

# AppleSeed: A Parallel Macintosh Cluster for Scientific Computing

by

Viktor K. Decyk  
Department of Physics and Astronomy  
University of California, Los Angeles  
Los Angeles, CA 90095-1547

and

Dean E. Dauger  
Dauger Research  
Huntington Beach, CA

email: decyk@physics.ucla.edu  
dauger@daugerresearch.com

## Abstract

We have constructed a parallel cluster consisting of Apple Macintosh G4 computers running both Classic Mac OS as well as the Unix-based Mac OS X, and have achieved very good performance on numerically intensive, parallel plasma particle-in-cell simulations. Unlike other Unix-based clusters, no special expertise in operating systems is required to build and run the cluster. This enables us to move parallel computing from the realm of experts to the mainstream of computing.

## Introduction

In recent years there has been a growing interest in clustering commodity computers to build inexpensive parallel computers. A number of projects have demonstrated that for certain classes of problems, this is a viable approach for cheap, numerically intensive computing. The most common platform for building such a parallel cluster is based on the Pentium processor running the Linux version of Unix [1]. Since 1998, we have been using a cluster based on Apple Macintosh computers.

Our plasma simulation group uses a suite of Particle-in-Cell (PIC) codes to model plasmas [2-3]. These numerically intensive codes are used in a number of High-Performance Computing Projects, such as modeling fusion reactors [4] and advanced accelerators [5]. For these projects massively parallel computers are required, such as the 6,656 node IBM SP3 at the National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley Laboratory. However, it is very convenient if code development and student projects could be performed on more modest but user-friendly parallel machines such as the Macintosh clusters. It is also preferable that the resources of the large computers are devoted only to the large problems which require them. Indeed we discovered additional advantages in having a local cluster that we had

not anticipated.

## **Software Implementation for Mac OS Classic**

All of our parallel scientific codes use the industry standard Message-Passing Interface (MPI) [6] for communication. When we first began building Macintosh clusters in 1998, the Macintosh operating system at that time, Mac OS 8, did not have MPI. However, it did have a communications library available called PPC Toolbox, which used the AppleTalk protocol. We wrote a partial implementation of MPI (34 subroutines) called MacMPI, which translated the MPI calls we needed into the native PPC Toolbox. In addition, Dean Dager, a student in our group at that time, wrote a utility called Launch Den Mother to copy and launch the executables on distributed Macintoshes. We were then able to run our parallel PIC codes on the Apple Macintosh cluster without additional modifications. With a fast ethernet switch, an 8 node cluster of Macintosh G3/266s obtained performance better than that achieved with 8 nodes of the then current Cray T3E-900 at NERSC.

For high performance, Apple provided a protocol-independent communications library called Open Transport, which could be used either with AppleTalk or TCP/IP. After Mac OS 9 was introduced in 1999, we developed a second version of MacMPI (45 subroutines) based on the TCP/IP implementation of Open Transport, called MacMPI\_IP. With this library, a 16 node Macintosh G4/450 cluster connected with fast ethernet was able to achieve about half of the performance of 16 nodes of the then current IBM SP3/200 at NERSC. Further details about the early history of Macintosh clusters can be found in [7].

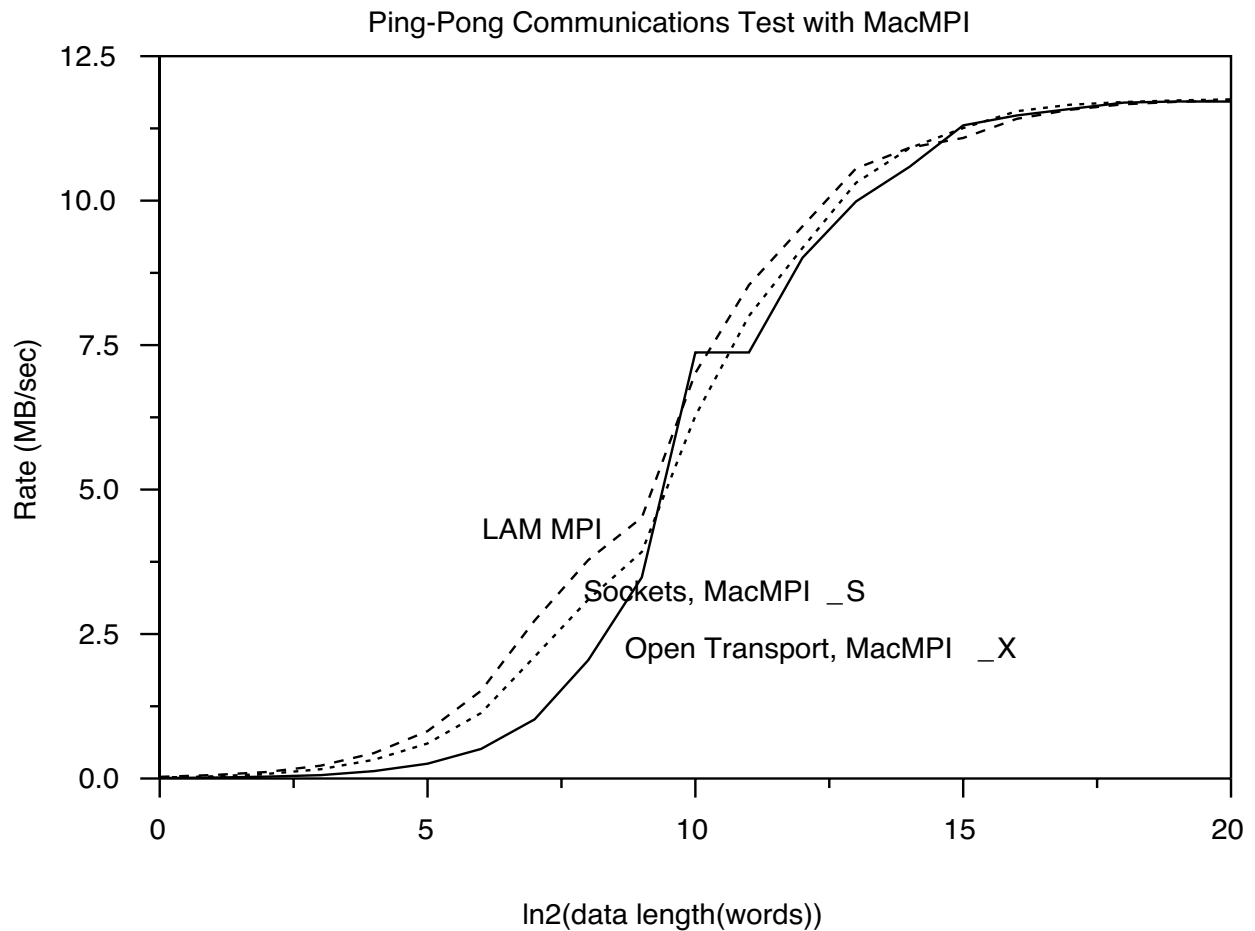
These libraries (for both Fortran77 and C) and related files and utilities are available at our web site: <http://exodus.physics.ucla.edu/appleseed/appleseed.html>.

## **MacOS X versions of MacMPI**

In 2001, Apple introduced a BSD Unix-based operating system called Mac OS X [8]. To ease the transition to the new OS, they provided a compatibility library called CarbonLib, which contains most of the features of the Classic Mac OS. This library allows the same compiled code to be used in either environment. With a relatively minor rewrite, the MacMPI\_IP library was made Carbon compliant, and we called this version MacMPI\_X. A new startup utility, called Pooch, was also written by Dean Dager. Pooch was a major rewrite, which added many new discovery features based exclusively on TCP/IP. A free version of Pooch is available from <http://dagerresearch.com/pooch/>. Today, we primarily use MacMPI\_X and Pooch to operate our cluster.

Apple suggested to us that using sockets rather than Open Transport might give us better performance on Mac OS X, so at the end of 2002, a sockets version of MPI, called MacMPI\_S, was also implemented. For machines connected with a fast ethernet switch, we discovered MacMPI\_S had better latency than MacMPI\_X, but about the same peak bandwidth.

Three other implementations of MPI are also available for Mac OS X, LAM MPI [9], mpich[10], and MPI/Pro [11]. The first two are freely available, the latter is a commercial product. The Pooch program works with all of them except LAM MPI. To determine what message sizes gave good performance, we developed a ping-pong and swap benchmark (where pairs of processors exchange packets of equal size) where the bandwidth was defined to be twice the packet size divided by the time to exchange the data. We found the peak performance of all of these with fast ethernet switches to be similar. A graph showing bandwidth as a function of message size is shown in Figure 1.



Macintosh G4/1000, Mac OS X, fast ethernet switch

Figure 1: Bandwidth (MBytes/sec) for 2 processors exchanging data as a function of message size, with a Fast Ethernet Switch.

### Recipe for Building a Simple Cluster

The easiest way to build a Macintosh cluster is to first obtain a number of Macintosh G4 computers. All the current models have built-in Gigabit or Fast Ethernet adapters. Next, obtain an appropriate Ethernet Switch containing at least one port for each Macintosh and a corresponding number of Category 5 Ethernet cables with RJ-45

jacks. Plug one end of each cable to the Ethernet jack on each Mac and the other end to a port on the switch. Turn everything on. As far as hardware is concerned, that's all there is. Then make sure the cluster is connected properly to the Internet as specified by your Internet Service Provider (ISP) or network administrator. (If the Mac is on an isolated network, you can manually configure TCP/IP to use a unique IP address from 192.168.1.1 to 192.168.1.254.)

It is recommended that in the Energy Saver System Preference, the sleep time be set to Never (although it is okay to let the monitor go to sleep). This prevents the Mac OS from going to sleep while running a Fortran or C program which does not interact with the user.

### **AppleSeed Hardware Implementation**

Our Macintosh cluster consists of various models purchased at different times during the last three years. The current configuration consists of 22 dedicated machines (16 G4/450s, 2 G4/450 duals, and 4 G4/1000 duals). A typical machine we would purchase today (May 2003) would be a G4/1000 with 1.25 GB RAM and 60 GB disk for about \$1600. It comes with Gigabit Ethernet and a CD-RW drive.

If only two Macs are being clustered, the only additional equipment needed is a Category 5 cable. A hub or switch is required to cluster more than two Macintoshes. If more than four nodes are being networked, a switch gives better performance but costs more. We currently have a 24 port Cisco and a 16 port Asanté switch. Fast Ethernet switches are now quite inexpensive (\$60 for an 8 port switch), but Gigabit switches are not (\$650 for an 8 port switch).

The 22 dedicated machines are currently clustered in subgroups: an eight node group, 3 four node groups, and a two node group. Subgroups can be used together to form larger groups. Each subgroup shares a single keyboard and monitor, via a Black Box or Master View USB KVM Switch. Such a subgroup is convenient for students writing parallel programs. If the machines were physically distant, then software such as Apple's Network Assistant or Farallon's Timbuktu can be used to observe and control Macintoshes remotely. We have used the Network Assistant software successfully, although it takes network bandwidth away from the application. Figure 2 shows a subcluster of G4 computers.



Figure 2: AppleSeed cluster of 4 Apple Macintosh computers.

### Using Pooch

For MacOS 9 and Mac OS X, a utility called Pooch has been written to automate the procedure of selecting remote computers, copying the executable and associated input files, and starting the parallel application on each computer. This utility can be downloaded from <http://daugerresearch.com/pooch/>.

To run in parallel, begin by selecting New Job... from the File menu of Pooch. This opens up a new Job Window. Select an application to run, either by selecting it from the dialog box (Select App...), or by dragging it from the Finder to this Job Window. In Figure 3, we show the AltiVec Fractal Carbon demo, available from our web site, being

selected by dragging. Adding input data is a matter of dragging in more files.

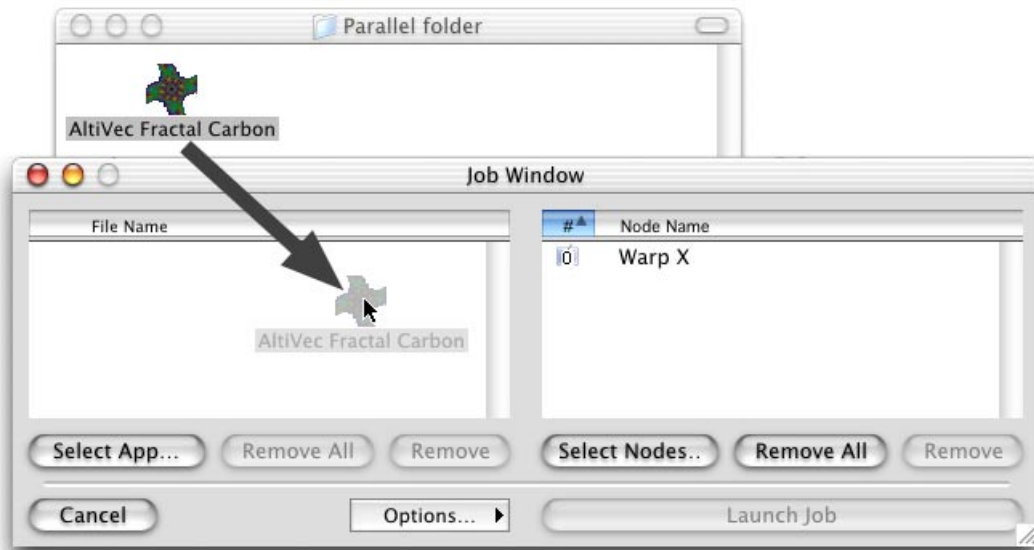


Figure 3. To set up a parallel computing job, drag a parallel application, in this case the AltiVec Fractal Carbon demo, and drop it in the Job Window of Pooch.

Next, choose the nodes to run in parallel. By default, Pooch includes the node where the job is being specified. To add more, click on Select Nodes..., which invokes a Node Scan Window, shown in Figure 4. It is not required that the user's computer be one of those selected for running the parallel application. If a Macintosh has two processors, then Pooch can treat each of them as a separate node.

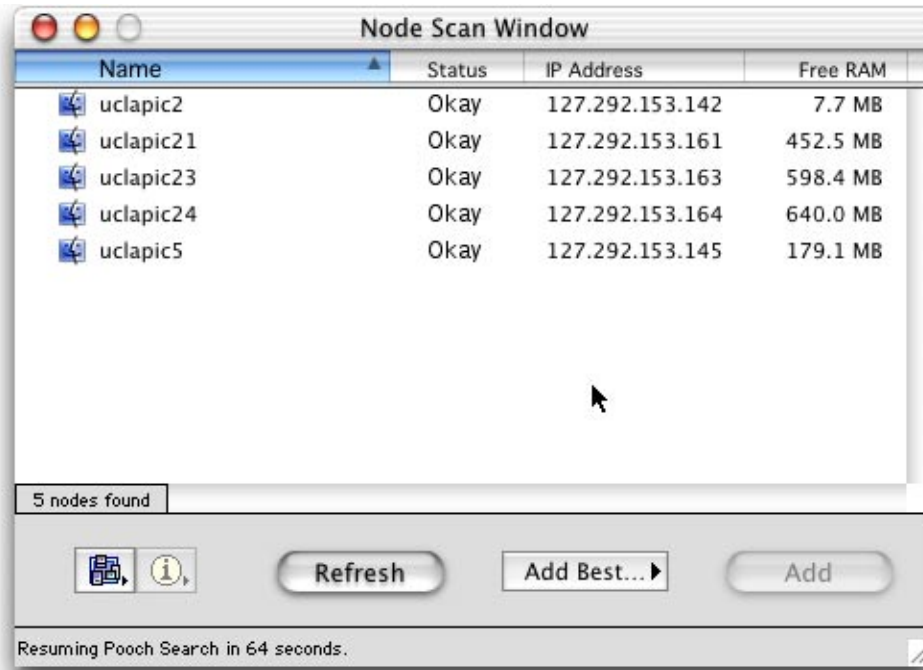


Figure 4. Selecting nodes is performed using the Node Scan Window, invoked by clicking on Select Nodes... from the window in the previous figure.

Double-clicking on a node moves it to the node list of the Job Window. Finally, the parallel job must be started by clicking on Launch Job, shown in Figure 5.

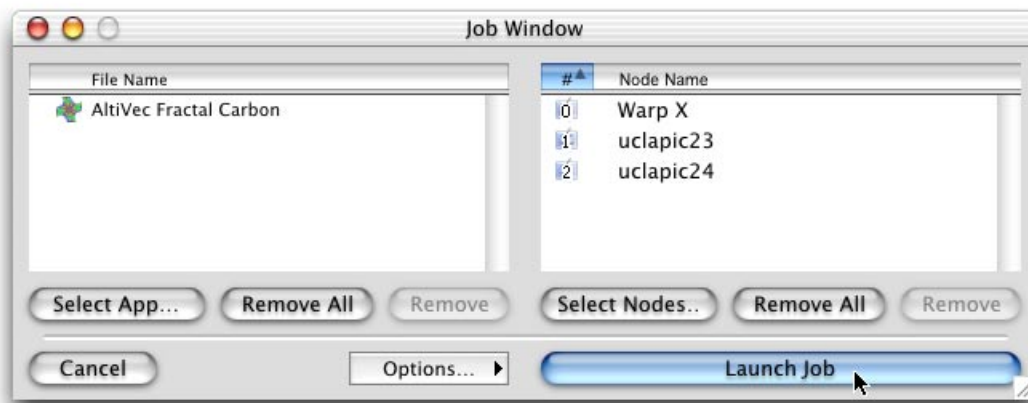


Figure 5. With the job ready, it begins with one more click.

Pooch should now be distributing copies of the parallel application to the other nodes and initiating them in parallel. MacMPI controls any further communication between nodes. That's all there is to it.

Upon completion of its computational task, the AltiVec Fractal Carbon demo then calculates its achieved performance, which should be significantly greater than single-node performance. If the user selected his or her own machine to participate, that machine becomes the master node (node 0 in MPI). Otherwise, the first node on the list becomes a master node.

The Pooch software provides discovery services, that is, a service to discover the existence and addresses of other nodes on the network. It also has the ability to determine up-to-the-minute information about nodes, including their availability and capability. It contains mechanisms for queuing jobs and launching them only when certain conditions have been met. It also has the ability to kill running jobs, launching jobs, and queued jobs. It can organize the job's files into subdirectories on the other nodes and retrieve files on those nodes containing output from completed jobs. Its components have the capability of automatic node discovery and selection. In fact, Pooch has been used to combine nodes at UCLA in Los Angeles, California, with machines in Munich, Germany, 10,000 km. apart. Further details can be found in the documentation available with the distribution.

Pooch features four user interfaces. In addition to its drag-and-drop graphical user interface illustrated above, Pooch has an AppleScript interface. This user interface makes it possible to write automatic and interactive scripts that direct Pooch to query the cluster or launch jobs for customized job queuing and other cluster operations. A command-line program, named `plaunch`, was written that translates Unix command-line options into AppleScript, providing a command-line user interface to Pooch. This utility makes it possible for users to log in from other platforms to control cluster operations. In addition, other applications can directly control Pooch through interapplication messages called AppleEvents. The AltiVec Fractal Carbon demo and the Fresnel Diffraction Explorer, another demo available on our web site, feature an item on their Parallel menus that initiate their launch onto a local cluster. These desktop applications can automatically take advantage of resources elsewhere on the cluster with only a menu selection. Such powerful yet easy to use features are the prerequisites for parallel computing to become mainstream.

## **Performance**

The performance of this cluster was excellent for certain classes of problems, mainly those where communication was small compared to the calculation and the message packet size was large. Results for the large 3D benchmark described in Ref. [3] are summarized in Table I. One can see that the Mac cluster performance is comparable to that of other computers we use. Indeed, the recent advances in computational performance is astonishing. A cluster of 4 Macintoshes now has more computational power and memory than a 4 processor Cray C-90, one of the best supercomputers of a decade ago, for less than one thousandth of the cost!

Table I.

3D Particle Benchmarks

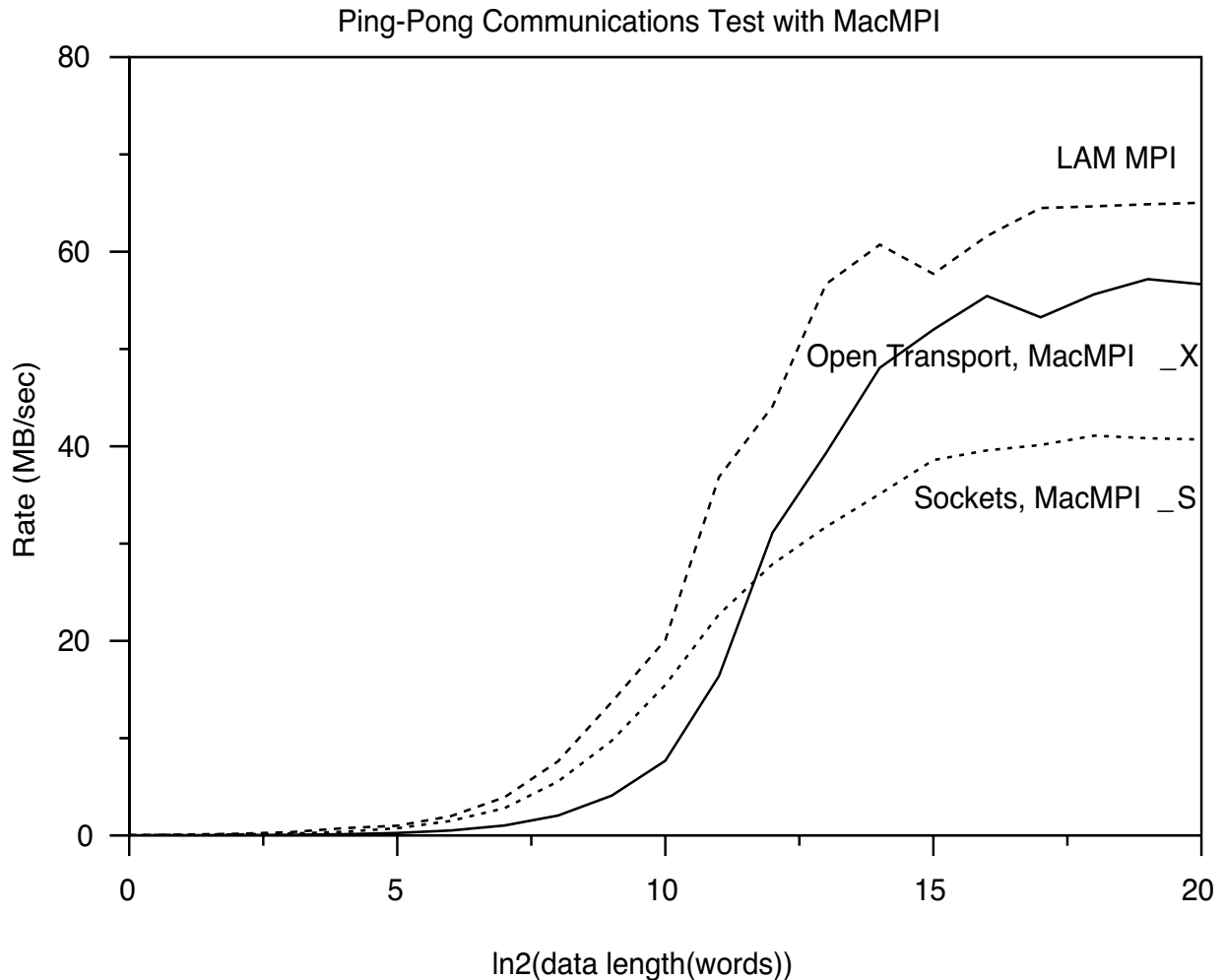
-----  
 The following are times for a 3D particle simulation, using 7,962,624 particles and a 64x32x128 mesh for 425 time steps. Push Time is the time to update one particle's position and deposit its charge, for one time step. Loop Time is the total time for running the simulation minus the initialization time.  
 -----

Computer	Push Time	Loop Time
Intel 2.2 GHz P4 Xeon, 4 proc:	108 nsec.	415.2 sec.
IBM SP3/375, w/MPI, 4 proc:	134 nsec.	476.5 sec.
Mac G4/1000 cluster, 4 proc:	238 nsec.	933.2 sec.
Cray T3E-900, w/MPI, 4 proc:	334 nsec.	1212.9 sec.
SGI Origin2000/R12000, 4 proc:	396 nsec.	1405.4 sec

-----

Recently, we established a new milestone with AppleSeed, 3D PIC simulation of 127 million interacting particles on a 4 node Macintosh G4/1000 dual processor cluster. The total time was 17.6 seconds/time-step, with a grid of 128x128x256. The current cost of such a cluster is less than \$10,000. Very interesting physics can now be done with limited resources.

We have just begun early experiments with Gigabit Ethernet. Figure 6 shows a typical curve obtained for the built-in gigabit Ethernet adapter, connected by an ethernet cable. One can see that the peak bandwidth varies between about 40 and 55 MB/sec, depending on the MPI implementation, which is about half the peak bandwidth supported by the hardware.



Macintosh G4/1000, Mac OS X, cross-over cable

Figure 6: Bandwidth (MBytes/sec) for 2 processors exchanging data simultaneously as a function of message size, with built-in gigabit Ethernet adapter and a crossover cable.

### Enhancements to MacMPI

In order to make the Macintosh cluster more useful for teaching students how to develop parallel programs, we have added some enhancements to MacMPI. One of these is the monitoring of MPI messages, controlled by a monitor flag in MacMPI. If this flag is set to 1 (the default), a small status window appears, as shown in Figure 7. In this window, status lights indicate whether the node whose screen is being examined is sending and/or receiving messages from any other node. Since messages normally are sent very fast, these lights blink rapidly. However, if there are communications difficulties, one can see visible pauses in the lights. Furthermore, if a deadlock occurs, which is a common occurrence for beginning programmers, the lights will stay lit, indicating which node was waiting for messages. This information can be used to debug the parallel code. The status window also shows a histogram of the size of

messages being sent or received. Since network-based cluster computers have a relatively large overhead (latency) in sending messages, it is better to avoid sending many short messages. This histogram indicates whether many short messages are being sent. The bottom of the status window has room for a one-line message of the users choice. This is useful for displaying the current procedure or time step being executed.

Two additional dials are also shown in the status window. One of these shows the percent of time spent in communication. If this number is very high, the code has too much communication. Both the time average over the whole run and an instantaneous value are shown. The second dial is a speedometer which shows the average and instantaneous speeds achieved during communication. These dials are only approximate: they measure the time between when a send or receive is first posted and when it actually completes. The speedometer indicating communication speed is a useful indicator of network problems if it suddenly starts to run slowly.

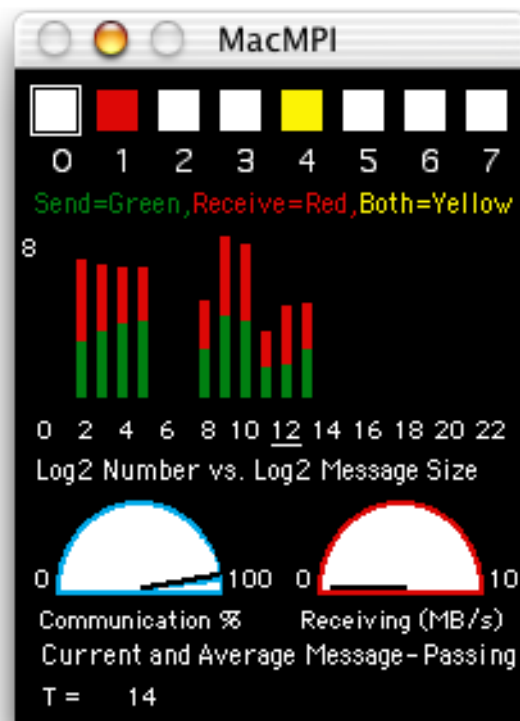


Figure 7. Monitor status window from MacMPI\_X

### Physics Applications

The AppleSeed cluster is primarily used for plasma physics projects. One of these is the Plasma Microturbulence Project. The goal of this project is to predict plasma and heat transport in fusion energy devices. Recent calculations by James Kniep and Jean-Noel Leboeuf have concentrated on studying various mechanisms of turbulence

suppression in devices such as the Electric Tokamak at UCLA and the DIII-D tokamak at General Atomics [12]. These researchers use AppleSeed for smaller problems when they need fast turnaround for fast scoping, as well as for production calculations which can be accommodated on our eight-node clusters with completion times on the order of weeks. They also run the same code on remote supercomputer centers for yet larger calculations. The results from all of these calculations have been favorably compared to experimental observations of plasma microturbulence characteristics in DIII-D discharges [13].

A second project which uses the AppleSeed cluster is a quantum particle-in-cell code. This code models multiparticle quantum mechanical problems in two dimensions (2d). This method was originally developed as part of Dean Dauger's Ph.D. thesis [14], and the current 2d code is developed and used by John Tonge, Dean Dauger, and Viktor Decyk [15]. This code represents the trajectory of a quantum particle using a Feynman path integral formulation in the semi-classical limit. Code development and small simulations are carried out on an Appleseed cluster and large simulations are performed at supercomputer centers. In addition, sophisticated interactive diagnostics have been developed for use on the Macintosh, including tools that can generate QuickTime movies "playing" quantum wavefunctions as sound as well as graphics.

A third project using AppleSeed is a study of wave-particle interactions in magnetized plasmas in space and laboratory plasmas. Using the particle-in-cell (PIC) codes OSIRIS and Parsec on a dedicated cluster of 8 single processor Mac G4/450's, the group has gain insights into the physics of wave-particle interactions in magnetized plasmas of very small transverse scale, which is of great importance in auroral physics. The early success on the AppleSeed enabled the researchers (F. S. Tsung, J. W. Tonge, and G. J. Morales) to get additional allocations on the supercomputer at the San Diego Supercomputer Center and some of the results obtained on the cluster have already been published [16].

## **Educational Applications**

One of the unexpected benefits of the AppleSeed cluster has been its usefulness for teaching advanced concepts in plasma physics and particle-in-cell plasma simulation techniques. In a Physics 260 class, "Exploring plasma physics using computer models", we were able to use complex 3D plasma models in the classroom. With 4 dedicated Macs a plasma simulation could be completed within a day. Using the IDL visualization environment students would study the results with the instructor during the class period, and a new run would be available by the next class. The students obtained interesting new results which have since been submitted for publication [17]. An earlier attempt to use a Unix-based supercomputer center was unsuccessful, because it took too much class time to teach the intricacies of Unix, and the center was so busy that the simulation often would not complete in time. Currently the AppleSeed cluster is greatly facilitating an introduction to Unix through Mac OS X in a less foreboding environment than that provided by a supercomputer center and is also enabling an initiation to parallel programming and in situ data analysis and visualization as part of a Physics 160 class, "Numerical analysis techniques and particle simulations of plasmas".

Project AppleSeed is also connected to the Visualization Portal [18] at the UCLA Computer Center. The Portal is an immersive virtual reality display that uses three 3-gun projectors to display images on a 160 x 40 degree spherical screen. The AppleSeed cluster can perform a large parallel calculation and display it interactively on the large 24' by 8' screen. Although this is still new technology, we have already found this immersive environment to be very useful for presentations.

## Evaluation

The inexpensive, powerful cluster of Macintosh G4s has become a valuable addition to our research group. It is especially useful for student training and running large calculations for extended periods. We have run simulations on 4 nodes for 100 hours at a time, using 1 GByte of memory. This is especially useful for unfunded research or exploratory projects, or when meeting short deadlines. The turnaround time for such jobs is often shorter than on supercomputer centers with more powerful computers, because we do not have to share this resource with the entire country. (However, we still need to use the supercomputer centers for problems too large for the Macintosh cluster.)

The cluster encourages a more interactive style of parallel programming, in contrast to the more batch-oriented processing encouraged by traditional supercomputer centers. This was an unexpected benefit. Because the cluster is used only by a small research group, we do not need sophisticated job management or scheduling tools. Indeed, we function entirely without a system administrator.

Why are we using Macintoshes? We have found that Mac OS X is an ideal platform for scientists. It combines the robustness of Unix with the traditional user-friendliness of the Macintosh OS. The hardware and software are well integrated and work seamlessly together. Furthermore, Apple is quick to introduce new technology when it becomes affordable. Both traditional high quality software such as word processors or IDL as well as a vast amount of open source Unix software are available.

Linux, in comparison, is far more difficult for the novice to use than the Mac. Substantial Unix expertise is required to correctly install, maintain, and run a Unix cluster. Indeed, reference [1] discusses many of the details one needs to worry about. In contrast, with the Mac cluster, the only required nonstandard item is a single library, MacMPI, and a single utility, Pooch. Everything else is ready to go right out of the box; just plug it in and connect it.

Because of its ease of use, the Macintosh cluster is particularly attractive to small groups with limited resources. For example, high school students are learning how to run parallel applications on clusters they built themselves [18]. Even a sixth grader has built his own Macintosh cluster [20].

What are the problem areas? The most common failure has nothing to do with the Macintosh computers: it is the network. Although network problems can be caused by bad cables or bad networking hardware, the most common cause is mismatched duplex settings. Fast Ethernet can send either full duplex, where a node can simultaneously

send and receive, or half duplex, where it cannot. The Ethernet adapters on each computer should have the same settings as the switch or hub. Hubs can only send half duplex, so the adapters should be set accordingly. Similar networking problems occur on non-Macintosh computers. They are typically more noticeable with tightly coupled clusters because they are taxing the network more severely.

### **Acknowledgments**

Many people have given us useful advice over the course of the last two years. We wish here to acknowledge special help given to us by Bedros Afeyan from Polymath Research, Inc., Ricardo Fonseca from IST, Lisbon, Portugal, and Frank Tsung from UCLA. This work has supported by the U. S. Department of Energy and the National Science Foundation.

## References

- [1] T. L. Sterling, J. Salmon, D. J. Becker, and D. F. Savarese, *How to Build a Beowulf*, [MIT Press, Cambridge, MA, USA, 1999].
- [2] V. K. Decyk, "Benchmark Timings with Particle Plasma Simulation Codes," *Supercomputer* 27, vol V-5, p. 33 (1988).
- [3] V. K. Decyk, "Skeleton PIC Codes for Parallel Computers," *Computer Physics Communications* 87, 87 (1995).
- [4] R. D. Sydora, V. K. Decyk, and J. M. Dawson, "Fluctuation-induced heat transport results from a large global 3D toroidal particle simulation model", *Plasma Phys. Control. Fusion* 38, A281 (1996).
- [5] K.-C. Tzeng, W. B. Mori, and T. Katsouleas, "Electron Beam Characteristics from Laser-Driven Wave Breaking," *Phys. Rev. Lett.* 79, 5258 (1997).
- [6] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, *MPI: The Complete Reference* [MIT Press, Cambridge, MA, 1996].
- [7] V. K. Decyk and D. E. Dauger, "Supercomputing for the Masses: A Parallel Macintosh Cluster," *Proc. of 4th Intl. Conf. on Parallel Processing and Applied Mathematics*, Naleczow, Poland, Sept., 2001, ed. by R. Wyrzykowski, J. Dongarra, M. Paprzycki, and J. Wasniewski [Springer Verlag Lecture Notes in Computer Science 2328, Berlin, 2002], p. 10.
- [8] Lon Poole and Dennis R. Cohen, *MacWorld Mac OS X Bible* [Hungry Minds, Inc., New York, 2002].
- [9] <http://www.lam-mpi.org/>.
- [10] <http://www-unix.mcs.anl.gov/mpi/mpich/>.
- [11] <http://www.mpi-softtech.com/>.
- [12] James C. Kniep, Jean-Noel Leboeuf and Viktor K. Decyk, "Gyrokinetic Particle-In-Cell Calculations of Ion Temperature Gradient Driven Turbulence with Parallel Nonlinearity and Strong Flow Corrections", Poster 1E25, Monday April 28, 2003, 2003 International Sherwood Fusion Theory Conference, Corpus Christi, Texas, April 28-30, 2003. <http://www.sherwoodtheory.org/sherwood03/agenda.html>.
- [13] T. L. Rhodes, J.-N. Leboeuf, R.D. Sydora, R. J. Groebner, E. J. Doyle, G. R. McKee, W. A. Peebles, C. L. Rettig, L. Zeng, and G. Wang, "Comparison of turbulence measurements from DIII-D L-mode and high performance plasmas to turbulence simulations and models", *Phys. Plasmas* 9, 2141-2148 (2002).

[14] Dean E. Dauger, "Semiclassical Modeling of Quantum Mechanical Mutiparticle Systems using Parallel Particle-in-Cell Methods," Ph.D. Thesis, University of California, Los Angeles, 2001. <http://dauger.com/DaugerDissertation.pdf>.

[15] V.K. Decyk, J. Tonge, D.E. Dauger, "Two Dimensional Particle-In-Cell Code for Simulation of Quantum Plasmas", Poster BP1. 118, 44th Annual Meeting of the Division of Plasma Physics, American Physical Society, November 11-15, 2002, Orlando, FL. Bull. Am. Phys. Soc., Vol. 47, No. 9, p. 52 (2002).

[16] F. S. Tsung, G. J. Morales, and J. N. Leboeuf, "Dynamics of a Supersonic Plume Moving along a Magnetized Plasma", Phys. Rev. Lett., 90, 5, p. 055004(2003).

[17] John Tonge, Jean-Noel Leboeuf, Chengkun Huang, John M. Dawson, "Kinetic simulations of the stability of a plasma confined by the magnetic field of a current rod", Submitted to Phys. Plasmas, April 2003.

[18] See <http://www.ats.ucla.edu/portal/>.

[19] Dennis Taylor, "Apples are the Core of These Clusters," IEEE Concurrency, vol. 7, no. 2, April-June, 1999, p. 7.

[20] <http://daugerresearch.com/pooch/users.html>.