Avalon: An Alpha/Linux Cluster Achieves 10 Gflops for $150k

Finalist for the 1998 Gordon Bell Price/Performance Prize

Michael S. Warren * Timothy C. Germann † Peter S. Lomdahl ‡ David M. Beazley §
John K. Salmon ¶

Abstract

As an entry for the 1998 Gordon Bell price/performance prize, we present two calculations from the disciplines of condensed matter physics and astrophysics. The simulations were performed on a 70 processor DEC Alpha cluster (Avalon) constructed entirely from commodity personal computer technology and freely available software, for a cost of 152 thousand dollars.

Avalon performed a 60 million particle molecular dynamics (MD) simulation of shock-induced plasticity using the SPaSM MD code. The beginning of this simulation sustained approximately 10 Gflops over a 44 hour period, and saved 68 Gbytes of raw data. The resulting price/performance is $15/Mflop, or equivalently, 67 Gflops per million dollars. This is more than a factor of three better than last year's Gordon Bell price/performance winners. This simulation is similar to those which won part of the 1993 Gordon Bell performance prize using a 1024-node CM-5. This simulation continued to run for a total of 332 hours on Avalon, computing a total of 1.12×10^{16} floating point operations. This puts it among the few scientific simulations to have ever involved more than 10 Petaflops of computation.

Avalon also performed a gravitational treecode N-body simulation of galaxy formation using 9.75 million particles, which sustained an average of 6.78 Gflops over a 26 hour period. This simulation is exactly the same as that which won a Gordon Bell price/performance prize last year on the Loki cluster, at a total performance 7.7 times that of Loki, and a price/performance 2.6 times better than Loki. Further, Avalon ranked at 315th on the June 1998 TOP500 list, by obtaining a result of 19.3 Gflops on the parallel Linpack benchmark.
Keywords: *Molecular Dynamics, SPaSM, N-body problem, treecode, Beowulf, price/performance*

1 Introduction

Building upon the foundation of the BEOWULF project [4] and our own success with Loki [9], it has become possible to construct high-performance computers entirely out of commodity components and open source software, thus obtaining a significant price/performance advantage over typical parallel machines. Last year Loki and Hyglac, clusters of 16 Pentium Pro processors, were the first such machines to win a Gordon Bell price/performance prize [14]. This year, changing to the the DEC Alpha microprocessor (which is also found in the Cray T3E series) and using a more advanced fast ethernet switch, we have improved total performance by almost a factor of ten, and improved price/performance by over a factor of three.

In 1992, Warren and Salmon were awarded a Gordon Bell Performance Prize [10] for "Astrophysical N-body Simulations Using Hierarchical Tree Data Structures." In 1993, Lomdahl and Beazley were awarded a Gordon Bell Performance Prize [8] for "50 GFlops Molecular Dynamics on the Connection Machine 5." It

*Theoretical Astrophysics, Los Alamos National Laboratory, Los Alamos, NM 87545, Email: *msw@lanl.gov*
†Condensed Matter Theory, Los Alamos National Laboratory, Los Alamos, NM 87545, Email: *tcg@lanl.gov*
‡Condensed Matter Theory, Los Alamos National Laboratory, Los Alamos, NM 87545, Email: *pxl@lanl.gov*
§Department of Computer Science, University of Chicago, Chicago, IL 60637, Email: *beazley@cs.uchicago.edu*
¶CACR, California Institute of Technology, Mail Code 206-49, Pasadena, CA 91125, Email: *johns@cacr.caltech.edu*

is now possible to run similar simulations on a machine constructed out of mail-order parts and free software for a cost of $152k.

2 Avalon

As a co-operative venture at the Los Alamos Center for Nonlinear Studies (CNLS) and Theoretical Division, Avalon was constructed from 70 nodes for a total cost of $152,175 as described in Table 1. All of the operating system software (RedHat Linux 5.0), software tools (GNU) and compilers (egcs-1.02) used for these results are freely available. MPI was used for the message passing layer, using our own TCP socket based implementation of the basic MPI functions (SWAMPI). The individual nodes were purchased from Carrera Computers and delivered to Los Alamos on April 10 completely assembled with the operating system already installed and configured. The ethernet equipment was bought under government contract, but a quick search found at least three Internet mail-order companies offering this hardware to the public for nearly the same price as we paid, and at least one offered it for less. The only labor required to complete assembly of the cluster was unpacking the nodes from their shipping boxes and the attachment of power and network cables. This took 28 man-hours of labor, which we have included in the price at $100/hour. The machine was operational on April 13, three days after delivery. A photo which shows forty of the processors is presented below. Further information and pictures of the machine are available at `http://cnls.lanl.gov/avalon`.

Qty.	Price	Ext.	Description
70	1701	119070	DEC Alpha 164LX 533 MHz 21164A, with 2x64Mb SDRAM DIMM ECC memory (128 Mbyte/node), Quantum 3240 Mbyte IDE Hard Drive, Kingston 100 Mb Fast Ethernet PCI Card, cables, assembly, Linux install, 3 year parts/labor warranty
2	6027	12054	3Com SuperStack II 3900, 36-port Fast Ethernet
4	968	3872	Gigabit uplink modules for 3900s
1	10046	10046	3Com SuperStack II 9300, 12-port Gigabit Ethernet
3	1055	3165	Cyclades Cyclom 32-YeP serial concentrators
70	10	700	Serial cables (20 ft)
4	117	468	Shelving
28	100	2800	Final assembly labor
Total		$152,175	$2174 per node 1.066 Gflops peak per node

Table 1: Avalon architecture and price.

We emphasize that the simulations we report here are only a few of the programs which Avalon has successfully run. In the three months since the submission of our Gordon Bell entry, Avalon has begun to perform as a general-purpose computational resource. There are currently about 35 user accounts on the machine, with more added every week. Avalon's initial success has led to additional monetary resources, and Avalon will have expanded to 140 processors and 40 Gbytes of memory by the time this paper is published. The memory upgrade was encouraged by the fact that the cost of memory has fallen by a factor of three since Avalon was initially constructed.

The machine has been more reliable than even our initially optimistic expectations. Since the initial burn-in, there has not been a single hardware component replaced on any of the nodes (a three month period). There have been three cases of a node becoming unresponsive and requiring a reboot, due in one

Figure 1: Michael Warren is shown in front of 40 of the Avalon processors.

case to a transient disk error, and for unknown reasons in the other two cases. We have rebooted the entire system on two occasions for a scheduled kernel upgrade, and in one case the system was shut down for 12 hours due to the failure of the air-conditioning system in the machine room. This reliability reinforces our experience with Loki, in which 8 of the 16 nodes have been running without a scheduled or unscheduled reboot for 455 days.

As an unexpected result of the use of a complete and efficient time-sharing operating system on each node, a trivially parallel cryptographic application was able run at very low priority on Avalon, at the same time as parallel scientific applications were running. The cryptographic application was able to utilize the cycles that otherwise would have been lost to load-imbalance, with a less than 5% effect on the run-time of the primary application. The end result was that Avalon made the largest contribution of any group in the world to the solution of the Certicom Cryptographic Challenge. Avalon participated in the team led by Robert J. Harley of the Institut National de Recherche en Informatique et Automatique (INRIA), France. The solution to the ECC2K-95 problem was found after 21.6 trillion elliptic curve operations, carried out in 25 days by 47 people. The $4000 prize for finding the solution was donated to the Free Software Foundation.

3 The SPaSM molecular dynamics code

SPaSM (Scalable Parallel Short-range Molecular dynamics) is a message-passing C code originally written [1] for the CM-5, and was recognized with a 1993 Gordon Bell performance prize for achieving a sustained rate of 50 Gflops in simulating 131 million atoms on a 1024-node CM-5 [8]. Since then, we have greatly increased the portability of the code by replacing CMMD and CM I/O calls with a set of wrapper functions, removed all assembler code, developed a visualization library, and combined these elements into an interactive package using the Python scripting language (see, e.g., [2, 3]).

The general molecular dynamics algorithm used in SPaSM has been presented in [1]; we outline a brief overview and the important features here. The basic steps in an MD timestep involves the following three elements:

1. Calculations of all forces on each atom.
2. Numerical integration of the equations of $F = ma$.
3. Updating the data structures.

In addition, we will periodically perform I/O to dump particle data and calculate important quantities like energy, etc. By far, the most time consuming step is number 2 above, it accounts for more than 90% of the total CPU time spent. We deal exclusively with this step below.

The data-layout used in SPaSM is based on an approach where space is decomposed into domains that are assigned to each processing node of the cluster. Each node then further subdivides its domain into a large collection of small cells. The size of the cells is chosen equal to the maximum interaction distance for the interatomic potential, R_{max}. Atoms are assigned to a particular processing node and cell according the atom's coordinates. This particular approach allows forces to be readily calculated since it places atoms that are physically close into the same cell or the neighboring cell. To calculate the total force on a given atom we simply look at all of the atoms in the same cell and those in the neighboring cell. When neighboring cells are on different processing nodes, we use message passing to exchange the data needed to complete the force calculation.

The critical code in the force calculation loop is written entirely in C, and hand-optimized for modern superscalar architectures by loop unrolling, macro expansion, and careful use of `register` variables. We have used SPaSM to study a number of materials science problems at an experimentally inaccessible atomistic level, including fracture mechanisms [15], dislocation interactions [16], and the mechanisms of shock-induced plasticity [6]. Further information about SPaSM and simulations carried out with it are available at `http://bifrost.lanl.gov/MD/MD.html`.

3.1 Timings and performance

To demonstrate the scalability of SPaSM on Avalon, we have carried out a series of short (10 timestep) simulations with various numbers of particles and processors. We use the usual Lennard-Jones 6-12 potential, truncated at a cutoff distance of $R_{max} = 4\sigma$, where σ is the nearest-neighbor spacing. The particles are in an fcc crystal structure initially moving at a uniform velocity into an infinitely massive piston face ("momentum mirror"), thus generating a shock wave through the sample [6].

The average wall clock times per timestep (including force calculation, timestep integration, and particle redistribution) are shown in Table 2, along with the corresponding Gflop rates obtained by counting the actual number of floating point operations in both the force and integration routines. (Add, subtract, multiply, and comparison are each counted as one operation, and divide as five operations.)

Number of Particles	Number of Processors 16	32	64	70
1,000,000	4.94 (2.75)	2.86 (5.22)	1.66 (9.89)	1.47 (10.93)
2,000,000	9.39 (2.79)	4.89 (5.57)	3.07 (10.27)	2.76 (10.82)
4,000,000	16.84 (2.99)	9.12 (5.73)	4.76 (11.43)	4.68 (11.70)
8,000,000	31.59 (3.07)	17.17 (5.87)	8.92 (11.74)	8.70 (11.73)
16,000,000	62.28 (3.12)	31.69 (6.12)	17.47 (12.00)	17.03 (11.99)
32,000,000		61.53 (6.30)	31.35 (12.37)	31.11 (12.47)
64,000,000			61.39 (12.64)	60.48 (12.83)

Table 2: Average time per timestep in seconds, with Gflops in parentheses.

To give an idea of how large the fraction of time spent in the force calculation is, the 64 million atom calculation on 70 nodes required 57.27 s to compute forces, 1.27 s for the (leapfrog Verlet) integration, and 1.97 s to redistribute particles after the integration step. The force calculation alone, which involves message-passing of cells of particles between processors (8975 message passing calls per processor per timestep), performs at an overall rate of **13.52 Gflops**, which drops to the quoted value of 12.83 Gflops after the time spent in the integration and redistribution steps is included. We measure a total message-passing time of 12.65 s, 21% of the total 60.48 s. As the number of particles is reduced this fraction increases, to 38% for the 1 million particle run. Despite this, the overall performance is still nearly 11 Gflops, and the iteration time of 1.47 s represents a speedup of 3.4 over the 16 node run, or 77% parallel efficiency. For larger problems the parallel efficiency is between 80 and 90%, demonstrating that the near-perfect scalability of the SPaSM code found on the CM-5 [8] is reasonably well preserved. For an additional point of reference, an identical 64 million particle calculation on 70 nodes of a Cray/SGI Origin 2000 (with 195 MHz R10000 CPUs) requires 76.22 seconds per timestep, or 10.18 Gflops (26% slower than Avalon). The same code was used in both cases, with use of the best available compiler optimization flags. Erring on the side of SGI, we assume that with 250 Mhz processors an Origin 2000 on this code would perform at the same rate as Avalon. Taking the ratio of the list prices, we find that Avalon has a price/performance advantage of about 12 for this code. A discount from list price might lower this factor to about 9. On the other hand, these figures do not take into account the the $125k/yr hardware maintenance contract for the Origin.

3.2 A 60.8 million atom shock wave simulation

To demonstrate the capability of Avalon to sustain an actual large-scale scientific calculation, we carried out a simulation of 60.8 million atoms with the same initial conditions as used for the timing measurements. This simulation is similar to a series of 10 million atom simulations recently carried out [6], but with a cross-sectional area four times as large (200×200 fcc unit cells). We ran on 68 nodes for a total of 13600 timesteps, which required nearly 2 weeks (332 hours) of CPU time, spread over 19 separate runs between April 24 and June 11. (Only 2 of these runs ended abnormally; the remainder were intentionally short enough to only run over a single night or weekend in most cases.)

As this was an actual production run, we carried out the usual "extras," such as computing the potential and kinetic energies every 10 or 20 timesteps, and checkpointing the data every 100 timesteps. Additionally, GIF images were periodically generated (every 100 timesteps during the early part of the simulation, and every 10 timesteps for the remainder) using the built-in visualization library [2, 3]. The checkpointing was carried out independently by each processor, writing out its particle data (50 MB on average) to the local disk. The 68-node subset of processors was also changed during the simulation, by manually transferring

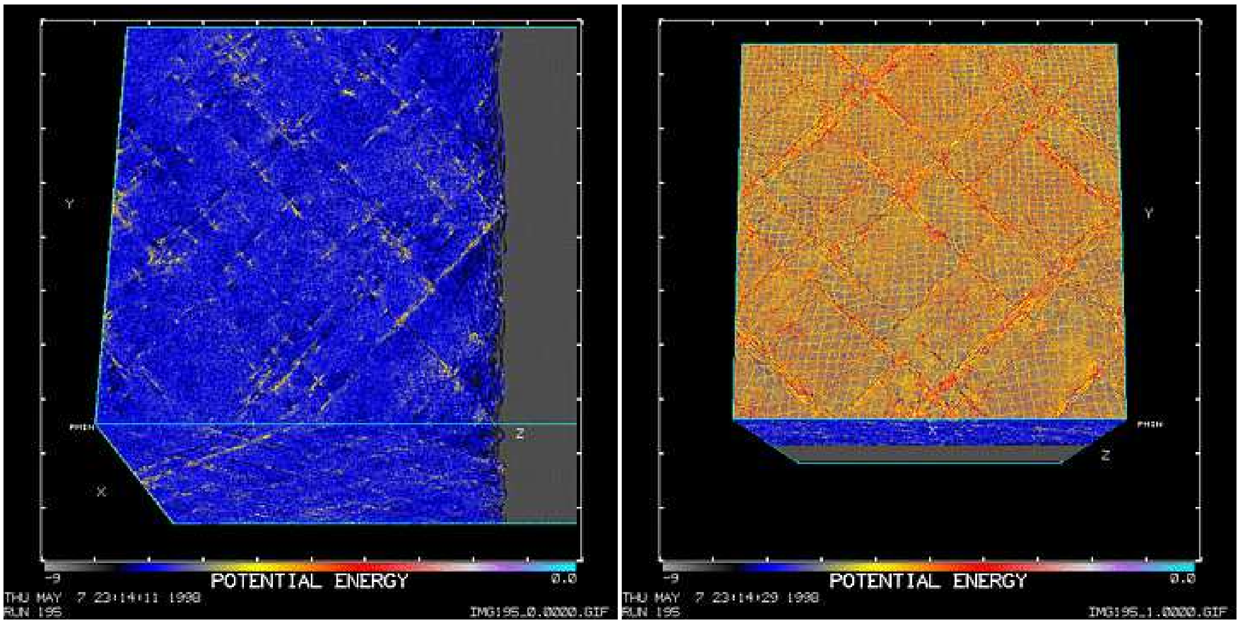

Figure 2: Side (left) and end (right) views of the sample at the piston face, after several stacking faults have been generated. Animations of the simulation are available at `http://bifrost.lanl.gov/MD/Avalon_movies/`

checkpoint files from the "old" node(s) to the "new" one(s). Despite all of these add-ons, the simulation still ran at an average of 88 seconds per timestep, with a total of 1.12×10^{16} floating point operations over a wall clock time of 1.19×10^6 seconds. We thus find a sustained throughput of **9.4 Gflops** and a price/performance of $16/Mflop. It also bears pointing out that this total of more than 11 Petaflops is greater than the number of operations carried out on ASCI Red in last year's Gordon Bell Performance Prize winner [14].

During the early part of the simulation (the first 2000 timesteps), when the sample was more uniform while energies and GIF images were calculated less frequently (every 20th and 100th timestep, respectively), the average time per timestep was 79 seconds. This part of the calculation required a total of 1.56×10^{15} floating point operations over a wall clock time of 1.58×10^5 seconds, giving a sustained throughput of **9.9 Gflops** and a price/performance of $15/Mflop. (Another cause for the slightly reduced performance of the full run was that during a two-week period when 5100 timesteps were carried out, the elliptic curve solver mentioned in section 2 was running at low priority on all 70 nodes.)

Selected snapshots and movies (reduced versions of the 600×600 and 1000×1000 GIF images generated during the simulation) are shown in Figures 2 and 3. Figure 2 shows two views near the "momentum mirror" during the early part of the simulation As the shock wave propagates, stacking faults are generated along all four of the available $\{111\}$ slip planes, and the shock front becomes increasingly non-planar. After the shock wave passes completely through the sample, it finally reaches the free end opposite the momentum mirror and generates stacking faults and some ejection of atoms and clusters of atoms at this surface, as seen in Figure 3.

4 The Hashed Oct-Tree Library

Our parallel N-body code has been evolving for several years, and on many platforms. This original version of the code was abandoned after it won a Gordon Bell Performance Prize in 1992 [10], due to various

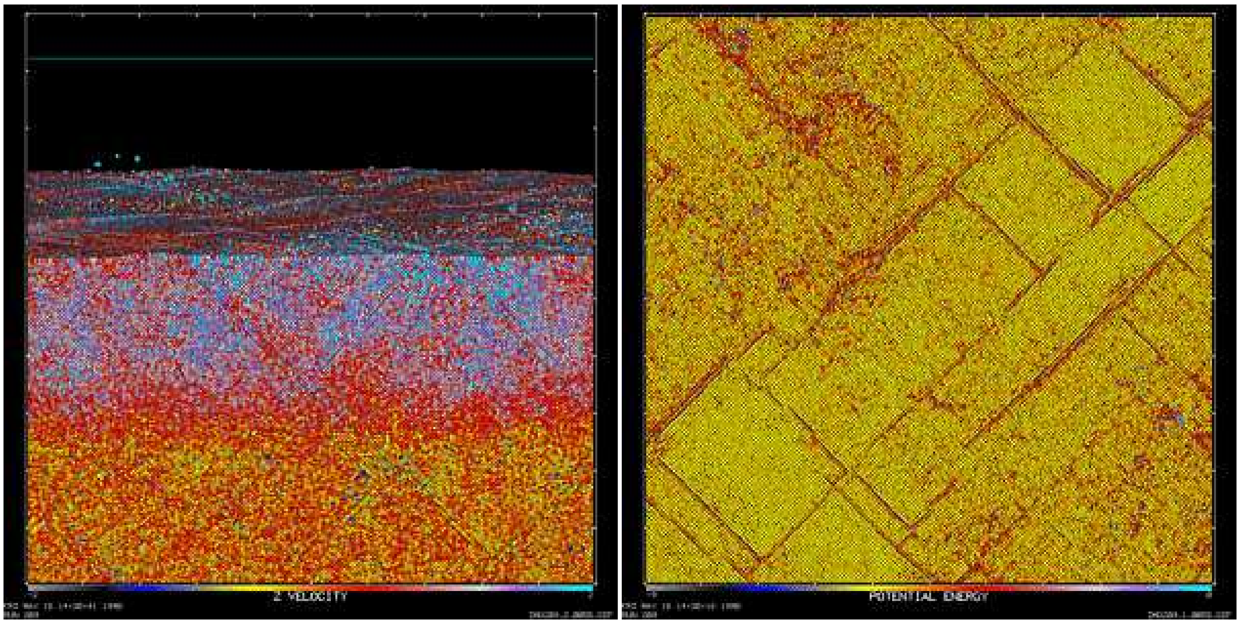

Figure 3: Side (left) and end (right) views of the free end of the sample, after the shock has propagated completely through and been reflected from the free end.

flaws inherent in the code, which was ported from a serial version. A new version of the code was initially described in [11], and further simulations and other details are reported in [13, 12].

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 Alpha microprocessor 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 [7]. 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.

4.1 A 9.75 million particle gravitational N-body simulation.

On April 28–29 1998, we ran a simulation with $9,753,824$ particles on the 70 processors of Avalon for 700 timesteps. 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. Overall, the simulation carried out 6.4×10^{14} floating point operations.

We quote two performance results from this simulation. The entire simulation required the computation of 1.68×10^{13} interactions over a wall clock time of 94308 seconds (26.2 hours), for an overall throughput of 6.78×10^9 floating point operations per second **(6.78 Gflops)**. This 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 26 hour simulation of \$22/Mflop. The initial 30 timesteps of the simulation (including reading the initial data and saving the first checkpoint) obtained **8.55 Gflops**.

4.2 Other treecode simulations

To obtain a direct comparison to other parallel machines using the same optimized code, we ran a short simulation using 10 million particles randomly distributed in a sphere, using a force accuracy of one part in 10^3. The result for Avalon was 10.1 Gflops, a 64 processor 195 Mhz Origin 2000 obtained 13.1 Gflops, a 512 processor CM-5 obtained 14.1 Gflops (with assembly code), a 256 processor T3D obtained 7.9 Gflops, a 64 processor SP-2 66 Mhz-wide machine obtained 9.5 Gflops and 4096 processors of ASCI Red obtained 164.3 Gflops.

We have recently developed a new code using the treecode infrastructure. It solves the scalar Helmholtz equation using the fast multipole method. An example of this type of problem would be to calculate the fields due a large set of sources emitting electromagnetic radiation. Radar scattering from vehicles such as planes can be efficiently calculated by discretizing the surface into small panels and using an algorithm of this type. The code consists of the treecode, which handles domain decomposition and tree traversal, and a series of Fortran subroutines which do most of the floating point arithmetic. On a problem which consists of a spherical distribution of 131072 sources, 32 Avalon processors can compute the fields in 367 seconds, while a 32 processor Origin 2000 takes 452 seconds.

5 Price/Performance

The problems described here are not the only ones on which these types of machines perform well. To enable additional price/performance comparisons with commercial machines using known benchmarks, we have computed results for parallel Linpack and the NAS Class B benchmarks. While these results are not directly relevant to the price/performance prize, we include them in order to convince the reader that this is a general purpose parallel machine, which has superior price/performance on a wide range of problems. Below we list machines of capability roughly similar to that of Avalon on the parallel Linpack benchmark. All data were taken from [5]. We do not have precise figures for the current list prices of these machines, but we estimate they are in the range of 1-4 million dollars or more, resulting in a Linpack price/performance advantage for Avalon of a factor 5-20.

Computer	Num Procs	Rmax(GFlops)	Nmax(order)	Rpeak(GFlops)
IBM SP2 (160 MHz)	64	29.45	27500	41
CRAY T3E-1200 (600 MHz)	32	25.98	42240	38
Cray T3D 256 (150 MHz)	256	25.3	40960	38
Sun Ultra HPC 10000(250 MHz)	52	21.68	19968	26
Cray C90 (240 MHz)	16	20.65	13312	15
DEC AlphaServer 8400 5/612 (625 MHz)	40	20.54	24552	50
SGI Origin 2000 (195 MHz)	64	20.1	40000	25.0
Avalon	68	19.33	30464	72.5

Table 3: Parallel Linpack performance.

We have produced results for the NAS Parallel Benchmarks, version 2.2, Class B, which demonstrate a price/performance advantage over what we believe to be the closest competitors (the Cray/SGI T3E and Origin series) of a factor of three or more. Table 4 shows a comparison between Avalon and a 64 processor T3E and Origin 2000 running these benchmarks, based on Fortran 77 and the MPI standard, which are intended to approximate the performance a typical user can expect for a portable parallel program on a distributed memory computer.

	Class	Procs	Avalon	T3E-900	SGI Origin
BT	B	64	2222	3483	4103
SP		64	1038	2655	3690
LU		64	3030	4032	6212
MG		64	2080	4865	3971

Table 4: Performance in Mflops for Avalon, a 64 processor Cray T3E-900 and a 64 processor SGI Origin 2000 are presented for the NAS v2.2 Class B benchmarks.

6 Conclusion

We believe that the simulations reported above are clearly of the "supercomputer" class, requiring nearly 8 Gbytes of memory, and performing in the range of 10^{16} floating point operations. The absolute performance of Avalon (running irregular production-type simulations) is factors of two and ten higher than the two price/performance winners last year, while improving overall price/performance by over a factor of three. Even more extraordinary, on these applications Avalon demonstrates price/performance an order of magnitude superior to commercial machines of equivalent performance.

Beowulf is far from an ideal resource for high performance computing. However, it encapsulates many features that are unique, and offers hope of providing a solution to the needs of many supercomputer users. The Beowulf architecture provides a standard message-passing hardware and software environment, with a low cost of entry. Hardware designs are driven by end-users. The software community that develops around these machines (if it remains open and follows the Linux development philosophy) will allow users, vendors, and the research community to each contribute to a robust software environment.

There is no "silver bullet" that will solve the problems posed by programming parallel machines. The fastest machines and the best software will always be right on the edge of being broken. Those that thrive in such an environment will be the first to solve the important computational problems of the future. We hope the example we have provided with Avalon will encourage others to investigate the applicability of Beowulf-class hardware as a solution to their particular problems.

References

[1] D. M. Beazley and P. S. Lomdahl. Message-passing multi-cell molecular dynamics on the connection machine 5. *Parallel Computing*, 20:173–195, 1994.

[2] D. M. Beazley and P. S. Lomdahl. Lightweight computational steering of very large scale molecular dynamics simulations. In *Proceedings of Supercomputing 96*, 1996.

[3] D. M. Beazley and P. S. Lomdahl. Controlling the data glut in large-scale molecular-dynamics simulations. *Computers in Physics*, 11(3):230–238, 1997.

[4] 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.

[5] Jack J. Dongarra. Performance of various computers using standard linear equations software, (linpack benchmark report). Technical report, University of Tennessee, 1998.

[6] B. L. Holian and P. S. Lomdahl. Plasticity induced by shock waves in nonequilibrium molecular-dynamics simulations. *Science*, 280:2085–2088, 1998.

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

[8] P. S. Lomdahl, P. Tamayo, N. Grønbech Jensen, and D. M. Beazley. 50 gflops molecular dynamics on the connection machine 5. In *Supercomputing '93*, pages 520–527, Los Alamitos, 1993. IEEE Comp. Soc.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] M. S. Warren, J. K. Salmon, D. J. Becker, M. P. Goda, T. Sterling, and G. S. Winckelmans. Pentium Pro inside: I. A treecode at 430 Gigaflops on ASCI Red, II. Price/performance of $50/Mflop on Loki and Hyglac. In *Supercomputing '97*, Los Alamitos, 1997. IEEE Comp. Soc.

[15] S. J. Zhou, D. M. Beazley, P. S. Lomdahl, and B. L. Holian. Large-scale molecular dynamics simulations of three-dimensional ductile failure. *Phys. Rev. Lett.*, 78:479, 1997.

[16] S. J. Zhou, D. L. Preston, P. S. Lomdahl, and D. M. Beazley. Large-scale molecular dynamics simulations of dislocation intersection in copper. *Science*, 279:1525–1527, 1998.