

# How to Build an AppleSeed: A Parallel Macintosh Cluster for Numerically Intensive Computing

by

Viktor K. Decyk, Dean E. Dauger, and Pieter R. Kokelaar

Department of Physics and Astronomy  
University of California, Los Angeles  
Los Angeles, CA 90095-1547

email: decyk, dauger, and pekok@physics.ucla.edu

## **Abstract**

We have constructed a parallel cluster consisting of 25 Apple Macintosh G3 and G4 computers running the MacOS, 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. For large problems where message packets are large and relatively few in number, performance of 50-150 MFlops/node is possible, depending on the problem. Unlike Unix-based clusters, no special expertise in operating systems is required to build and run the cluster.

## **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 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 of the 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.
Macintosh G3/350:	5966 nsec.	781.2 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.
Macintosh G3/266:	9185 nsec.	1195.3 sec.
IBM SP2, 1 proc:	10342 nsec.	1335.4 sec.
iMac/233:	10410 nsec.	1358.5 sec.
Cray T3E-900, 1 proc:	13364 nsec.	1702.0 sec.

-----

A further motivation to build the cluster came when we realized that the MacOS 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.

### Initial Software Implementation

Although a complete implementation of MPI has many high-level features (such as user defined datatypes and division of processes) 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 currently 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, due to requests by others.

The only complicated subroutine was the initialization procedure, MPI\_INIT. To initialize the cluster, a nodelist file is read which contains a list of n computer names and zones which are participating in the parallel computation. The node which has this file is designated the master node (node 0). The master initiates a peer connection with each of the other participating nodes (1 through n-1), and then passes to node 1 the list of remaining nodes (2 through n-1). Node 1 then establishes a peer connection with them, and passes on the list of remaining nodes (3 through n-1) to node 2, and so on. The last node receives a null list and does not pass it on further. Each node also establishes a connection to itself, as required by MPI. We have written a utility called Launch Den Mother to create the nodelist file and to copy and launch the executables on the nodes automatically. 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. This library and related files and utilities are available at our web site: <http://exodus.physics.ucla.edu/appleseed/appleseed.html>. For a simple introduction to MPI programming, we recommend [Using MPI](#) [9].

### **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 Fast Ethernet adapters. Next obtain a Fast 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.

To setup the Macintosh for parallel processing in MacOS 8.1 and higher, one must set the AppleTalk Control Panel to use the appropriate Fast 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. (In MacOS 9, Users and Groups is part of the File Sharing Control Panel.) In addition, it is strongly recommended that in the Energy Saver Control Panel the sleep time is set to Never (although it is okay to let the monitor go to sleep). A running Fortran or C program may appear to be inactive to the MacOS, and if the Mac is put to sleep, it will suddenly run very slowly.

To run a parallel program on the cluster, you should download 3 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, we recommend you first run the Parallel Fractal Demo on a single node by double clicking. Make a note of the execution time and speed. Then run it in parallel by dragging and dropping the Parallel Fractal Demo program to the Launch Den Mother, select the computers you want to run it on, and click on "Launch Executables". By running the

code in parallel, you should observe a noticeable speedup. To write your own parallel software, look at the section later in this paper entitled Using MacMPI.

## **AppleSeed Hardware Implementation**

Our cluster of 25 Macintosh machines consists of various models purchased at different times during the last 2 years. The current configuration consists of 7 G3/266s, 4 G3/300s, 3 G3/350s, and 11 G4/450s. A recommended baseline Macintosh G4 running at 400 MHz currently (April, 2000) costs \$1439 at UCLA. This machine comes with 64 MB RAM, 1MB L2 Cache, a 10 GB Hard Drive and DVD-ROM. We generally upgrade the memory of each Macintosh by adding 256 MB (current cost \$335). All current Macintoshes come with a built-in 100BaseT Ethernet adapter. We have also found it convenient to add an additional 100BaseT PCI Fast Ethernet adapter, currently from Asanté (cost \$49).

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 cross-over cable. We made our own cables, which otherwise would have cost about \$8 apiece. A hub or switch is required to cluster more than 2 Macintoshes. If more than 4 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 (the cost of such switches is around \$1500).

Our Appleseed cluster consists of 14 user-owned machines and 11 common machines. The user-owned machines are used for normal daily activities in the daytime, but are generally available at night for numerical computing. The common machines are always available for numerical computing. All the Macintoshes have two 100BaseT Ethernet adapters and are connected to two networks simultaneously. One adapter is set to TCP/IP and the machines are all connected via a Cisco switch, which is then connected to a fast campus ATM network. The other adapter is set to AppleTalk, and most of the machines are connected via an Asanté switch, which is then connected to a slower campus network. The main reason for the two networks is that our campus network authorities will not allow AppleTalk packets on the fast network. When the machines in the cluster are communicating only with each other, their communications are handled entirely by the switches and do not go on the campus networks.

The 10 common machines are currently clustered in groups of 4, 4 and 3 in different offices. Each sub-cluster shares a single keyboard and monitor, by connecting each computer in the sub-cluster to a Master View USB KVM Switch. This is convenient since the machines are physically close together. 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.



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

### 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. PPC Toolbox is based on the AppleTalk protocol only. For high performance, Apple has available 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 new 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. Both a Fortran77 and C version of MacMPI\_IP are available on our web site, and the current 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. If the computers are not connected to an Internet, then one can choose unique numbers from 192.168.1.0 to 192.168.1.255. If the Macintoshes are sometimes connected to an Internet and sometimes not, then one can create two TCP/IP configurations and select between them as appropriate. MacMPI\_IP requires MacOS 8.5 or later.

Since Open Transport is protocol-independent, we also implemented a new AppleTalk version as well. This version, called MacMPI\_OT, also gives good performance, but it is not entirely reliable. When sending large messages with many nodes simultaneously, it is possible to deadlock. This appears to be due to a bug in the AppleTalk version of Open Transport that results in lost acknowledgments when data is being sent under flow control.

## **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!

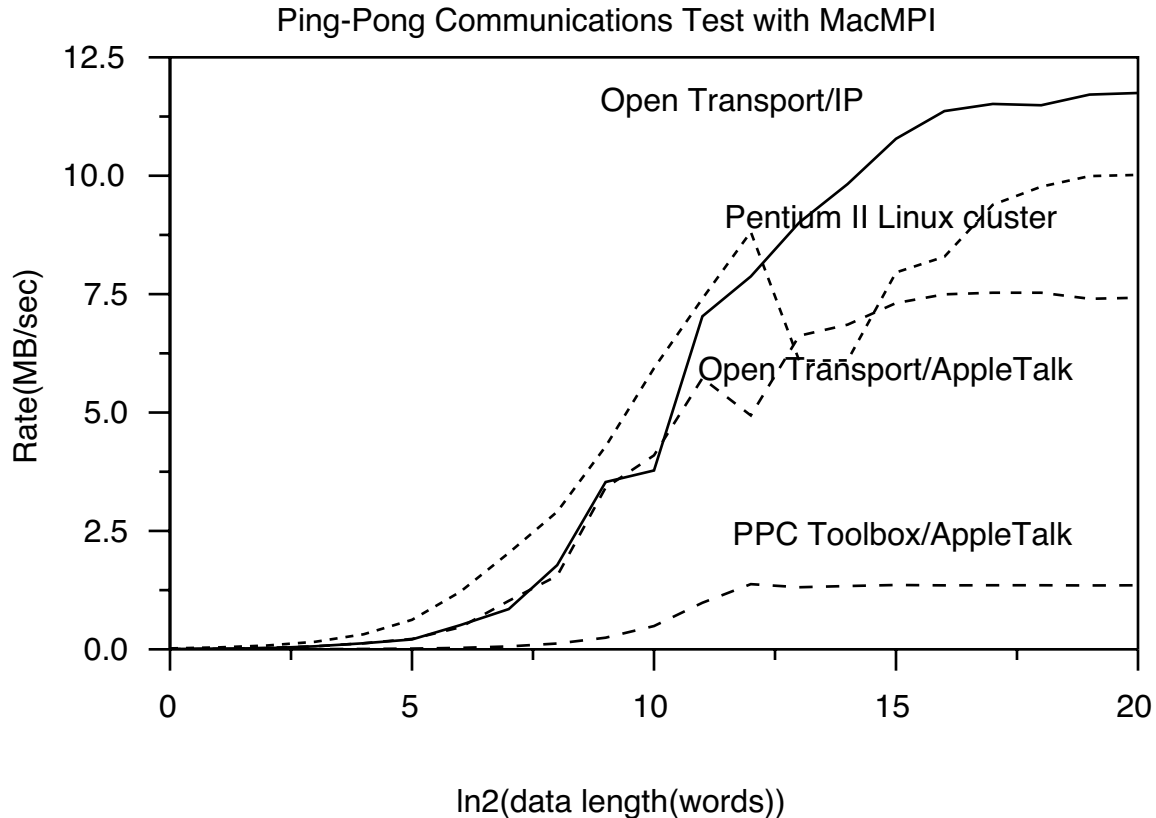
Table II.

## 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.
-----		
Mac G3/266, AT cluster, 8 proc:	1496 nsec.	5891.2 sec.
Mac G3/266, AT cluster, 4 proc:	3231 nsec.	11929.6 sec.
Mac G3/266, AT cluster, 2 proc:	7182 nsec.	25738.5 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.
-----		

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.



Macintosh G3/350, 100BaseT with Switch

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

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.

### Using MacMPI

To compile and run your own Fortran source code, two additional files are needed, a communications library, either `MacMPI.f` or `MacMPI_IP.f`, and the include file `mpif.h`. Creating an executable with the Absoft compiler is straightforward. First make sure the include file `mpif.h` is present, then compile the appropriate library with the `f77` compiler:

```
f77 -O -Q92 -c MacMPI.f
f77 -O -Q92 -c MacMPI_IP.f
```

Linking depends on which library is chosen and the language of the main program. If `test.f` is the main program, one links with the original MacMPI as follows:

```
f77 -O -Q92 test.f MacMPI.f.o
f90 -O -604 test.f MacMPI.f.o
```

For the TCP/IP version, additional libraries must be included, which depend on the version of the operating system being used. For MacOS prior to 9.0, the link command is:

```
f77 -O -Q92 test.f MacMPI_IP.f.o @
"{PPCLibraries}"OpenTransportAppPPC.o @
"{PPCLibraries}"OpenTptAtalkPPC.o @
"{SharedLibraries}"OpenTransportLib @
"{SharedLibraries}"OpenTptInternetLib @
"{SharedLibraries}"OpenTptAppleTalkLib
```

Unfortunately, the Shared Library OpenTransportLib included with the MPW Absoft compiler is out of date. To fix this, one should make a copy of the file OpenTransportLib from the Extensions folder in System folder and place it in the SharedLibraries folder in Libraries folder of MPW. Linking with Fortran90 is similar. If one has an older version of the Absoft compiler (version 6.0), make sure to download the latest patches from their web site.

For MacOS 9.0 and later, Apple has combined some of these Shared Libraries into a new library called Open Transport, and the link command is simplified:

```
f77 -O -Q92 test.f MacMPI_IP.f.o @
"{PPCLibraries}"OpenTransportAppPPC.o
"{PPCLibraries}"OpenTptAtalkPPC.o @
"{SharedLibraries}" "Open Transport"
```

As before, one needs to copy the file Open Transport from the Extensions folder in the System folder into the Shared Libraries folder. One can also run Fortran77 code with automatic double precision, by using the -N113 and -N2 options:

```
f77 -O -N113 -N2 -Q92 test.f MacMPI.f.o
```

This option is not available for Fortran 90. It is possible to create a makefile both manually as well as via a graphical interface, although the makefiles differ from Unix style makefiles.

To compile and run a C source program, one needs the library MacMPI.c or MacMPI\_IP.c and the header file mpi.h. With the Absoft C compiler, one compiles the library with one of the following commands:

```
acc -O -A -N92 -c MacMPI.c
acc -O -A -N92 -c MacMPI_IP.c
```

Then links with a main program, as the following TCP/IP example shows:

```
acc -O -A -N92 test.c MacMPI.c.o @
"{PPCLibraries}"OpenTransportAppPPC.o @
"{PPCLibraries}"OpenTptAtalkPPC.o @
"{SharedLibraries}"OpenTransportLib @
"{SharedLibraries}"OpenTptInternetLib @
"{SharedLibraries}"OpenTptAppleTalkLib
```

With the Absoft compiler, C and Fortran programs can link with one another. We have also successfully tested the C version of MacMPI and MacMPI\_IP with the Metrowerks C compiler. Sample parallel programs for beginners are available on our web site.

## Launch Den Mother

A parallel application can be started either manually or automatically. A utility called Launch Den Mother (and associated Launch Puppies) 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 our web site.

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 [11] 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. Launch Den Mother operates best with MacOS 8.0 or later.

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 3.

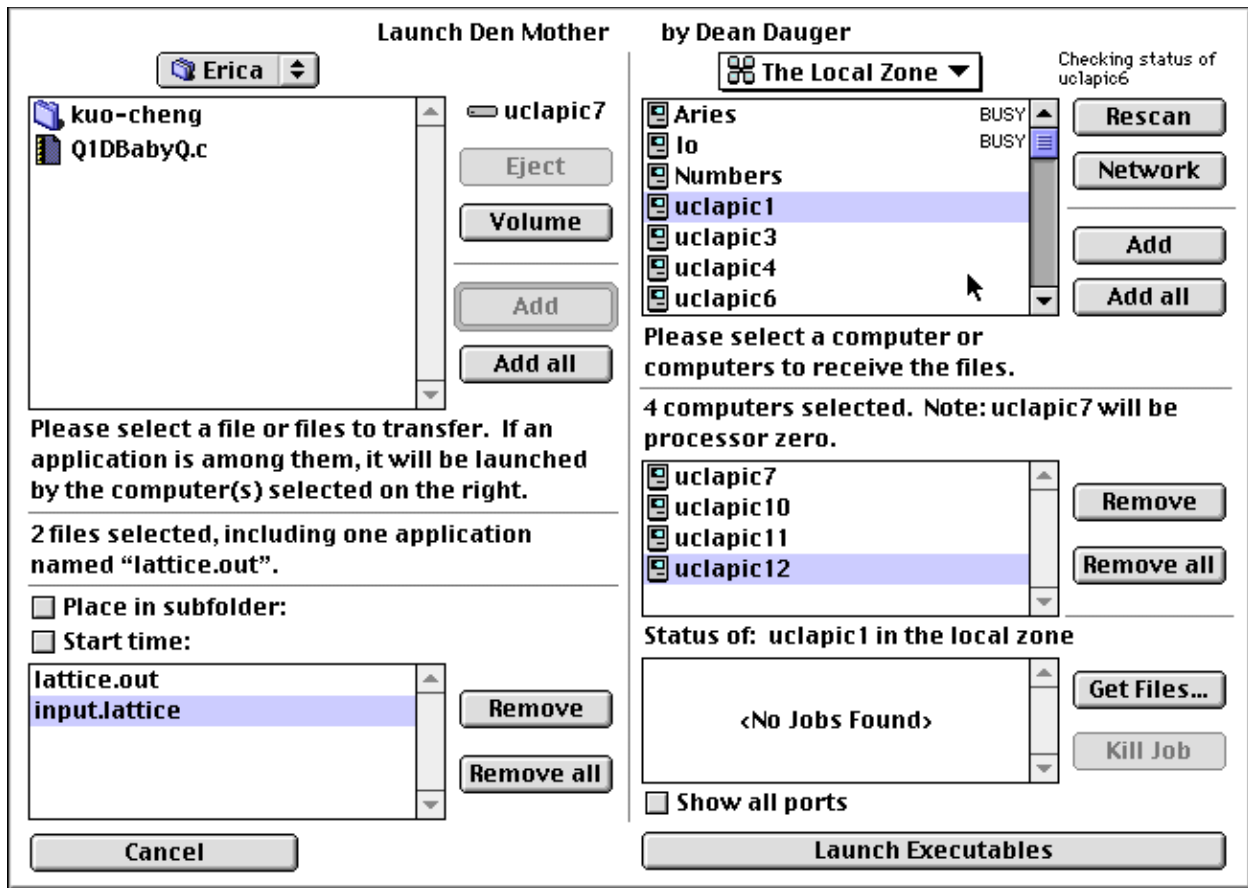


Figure 3. Launch Den Mother utility dialog box.

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 3, 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 3, 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.

## Manual Execution

Parallel applications can also be started manually. This is necessary if the Macintoshes are running a system earlier than MacOS 8.0, or if TCP/IP is being used and the machines are not accessible via AppleTalk.

MacMPI requires that the master node have a file called `nodelist` present in the same directory as the executable. If the parallel job is started manually, this file must be created by the user. (The Launch Den Mother utility creates this file automatically.) This is a straight text file, with different versions for Apple Talk and TCP/IP.

For AppleTalk, the first line contains a port name. If the name `ppc_link` is used, then the slave nodes do not need to have a host file. (If some other port name is used, then the slave nodes need to have a nodelist file which contains only the port name.) The second line contains the name `self`. This name is required only if the cluster contains a single processor. Finally the remaining lines consist of a list of computer names and zones, one name per line, in the form:

```
computer_name@zone_name
```

If there is only one zone in the AppleTalk network, the zone names are omitted. The names cannot have trailing blanks. A sample nodelist file is shown in Table III.

For TCP/IP, the first line contains a port number. If the number 5013 is used, then the slave nodes do not need to have a host file. (If some other number is used, then the slave nodes need to have a nodelist file which contains only the port number.) The second line contains the name `self`. This name is required only if the cluster contains a single processor. Finally the remaining lines consist of a list of ip addresses, one address per line, in dotted decimal form:

```
12.13.14.15
```

The addresses cannot have trailing blanks. A sample nodelist file for TCP/IP is shown in Table IV.

Table III.

## Sample nodelist file for AppleTalk

```
ppc_link
self
uclapic1
BarbH2@Physics-Plasma Theory
fried@Physics-Plasma Theory
```

### Table IV.

## Sample nodelist file for TCP/IP

```
5013
self
128.97.65.90
128.97.65.91
128.97.65.92
```

To start the parallel job manually, one has to copy the executable to each node (via floppy disk, AppleShare, or ftp), and start up (by double clicking) each executable. The master must be started last.

### **Additional Software**

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. 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 users 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`.

When `MacMPI_IP` 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.

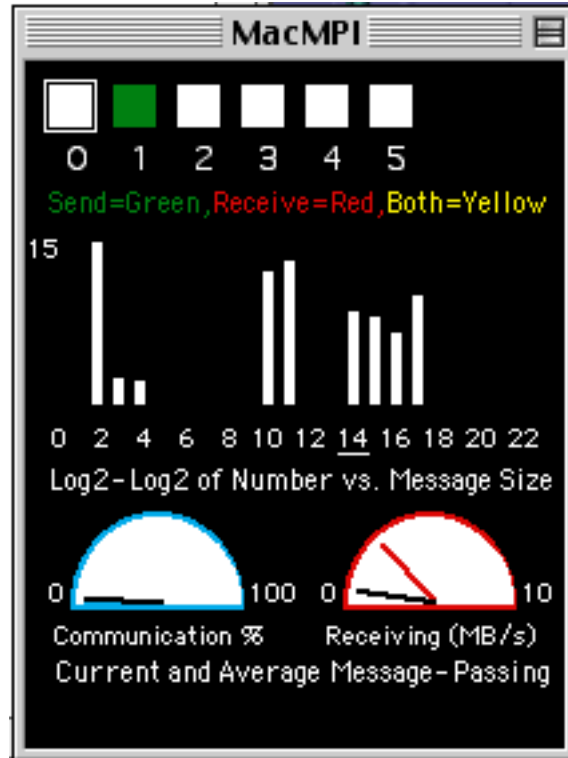


Figure 4. Monitor status window from MacMPI\_IP

We have also made available a number of working applications. The most interesting of these is the Parallel Fractal Demo (including an Interactive version) which runs on an arbitrary number of nodes. This demo is useful for beginners to make sure their network connections are all properly working, as well as for illustrating the speedups one obtains with parallel processing.

## 6. 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 currently using the AppleSeed cluster is a quantum particle-in-cell code, which models multiparticle quantum mechanical problems. 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, and is the thesis work of Dean Dager. This is a typical, underfunded student project, whose current problem size makes it highly appropriate for the Mac cluster. In addition, this project makes 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.

Project AppleSeed is also connected to the Visualization Portal [13] 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 bringing small details to one's attention.

## 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, which use 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.

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 MacOS? 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 Mathematica, which run better on the Macintosh G3 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 MacOS. 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 MacOS, such as Fortner's graphics packages or IDL or QuickTime, or to programs we use for presentation, such as ClarisWorks or Microsoft Word. Finally, the MacOS has encouraged us to write software to a higher standard, that has more of a Mac "look and feel" (such as the Launch Den Mother).

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, Launch Den Mother. 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 [14].

What are the problem areas? The most common failure has nothing to do with the Macintosh computers: it is the network. Symptoms of this failure are nodes appearing and disappearing in the list of available nodes in the Launch Den mother utility, errors returned by MacMPI indicating a node is unreachable, or unexpectedly slow performance. Although such errors can be caused by bad cables or bad networking hardware, the most common cause is a mismatch in the 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. If problems are occurring, one should first check the actual duplex settings. Generally switches have lights on their front panels indicating whether full or half duplex is being used. More sophisticated switches also have software which will tell you the settings of each port and if a high number of errors are occurring. Ethernet adapters sometimes also have lights and software that can tell you the settings. If they are mismatched, it is sometimes necessary to reset the switch and/or the Macintosh to fix it. On less sophisticated switches, the only way to reset them is to pull the plug. 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.

Another problem area is the generation of remote Fortran or C run time errors by running applications, especially if they happen on computers which are located behind closed doors. Some run time errors gracefully abort, but they require user interaction to terminate (e.g, Absoft run-time errors will say, "Hit carriage return to continue"), which may not be possible on a remote machine. Others are less graceful and hang the machine. The Apple Network Assistant software allows one to click OK or reboot a remote Macintosh, if it is not "too dead." However, Network Assistant can be very invasive of the computer owner's privacy, and some people do not wish to have it installed on their machine. We do not have any good solutions for these problems, except to encourage students to debug their codes on the sub-clusters which are located together and only run on remote machines after the code has been tested. Nevertheless, unexpected errors still happen, especially with student written software.

The future continues to look bright. The Macintosh G4 has a vector co-processor (designed for multimedia) that can calculate at a rate of 4 GFlops, single precision only (1 GigaFlop=1000 MegaFlops). We have written some sample codes using this processor and are beginning to investigate its use in our scientific codes. Networking is also becoming cheaper and faster, with Gigabit networks and FireWire two interesting candidates to improve performance. Finally, with MacOS X, a large body of Unix-based software should also be available, including full implementations of MPI. At least one company, MPI Software Technology [15] has announced support for Apple Orchards, which includes a fully compliant and optimized version of MPI for MacOS X.

## **Acknowledgements**

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[16]. We wish here to acknowledge subsequent help given to us by Steve Zimmer, Stan Ng, Ernie Prabhakar, and John Tucker from Apple, Frank Tsung and Joan Slottow 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.

## 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] See [ftp://ftp.apple.com/developer/Tool\\_Chest/OS\\_Uutilities/](ftp://ftp.apple.com/developer/Tool_Chest/OS_Uutilities/)
- [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] See <http://www.oac.ucla.edu/portal/>.
- [14] Dennis Taylor, "Apples are the Core of These Clusters," IEEE Concurrency, vol. 7, no. 2, April-June, 1999, p. 7.

[15] See <http://www.mpi-softtech.com/>

[16] 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.