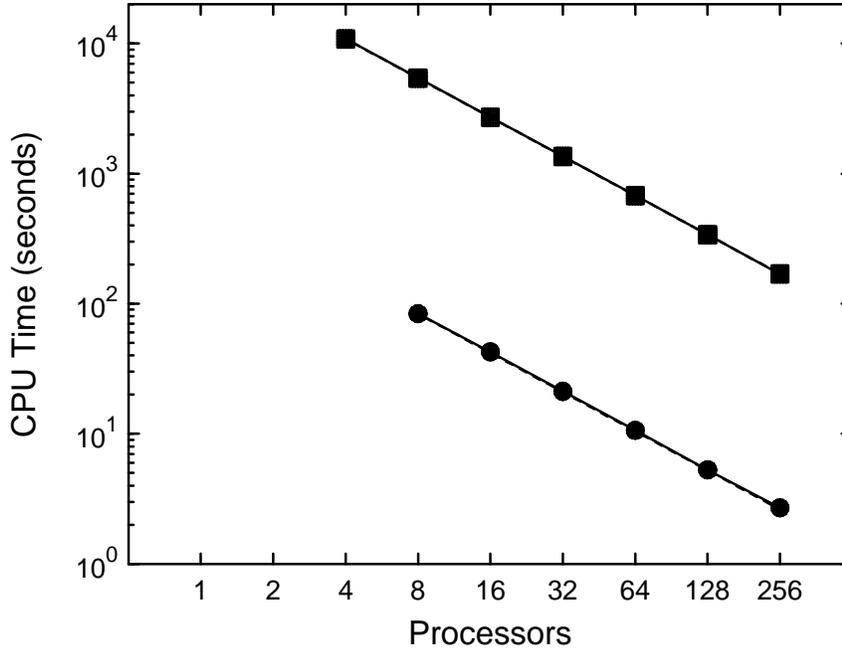


FIG. 4. Total runtime for the EP benchmark on TFLOPS (squares) and Cplant (circles).

a square number of processors, and the size of the problem for each benchmark is fixed as the number of processors is increased.

Figure 4 shows the total runtime results of the EP benchmark on Cplant and TFLOPS out to 256 processors. This benchmark is totally computation bound and performs no message passing. Since there are no communication effects on scaling, this benchmark only characterizes the difference in compute performance between the two platforms.

Figure 5 and Figure 6 show the total runtime for the MG and BT benchmarks respectively on Cplant and TFLOPS up to 256 processors. The dotted line represents perfect speedup relative to the runtime on the fewest processors. Both of these benchmarks are computation bound [30]. Given that the peak theoretical compute performance is 1 GFLOPS for a Cplant processor and 300 megaFLOPS (MFLOPS) for a TFLOPS processor, Cplant outperforms TFLOPS all the way out to 256 processors. However, because the problem size is fixed, each node has less computation to perform as the number of processors is increased. This allows the compute performance of TFLOPS to approach that of Cplant on large numbers of processors.

Figure 7 shows the results of the LU benchmark on Cplant and TFLOPS out to 256 processors. TFLOPS outperforms Cplant on this benchmark, except for the 256 processor case. The LU benchmark uses a finer-grain communication scheme than the other benchmarks, making it sensitive to message passing latency [30]. This may explain why TFLOPS is able to outperform Cplant at smaller numbers of processors. It is unclear why Cplant outperforms TFLOPS for the 256 processor case.

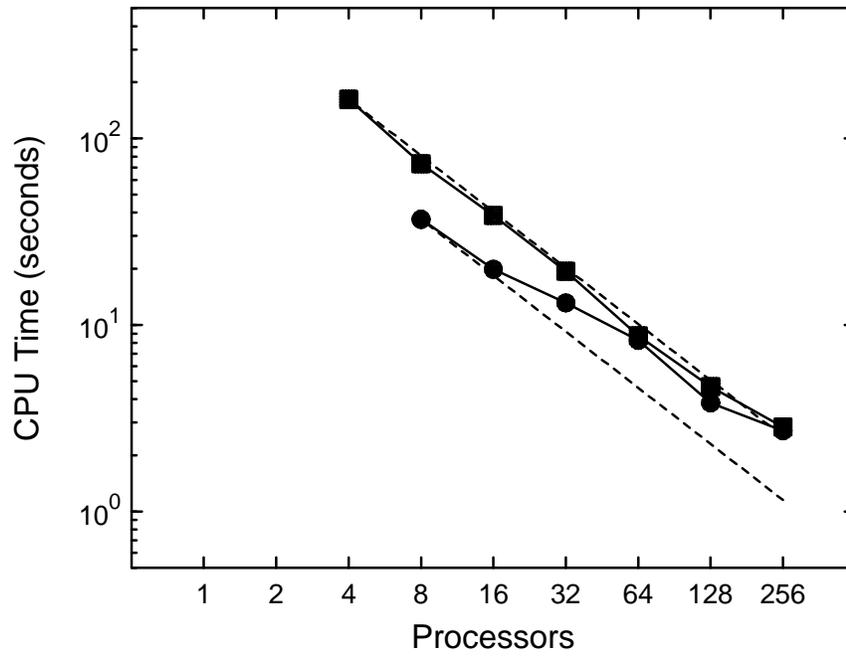


FIG. 5. Total runtime for the MG benchmark on TFLOPS (squares) and Cplant (circles).

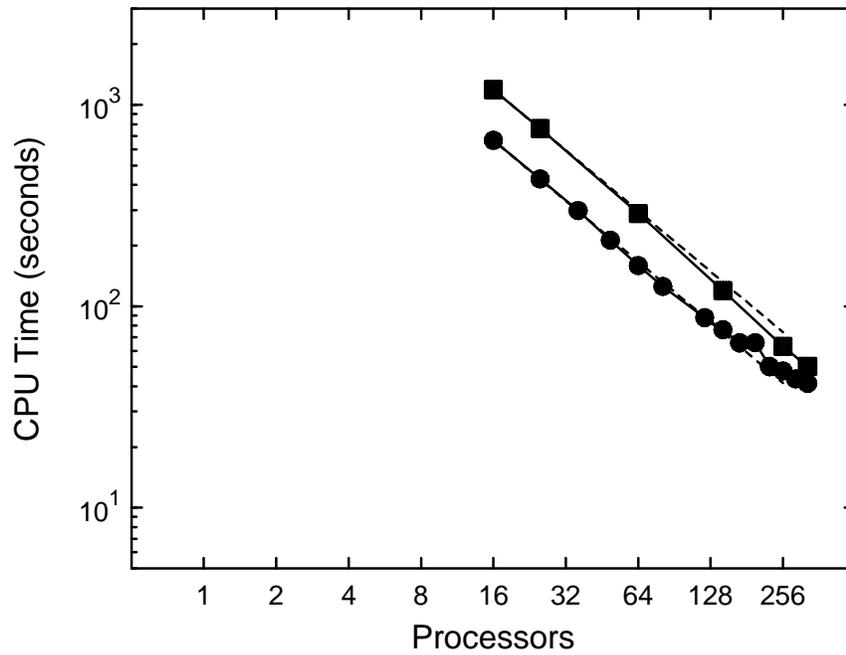


FIG. 6. Total runtime for the BT benchmark on TFLOPS (squares) and (circles).

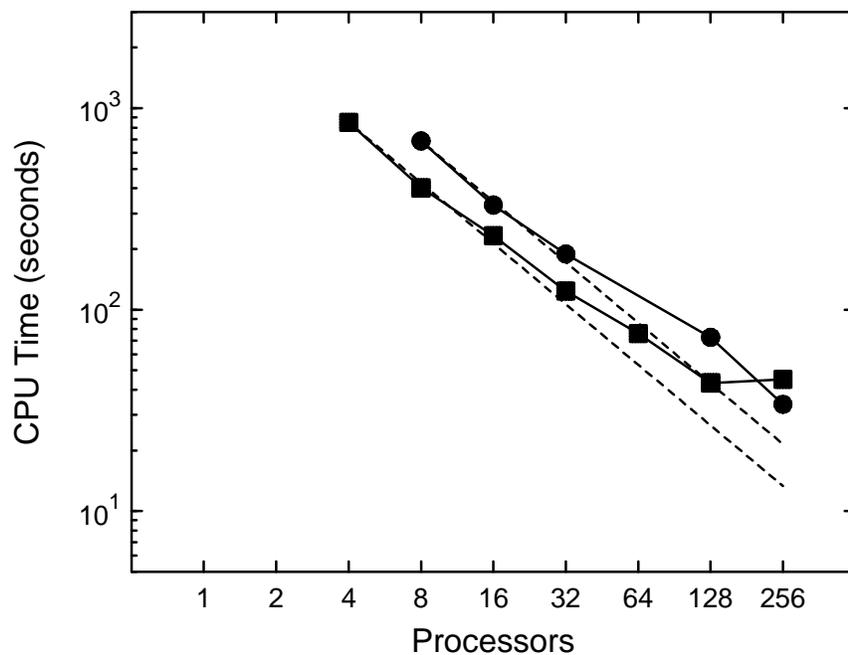


FIG. 7. Total runtime for the LU benchmark on TFLOPS (squares) and Cplant (circles).

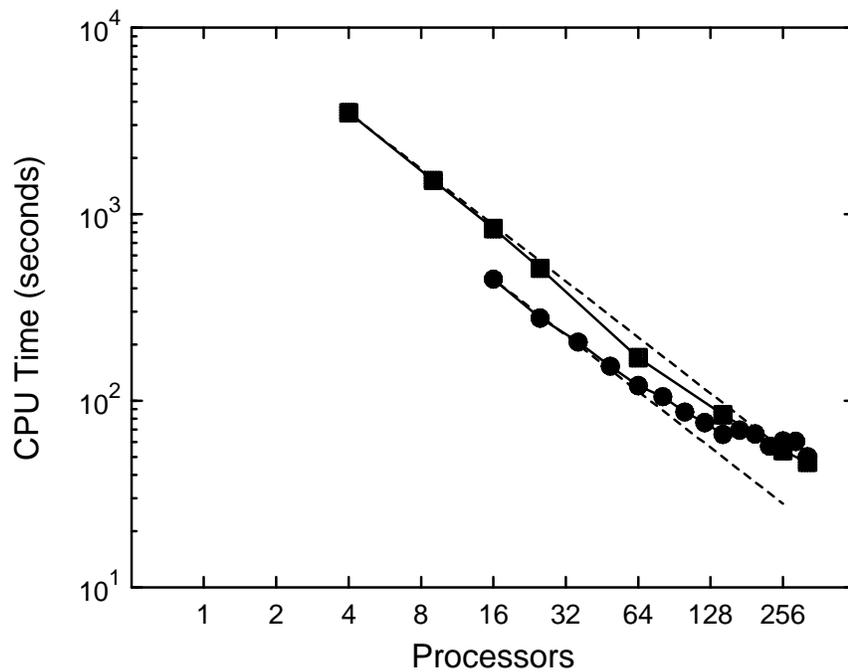


FIG. 8. Total runtime for the SP benchmark on TFLOPS (squares) and Cplant (circles).

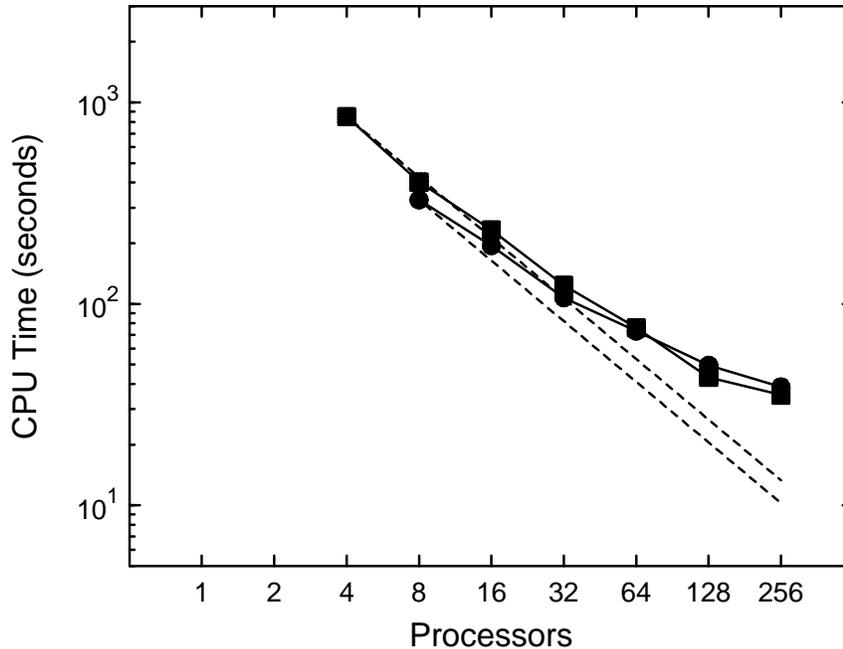


FIG. 9. Total runtime for the CG benchmark on TFLOPS (squares) and Cplant (circles).

Figure 8 shows the results of the SP benchmark on Cplant and TFLOPS out to 256 processors. Cplant outperforms TFLOPS to just before 256 processors, at which point TFLOPS has a faster runtime. For this benchmark, each processor owns a spatial sub-domain where the surface-to-volume ratio increases with the number of processors. Since shared surfaces between processors require communication, superior communication performance of TFLOPS allows it to win out on large numbers of processors.

Figure 9 shows the results of the CG benchmark on Cplant and TFLOPS out to 256 processors. These results are similar to the previous results on the SP benchmark. The crossover point occurs much earlier, at about 70 processors. The communication requirements of this benchmark also favor TFLOPS for large numbers of processors.

## 5. APPLICATIONS

In this section we present performance results for three full-scale applications run on both the Cplant and TFLOPS machines. The applications are all different in nature, both from a scientific and computational standpoint. Hence they test Cplant's capabilities versus a traditional MPP machine in different ways. In each of the three cases we examine performance trade-offs for scalability on both fixed-size and scaled-size problems.

The first simulations model the motion of an ensemble of strongly interacting particles. The second code solves a set of partial-differential equations (PDEs) (Maxwell's equations) on structured grids using finite-difference methods. The

final application is for a PDE (Boltzmann radiation transport equation) solved via directional sweeps on an unstructured grid.

### 5.1. Particle Simulation

The first application is a molecular dynamics (MD) model of a Lennard-Jones (LJ) liquid. In an MD simulation, Newton’s equations of motion are integrated for a collection of interacting particles. In a LJ system, on a given timestep a particle interacts in a pairwise fashion with every other particle that is nearby (within some cutoff distance).

The simulations we discuss here are for a 3-d box of particles, decomposed spatially across processors so that each owns a small 3-d “brick” or subsection of the global simulation domain. Each processor computes the forces on its atoms and advances their positions and velocities. This requires each processor to communicate every timestep with neighboring processors to exchange particle information (coordinates, forces). Periodically, particles are also migrated to new processors as they diffuse through the simulation box.

In these simulations the fluid density and cutoff distance were set so that each particle interacts with approximately 55 neighbors every timestep. More details about the benchmark itself and the algorithms used to efficiently model such a system, both from a parallel and serial standpoint, are discussed in Plimpton [19]. This is a model of a simple monotomic fluid, but the same parallel methodology and communication routines are used in a variety of more sophisticated codes we have developed at Sandia for modeling polymeric, biological, and metallic systems [27, 20, 13]. These simulations are a good test for our purpose here in that they are a good “stress-test” of interprocessor communication.

The first set of results in Figure 10 are for fixed-size simulations of an  $N = 32000$  particle system. This is a modest system size that one might typically wish to simulate for very long timescales, e.g. millions of timesteps. A single TFLOPS Pentium II processor runs this simulation in 0.659 secs/timestep. An Alpha processor on Cplant is about 2 times faster at 0.344 secs/timestep. In the figure, one-processor timings on both machines are shown as 100% efficient. Timings on multiple processors were scaled to the one-processor time for the respective machine. Thus a parallel efficiency of 60% on 16 Cplant processors means the simulation ran 9.6 times faster than it did on one Cplant processor.

As expected, the results show that parallel efficiency falls off more quickly on Cplant than on TFLOPS, due to Cplant’s faster processors and slower communication. Still, in terms of absolute CPU time, Cplant remains competitive with TFLOPS out to 64 processors, where the CPU time/timestep is 0.0132 and 0.0128 secs for Cplant and TFLOPS respectively.

In Figure 11, we show timing results for a scaled version of the same physical model. On one processor we run the same 32000-particle simulation as in the previous figure. On  $P$  processors we simulate a system that is  $P$  times larger. Thus on 300 Cplant processors (rightmost data point in figure), we are simulating 9.6 million atoms. Perfect efficiency (100%) now represents a simulation that runs in the same CPU time/timestep as the one-processor timing (0.344 secs/timestep on Cplant, 0.659 secs/timestep on TFLOPS).

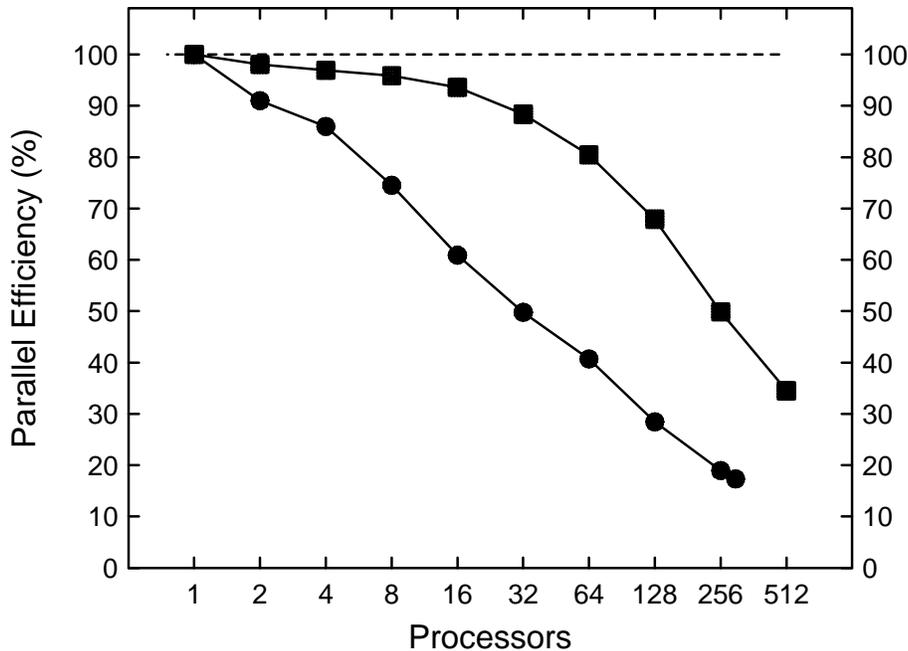


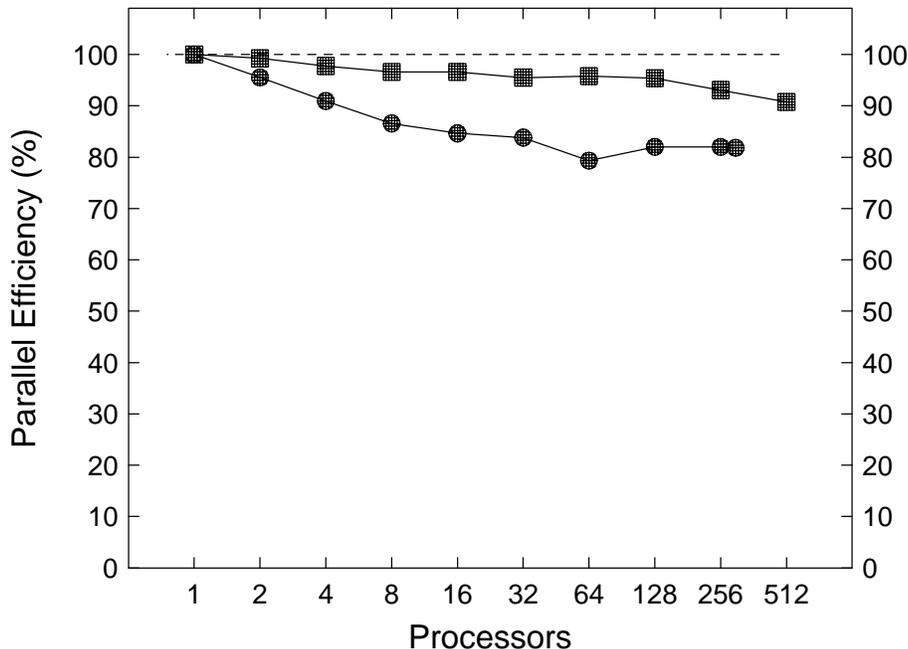
FIG. 10. Parallel efficiencies for the same molecular dynamics simulation run on varying numbers of processors on TFLOPS (squares) and Cplant (circles). This is a fixed-size simulation of 32000 particles.

These timings show Cplant runs this simulation in a reasonably scalable fashion. Since its processors are twice as fast as TFLOPS processors on this code, Cplant is running these large simulations 1.7 to 2 times faster than TFLOPS for all processor counts shown in the figure.

## 5.2. Electromagnetics on a Structured Grid

QUICKSILVER is a large particle-in-cell (PIC) electromagnetics code, developed and parallelized at Sandia over the last several years. It models relativistic charged particle transport and the evolution of electric and magnetic fields via solutions to Maxwell’s equations. The field computations are performed on one or more structured grid blocks using a finite-difference time-domain integrator in either an explicit or implicit formulation. In this section we run QUICKSILVER in a fields-only mode where we model the propagation of incident wave pulses across a large domain containing embedded conductors. We run these calculations in explicit mode because this maximizes the communication/computation ratio within each timestep, and thus is again a good “stress-test” of Cplant’s performance. More details about QUICKSILVER and the various algorithms used in its parallelization can be found in Plimpton [21].

The basic parallel strategy is to break up the global domain into small 3-d sub-blocks, one (or more) owned by each processor. A single processor performs field updates within its sub-domain(s) and exchanges electric and magnetic field components with surrounding processors. Because the field components are edge- and face-centered quantities within the grid cells, in these benchmark runs each processor ends up communicating with 26 neighbor blocks each timestep.

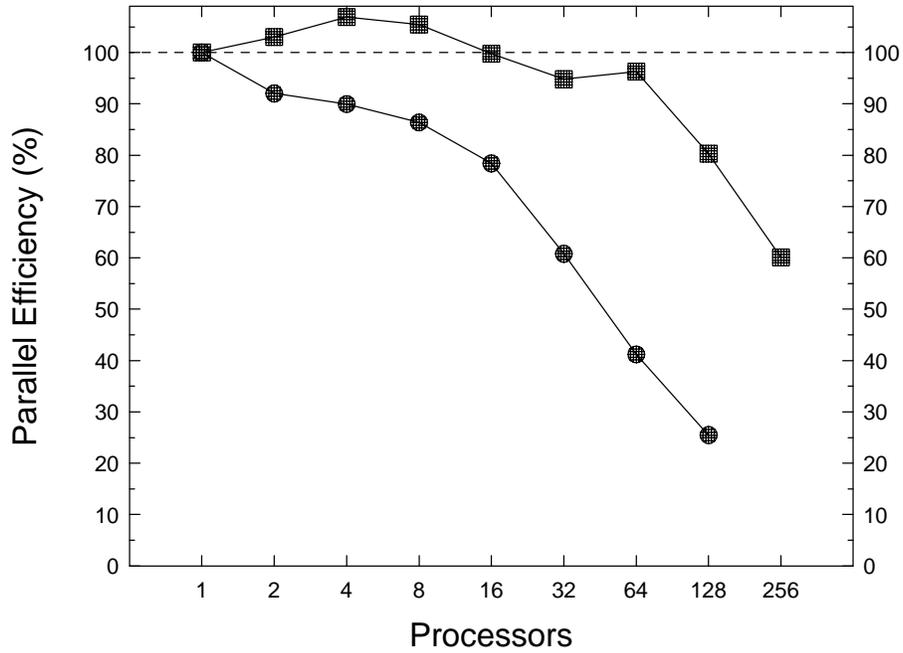


**FIG. 11.** Parallel efficiencies for scaled-size molecular dynamics simulations run on TFLOPS (squares) and Cplant (circles). The simulation size was 32000 particles/processor. Thus the leftmost point are simulations of 32000 particles; the rightmost points are simulations of 16.3 million particles on TFLOPS and 9.6 million on Cplant (300 processors).

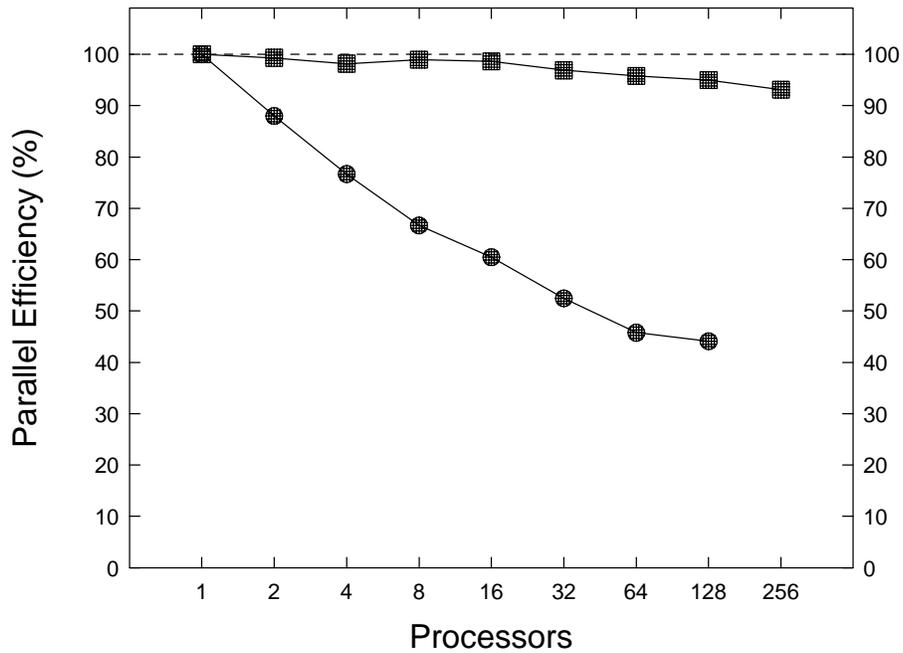
In Figure 12 we show parallel efficiencies for a simulation of wave propagation in a 768,000 grid-cell model. This is a problem size that might typically be run for tens of thousands of timesteps over many hours on a high-end desktop machine. As in the previous section, one-processor timings on both machines are normalized to 100% efficiency. A Cplant processor is about 2.5 times faster than a TFLOPS processor on this simulation (0.56 sec/timestep versus 1.38 sec/timestep). Timings on TFLOPS show super-linear speed-up on 2-8 processors due to cache effects when the field arrays can be fit into cache as the problem size (per processor) is reduced.

In Figure 13 a scaled-size problem was run where each processor owned a sub-block with 27000 grid cells. Thus the largest simulations were roughly 3.5 million grid cells on Cplant (128 processors) and nearly 7 million cells on TFLOPS (256 processors). Here the results on TFLOPS show excellent scalability (over 90% efficient), while the Cplant scalability degrades about as quickly as it did in previous plot for fixed-size scaling.

The reason for this is again due to the disparity in computation/communication ratios for the two machines. The relatively slow communication on Cplant, particularly the high latencies for the small messages being exchanged (with some of the 26 neighboring processors) imposes a high overhead cost. Also, on this problem the one-processor timings are more than a factor of 3 different (0.0412 sec/timestep on TFLOPS versus 0.0129 sec/timestep on Cplant). This corresponds to a 40 MFLOPS versus 120 MFLOPS compute rate on the two machines for the field updating. This means that in terms of raw performance on 128 processors, Cplant is still running this simulation about 1.5 times faster than TFLOPS.



**FIG. 12.** Parallel efficiencies for the same electromagnetic simulation run on TFLOPS (squares) and Cplant (circles). This calculation was a production-scale model of a wave pulse propagating through a structured grid of 768,000 cells for 2000 timesteps.



**FIG. 13.** Parallel efficiencies for scaled-size electromagnetic simulations run on TFLOPS (squares) and Cplant (circles). The model ran for 10000 timesteps on a grid size of 27000 grid cells/processor. The 128-processor data points are for simulations on a grid of 3.5 million cells.

### 5.3. Radiation Transport on an Unstructured Grid

The final application we discuss is a radiation transport simulator which computes the solution to Boltzmann’s equation for radiative flux on an unstructured grid. The code computes the energy distribution of the flux in six dimensions (3 spatial, 2 angular, and 1 energy) via the method of discrete ordinates which partitions the angular dependence of the solution over a finite number of (ordinate) directions [10, 7]. The solution in any one direction can be computed by a sweep across the grid where each finite element (grid cell) solves for its contribution only when neighboring cells that are “upwind” from it (relative to the sweeping direction) have already completed their solution. A finished cell then passes flux along to its “downwind” neighbors.

Our parallel implementation of this algorithm employs a spatial decomposition of the 3-d unstructured grid across processors (via the CHACO partitioning tool [12]). As a processor computes the solution in one of its grid cells it sends flux information to any downwind cells that may be owned by neighboring processors. This means that a processor owning a sub-region of the global grid that is far downwind must wait for all upwind processors to complete their computations. This natural load-imbalance is (partially) alleviated by working on many dozens or hundreds of directional solutions simultaneously. As a sweep is computed, the net effect is that a very large number of small messages are sent asynchronously in a point-to-point fashion between all different pairs of neighboring processors. This code is thus an excellent test of a cluster’s ability to do small-message unstructured communication.

In Figure 14 we show the results for a fixed-size simulation with 6360 hexahedral finite elements, 80 ordinate directions, and 2 energy groups. Despite the small grid size, due to the 6-dimensionality of the data structures, this is actually a fairly large one-processor problem. As before, we normalize the one-processor timings on both platforms as 100% efficient to compute the efficiency results for multiple processors. In absolute CPU time, a Cplant processor is a little less than two times faster than a TFLOPS processor on this calculation (113.7 seconds for a sweep of all 80 ordinates versus 201.7 seconds). Both machines show reasonable scalability out to large numbers of processors, especially considering the fact there are only 25 grid/cells per processor in the 256-processor runs. Note that Cplant takes an immediate efficiency hit on two processors as inter-processor communication is required, but performance degrades more slowly thereafter.

It is difficult to construct scaled-size problems of ideal size on unstructured meshes. Instead, in Figure 15, we show CPU timings for both machines running 3 different sized problems, where  $N$  is the number of finite element grid cells, ranging from 6360 to nearly 100,000. Because these data are actual timings, the difference in processor speeds between the two machines is now evident. In contrast to the previous 5 figures, perfect scalability would now be a sloped line (offset for each machine), going through the one-processor data point for that machine. Dotted reference lines are shown for Cplant on the figure.

These data again show that Cplant is not scaling as well as TFLOPS. However the crossover point in raw performance is at about 64-128 processors on the smaller problems and looks like it will be pushed even further out on larger problems. This

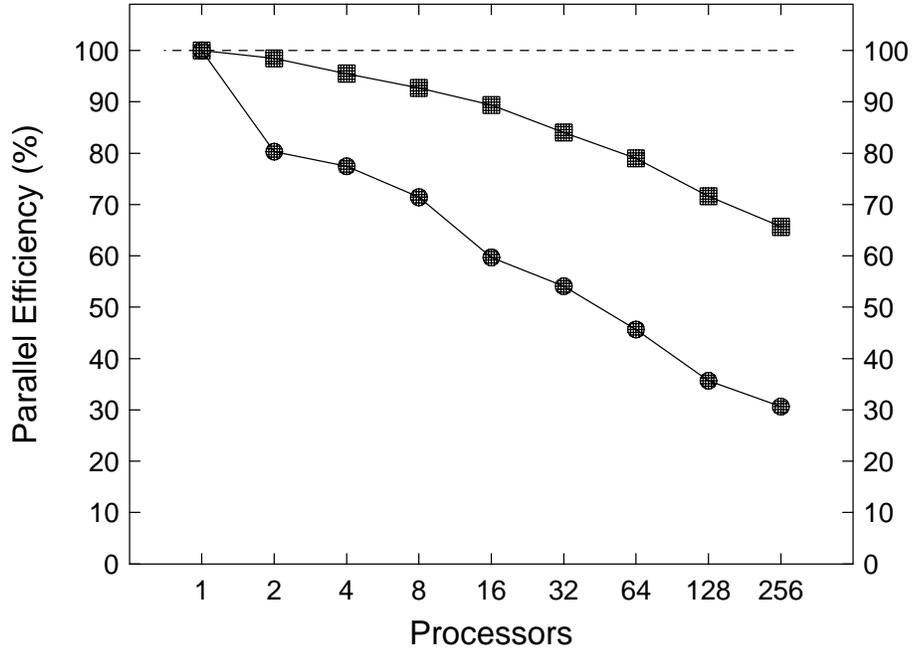


FIG. 14. Parallel efficiencies for the same radiation transport calculation run on TFLOPS (squares) and Cplant (circles). The radiation equations were solved for 80 ordinate directions and 2 energy groups on a grid of 6360 hexahedral elements.

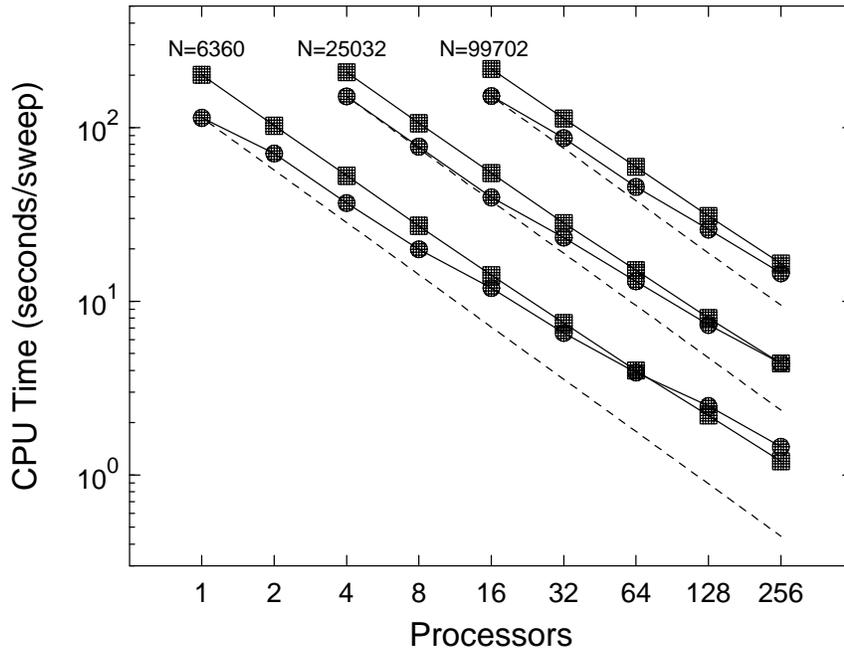


FIG. 15. CPU times for a radiation transport simulation on three different grid sizes from 6360 to 99702 elements. Each model size was run on varying numbers of processors on TFLOPS (squares) and Cplant (circles). The dotted lines represent perfect speed-up on Cplant.

will enable some truly large radiation transport calculations to be done effectively on the Cplant platform.

## 6. CONCLUSIONS

From a system software point of view, our experiences in porting the TFLOPS environment to a large Linux cluster have allowed us to better understand the Linux operating system and its interaction with a high-performance network interface. We have identified some limitations in hardware, software, and topology that prevent us from achieving the communication performance that will allow Cplant to scale out to thousands of nodes. The design of Portals 3.0 will allow us to better utilize the programmable Myrinet cards. We hope to make modifications to the memory management structures in Linux to allow for physically contiguous regions of memory, which should simplify and increase the performance of our implementation of Portals 3.0.

From an application standpoint, we have shown that despite a significant difference in the ratio of compute-to-communication performance, a large Linux cluster can often meet and sometimes exceed the performance of a large MPP up to a few hundred processors. Our experience with the applications discussed here is typical of other Cplant users at Sandia who have tested a variety of applications on the machine over the last six months. These applications include shock hydrodynamics codes, fluid dynamics and reacting flow simulators, and quantum electronic structure models, to name a few.

In many cases on our cluster, the key feature that appears to be limiting application scalability is message-passing latency. This can sometimes be overcome by increasing the problem sizes that are simulated, but this is often not a good solution from the analyst's point of view. Our hope is that the re-design of Portals 3.0 will provide a significant decrease in latency times and thus alleviate many of these problems. However, we note that our next-generation cluster, which we describe in the next section, has faster processors. Thus we are again changing the machine's compute-to-communication ratio in a direction that will degrade overall scalability. It remains to be seen whether we can achieve scalable performance at the near-1000 processor level that this new machine will offer.

## 7. FUTURE WORK

In October of 1999, Compaq Computer Corporation installed a 592-node 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 [9] benchmark, which would place it at number 40 on the November 1999 list of the Top 500 [18] 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 results for the benchmarks and applications presented in this paper on this platform in early 2000 when the machine is deployed into the Sandia production computing environment. We will add these results to the final version of this paper as it goes through the review process.

## REFERENCES

1. D. A. Bader, A. B. Maccabe, J. R. Mastaler, J. K. McIver III, and P. A. Kovatch. Design and Analysis of the Alliance / University of New Mexico Roadrunner Linux SMP Super Cluster. In R. Buyya, M. Baker, K. Hawick, and H. James, editors, *Proceedings of the IEEE Computer Society International Workshop on Cluster Computing*, pages 9–18, Melbourne, Australia, December 1999.
2. D. H. Bailey et al. The NAS Parallel Benchmarks. *International Journal of Supercomputer Applications*, 5(3):63–73, 1991.
3. N. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. Su. Myrinet—a gigabit-per-second local-area network. *IEEE Micro*, 15(1):29–36, February 1995.
4. 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*, 26(2-3):243–266, February 2000.
5. R. B. Brightwell, T. B. Hudson, A. B. Maccabe, and R. E. Riesen. The Portals 3.0 Message Passing Interface. Technical Report SAND99-2959, Sandia National Laboratories, December 1999.
6. R. B. Brightwell and P. L. Shuler. Design and implementation of MPI on Puma portals. In *Proceedings of the Second MPI Developer's Conference*, pages 18–25, July 1996.
7. S. P. Burns. Spatial domain-based parallelism in large scale, participating-media, radiative transport applications. Technical Report SAND96-2485, Sandia National Laboratories, Albuquerque, NM, 1996.
8. Compaq, Microsoft, and Intel. Virtual Interface Architecture Specification Version 1.0. Technical report, Compaq, Microsoft, and Intel, December 1997.
9. J. J. Dongarra. Performance of Various Computers Using Standard Linear Equations Software. Technical Report CS-89-85, Department of Computer Science, University of Tennessee, 1994.
10. W. A. Fiveland. Three-dimensional radiative heat-transfer solutions by the discrete ordinates method. *J. Thermophysics and Heat Transfer*, 2:309–316, 1988.
11. D. S. Greenberg, R. B. Brightwell, L. A. Fisk, A. B. Maccabe, and R. E. Riesen. A System Software Architecture for High-End Computing. In *Proceedings of SC'97*, 1997.
12. B. Hendrickson and R. Leland. The Chaco user's guide: Version 2.0. Technical Report SAND94-2692, Sandia National Labs, Albuquerque, NM, June 1995.
13. C. L. Kelchner, S. J. Plimpton, and J. C. Hamilton. Dislocation nucleation and defect structure during surface indentation. *Phys. Rev. B*, 58:11085, 1998.
14. M. Lauria, S. Pakin, and A. Chien. Efficient Layering for High Speed Communication: Fast Messages 2.x. In *Proceedings of the IEEE International Symposium on High Performance Distributed Computing*, 1998.
15. A. B. Maccabe, K. S. McCurley, R. E. Riesen, and S. R. Wheat. SUNMOS for the Intel Paragon: A brief user's guide. In *Proceedings of the Intel Supercomputer Users' Group. 1994 Annual North America Users' Conference.*, pages 245–251, June 1994.
16. Message Passing Interface Forum. MPI: A Message-Passing Interface standard. *The International Journal of Supercomputer Applications and High Performance Computing*, 8, 1994.
17. Myricom, Inc. The GM Message Passing System. Technical report, Myricom, Inc., 1997.
18. Netlib. *Top 500 Supercomputers*, 1998. <http://www.top500.org>.
19. S. J. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *J. Comp. Phys.*, 117:1–19, 1995.
20. S. J. Plimpton, R. Pollock, and M. Stevens. Particle-mesh ewald and rREPSA for parallel molecular dynamics simulations. In *Proc. 8th SIAM Conference on Parallel Processing for Scientific Computing, Minneapolis, MN*. SIAM, 1997.

21. S. J. Plimpton, D. B. Seidel, M. F. Pasik, and R. S. Coats. Novel load-balancing techniques for an electromagnetic particle-in-cell code. Technical Report SAND2000-0035, Sandia National Laboratories, Albuquerque, NM, 2000.
22. L. Prylli. BIP Messages User Manual for BIP 0.94. Technical report, LHPC, June 1998.
23. Sandia National Laboratories. *ASCI Red*, 1996. [http://www.sandia.gov/ASCI/TFLOP/Home\\_Page.html](http://www.sandia.gov/ASCI/TFLOP/Home_Page.html).
24. Sandia National Laboratories. *Computational Plant*, 1997. <http://www.cs.sandia.gov/cplant>.
25. Sandia National Laboratories. *Puma Portals*, 1997. <http://www.cs.sandia.gov/puma/portals>.
26. 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.
27. M. J. Stevens and S. J. Plimpton. The effect of added salt on polyelectrolyte structure. *Eur. Phys. J. B*, 2:341-345, 1998.
28. Task Group of Technical Committee T11. Information Technology - Scheduled Transfer Protocol - Working Draft 2.0. Technical report, Accredited Standards Committee NCITS, July 1998.
29. M. S. Warren, M. P. Goda, D. J. Becker, J. K. Salmon, and T. Sterling. Parallel Supercomputing with Commodity Components. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, 1997.
30. F. C. Wong, R. P. Martin, R. H. Arpaci-Dusseau, and D. E. Culler. Architectural Requirements and Scalability of the NAS Parallel Benchmarks. In *Proceedings of SC'99*, November 1999.

## ACKNOWLEDGMENTS

The authors would like to acknowledge the contributions of the following people: