

How to Build an AppleSeed: A Parallel Macintosh Cluster for Numerically Intensive 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 a mixture of Apple Macintosh G3 and G4 computers running the Mac OS, and have achieved very good performance on numerically intensive, parallel plasma particle-in-cell simulations. A subset of the MPI message-passing library was implemented in Fortran77 and C. This library enabled us to port code, without modification, from other parallel processors to the Macintosh cluster. Unlike 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 main stream 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]. When Apple Computer introduced the Macintosh G3 computer based on the Motorola PowerPC 750 processor, we decided to investigate whether a cluster based on the Macintosh G3 was practical.

This investigation was initially motivated by the impressive single node performance we achieved on our well-benchmarked suite of plasma particle-in-cell (PIC) simulation codes [2-3] on the Macintosh G3, as shown in Table I. This was due in part to the availability of an excellent optimizing Fortran compiler for the Macintosh produced by the Absoft Corporation [4]. Not only was the performance faster than the Pentiums, but it was comparable to the performance achieved on some Crays.

Table I.

3D Particle Benchmarks

 The following are times for a 3D particle simulation, using 294,912 particles and a 32x16x64 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
SGI Origin2000/R10000:	3430 nsec.	447.8 sec.
Macintosh G4/450:	4273 nsec.	555.4 sec.
Intel Pentium III/500:	5253 nsec.	683.8 sec.
Cray Y-MP:	5650 nsec.	741.1 sec.
Intel Pentium II/450:	6230 nsec.	804.6 sec.
Macintosh G3/300:	6390 nsec.	837.3 sec.
Intel Pentium II/300:	9040 nsec.	1172.8 sec.
IBM SP2, 1 proc:	10342 nsec.	1335.4 sec.
Cray T3E-900, 1 proc:	13364 nsec.	1702.0 sec.

A further motivation to build the cluster came when we realized that the Mac OS had a native message-passing applications programming interface (API), called the Program-to-Program Communications (PPC) Toolbox [5]. It has been there since 1990 and is used by AppleScript and Apple Events. We already had many programs written using the Message-Passing Interface (MPI) [6], a common message-passing API used on high-performance parallel computers. The similarity of the native PPC Toolbox message-passing facility to the low-level features of MPI further encouraged us to build the Macintosh cluster.

Our PIC codes are used in a number of High-Performance Computing Projects, such as modeling fusion reactors [7] and advanced accelerators [8]. For these projects massively parallel computers are required, such as the 512 node Cray T3E at NERSC. However, it would be 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.

Initial Software Implementation

Although a complete implementation of MPI has many high-level features (such as user defined datatypes and topologies) not available in the PPC Toolbox, these features were generally not needed by our PIC codes. It was therefore straightforward to write a partial implementation of MPI (34 subroutines), based on the PPC Toolbox, which we call MacMPI. PPC Toolbox uses the AppleTalk protocol, which can run on Ethernet hardware, but does not require an IP address. The entire MacMPI library was first written in Fortran77, making use of Fortran extensions for creating and dereferencing pointers available in the

Absoft compiler. A C language interface to MacMPI was also later implemented. In addition to the MacMPI library, Dean Dauger wrote a utility called Launch Den Mother to copy and launch the executables on the nodes. The executable file can also be copied to each node and started manually. Once the MacMPI library was implemented, we were able to port the parallel PIC codes from the Cray T3E and IBM SP2 to the Apple Macintosh cluster without modification.

High Performance MacMPI

We have found that MacMPI based on PPC Toolbox is very robust and reliable, since PPC Toolbox is very mature. It works with virtually every Macintosh, even older machines using the Motorola 680x0 processors. However, it does not give optimum performance, since it was written in an earlier era when network speeds were much slower. The current performance is adequate for many of the large problems we do, but it limits the range and types of problems that can be run on the cluster. For high performance, Apple provides a protocol-independent communications library called Open Transport, which can be used either with AppleTalk or TCP/IP.

In order to obtain high performance, we developed a second version of MacMPI based on the TCP/IP implementation of Open Transport, called MacMPI_IP, which gives performance about 7 times faster for large messages than the original MacMPI and is reliable. We also added 11 additional subroutines. Both Fortran77 and C versions of MacMPI_IP are available on our web site, and the Launch Den Mother works correctly with either version of MacMPI. The only additional requirement for setting up a Macintosh cluster for use with TCP/IP is to ensure that the TCP/IP Control Panel is properly configured. MacMPI_IP requires MacOS 8.5 or later. Although TCP/IP is used during the program execution, AppleTalk is still used to start the application. This generally limits the clusters to local area networks only, since AppleTalk packets are rarely routed beyond a local network. These libraries and related files and utilities are available at our web site:
<http://exodus.physics.ucla.edu/appleseed/appleseed.html>.

MacOS X version of MacMPI

Apple has recently introduced a Unix-based operating system called Mac OS X. To ease the transition to the new OS, they provided a compatibility library called CarbonLib, which contains most, but not all, 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 call this version MacMPI_X. A new startup utility, called Pooch, was also written by Dean Dauger. Pooch was a major rewrite, which eliminated all dependence on AppleTalk and added many new discovery features based exclusively on TCP/IP. A demonstration version of Pooch is available from <http://daugerresearch.com/pooch/>.

Using MacMPI

For a simple introduction to MPI programming, we recommend Using MPI [9]. The various MacMPI libraries have been tested with the Absoft Fortran and C compilers and with the Metrowerks C compiler. Details about compiling and linking are available on our web site: <http://exodus.physics.ucla.edu/appleseed/dev/developer.html>. During execution, errors detected by Fortran versions of MacMPI are written to Fortran unit 2, which defaults to a file called FOR002.DAT. In C versions of MacMPI, the errors are written to a file called MPIerrs. These files (one on each Mac) should be examined if

problems occur. Most MPI errors are fatal and will cause the program to halt

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.

The software setup for the Macintosh cluster is a little easier when running in the Carbon environment, which requires Mac OS 9 and CarbonLib 1.2 or higher and is available from Apple. First 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 the TCP/IP control panel to use a unique IP address from 192.168.1.1 to 192.168.1.254.) Finally, a unique computer name must be set in the File Sharing control panel.

To run a parallel program on the cluster, you should download two additional items: Pooch, and the AltiVec Fractal Carbon demo. Double-click the Pooch Installer and select a drive for installation. Congratulations! You've just constructed your own parallel computer. To test the cluster, drag and drop the AltiVec Fractal Carbon demo program to the Pooch Application, click Select Nodes..., select the computers you want to run it on, and, in the Job Window, click on Launch Job. The demo application should now be running in parallel.

To setup the Macintosh for parallel processing in Mac OS prior to 9 is a little more involved. One must set the AppleTalk control panel to use the appropriate Ethernet Adapter and verify in the Chooser that AppleTalk is active. Next, a unique computer name must be set and Program Linking should be enabled in the File Sharing Control Panel. Finally, in the Users and Groups Control Panel, one must allow Guests to link.

To run a parallel program on the cluster, you should download three additional items from our web site: the Launch Den Mother utility, the AutoGuest Control Panel, and the Parallel Fractal Demo. Copy the AppleSeed *f* folder containing the Launch Den Mother to the top directory of each Mac in the cluster, and drag and drop the AutoGuest Control Panel to the System Folder. Restarting each computer completes the software installation. To test it, drag and drop the Parallel Fractal Demo program to the Launch Den Mother, select the computers you want to run it on, and click on "Launch Executables". The demo application should now be running in parallel.

It is recommended that in the Energy Saver Control Panel, 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. For long running applications (days or weeks) we have found the system is more stable if we turn off all third party extensions in the Extensions Manager (by choosing Mac OS All).

Although MacMPI_X and Pooch are Carbon compliant and in principle will run with Mac OS X, there are still a few bugs left in the OS X version of CarbonLib which prevent them from running properly. Apple is aware of these problems and has promised fixes.

AppleSeed Hardware Implementation

Our Macintosh cluster consists of various models purchased at different times during the last three years. The current configuration consists of 16 dedicated machines (12 G4/450s and 4 various G3s), plus 12 user-owned machines. A typical machine we purchase today (May 2001) would be a G4/466 with 640 MB RAM and 30 GB disk for about \$1700. It comes with Gigabit Ethernet and a CD-RW drive.

For a simple introduction to Macintosh networking, we recommend the [Mac OS 8 Bible](#) [10]. If only two Macs are being clustered, the only additional equipment needed is a Category 5 crossover cable. We made our own cables, which otherwise would have cost about \$8 apiece. 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 (\$100 for an 8 port switch), but Gigabit switches are not (\$1800 for an 8 port switch).

The 12 dedicated machines are currently clustered in groups of four in different offices. Each sub-cluster shares a single keyboard and monitor, via a Black Box or Master View USB KVM Switch. Such a sub-cluster is convenient for students writing parallel programs. If the machines were physically apart, 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 1 shows a cluster of G3 and G4 computers. The user-owned machines are used for normal daily activities in the daytime, but can be available at night or weekends for numerical computing with the owner's permission.



Figure 1: AppleSeed cluster of 8 Apple Macintosh computers of various types.

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 II. One can see that the Mac cluster performance was better than that achieved by the Cray T3E-900 and the IBM SP2/266. 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 Y-MP, one of the best supercomputers of a decade ago, for less than one thousandth of the cost!

Table II.

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
Mac G4/450, IP cluster, 8 proc:	772 nsec.	2756.9 sec.
Mac G4/450, IP cluster, 4 proc:	1928 nsec.	6715.3 sec.
Mac G4/450, IP cluster, 2 proc:	4676 nsec.	16234.3 sec.

Cray T3E-900, w/MPI, 8 proc:	1800 nsec.	6196.3 sec.
Cray T3E-900, w/MPI, 4 proc:	3844 nsec.	13233.7 sec.

IBM SP2, w/MPL, 8 proc:	2104 nsec.	7331.1 sec.

Recently, we established a new milestone with AppleSeed, 3D PIC simulation of 100 million interacting particles on an 8 node Macintosh G4/450 dual processor cluster. The total time was 17.8 seconds/time-step, with a grid of 128x128x256. We used the cluster at Polymath Research, Inc. [11], which has 1 GB memory per node, since the UCLA machines were not large enough for this simulation. The current cost of such machines is about \$2000/node. It was only 5 or 6 years ago that such calculations required the world's largest supercomputers, costing tens of millions of dollars.

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. Figure 2 shows a typical curve. As one can see, high bandwidth is achieved for message sizes of around 2^{12} (4096) words or larger. Best bandwidth rates achieved on this test are better than 90% of the peak speed of the 100 Mbps hardware. We have found that for small systems with 4 nodes or less, hubs actually give better performance than switches. For the 3D benchmark case described in Table II, the average packet size varied between 2^{13} and 2^{17} words, which is in the region of good

performance.

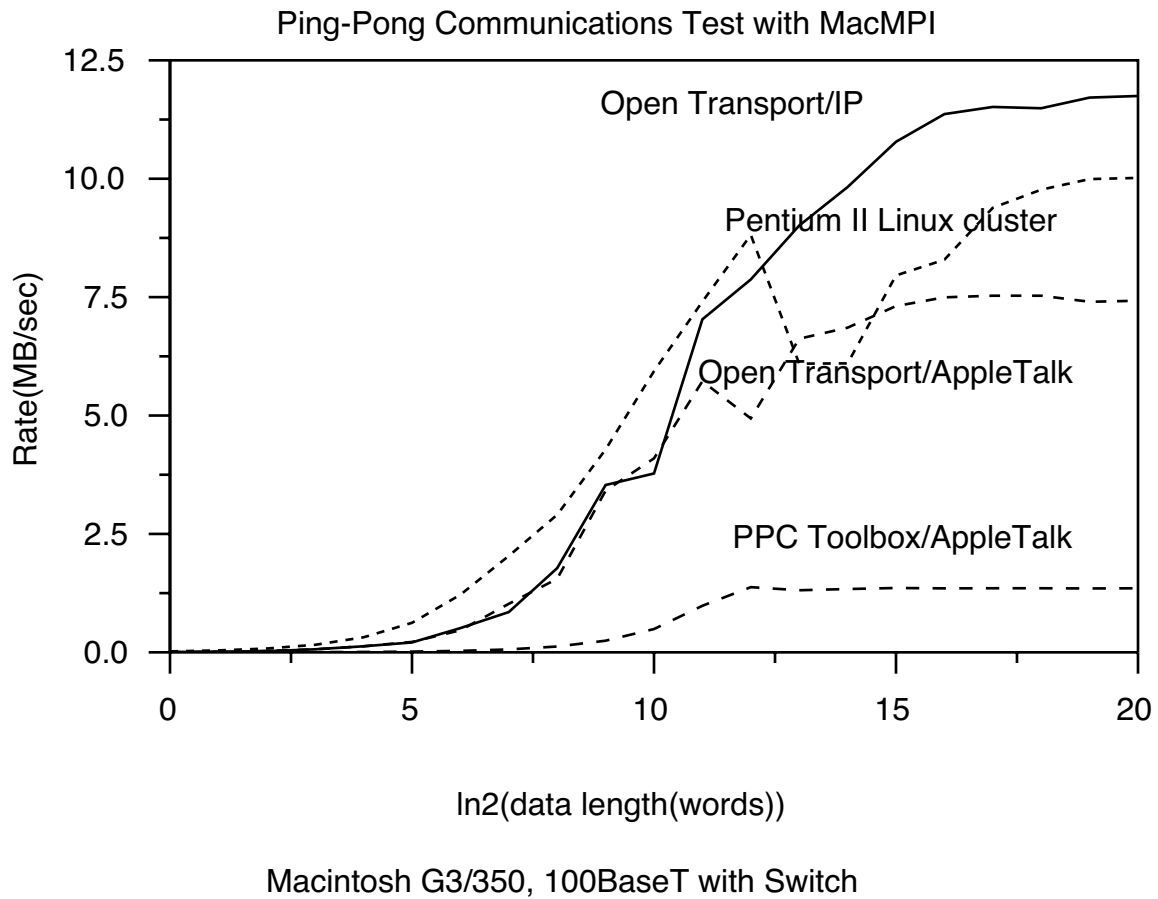
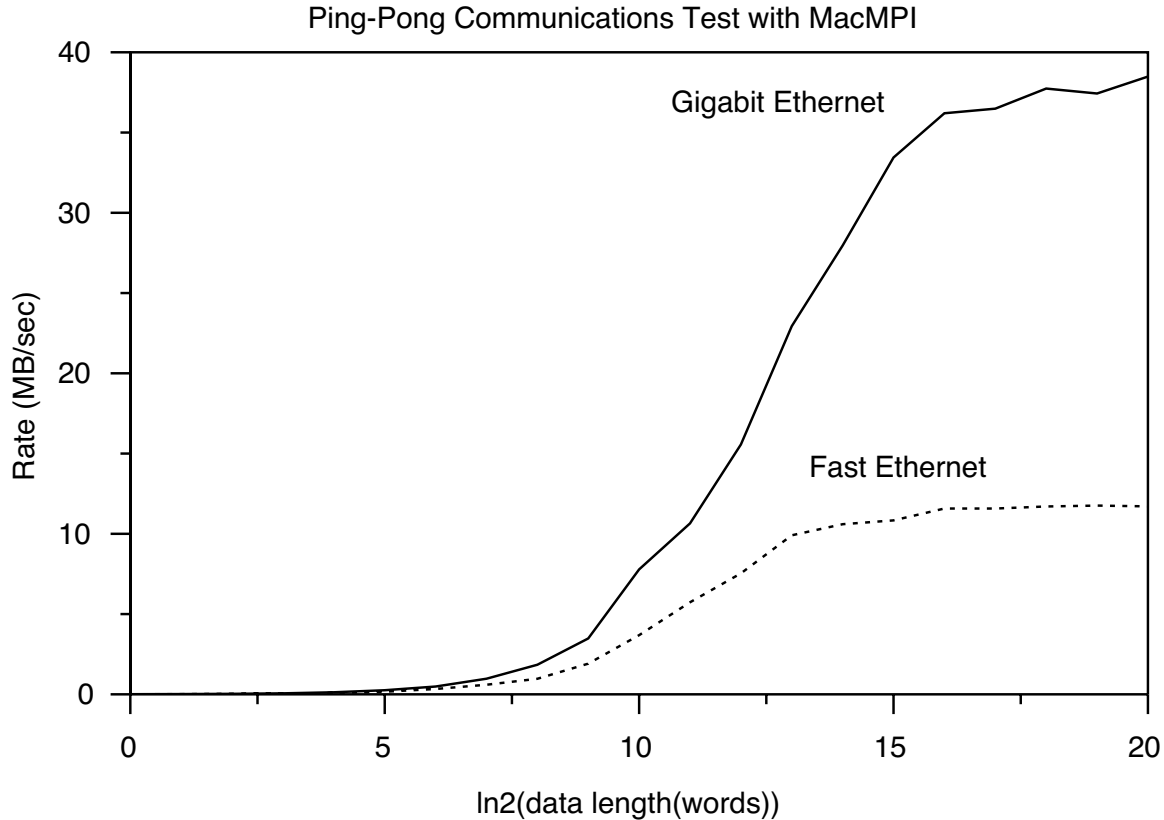


Figure 2: Bandwidth (MBytes/sec) for 2 processors exchanging data simultaneously as a function of message size, with a Cisco Fast Ethernet Switch.

Figure 3 shows a typical curve obtained for the built-in gigabit Ethernet adapter, connected by a cross-over cable. One can see that the peak bandwidth is almost 40 MB/sec, a speedup over Fast Ethernet by more than a factor of 3.



Macintosh G4/450, with cross-over cable

Figure 3: Bandwidth (MBytes/sec) for 2 processors exchanging data simultaneously as a function of message size, with built-in gigabit Ethernet adapter and a Cross-over cable.

Using Pooch

For MacOS 9 and CarbonLib 1.2 and later, 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 4, we show the AltiVec Fractal Carbon demo being selected by dragging. Adding input data is a matter of dragging in more files

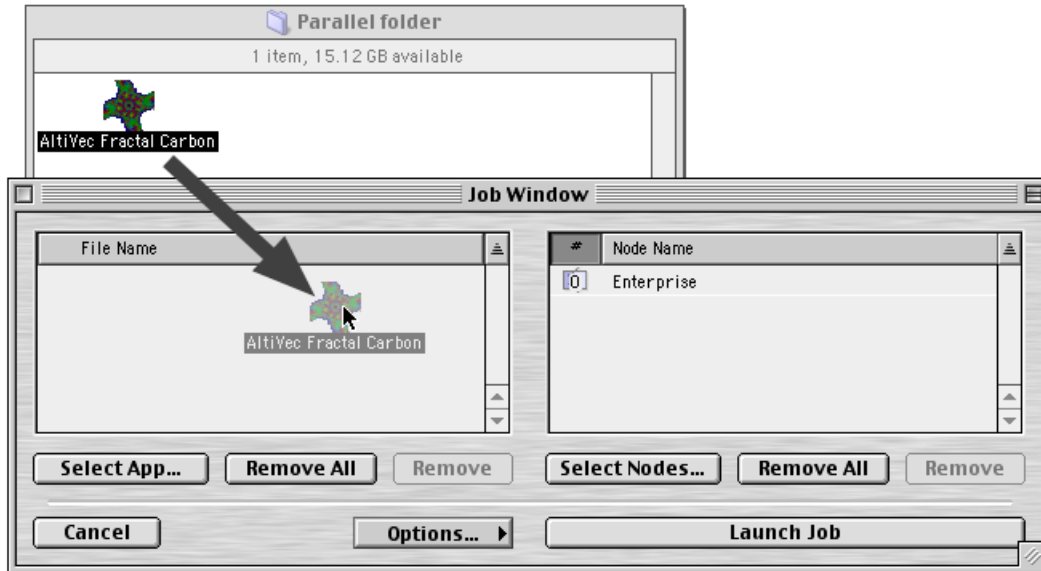


Figure 4. To set up a parallel computing job, drag a parallel application, in this case the AltiVec Fractal Carbon demo, and drop it in the Job Window of Pooch.

Next, choose the nodes to run on. By default, Pooch selects the node where the job is being specified. To add more, click on Select Nodes..., which invokes a Node Scan Window, shown in Figure 5. It is not required that the user's computer be one of those selected for running the parallel application

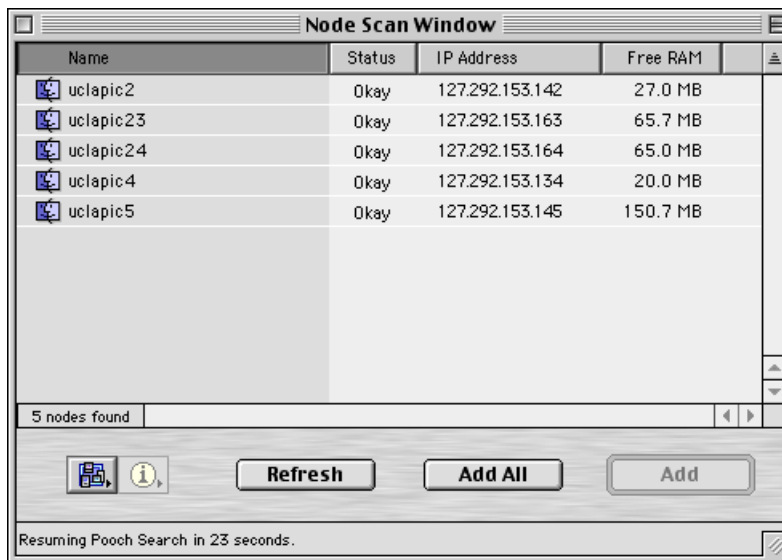


Figure 5. Selecting nodes is performed using the Node Scan Window, invoked by clicking on Select Nodes... from the window in the previous figure.

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 6.

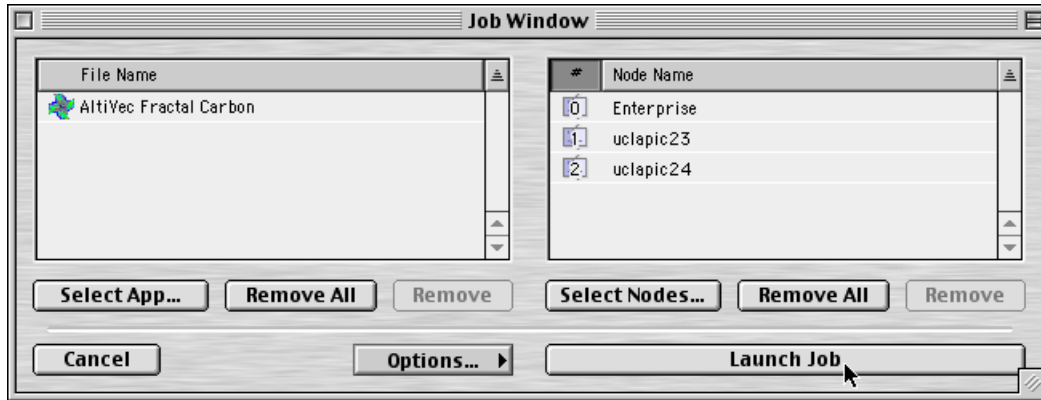


Figure 6. With the job ready, it begins with one more click.

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 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 remote master.

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. And all of its functions can operate via the Internet, and do not require AppleTalk. Further details can be found in the documentation available with the distribution.

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, whose value is set by a subroutine called `SET_MON` (`Set_mon` in C). If this flag is set to 1 (the default), a small status window appears. 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 a deadlock occurs, which is a common occurrence for beginning programmers, the lights will stay lit, indicating which node was waiting for messages, and this information can be used in debugging. The status window also shows a histogram of the size of messages being sent or received. Since network based cluster computers have a large overhead (latency) in sending messages, it is better to avoid sending many short messages. The purpose of this histogram is to indicate if many short messages are being sent. There is also room at the bottom of the status window for a one-line message of the user's choice. This is useful for displaying the current procedure or time step being executed. This is implemented by

a special subroutine added to MacMPI, called LOGNAME (Logname in C).

When MacMPI_IP or MacMPI_X is being used, two additional dials are 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. Finally, if the monitor flag is set to 2, a debug log of every message sent and received is written to the error file.

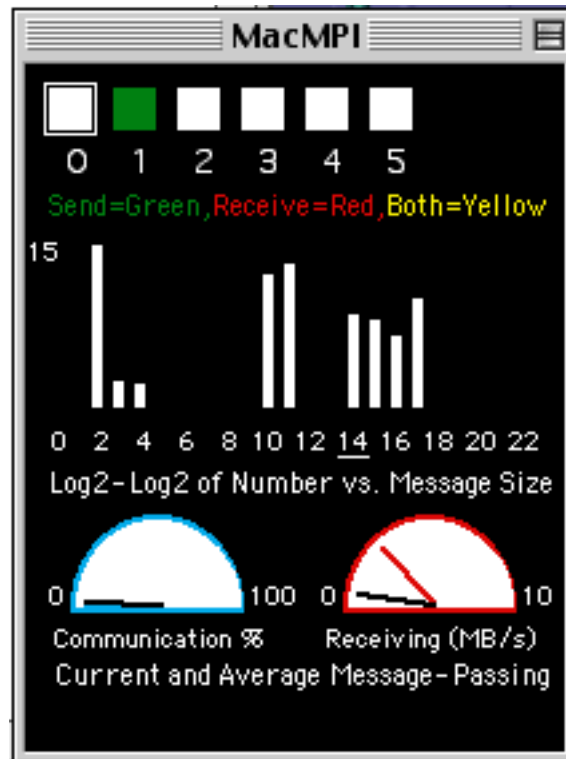


Figure 7. Monitor status window from MacMPI_IP or MacMPI_X

Physics Applications

The AppleSeed cluster is primarily used for plasma physics projects. One of these is the Numerical Tokamak Turbulence Project. The goal of this project is to predict plasma and heat transport in fusion energy devices. Recent calculations by Mike Kissick and Jean-Noel Leboeuf have concentrated on studying various mechanisms of turbulence suppression in devices such as the Electric Tokamak, under construction at UCLA [12]. These researchers use AppleSeed for smaller problems when they need fast turnaround. They also run the same code on remote supercomputer centers for the larger calculations.

A second project which used the AppleSeed cluster was a quantum particle-in-cell code, which models multiparticle quantum mechanical problems, which was the Ph.D. thesis

work of Dean Dauger [13]. It uses a semiclassical approximation of Feynman path integrals to reduce the calculation to a large number of classical charged particles moving in an electromagnetic field. This was a typical, underfunded student project, whose current problem size made it highly appropriate for the Mac cluster. In addition, this project made use of sophisticated interactive diagnostics developed on the Macintosh, including tools that generate QuickTime movies playing quantum wavefunctions as sound.

A third project using AppleSeed is a study by Frank Tsung and Ken Reitzel of wave-particle interactions with Alfvén waves in plasma. It uses a 3D electromagnetic PIC code called Osiris, developed by Roy Hemker at UCLA. A typical run uses 6 nodes and runs for 3 days. This is a preliminary study in preparation for a larger proposal.

Educational Applications

One of the unexpected benefits of the AppleSeed cluster was its usefulness for teaching advanced concepts in plasma physics. In a Physics 260 class, "Exploring plasma physics using computer models," we were able for the first time 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 and presented them a major international fusion conference [14]. 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.

Project AppleSeed is also connected to the Visualization Portal [15] 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.

Multiprocessor Macintosh

The Mac OS has multiprocessing capabilities, which allows computers with multiple processors to perform multiple tasks simultaneously. With Mac OS 9, however, both processors cannot call the OS simultaneously. This precludes the implementation of an MPI library which treats each CPU as a separate node. The programming interface provided by Apple for multiprocessing supports multitasking on a procedure level. It is low-level and unnecessarily complex for simple tasks. We have therefore written a simpler interface, which we call MacMP, that addresses more directly the typical needs of scientific programmers. This simplified interface is similar in spirit to the simple Multitasking library on the Cray parallel-vector computers of a decade ago. This library is freely available on our web site and is available in both Fortran and C. Thus with dual processor machines running Mac OS 9, two different programming models are used together, message-passing between machines, and multitasking on the machine. With Mac OS X, both processors should be able to call the OS directly, and using multiprocessors should be much simpler.

Evaluation

The inexpensive, powerful cluster of Macintosh G3s 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 turn around 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. (Although some problems can only run on the supercomputer centers because they are too large for the Macintosh cluster.)

The presence of the cluster has encouraged students and visitors to learn how to write portable, parallel MPI programs, which they can run later on larger computers elsewhere. In fact, since Fast Ethernet is slow compared to the networks used by large parallel supercomputers, our students are encouraged to develop better, more efficient algorithms that use less communication. Later, when they move the code to a larger parallel computer, the code scales very well with large numbers of processors. The cluster also 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 (which may not even exist). Everything is done in the spirit of cooperation, and so far that has worked. (We don't have uncooperative people in our group, anyway!)

Why are we using the Mac OS? Why not run Linux (a free Unix) on the Macs, for example? One reason is that we have always been Macintosh users and are very productive in the MacOS environment. There are good third party mathematical or numerical software packages, such as IDL or Mathematica, which run better on the Macintosh G4 than on our Unix workstations. Another reason is that many of the Macs are used for purposes other than numerical calculations and rely on software written for Mac OS. Furthermore, we find that the Mac environment makes it very easy to couple the output of our numerical codes to other software written in the Mac OS we use for presentation, such as Microsoft Word or AppleWorks. Finally, the Mac OS has encouraged us to write software to a higher standard, that has more of a Mac "look and feel," such as Pooch.

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 non-standard item is a single library, MacMPI, and a single utility, Pooch. Everything else is 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 [16].

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. They are more noticeable with AppleTalk networks which are constantly monitoring their state and will remove non-responsive nodes than with TCP/IP networks, which might merely degrade in performance.

Another cause of problems are the USB KVM switches used to share a single monitor and keyboard among multiple machines. USB requires a negotiation when a new device is connected, and interrupting the negotiation by switching between machines can sometimes cause a machine to crash. The Black Box switches are better in this regard than the Master View switches, they both have occasional problems. This problem does not occur with the older ADB bus Macintoshes.

The future continues to look bright. There are now several dozen Macintosh clusters around the world. The introduction of Mac OS X looks promising because it allows the use of a great deal of Unix software, such as public domain versions of MPI. Whether Apple can succeed in making Unix friendly is an interesting question. Apple is usually quick to introduce new technology when it becomes sufficiently cheap, such as gigabit networks. One does not have to search the trade literature to keep up with advanced technology, it just shows up.

Acknowledgments

Many people have given us useful advice over the course of the last two years. Some of them have been acknowledged in a previous publication[17]. We wish here to acknowledge subsequent 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 NSF contracts DMS-9722121 and PHY 93-19198 and DOE contracts DE-FG03-98DP00211, DE-FG03-97ER25344, DE-FG03-86ER53225, and DE-FG03-92ER40727.

Appendix: Launch Den Mother

A utility called Launch Den Mother (and associated Launch Puppies) was originally written for MacOS 8.1 to automate the procedure of selecting remote computers, copying the executable and associated input files, and starting the parallel application on each computer. For MacOS 9 and later, it was been replaced by Pooch.

Before running Launch Den Mother, each participating Macintosh must have the Launch Puppy utility located in a folder called AppleSeed *f*, that must reside in the top directory of the startup disk. In addition, the AutoGuest Control Panel available from Apple Computer [18] should be installed in the System folder on each participating Macintosh. AutoGuest permits the Finder of each computer to start the Launch Puppy if guest link access has been enabled in the Users and Groups Control Panel, without asking for further verification from the owner of each machine.

The Launch Den Mother utility needs to reside only on the computers which will be initiating a parallel application, although we normally install it on all the computers. After starting Launch Den Mother, a single dialog box appears, as shown in Figure 8.

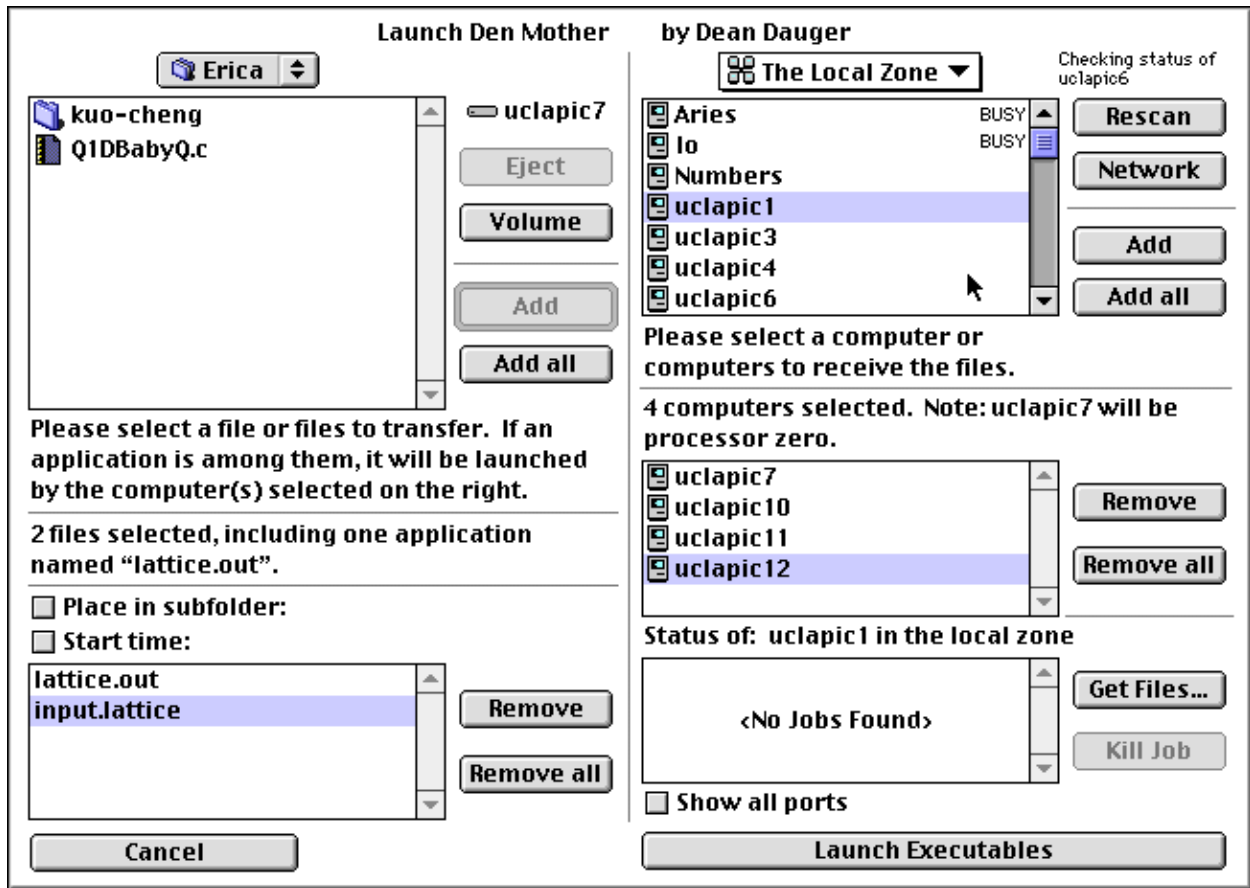


Figure 8. Launch Den Mother utility dialog box.

First one selects the application to run. In the upper left hand corner of the dialog box appears a list of files, from which one selects the files which will be copied to the other Macs and executed. This list of files selected is displayed in the lower left hand corner of the dialog box. In Figure 8, one can see that from the Erica folder, we have chosen two files, an executable file called `lattice.out`, and an input file called `input.lattice`, which is needed by `lattice.out`. Only one executable can be selected. To execute our Demo program, look for and select Parallel Fractal Demo.

Then one selects the computers to run on. In the upper right hand corner of the dialog box appears a list of available Macintoshes whose owners have permitted Program Linking in the File Sharing Control Panel. One selects from this list the computers one wishes to run on. In Figure 8, four computers from the Local Zone have been selected for execution, `uclapic7`, `uclapic10`, `uclapic11`, and `uclapic12`. Other computers were available, but two of them were already busy running a parallel job (Aries and `lo`). It is not required that the user's computer be one of those selected for running the parallel application.

Once the application and computers have been selected, one clicks on Launch Executables. The files are then copied to the AppleSeed *f* folder on each computer and each application is started up. MacMPI controls any further communication between nodes. That's all there is to it.

The Launch Den Mother works by sending an Apple Event to the Finder on each remote Macintosh, requesting that the Launch Puppy be started up. The Launch Puppy must be in the AppleSeed *f* folder so that the Finder can find it. This remote Apple Event requires MacOS 8.0 to work properly. Once all the remote Launch Puppies are started up, the Den Mother sends the requested files to each Puppy, which then copies them to the AppleSeed *f* folder on the local Macintosh. The Puppy starts its copy of the parallel application, sends a message to the Den Mother, and the Den Mother tells the Puppy to quit. After all the Puppies have been told to quit, the Den Mother launches the application on node zero and quits, and MacMPI takes over.

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 remote master, and the user's Launch Den Mother starts up and passes control to a remote Launch Den Mother.

During execution, errors detected by MacMPI.f are written to Fortran unit 2, which defaults to a file called FOR002.DAT. In MacMPI.c, the errors are written to a file called MPIerrs. These files (one on each Mac) should be examined if problems occur. Most MPI errors are fatal and will cause the program to halt.

After execution, there are usually output files created by the application. In most of our applications, only the master node produces any output. All the other nodes which have output data send it to the master node using MPI. Since the master node is usually either on the user's desk or in a common area available to everyone, there is no difficulty accessing these files. However, it is possible that there may be output files in the AppleSeed *f* folder on a remote computer. The simple way to retrieve remote output files is to use the "Get Files..." feature of Launch Den Mother. It is also possible, but more complex, to access them using AppleShare (via the Chooser), if the owner of the remote computer allows it. For user-owned machines, the Macintoshes are generally configured to allow Program Linking, but not allow File Sharing, so that the only way to retrieve remote files is via the Launch Den Mother. For common machines, File Sharing is generally turned on, and all files are accessible.

The Launch Den Mother currently uses AppleTalk to communicate with the remote Puppies regardless of the version of MacMPI being used. This is because browsing facilities (to find available Macintoshes) and sending remote Apple Events (to start up remote Puppies) are more easily implemented with AppleTalk. In order to support the TCP/IP version of MacMPI, the Puppies determine the IP address of their own machine and send the information to the Launch Den Mother. The Den Mother then creates two nodelist files, one for AppleTalk and one for TCP/IP, since it does not know which version will actually be used.

The Launch Den Mother has additional features not described here, such as the ability to kill remote jobs and to schedule jobs for later execution. Further details can be found in the README documentation available with the distribution on our web site.

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] See <http://www.absoft.com/>
- [5] Apple Computer, Inside Macintosh: Interapplication Communication [Addison-Wesley, Reading, MA, 1993], chapter 11.
- [6] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, MPI: The Complete Reference [MIT Press, Cambridge, MA, 1996].
- [7] 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).
- [8] K.-C. Tzeng, W. B. Mori, and T. Katsouleas, "Electron Beam Characteristics from Laser-Driven Wave Breaking," Phys. Rev. Lett. 79, 5258 (1997).
- [9] William Gropp, Ewing Lush, and Anthony Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface [MIT Press, Cambridge, MA, 1994].
- [10] Lon Poole, MacWorld Mac OS 8 Bible [IDG Books Worldwide, Foster City, CA, 1997], chapter 17.
- [11] <http://polymath-usa.com/>
- [12] M. W. Kissick, J. N. Leboeuf, S. Cowley, J. M. Dawson, V. K. Decyk, P. A. Gourdain, J. L. Gauvreau, L. W. Schmitz, R. D. Sydora, and G. R. Tynan, "Radial electric field required to suppress ion temperature gradient modes in the electric tokamak", Phys. Plasmas 6, 4722 (1999).
- [13] Dean E. Dauger, "Semiclassical Modeling of Quantum Mechanical Mutiparticle Systems using Parallel Particle-in-Cell Mehtods," Ph.D. Thesis, University of California, Los Angeles, 2001. <http://dauger.com/DaugerDissertation.pdf>.
- [14] Chengkin Huang, John Tonge, Jean-Noel Leboeuf, and John M. Dawson, "Particile Simulations of Plasma Confinement in a Levitated Dipole," Intl. Sherwood Fusion Theory Conference, Paper 1C46, April 2001.
- [15] See <http://www.ats.ucla.edu/portal/>.
- [16] Dennis Taylor, "Apples are the Core of These Clusters," IEEE Concurrency, vol. 7, no. 2, April-June, 1999, p. 7.

[17] V. K. Decyk, D. Dager, and P. Kokelaar, "How to Build An AppleSeed: A Parallel Macintosh Cluster for Numerically Intensive Computing," *Physica Scripta*, T84, 85, 2000.

[18] See ftp://ftp.apple.com/developer/Tool_Chest/OS_Uilities/