Pentium Pro Inside: I. A Treecode at 430 Gigaflops on ASCI Red
II. Price/Performance of $50/Mflop on Loki and Hyglac

Michael S. Warren
Theoretical Astrophysics
Mail Stop B288
Los Alamos National Laboratory
Los Alamos, NM 87545

John K. Salmon
Center for Advanced Computing Research
California Institute of Technology
Mail Code 206-49
Pasadena, CA 91125

Donald J. Becker
Goddard Space Flight Center
Code 930.5
Greenbelt, MD 20771

M. Patrick Goda
Theoretical Astrophysics
Los Alamos National Laboratory
Los Alamos, NM 87545

Thomas Sterling
Center for Advanced Computing Research
Caltech/JPL, Mail Code 158-79
Pasadena, CA 91125

Grégoire S. Winckelmans
Mechanical Engineering Dept.
Universite Catholique de Louvain
B-1348 Louvain-la-Neuve, Belgium

Abstract

As an entry for the 1997 Gordon Bell performance prize, we present results from two methods of solving the gravitational N-body problem on the Intel Teraflops system at Sandia National Laboratory (ASCI Red). The first method, an $O(N^2)$ algorithm, obtained 635 Gigaflops for a 1 million particle problem on 6800 Pentium Pro processors . The second solution method, a treecode which scales as $O(N \log N)$, sustained 170 Gigaflops over a continuous 9.4 hour period on 4096 processors, integrating the motion of 322 million mutually interacting particles in a cosmology simulation, while saving over 100 Gigabytes of raw data. Additionally, the treecode sustained 430 Gigaflops on 6800 processors for the first 5 timesteps of that simulation. This treecode solution is approximately 10^5 times more efficient than the $O(N^2)$ algorithm for this problem.

As an entry for the 1997 Gordon Bell price/performance prize, we present two calculations from the disciplines of astrophysics and fluid dynamics. The simulations were performed on two 16 Pentium Pro processor Beowulf-class computers (Loki and Hyglac) constructed entirely from commodity personal computer technology, at a cost of roughly $50k each in September, 1996. The price of an equivalent system in August 1997 is less than $30k. At Los Alamos, Loki performed a gravitational treecode N-body simulation of galaxy formation using 9.75 million particles, which sustained an average of 879 Mflops over a ten day period, and produced roughly 10 Gbytes of raw data. During the initial 10 hours of the simulation, Loki sustained 1.19 Gigaflops. This simulation is nearly identical to that which won a Gordon Bell performance prize in 1992 on the 512 processor Intel Delta. At Caltech, Hyglac performed a simulation of the fusion of two vortex rings using a vortex particle method which took place over a 20 hour period, sustaining about 950 Mflops over that time span. Loki and Hyglac were connected together on the floor of Supercomputing '96 in November 1996, obtaining 2.19 Gigaflops on an N-body treecode benchmark.

1 Introduction

We are at a unique point in the history of supercomputing, where the fastest computer in the world (the Intel Teraflops system at Sandia National Laboratory, ASCI Red) utilizes exactly the same processor as is found inside millions of desktop machines. Powerful mass-market economic forces led Intel to base the design of their flagship parallel machine on commodity parts. Presumably, economic forces also convinced them to withdraw from the world of supercomputing, turning Intel Supercomputing Systems into Intel Scalable Server Products.

The rate of change for all aspects of computing is extraordinary, and difficult to keep pace with. The fixed costs of system maintenance and software development remain roughly constant, while the performance of computer hardware keeps increasing by an order-of-magnitude every five years (a corollary of Moore's Law). Paradoxically, the success of supercomputing centers and the Internet has meant that the amount of computing available to each individual research group has actually declined over the past few years, since practically all large computational resources must be shared with an ever-larger group of people interested in high-performance computing. As computational scientists, we need access to usable, reliable, abundant and affordable computing power. We have expended considerable effort to develop and implement parallel algorithms, while at the same time watching Intel and others retreat from or be destroyed on the battlefield of high-performance computing.

This leads us to consider the danger that the computing systems that we need to do our research will either cease to be developed, or become prohibitively expensive. At present, the primary vendors of parallel computers are large companies which can subsidize their parallel computing research and development through profits from other business divisions. Until recently, economics has played a limited role in the evolution of supercomputing. Machines were designed to attract attention, through heroic efforts to achieve the highest speeds possible. The era of easily available subsidized supercomputing has come to an end, just as many computational scientists have developed efficient codes capable of taking advantage of parallel hardware. These forces have led to a quiet revolution in the design and construction of modest message-passing parallel machines. Building upon the foundation of the BEOWULF project [1], it is now possible to use off-the-shelf hardware and software to construct a parallel machine capable of exceeding 1 Gflop performance for only a few tens of thousands of dollars [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]. This paper describes our experiences at the highest-end of supercomputing with the ASCI Red system, as well as our experience with parallel machines we have constructed ourselves, out of commodity parts and software.

2 Architecture

2.1 The Intel Teraflops System

The complete ASCI Red machine contains 4,536 compute nodes, each containing two 200 Mhz Pentium Pro processors and 128 Mbytes of 4-way interleaved DRAM. A custom interconnect provides 800 Mbytes/sec of bi-directional bandwidth at the hardware level, using a 38x32x2 mesh. Using MPI, we measured uni-directional bandwidth out of one node of 290 Mbytes/sec, and round-trip latencies of 68 or 41 microseconds, depending on whether or not the second CPU was used as a communication co-processor. When the calculations described here were performed, the complete system was not yet installed, so the maximum number of nodes available was 3400 (6800 processors) with a theoretical peak speed of 1.36 Teraflops. For further information, refer to [15].

Intel claims [16] that "The [Intel Teraflops] system will be the first large-scale supercomputer to be built entirely of commodity, commercial, off-the-shelf (C-COTS) components — the same processors, memory, disks and other modules found in millions of desktop computers and servers." This is true, as far as the components which are mentioned. However, the network technology and operating system of that machine are far from off-the-shelf. We describe a truly off-the-shelf parallel architecture in the following section.

2.2 Loki and Hyglac: A Commodity Parallel Architecture

In 1992, two authors of this paper were awarded a Gordon Bell Performance Prize [17] for "Astrophysical N-body Simulations Using Hierarchical Tree Data Structures," which was run on the 512 processor Intel Delta machine. It is now five years later, and as we show below, it is possible to run that same simulation on a machine constructed out of mail-order parts and free software for a cost of less than $50k. We have invested our time and money in these systems because they are (at present) clearly superior to any other technology for solving the computational physics problems that we are most interested in. We have no particular desire to build and maintain our own computer hardware. If we could buy a better system for the money, we would be using it instead.

We have constructed two distinct 16 processor Pentium Pro machines, each having 2 Gbytes of RAM, and either 50 or 80 Gbytes of disk space. The machines differ primarily in their network topology. The cost of the machines in the fall of 1996 was roughly $50,000—$60,000. The price to construct an equivalent system as we go to press in August 1997 is less than $30,000. Loki and Hyglac use a commodity communication network based on fast ethernet [18] and the PCI bus [19], as well as the freely available Linux [20] operating system and development tools from the GNU project [21] and RedHat [22], making them true commodity, off-the-shelf machines.

Qty.	Price	Ext.	Description
16	595	9520	Intel Pentium Pro 200 Mhz CPU/256k cache
16	15	240	Heat Sink and Fan
16	295	4720	Intel VS440FX (Venus) motherboard
64	235	15040	8x36 60ns parity FPM SIMMS (128 Mb per node)
16	359	5744	Quantum Fireball 3240 Mbyte IDE Hard Drive
16	85	1360	D-Link DFE-500TX 100 Mb Fast Ethernet PCI Card
16	129	2064	SMC EtherPower 10/100 Fast Ethernet PCI Card
16	59	944	S3 Trio-64 1Mb PCI Video Card
16	119	1904	ATX Case
2	4794	9588	3Com SuperStack II Switch 3000, 8-port Fast Ethernet
		255	Ethernet cables
Total		$51,379	

Table 1: Loki architecture and price (September, 1996).

At the Theoretical Division of Los Alamos National Laboratory, Loki [5] was constructed from 16 nodes as described in Table 1. The whole machine contains 2 Gbytes of memory and 50 Gbytes of disk. All of the operating system software (Linux), software tools and compilers (GNU) used for these results are freely available. We have measured Fast Ethernet bandwidth of 11.5 Mbytes/sec (uni-directional, per port) and latencies of 208 microseconds (round-trip) at the user level with MPI [23]. At the hardware level, we have measured fast ethernet latencies of 55 microseconds (round-trip). An early lesson we learned is that the memory bandwidth of the Pentium Pro Natoma chipset is not sufficient to support more than two fast ethernet ports (about 20 Mbytes/sec of message traffic) per node when using TCP or UDP protocols, due to copies of data from the kernel to user space. Thus, for the results quoted in this paper, Loki was connected in a split-switch topology, using only two ethernet ports per node. It should be noted that each node also contains a 4-port Cogent Ethernet card, but those cards were not connected during the runs described here (the topology was two 8-way switches, with 8 cables connecting each half of the machine). The video card in each node is not strictly necessary, but is required to access the initial BIOS setup screen, and to see certain hardware error messages.

At Caltech/JPL, Hyglac [6] was constructed from 16 nodes which were almost identical to those of Loki. The primary differences were the use of D-Link DFE-500TX 100 Mb Fast Ethernet Cards ($85 each), a Bay Networks 28115 16-way Fast Ethernet Switch, two Western Digital 2.52 Gbyte drives per node, and the use of EDO DRAM. The total price of Hyglac (including 8.75% sales tax) was $50,498.

The current cost of a machine such as Loki or Hyglac is considerably less than the values listed in Table 1. Table 2 shows the costs for the relevant components in August 1997. It is likely that a machine with components equivalent to those in Loki or Hyglac will be constructed for less than $25,000 by the time this paper is published in the Fall of 1997, which would result in price/performance figures which exceed those listed here by a factor of two.

Item	Description	Price ($)
ASUS P/I-XP6NP5	motherboard	220
Pentium Pro	200 MHz, 256k L2	467
Pentium Pro	150 MHz, 256k L2	204
SIMM	FPM 8x36x60, 32 Mbyte	112
Disk	Quantum Fireball 3.2GB EIDE	215
Fast Ethernet	DFE-500TX 21140 PCI	53
Misc.	Case, Floppy, Heat Sink	150
BayStack 350T	16 port 10/100 Mbit switch	2500

Table 2: Spot prices for August, 1997. These were obtained from sites listed at `www.uvision.com`. A 16 processor 200Mhz–2 Gbyte memory–50 Gbyte disk system with BayStack switch would be $28k.

3 N-body methods

N-body methods are widely used in a variety of computational physics algorithms where long-range interactions are important. The $O(N^2)$ solution of the gravitational N-body problem with special purpose hardware has been recognized with a Gordon Bell prize for two years in a row. While that method is appropriate for some problems, we feel a need to vigorously promote the use of smarter algorithms and general purpose hardware.

Several methods have been introduced which allow N-body simulations to be performed on arbitrary collections of bodies in time much less than $O(N^2)$, without imposition of a lattice [24, 25]. They have in common the use of a truncated expansion to approximate the contribution of many bodies with a single interaction. The resulting complexity is usually determined to be $O(N)$ or $O(N \log N)$, which allows computations using orders of magnitude more particles. These methods represent a system of N bodies in a hierarchical manner by the use of a spatial tree data structure. Aggregations of bodies at various levels of detail form the internal nodes of the tree (cells). These methods obtain greatly increased efficiency by approximating the forces on particles. Properly used, these methods do not contribute significantly to the total solution error. This is because the force errors are exceeded by or are comparable to the time integration error and discretization error.

Using a generic design, we have implemented a variety of modules to solve problems in galactic dynamics [26] and cosmology [27] as well as fluid-dynamical problems using smoothed particle hydrodynamics

[28], a vortex particle method [29] and boundary integral methods [30]. Solving each of these problems with different varieties of special purpose hardware, such as GRAPE [31], is clearly intractable.

4 The Hashed Oct-Tree Library

Our parallel N-body code has been evolving for several years, and on many platforms. We began with an Intel ipsc/860, Ncube machines, and the Caltech/JPL Mark III [32, 26]. This original version of the code was abandoned after it won a Gordon Bell Performance Prize in 1992 [17], due to various flaws inherent in the code, which was ported from a serial version. A new version of the code was initially described in [33], and further simulations and other details are reported in [34, 35, 28, 36, 30].

The basic algorithm may be divided into several stages. Our discussion here is necessarily brief. First, particles are domain decomposed into spatial groups. Second, a distributed tree data structure is constructed. In the main stage of the algorithm, this tree is traversed independently in each processor, with requests for non-local data being generated as needed. In our implementation, we assign a `Key` to each particle, which is based on Morton ordering. This maps the points in 3-dimensional space to a 1-dimensional list, which maintaining as much spatial locality as possible. The domain decomposition is obtained by splitting this list into N_p (number of processors) pieces. The implementation of the domain decomposition is practically identical to a parallel sorting algorithm, with the modification that the amount of data that ends up in each processor is weighted by the work associated with each item.

The Morton ordered key labeling scheme implicitly defines the topology of the tree, and makes it possible to easily compute the key of a parent, daughter, or boundary cell for a given key. A hash table is used in order to translate the key into a pointer to the location where the cell data are stored. This level of indirection through a hash table can also be used to catch accesses to non-local data, and allows us to request and receive data from other processors using the global key name space. An efficient mechanism for latency hiding in the tree traversal phase of the algorithm is critical. To avoid stalls during non-local data access, we effectively do explicit "context switching". In order to manage the complexities of the required asynchronous message traffic, we have developed a paradigm called "asynchronous batched messages (ABM)" built from primitive send/recv functions whose interface is modeled after that of active messages.

All of this data structure manipulation is to support the fundamental approximation employed by treecodes:

$$\sum_j \frac{G m_j \vec{d}_{ij}}{|d_{ij}|^3} \approx \frac{G M \vec{d}_{i,cm}}{d_{i,cm}^3} + \ldots, \tag{1}$$

where $\vec{d}_{i,cm} = \vec{x}_i - \vec{x}_{cm}$ is the vector from $\vec{x}_i$ to the center-of-mass of the particles that appear under the summation on the left-hand side, and the ellipsis indicates quadrupole, octopole, and further terms in the multipole expansion. The monopole approximation, i.e., Eqn. 1 with only the first term on the right-hand side, was known to Newton, who realized that the gravitational effect of an extended body like the moon can be approximated by replacing the entire system by a point-mass located at the center of mass. Effectively managing the errors introduced by this approximation is the subject of an entire paper of ours [37].

Isolating the elements of data management and parallel computation dramatically reduces the amount of programming required to implement a particular physical simulation. For instance, only 2000 lines of code external to the library is required to implement a gravitational N-body simulation. The vortex particle method is implemented with 2500 lines interfaced to exactly the same library. Smoothed Particle Hydrodynamics is implemented with 3000 lines. This may be compared to the nearly 20,000 lines of code in the treecode library.

5 Recent simulations on ASCI Red

The statistics quoted below are based on internal diagnostics compiled by our program. Essentially, we keep track of the number of interactions computed. We obtain optimal performance on the Pentium Pro processor by decomposing the reciprocal square root function required for a gravitational interaction into a table lookup, Chebychev polynomial interpolation, and Newton-Raphson iteration, using the algorithm of Karp [38]. This algorithm uses only adds and multiplies, and requires 38 floating point operations per interaction. We *do not* use assembly language for any part of the code. The flop rates follow from the interaction counts and the elapsed wall-clock time. The flop counts are identical to the best available sequential algorithm. We *do not* count flops associated with decomposition or other parallel constructs. The reported times are for the entire application, including I/O, communication, program initialization, etc. For all results quoted below, we use both processors on each node as compute processors.

5.1 A 1 million body $O(N^2)$ benchmark.

We are not fans of the trivial $O(N^2)$ solution to the N-body problem. To its credit, several features of the N^2 algorithm make it easy to obtain high performance. First, the software implementation is simply a double loop, and is very easy to parallelize using a ring decomposition. Second, the number of operations scales like N^2, while the communication scales like N. This means that for reasonably large numbers of particles, the speed of the communication network is not a limiting factor on performance. Third, the computational intensity is fairly high, requiring 38 floating point operations to be performed on each 32 bytes of data, which is read in a regular fashion from memory. This means the usual memory hierarchy can easily keep the processor busy. It is possible to save a factor of two in the overall flop count by using the pair-wise symmetry of the forces, but this requires an additional memory write for each interaction, which slows down the computation to the point where you end up not saving any time.

Solving the N-body problem with an N^2 algorithm is not difficult, but as we show below, it takes a whole lot longer than necessary. Thus, we do not see much benefit to humanity from running an N^2 algorithm for hours and hours. However, we are willing to provide the results of a few minute benchmark in order to compare the raw speeds of the Intel Teraflops system vs the GRAPE system on almost identical problems. We ran a 1 million particle problem for four timesteps on 3400 nodes (6800 processors). The total time required was 239.3 seconds. Thus, we computed $10^6 \times 10^6 \times 38 \times 4$ flops in 239.3 seconds, for an overall throughput of 635×10^9 floating point operations per second **(635 Gflops)**.

5.2 A 322 million body $O(N \log N)$ simulation.

On April 26–27 1997, we ran a simulation with $322,159,436$ particles on a number of nodes which ranged between 1024-2048 for 437 timesteps. We believe this is the largest gravitational N-body simulation ever performed. For a cosmological simulation, having the largest number of particles possible is critically important. Our ability to identify galaxies which can be compared to observational results requires that each galaxy contain hundreds or thousands of particles, and galaxy catalogs will soon be available which contain the positions and redshifts of a million or more galaxies. An image of this simulation is shown in Figure 1. The simulation was of a spherical region of space 200 Mpc (Megaparsec) in diameter; a region large enough to contain several hundred thousand typical galaxies. The region inside a sphere of diameter 160 Mpc was calculated at high mass resolution, while a buffer region of 20 Mpc with a particle mass 8 times higher was used around the outside to provide boundary conditions. The initial conditions were extracted from a 1 billion point dataset, calculated using a 1024^3 point 3-d FFT from a Cold Dark Matter power spectrum of density fluctuations. Overall, the simulation carried out 9.7×10^{15} floating point operations (just shy of 10 Petaflops). We created 10 data files totaling 100 Gbytes. A single data file from this simulation exceeds 10

Gbytes. The only difficulty porting the code to the Teraflops system had to do with saving these large files. Since each data file exceeds 2^{31} bytes, several I/O routines in our code had to be extended to support 64-bit integers.

We quote two performance results from this simulation. The portion of the simulation between timestep 150 and 437 was accomplished during a single continuous run on 2048 nodes (no crashes, no restarts), and was tuned to the greatest extent possible to obtain useful scientific data (as opposed to the largest number of Gflops). During this period, the code computed 1.52×10^{14} interactions over a time span of 9 hours and 24 minutes, for an overall throughput of 170×10^9 floating point operations per second **(170 Gflops)**. It is perhaps interesting to note that the number of floating point operations and amount of data created during this single simulation is comparable to the aggregate amount of computing our group has performed during the five year period preceding this simulation.

We quote a better performance result using 3400 nodes (6800 processors) during the initial portion of the same simulation. For the first five timesteps, we computed 7.18×10^{12} interactions over a time span of 632 seconds for an overall throughput of 431×10^9 floating point operations per second **(431 Gflops)**. This result is better than the 170 Gflops quoted above for two reasons. First, a larger compute partition was available for a limited period of time, and the initial stages of the calculation are more well-behaved in terms of load balance and the amount of time spent in tree traversal overhead.

The demands of the simulation after significant clustering develops are extreme. The load balancing problem associated with galaxy formation is probably more severe than any other conventional computational physics algorithm. The problem domain is irregular and time-dependent, with some regions of space having a density of particles more than 1 million times the mean density. The second factor which results in a lower computational throughput (as clustering develops) is that it takes more operations to traverse the tree and identify which interactions to compute. Much of the useful work accomplished by our algorithm has nothing to do with floating point operations, and is not reflected in the number of Gflops which we report. The only purpose of using a treecode is to *avoid doing* as many floating point operations as possible. While more and more attention is focused on codes which obtain huge “Gflops”, real computational physics is advanced to a far greater degree through the use of sophisticated algorithms which solve a problem faster, with the usual side-effect of doing less floating point operations per second.

6 Recent simulations on Loki and Hyglac

6.1 A 9.75 million particle gravitational N-body simulation.

We compute the statistics below in manner identical to that described in the preceding section. On April 18–30 1997, we ran a simulation with $9,753,824$ particles on the 16 processors of Loki for 750 timesteps. An image of this simulation is shown in Figure 2. The simulation was of a spherical region of space 100 Mpc (Megaparsec) in diameter; a region large enough to contain a few hundred thousand typical galaxies. The region inside a sphere of diameter 100 Mpc was calculated at high mass resolution, while a buffer region of 50 Mpc with a particle mass 8 times higher was used around the outside to provide boundary conditions. The initial conditions were extracted from a 134 million point initial dataset, calculated using a 512^3 point 3-d FFT on Loki, from a Cold Dark Matter power spectrum of density fluctuations. Overall, the simulation carried out 6.6×10^{14} floating point operations. We created 33 data files totaling just over 10 Gbytes. A single data file from this simulation is 312 Mbytes is size, and they were written striped over the 16 disks in the system, obtaining an aggregate I/O bandwidth of well over 50 Mbytes/sec.

We quote two performance results from this simulation. The simulation up to April 30 required the computation of 1.97×10^{13} interactions over a wall clock time of 850000 seconds (236 hours, just shy of ten days), for an overall throughput of 879×10^6 floating point operations per second **(879 Mflops)**. This

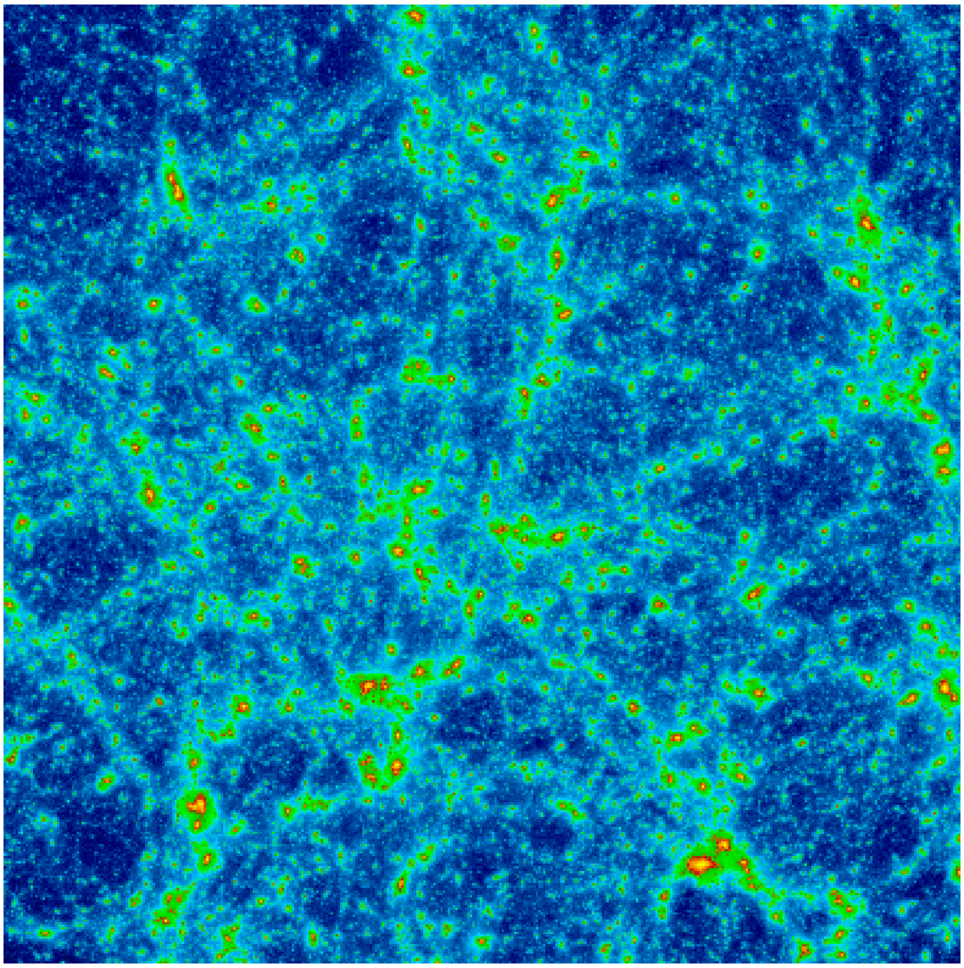

Figure 1: The image shows an intermediate stage of a gravitational N-body simulation using 322 million particles, computed on ASCI Red. Using initial conditions derived from fundamental theories, the evolution of matter in the universe is simulated from a time shortly after the Big Bang through its non-linear evolution to the present epoch. The region shown is about 400 million light years across, and the color of each pixel represents the logarithm of the projected particle density along the line of sight through the periodic computational volume. The particles have formed clumps which represent dark matter halos, which are the sites where galaxy formation occurs. .

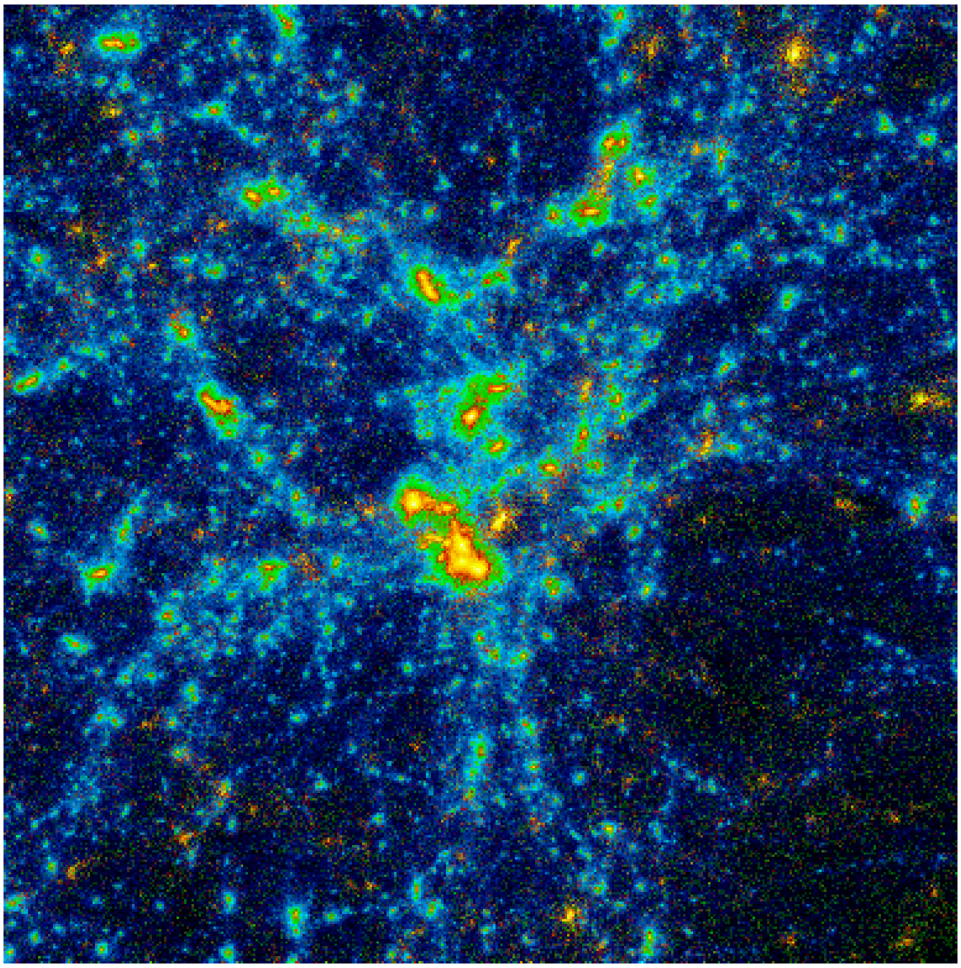

Figure 2: The image shows an intermediate stage of a gravitational N-body simulation using 9.7 million particles, computed on Loki. The region shown is about 150 million light years across.

simulation was tuned to the greatest extent possible to obtain useful scientific data (as opposed to the largest number of Gflops). We quote a price/performance for this 10 day simulation of $58/Mflop. We quote a better performance result during the initial 30 timesteps of the same simulation. We computed 1.15×10^{12} interactions in 36973 seconds of wall-clock time, for an overall throughput of 1.186×10^{9} floating point operations per second **(1.19 Gflops)**.

Between April 25 and May 8, the code ran continuously for 13.5 days, with no restarts. This accomplishment is a clear evidence that Loki is capable of performing large calculations reliably. The entire simulation of over 1000 timesteps was completed on May 8, after a total computation of 1.2×10^{15} floating point operations (1.2 Petaflops). This simulation on Loki consisted of as many operations as any single simulation we had performed on any parallel supercomputer prior to April 1997.

6.2 A 360 thousand vortex-particle fluid dynamics simulation

To demonstrate the flexibility of our code, we have simulated the fusion of two vortex rings using the vortex particle method on Hyglac. The simulation started with 57,000 vortex particles in two "smoke rings". During the computation, the particles are occasionally "remeshed" in order to satisfy the core-overlap condition. This creates additional particles, so that by the end of the 340 timestep simulation, there were 360,000 vortex particles. Each vortex particle interaction is substantially more complex than a gravitational interaction, and directly counting the flops by looking through the code is prone to error. Through the use of the Pentium Pro hardware performance monitors, we have directly counted the actual number of floating point operations involved while the code was running. Measuring single processors during a parallel run, the code maintains somewhat over 65 Mflops per processor during the majority of the computation, with the tree building, domain decomposition and I/O contributing an overhead of less than 10 percent. Thus, the overall throughput of the code running on 16 processors is close to 950 Mflops.

6.3 Loki and Hyglac at Supercomputing '96

Loki and Hyglac were physically present at Supercomputing '96 in Pittsburgh. During that time period, with the addition of 32 Fast ethernet cards and 16 cables the two machines performed a 10 million particle treecode benchmark at the rate of 2.19 Gflops. The cost of the combined system (including the $3000 of additional hardware to connect the two machines) was $103k. Thus, we quote a price/performance result for this benchmark of $47/Mflop, or equivalently, 21 Gflops per million dollars. Unfortunately, although these systems were available prior to last year's price/performance award, they were not available at the entry deadline, which was six months prior to the conference. We note that this result and the application results above are about a factor of three better than last years Gordon Bell price/performance winner. If we were to construct a similar system today, it would have a price/performance almost a factor of two better than that which we quote here.

7 NAS Parallel Benchmarks

The N-body problems described above are not the only ones on which these types of machines perform well. The results shown in Table 3 use the NAS Parallel benchmarks version 2 [39]. These benchmarks, based on Fortran 77 and the MPI standard, are intended to approximate the performance a typical user can expect for a portable parallel program on a distributed memory computer.

	Loki		ASCI	SGI
	PGI	GNU	Red	Origin
BT	354.6	331.4	445.5	925.5
SP	255.5	224.5	334.8	957.0
LU	428.6	403.7	490.2	1317.4
MG	296.8	267.1	363.7	1039.6
FT	177.8			648.2
EP	8.9	12.7	7.1	68.7
IS	14.8	14.6	38.0	33.9

Table 3: Sixteen processor performance (Mops) for Class B NPB 2.2 benchmarks. Data from Loki with the Portland Group (PGI) pgf77 Rel 1.3-3 Fortran 77 compiler, the GNU 2.7.2.f.1 compilers, and the ASCI Intel Teraflops system with PGI Rel 1.3-4a, and an SGI Origin 2000 are presented. Results for Loki and ASCI Red were obtained by the authors; results for the SGI were obtained from the NPB Web page [40].

NC	BT	SP	LU	FT	MG	IS
1		19	31			2.5
2			59			3.2
4	94	71	118	73	78	5.7
8			222	134	161	9.3
9	202	122				
16	358	242	453	250	281	15.0

Table 4: Data for the NPB 2.2 Class A benchmarks on Loki. The first column denotes the number of processors, and the data are Mflops/sec (Mops/sec for IS). This data is plotted in Figure 3.

8 Comparing machines

We can compare Loki to the results obtained on 16 nodes of the ASCI Red system (Janus, which incidentally is binary compatible with Loki at the object file level). Janus has exactly the same Pentium Pro processor, amount of memory, and compiler as Loki. The differences were the network on Janus is about 15 times faster (160 Mbytes/sec), the latency is less (60 microseconds round-trip), and the memory bandwidth is higher. Overall Janus has an advantage at the 16 processor level that ranges from 10%-30% (Table 3). We estimate that roughly half of this improvement comes from the better memory bandwidth of the Janus nodes. Thus, we conclude (surprisingly!) that using switched fast ethernet instead of an exotic networking technology appears to have a fairly small effect on performance for a 16 processor machine on the NAS benchmarks (with the exception of the message bandwidth hungry IS benchmark). The effect of improved memory bandwidth from the interleaved memory on Janus is practically the same as its network advantage. We have observed that memory bandwidth is critical to both floating point performance, as well as network performance. It is for this reason that using multiple processors within each node (SMP) is unlikely to be a good idea for many applications. It simply makes the shortage of memory bandwidth even worse.

We are aware that current machines such as the SGI Origin series can approach the price/performance figure above, but any such machine with enough memory and disk space to perform the simulations reported here would come nowhere near our price/performance. Until the summer of 1997, it was impossible to obtain memory for the Origin series for less than \$40/Mbyte. At list price, *just the memory* to do this problem on an

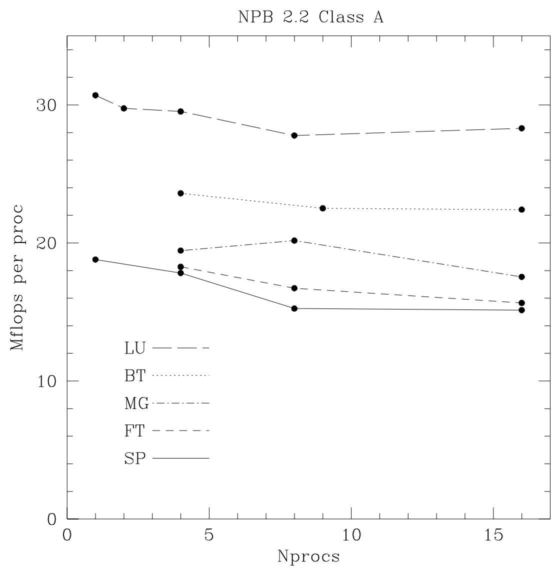

Figure 3: Scaling of the NAS Class A benchmarks on Loki.

Origin machine would exceed the total cost of Loki or Hyglac. It is currently possible to obtain unlicensed Origin memory for about $12 Mbyte, while SGI approved third party memory is $24 per Mbyte. This may be compared to the parity DRAM in Loki, which is currently less than $4 per Mbyte.

Taking a look at the vendor reported prices in Nov. 1996 for NPB Class B capable machines as reported in [41] ($960,000 for a 24 processor Origin 2000, $3,520,000 for a 64 processor IBM SP-2 P2SC, and $580,000 for a DEC AlphaServer 8400 5/440), we find that the price/performance of our commodity parallel processor (CPP) machines on the NPB 2.2 MPI/Fortran 77 benchmarks are better by a factor of three or more. For example, the time for a 64 processor SP-2 P2SC-120 machine to run the NPB version 2 Class B BT benchmark is 118 seconds, beating Loki in speed by a factor of 17, at a cost about 60 times higher. The SGI runs the benchmark in 471 seconds, 4.2 times faster, at a cost 16 times higher.

9 Conclusion

We feel the simulations reported above which were computed on Loki and Hyglac are clearly of the "supercomputer" class. Both simulations required nearly 2 Gbytes of memory, and the gravitational simulation required 1.2×10^{15} floating point operations. If these machines are not "supercomputers", then one must admit that a state-of-the-art problem which won the Gordon Bell Performance Prize five years ago is now a mundane problem solvable on run-of-the-mill computers. We believe that we have demonstrated that moderately parallel computers constructed from commodity parts which can perform at a level of 10-100 times the capability of a desktop machine deserve a respected position in the hierarchy of computer hardware that a computational scientist should consider for production simulations.

For ASCI Red, the real metric for evaluating gravitational N-body simulations is not Teraflops, but how many particles are updated per second, at an accuracy sufficient to accurately represent the physics involved. For our 322 million particle simulation, we can update 3 million particles per second on 3400 nodes with an RMS force accuracy of better than 10^{-3}. Using an N^2 algorithm on 3400 nodes we can update 52 particles per second. We conclude that the ASCI Teraflops system (in its partial configuration of April 1997) exceeds the performance of the GRAPE system (which won the Gordon Bell performance prize last year) when using an identical solution algorithm, and exceeds the performance that the GRAPE system could achieve on this problem by a factor of roughly one hundred thousand through the use of a sophisticated treecode algorithm. For this problem, our treecode on the Intel Teraflops system is equivalent to special purpose hardware running an N^2 algorithm at 25 million Gigaflops, or 25 Exaflops. We make this point in order to firmly emphasize the advantages of a good algorithm.

Tree-based codes can solve a very general class of problems that can be expected to grow in importance as the need for spatial adaptivity becomes necessary for the simulation of ever more difficult problems. We believe that our most important demonstration is that a good algorithm can achieve an increase of many orders of magnitude in performance, while hardware is limited to a mere factor of ten every five years.

References

[1] D. J. Becker, T. Sterling, D. Savarese, J. E. Dorband, U. A. Ranawake, and C. V. Packer. BEOWULF: A parallel workstation for scientific computation. In *Proceedings of the 1995 International Conference on Parallel Processing (ICPP)*, pages 11–14, 1995.

[2] S. Elbert. Pentium pro cluster workshop. `http://www.scl.ameslab.gov/workshops/`.

[3] H. Dietz. Linux parallel processing sites. `http://yara.ecn.purdue.edu/\~{}pplinux/Sites/Index.html`.

[4] Beowulf project. `http://cesdis.gsfc.nasa.gov/linux-web/beowulf/beowulf.html`.

[5] M. S. Warren and M. P. Goda. Loki – commodity parallel processing. `http://loki-www.lanl.gov/`.

[6] J. Lindheim, J. K. Salmon, and T. Sterling. CACR's Beowulf machine. `http://www.cacr.caltech.edu/research/Beowulf/`.

[7] Grendel: The Clemson Beowulf cluster. `http://ece.clemson.edu/parl/grendel.htm`.

[8] A Beowulf-class parallel supercomputer for Drexel University. `http://einstein.drexel.edu/beowulf/Beowulf.html`.

[9] The HERMES linux production cluster. `http://www-hermes.desy.de/ww/cluster.html`.

[10] AnimalFarm: The MPII linux cluster project. `http://bigbrother.ag1.mpi-sb.mpg.de/`.

[11] Megalon—a massively parallel computer resource. `http://megalon.ca.sandia.gov/`.

[12] The Argonne Chicago pile 6 project. `http://www.mcs.anl.gov/ntcluster/`.

[13] ALICE 2: A cluster of 16 200 mhz pentium pro's. `http://www.scl.ameslab.gov/Resources/cluster2.html`.

[14] The Whitney project. `http://parallel.nas.nasa.gov/Parallel/Projects/Whitney/`.

[15] ASCI red ultracomputer. `http://www.sandia.gov/ASCI/Red.htm`.

[16] Intel Corporation. The DOE accelerated strategic computing initiative TFLOPS system. `http://www.ssd.intel.com/sc95booth/tflop2.html`.

[17] M. S. Warren and J. K. Salmon. Astrophysical N-body simulations using hierarchical tree data structures. In *Supercomputing '92*, pages 570–576, Los Alamitos, 1992. IEEE Comp. Soc.

[18] Charles Spurgeon. Welcome to Charles Spurgeon's ethernet page. `http://wwwhost.ots.utexas.edu/ethernet/`.

[19] T. Shanely and D. Anderson. *PCI System Architecture*. MindShare, Inc., Richardson, TX, 1992.

[20] The Linux documentation project. `http://sunsite.unc.edu/mdw/linux.html`.

[21] GNU's Not Unix! - the GNU project and the free software foundation (FSF). `http://www.gnu.org/`.

[22] Red Hat Software, Inc. `http://www.redhat.com/`.

[23] M. S. Warren, D. J. Becker, M. P. Goda, J. K. Salmon, and T. Sterling. Parallel supercomputing with commodity components. In H. R. Arabnia, editor, *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97)*, pages 1372–1381, 1997.

[24] J. Barnes and P. Hut. A hierarchical O(NlogN) force-calculation algorithm. *Nature*, 324:446, 1986.

[25] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comp. Phys.*, 73:325–348, 1987.

[26] M. S. Warren, P. J. Quinn, J. K. Salmon, and W. H. Zurek. Dark halos formed via dissipationless collapse: I. Shapes and alignment of angular momentum. *Ap. J.*, 399:405–425, 1992.

[27] W. H. Zurek, P. J. Quinn, J. K. Salmon, and M. S. Warren. Large scale structure after COBE: Peculiar velocities and correlations of cold dark matter halos. *Ap. J.*, 431:559–568, 1994.

[28] M. S. Warren and J. K. Salmon. A portable parallel particle program. *Computer Physics Communications*, 87:266–290, 1995.

[29] J. K. Salmon, M. S. Warren, and G. S. Winckelmans. Fast parallel treecodes for gravitational and fluid dynamical N-body problems. *Intl. J. Supercomputer Appl.*, 8:129–142, 1994.

[30] G. S. Winckelmans, J. K. Salmon, M. S. Warren, and A. Leonard. The fast solution of three-dimensional fluid dynamical N-body problems using parallel tree codes: vortex element method and boundary element method. In *Seventh SIAM Conference on Parallel Processing for Scientific Computing*, pages 301–306, Philadelphia, 1995. SIAM.

[31] J. Makino, M. Taiji, and D. Sugimoto. GRAPE-4 - a massively-parallel special-purpose computer for collisional N-body simulations. *Ap. J.*, 480:432–446, 1997.

[32] J. K. Salmon, P. J. Quinn, and M. S. Warren. Using parallel computers for very large N-body simulations: Shell formation using 180k particles. In A. Toomre and R. Wielen, editors, *Proceedings of 1989 Heidelberg Conference on Dynamics and Interactions of Galaxies*. Springer-Verlag, New York, 1990.

[33] M. S. Warren and J. K. Salmon. A parallel hashed oct-tree N-body algorithm. In *Supercomputing '93*, pages 12–21, Los Alamitos, 1993. IEEE Comp. Soc.

[34] W. H. Zurek, M. S. Warren, P. J. Quinn, and J. K. Salmon. The second coming of cold dark matter? In F. R. Bouchet, editor, *Proceedings of the 9th IAP Meeting*, Paris, France, 1993. IAP.

[35] Grégoire S. Winckelmans, J. K. Salmon, and M. S. Warren. The fast solution of three-dimensional boundary integral equations in potential flow aerodynamics using parallel and sequential tree codes. 1995. (technical report).

[36] M. S. Warren and J. K. Salmon. A parallel, portable and versatile treecode. In *Seventh SIAM Conference on Parallel Processing for Scientific Computing*, pages 319–324, Philadelphia, 1995. SIAM.

[37] J. K. Salmon and M. S. Warren. Skeletons from the treecode closet. *J. Comp. Phys.*, 111:136–155, 1994.

[38] Alan H. Karp. Speeding Up N-body Calculations on Machines without Hardware Square Root. *Scientific Programming*, 1:133–140, 1993.

[39] D. Bailey, T. Harris, W. Saphir, R. van der Wijngaart, A. Woo, and M. Yarrow. The NAS parallel benchmarks 2.0. Technical Report NAS-95-020, NASA Ames Research Center, 1995.

[40] NPB 2 detailed results: Graphs and raw data 04/21/97. `http://science.nas.nasa.gov/Software/NPB/`.

[41] Subhash Saini and David H. Bailey. NAS parallel benchmark (version 1.0) results. Technical Report NAS-96-018, NASA Ames Research Center, 1996.