

Numerically-Intensive "Plug-and-Play" Parallel Computing

Dean E. Dauger
Dauger Research
<http://daugerresearch.com/>

Viktor K. Decyk
Department of Physics
University of California, Los Angeles
<http://exodus.physics.ucla.edu/>

Abstract

At UCLA's Plasma Physics Group, we have been successful in building and using a numerically-intensive parallel computing cluster using Power Macintosh hardware and the Macintosh Operating System. Our solution makes the problem of building and operating a parallel computer far easier than using other technologies, allowing the user, without expertise in the operating system, to efficiently develop and run parallel code. That advantage enables the user to most effectively advance scientific research. At the same time, by achieving over 20 Gigaflops on 16 400-MHz G4s, our team has proved the computational potential of the underlying PowerPC hardware. The ongoing widespread deployment of OS X, a Unix-based Mac OS, will provide scientists access to the best tools of the Mac and Unix in one computing solution. In the midst of this development, clustering Macs is poised to become a technology that will move parallel computing into the mainstream. For details: <http://exodus.physics.ucla.edu/appleseed/>

I. Introduction

To answer the need for accessible computing power, cluster computing is becoming an increasingly popular suggestion. Some find inspiration in the proliferation of desktop computing, while others seek that solution because they find access to large supercomputing centers to be difficult or unattainable. Both are led to ask if desktop machines can be combined to satisfy their computational needs. In this article, we describe our approach to cluster computing and demonstrate what we use to operate it.

The increase of desktop computing power in the last decade has made that idea a practical possibility. That

increase has largely been an indication of a trend identified as Moore's law. During that period, the price to performance ratio of desktop computing hardware has been decreasing, enough to suggest that such cluster computing solutions could easily have a price benefit compared to large supercomputers. That observation became an inspiration for a new form of parallel computing.

One of the first embodiments of that idea was the "Beowulf"-style cluster computing introduced in the mid-1990's. [1] Beowulf clusters used a parallel computing message passing library with the Linux operating system. As Beowulf developed over the years, it has been implemented on a few different hardware architectures such as Alpha and PowerPC, but Intel hardware has clearly become the most popular choice for Beowulf clusters. Also, Message-Passing Interface (MPI), [2] standardized by manufacturers of parallel computers used at supercomputing centers, has become the primary application programming interface (API) for message passing between nodes of such clusters. MPI has become a dominant industry standard, and many MPI implementations are available under open source license. Proponents of that approach often quote only the cost of open source code (free) with commodity Intel hardware (commodity of the shelf (COTS)) in their price to performance ratios.

Cost seems to be a major selling point for proponents of Beowulf clusters. What their proponents fail to address, however, are other costs incurred due to the fundamental nature of Beowulf hardware and software. Beowulf clusters can take significant time to build and to maintain. For novices, those tasks are especially daunting. Except for the top experts in the field, system administrators and other specialists spend weeks or months just assembling the hardware and installing the software. Those people also must spend time updating and fixing the software and hardware if the cluster fails,

because of a breakdown, security breach, software incompatibilities, or other reasons.

Since no one controls the complete environment, incompatibilities between hardware manufacturers and subtle differences in the operating system often become sufficient to cause failure. Low-cost COTS hardware can be inconsistent or unreliable, making the cluster fail to operate. Because they must sell at commodity prices, commodity hardware manufacturers cannot afford to bear the responsibility to make sure their hardware reliably operates in a user's permutation of hardware and software. The user must compensate for such problems, increasing the end-user's cost.

Linux-based operating systems, while being open source, are also complicated. Because they are open source, the software components have a variety of different authors of varying quality. That fact easily results in software mismatches while none of them is obligated to fix bugs or give official support. As in hardware, those responsibilities must then be assumed by the users of the software. Tracking down such problems is time consuming and difficult and requires paid, difficult-to-find, and therefore expensive, expertise (either hired in-house or from a consulting company).

In the end, those users learn that Beowulfs can be fragile. Hence, once their Beowulf is working, users often resist making any adjustments, much less an operating system upgrade, for fear of breaking the application. New hardware generally is not integrated with old.

Those practical problems severely limit Beowulf's potential acceptance by mainstream users. The Beowulf community does not appear to recognize those and other practical issues such as accessibility. For example, an operational Beowulf cluster is controlled at a command line. The Beowulf user-interface has remained at a command-line level for years and shows little sign of improving. Thus, it is evident that few computer scientists are interested in producing a tool for the end user.

It is time to move parallel computing out of the realm of experts and into the mainstream. A development like that would allow parallel computing to have a greater impact for the end user. This alternative to Linux-based and other forms of clustering must clearly have the potential to accomplish this technology transfer. It must be more productive and cost-effective by requiring only a minimum of expertise to build and operate the parallel computer. For example, command-line access *should not be necessary at any time*. The simplicity and straightforwardness of this solution is just as important as its processing power because power provides nothing if it cannot be used effectively. This solution would provide a better price to performance ratio and a higher commitment

to the purpose of such systems: provide the user with large amounts of *accessible* computing power.

At UCLA's Plasma Physics Group, we have been providing a solution that meets those criteria since 1998. It is based on the Macintosh Operating System (Mac OS) using PowerPC-based Macintosh (Power Mac) hardware; we call it an "AppleSeed"-type cluster. [3] We have seen the price-of-hardware to performance ratio of Power Macs to be at least as good as for Intel hardware, but the simplicity of using this technology easily tips the scales in AppleSeed's favor. In April 2001, new software debuted that streamlined the user experience and added numerous new features. That development adds features to AppleSeed clusters that are common on Beowulf clusters.

We have extended the Macintosh's famous ease-of-use to parallel computing, and we have extensively used those technologies for research in physics. Graduate student researchers in physics and their professors are typically not interested in computer science issues and would prefer to spend a maximum of their time researching physics. Thus, our efforts have been focused on *both* performance and streamlining the user experience. In the following, we describe how we build an AppleSeed cluster and demonstrate what we use to operate it. By describing the experience of using the cluster in application to particle-in-cell (PIC) codes, we show what it is like to use and what we achieve with it. Not only do we achieve high-performance results, but we also perform the research we set out to accomplish and perform it most effectively. Finally, we briefly describe what we see for the future of this type of cluster computing, also in light of upcoming changes in the platform.

II. The Cluster

A. Building a Mac Cluster

The following paragraphs completely define the components and procedures for setting up an AppleSeed-type cluster:

Building an AppleSeed cluster begins by collecting the hardware: Power Mac G3s or G4s, one Category 5 Ethernet cable with RJ-45 jacks per Mac, and an Ethernet switch. The latest Power Mac models have either Fast (100BaseT) or Gigabit Ethernet, so a switch of either type with at least as many ports as there are Macs functions well. For each Mac, one end of a cable plugs into the Ethernet jack on the Mac and the other end to a port on the switch.

The required system software is a simple matter. Macs (as of this writing) come preinstalled with Mac OS 9 or later. The latest Mac clustering software needs CarbonLib 1.2 or later, which may or may not already be

present. The latest CarbonLib is available for free download from Apple. [4]

Configuring the Macs generally involves making sure each Mac has an working Internet connection and a unique name, specified in the TCP/IP and File Sharing control panels, respectively. If the Macs are on an isolated network, manually configuring the TCP/IP control panel to use a unique IP address from 192.168.1.1 to 192.168.1.254 is sufficient. (On the Plasma Physics cluster, we set the sleep time to Never in the Energy Saver control panel to prevent the Mac from going to sleep while running a code.)

Finally, a software package called Pooch is used to operate the cluster. A free demo is available for download. [5] Running the installer on a hard drive of each Mac completes the parallel computer.

The reader should deduce two major points from its simplicity of the above description. First, the time spent by the end-user is short: the typical time required per node is a few minutes. Second, the absence of further details about the cluster expresses how reliably it tolerates variations in configuration while interfacing and operating with hardware and software. The hardware need not be identical. The network interfaces can vary (100BaseT, 10BaseT, Gigabit, IrDA (infrared), Airport (wireless)).

Motherboards can be different (G3s of any speed, G4s of any speed, multiple processors, desktops, portables). Also, the above installation and configuration easily coexists with almost all other applications because the existence of extra applications and system extensions are generally unimportant to cluster functions. Even the operating system on some cluster machines can be any variant of OS 9, a Mac OS descendent, while others are running the fundamentally-different, Unix-based OS X. The AppleSeed design has great implications for the mainstream because end users (including dedicated physicists) have little concern for such details.

B. Running a Mac Cluster

For the purpose of testing an AppleSeed-type parallel computer, the AltiVec Fractal Carbon demo, a demonstration parallel application, is available for free download. [5] This demonstration of high-performance computing can run on a single-node as well.

The user runs this application in parallel by selecting New Job... from the File menu of Pooch. This action opens up a new Job Window. The user may drag the AltiVec Fractal Carbon demo from the Finder to this Job Window, depicted in Figure 1.

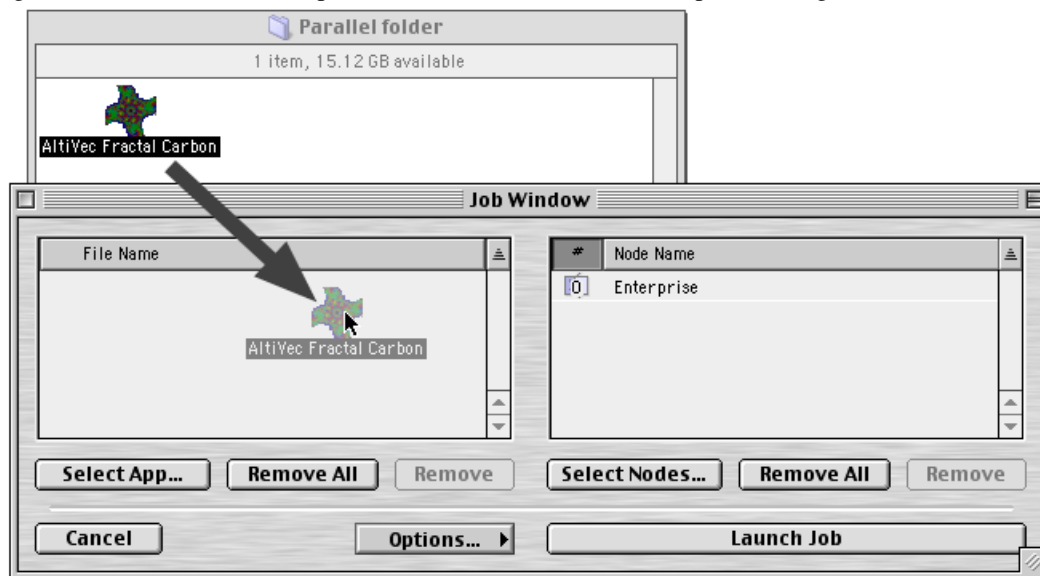


Figure 1. To set up a parallel computing job, the user drags a parallel application, in this case the AltiVec Fractal Carbon demo, and drop it in the Job Window of Pooch.

Next, the user chooses nodes to run on. By default, Pooch selects the node where the job is being specified. To add more, the user clicks on Select Nodes..., which invokes a Node Scan Window, shown in Figure 2. 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 3. Pooch should now be distributing copies of the parallel application to the other nodes and initiating them in parallel. Upon completion of its computational task, the demo then calculates its achieved performance, which should be significantly greater than single-node performance.

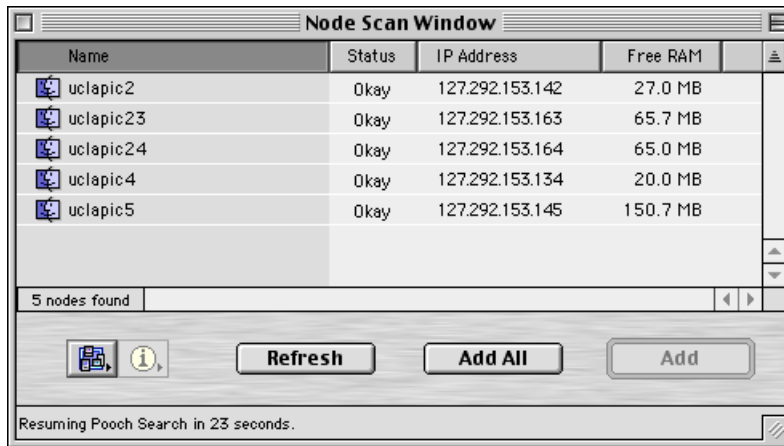


Figure 2. Selecting nodes is performed using the Node Scan Window, invoked by clicking on Select Nodes... from the window in the previous figure.

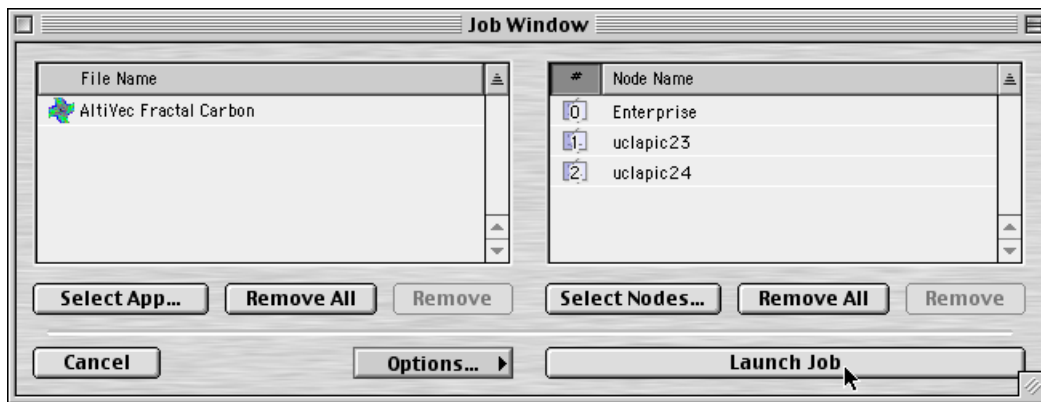


Figure 3. With the job ready, it begins with one more click.

Pooch debuted in April 2001. It is designed to provide users maximum accessibility to parallel computing. Like its predecessor, Pooch never requires a command-line instruction. It can organize the job's files into subdirectories on the other nodes and retrieve files on those nodes containing output from completed jobs. But Pooch adds new features to AppleSeed clusters. Some of these new features were inspired by those common to Beowulf clusters and large supercomputers. 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. And all of its functions can operate securely via the Internet.

In addition, Pooch provides new features not commonly available on Beowulf clusters. This software provides discovery services, that is, a service to discover the existence and addresses of other nodes on the network, via TCP/IP on any subnet of the Internet. It has the ability to determine up-to-the-minute information about nodes, including their availability and capability. Finally, Pooch's components have the capability of automatic

node discovery and selection.

III. Results

A. Network Performance

As on other parallel computing platforms, a library for message passing between nodes is available. We call the 45 routine subset of Message-Passing Interface (MPI) supported on the Macintosh platform MacMPI. MacMPI, freely available from the AppleSeed site at UCLA Physics, is a wrapper library that calls the Mac OS networking APIs. The latest version of MacMPI is called MacMPI_X and uses Apple's latest Open Transport implementation of TCP/IP. [6]

Using MacMPI, we achieve excellent network performance comparable to other networking implementations. We use a ping-pong and swap benchmark (where pairs of processors exchange packets of equal size) to probe that performance. With 100BaseT Ethernet and Power Mac G3/350 hardware we achieve near peak speed of 100BaseT for large messages. Figure 4

shows that the high bandwidth is achieved for message sizes of around 4096 words or larger. The best bandwidth measured is about 90% of the peak theoretical speed of 100BaseT hardware.

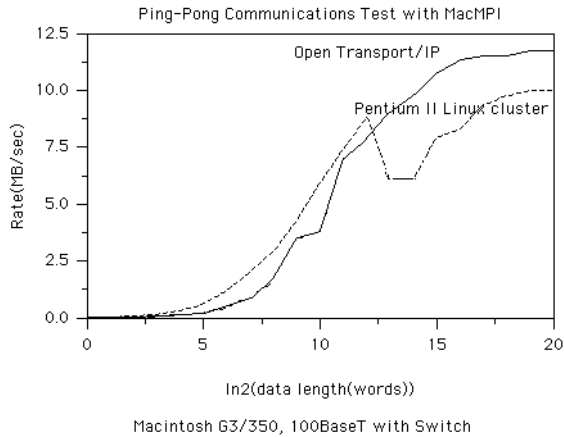


Figure 4. Networking benchmark using TCP/IP through MacMPI on Power Mac G3 hardware connected via a Cisco Fast Ethernet switch. The performance on a Pentium II Linux cluster is also shown for comparison.

Apple's most recent versions of their Power Mac G4 hardware also come with built-in Gigabit Ethernet ports. Running the ping-pong test using two such Macs connected via a crossover Ethernet cable are shown in Figure 5. These results show over three times the

performance of 100BaseT.

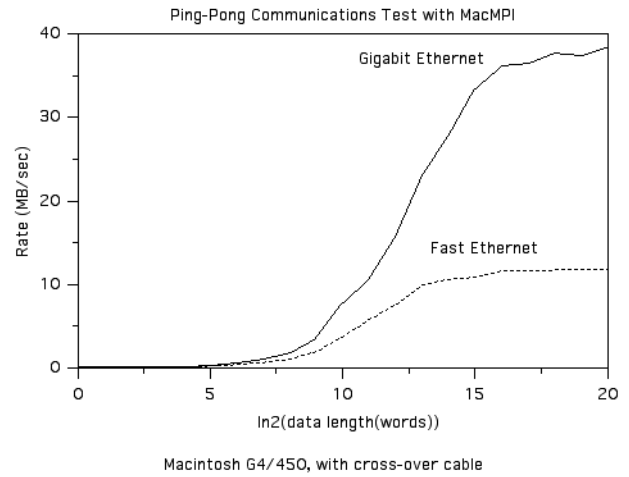


Figure 5. Ping-pong networking benchmark using two G4/450's using Gigabit networking on a crossover cable. The previous 100BaseT results are shown for comparison.

B. Parallel Computing Performance

The performance of the 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.

3D Particle Benchmarks

Computer Type	Push Time (nsec)	Loop Time (sec)
Mac G4/450, IP cluster, 8 proc:	772	2756.9
Mac G4/450, IP cluster, 4 proc:	1928	6715.3
Mac G4/450, IP cluster, 2 proc:	4676	16234.3
Cray T3E-900, w/MPI, 8 proc:	1800	6196.3
Cray T3E-900, w/MPI, 4 proc:	3844	13233.7
IBM SP2, w/MPL, 8 proc:	2104	7331.1

Table I. The above 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 taken to update one particle's position and deposit its charge (but without the field solve), for one time step. Loop Time is the time to run the entire simulation minus the initialization time.

Both measurements include the communications time necessary to complete the tasks.

The AltiVec Fractal demo, used for the example job in Section II, uses the Single Instruction Multiple Data (SIMD) instruction unit known as the Velocity Engine (or AltiVec) to compute a color graphical representation of a Mandelbrot-style fractal. It decomposes the problem according to scan lines of the image. Each processor is assigned a different set of scan lines, and the results are collected for display at node zero. That demo has achieved

over 6 Gigafllops on a pair of dual-processor G4/450's and over 20 Gigafllops on the UCLA Department of Statistics' 16 G4/400 cluster.

Results for the large 3D benchmark described in Ref. [7,8] are summarized in Table I. 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.

In addition, a recent (February 2001) milestone was set with AppleSeed software. We were able to run a 100 million particle 3D electrostatic PIC simulation on an eight-node Macintosh G4/450 dual processor cluster. Since the Plasma Physics cluster at UCLA did not have machines large enough to do the job, we used Bedros Afeyan's Polymath 2000 cluster instead, [9] which had 1 GB of memory per node. The total time was 17.8 seconds per time step, with a grid of 128x128x256. As of this writing, the cost of such machines was less than \$2500 per node. It was only six years ago that such calculations required the world's largest supercomputers!

IV. Real-World Application

A. Flexibility

The inexpensive and powerful cluster of Power Mac G3s and G4s has become a valuable addition to the UCLA Plasma Physics group. We use it to introduce new members of our group to parallel computing and run large calculations for extended periods. Although some problems can only run on the supercomputer centers due to their large size, the turnaround time for jobs on the cluster is often much shorter than those in the supercomputer centers because we do not have to share this resource with the entire country. That feature makes the cluster very appropriate for small- and medium-sized jobs, allowing the supercomputing centers to focus on the largest jobs, a task for which they are much better suited.

The solution at UCLA Physics is fairly unique in that half of the nodes are not dedicated for parallel computing. We purchase high-end Macs and devote them for computation while reassigning the older, slower Macs for individual (desktop) use and data storage. Thus, we are reusing the Macs in the cluster, making for a very cost-effective solution to satisfy both our parallel computing *and* desktop computing needs. The AppleSeed-type cluster is unique in this regard, made possible by how tolerant the software is of variations in configuration.

In addition, the flexibility of the AppleSeed-type cluster allows us to redirect computational resources very quickly within the group. That ability is useful for unfunded research or exploratory projects, so we can better prepare for an official proposal later. If one investigator needs to meet a short deadline, such as for a conference, that person can ask the research group, borrow their desktop Macs, and combine them with the dedicated Macs for one large job or many smaller ones.

The presence of the cluster has encouraged new members of our group 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, beginning parallel programmers 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 larger 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. We are able to display on desktop machines the results of calculations made elsewhere in the cluster. That even allows us to study a simulation partway through the calculation, much like checking the oven before the turkey is done. Checking for mistakes early allows one to save a great deal of computation time that might otherwise be wasted.

B. Parallel Code Development

So that the Plasma group's physics researchers (students, post-docs, etc.) can maximize their time studying physics, we have added enhancements to MacMPI that make it easier for them to develop parallel programs. Those enhancements are beyond MacMPI's basic message-passing functions.

One of these is the monitoring of MPI messages, controlled by a monitor flag in MacMPI. This flag may be set to log every message sent or received. In its default setting, a small monitor window appears, shown in Figure 6. 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. The moment such a problem occurs, a particular color pattern is immediately visible to the user, who can then apply the new information to debugging the code.

The monitor 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 draw the user's attention to the length of the messages the code is sending.

Other features of this window include room at the bottom of the status window for a one-line message of the user's choice. That feature is useful for displaying the current procedure or time step being executed.

Two dials are also shown in MacMPI_X's monitor window. One dial shows the approximate percent of time spent in communication, a quantity that should be minimized. Both the time average over the whole run and

the instantaneous value are shown. The second dial is a speedometer which shows the average and instantaneous speeds achieved during communication. While approximate, those indicators have been invaluable in revealing problems in the code and the network.

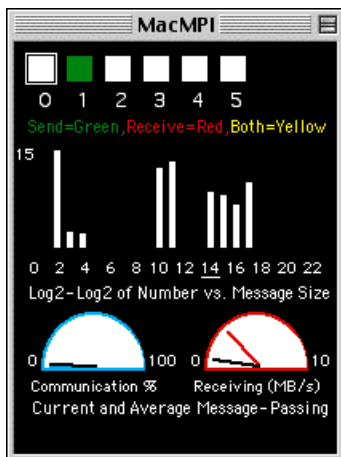


Figure 6. The monitor window of MacMPI_X, which keeps track of statistics about the execution of the running parallel application.

C. Physics

The PIC codes at the UCLA Plasma Physics Group are used in a number of High-Performance Computing projects, such as modeling fusion reactors [10] and advanced accelerators [11]. For those projects massively parallel computers are required, such as the 512-node Cray T3E at NERSC. However, the group has found it very convenient to perform research projects on more modest but user-friendly parallel machines such as the Macintosh clusters. The resources of the large computers can then focus on large problems for which they are better suited.

Simplifying the problem of building, operating, and maintaining a parallel cluster allows our group to use its cluster to focus on physics research. The AppleSeed cluster at UCLA Physics is primarily used for plasma physics projects. One of those is the Numerical Tokamak Turbulence Project. The goal of that project is to predict plasma and heat transport in fusion energy devices. Recent calculations have concentrated on studying various mechanisms of turbulence suppression in devices such as the Electric Tokamak, under construction at UCLA. [12] The researchers involved use AppleSeed for smaller problems when they need fast turnaround. They also run the same code on remote supercomputer centers for their largest calculations.

A second project that used this cluster is a quantum particle-in-cell code, which models multiparticle quantum mechanical problems. That code, a recently completed

doctoral work, used a semiclassical approximation of Feynman path integrals to reduce the calculation to a large number of classical charged particles moving in an electromagnetic field. [13] That was a typical underfunded student project, whose problem size makes it highly appropriate for the Mac cluster. The largest runs of that work, some tracing over 100 million classical paths per time step, were performed on the UCLA Department of Statistics' dedicated 16-node G4/400 cluster. [14] Those runs studied the quantum effects in hot plasmas, such as those inside the sun. [15] In addition, that project makes use of sophisticated interactive diagnostics developed on the Macintosh, including tools that generate QuickTime movies playing quantum wavefunctions as sound.

We were also highly successful in applying the cluster's computational power to a classroom environment, but the experience serendipitously led to a plasma physics conference presentation. At UCLA, Physics 260, entitled "Exploring plasmas using computer models", is a computational plasma physics course where the students learn about plasmas and operate and analyze data from a plasma physics code. In Fall 2000, Dawson, et al, assigned students to use the cluster for their course work. The code, called Parsec and written by John Tonge, was a 3D fully electromagnetic particle-in-cell code suitable for investigating the physics of plasma confinement in a levitated internal conductor device. Some of the runs contained as many as 50 million particles on up to a 256x256x128 grid. Such assignments would have been impossible without the local cluster. The runs could be started at the end of class on one day, and then the output could be retrieved for analysis at the next class meeting. Because of their large queues and run-time limitations, it was impossible for supercomputing centers to meet such a need. The course of the student work led to an academic contribution to the field. [16] The researchers involved consider their experience an indication of the power of the AppleSeed approach to physics research and, in their words, call it "a real success story."

V. Future

We see the future for computation on Macintosh being very bright. As with all parallel computing, the problem remains finding the best way to program in parallel. Recent developments bring AppleSeed clusters' feature set on par with other cluster types. However, unlike other cluster types, the AppleSeed solution makes the problem of building and operating a parallel computer easy and therefore enables the user to most efficiently write, debug, and run parallel codes. That advantage is unique to Mac clusters. We believe our work shows that Mac clusters, relative to other cluster types, enable its

users to most effectively advance their scientific research.

By combining parallel computing with a well-established platform like Macintosh, we can use shrink-wrap applications, such as IDL and Mathematica, to visualize the output of our code. We have even used IDL to display the live output of a plasma physics parallel application and display it in the Visualization Portal [17], a large 24' by 8' screen at the UCLA Computer Center. As the physics code ran, it produced a series of output files containing electric potential data and labeled according to their time step. Then an AppleScript was used to monitor the appearance of these files, and feed them at specified intervals via AppleEvents to IDL. IDL then presented the data, frame by frame, on a Power Mac connected to the Visualization Portal's projectors. That set up is a proof-of-principle for live presentation of simulation data from a running parallel computation. An experimentalist, unconcerned with computational details, could use such a solution in tandem with a physical apparatus.

The Power Mac G4 hardware comes with a SIMD instruction unit named AltiVec, with a peak speed of eight flops per clock. We intend to better utilize that powerful processor with our physics codes, starting with the time-critical parts of the physics code, as others at NASA Langley Research Center have done. [18] Also, many of the new Power Mac G4s come contain dual G4s. We have modified portions of our codes to use that feature, and we intend to continue such optimizations.

We also expect to make enhancements to Pooch in the coming months. We intend to enhance AppleSeed clusters' ability to interface with existing applications by adding standardized mechanisms for Pooch to respond to commands from other applications. Such features could allow, for example, IDL to direct Pooch to launch a parallel physics simulation code, which then supplies data for live display in IDL. That ability would allow for greater automation of the above proof-of-principle demonstration. We plan on adding synergistic features to MacMPI and Pooch to report live diagnostic information to remote machines. We intend to add automatic node selection features, based on node load and capabilities, to enable metacomputing-grid possibilities, much like that promised by Globus. [19] And, of course, Pooch will evolve in response to user feedback.

The Macintosh platform is in the midst of a change. Mac OS X, the future of the Mac OS, is based on Unix. [20] Many Unix applications are already being ported, or have been ported, to the new operating system. The latest AppleSeed software runs on OS X. As we have done with IDL, those Unix applications could be combined with parallel computing on OS X. Or they could be made into

parallel applications themselves. Even libraries used for Beowulf's message passing, could in principle be ported to OS X. As of this writing, LAM-MPI, with modification, is functioning on X. Combining the best of Unix with the best of the Mac is upon us. Macintosh clusters have the potential to be a boon for scientific and computational work of all kinds.

VI. 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] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, *MPI: The Complete Reference* [MIT Press, Cambridge, MA, 1996]; William Gropp, Ewing Lush, and Anthony Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface* [MIT Press, Cambridge, MA, 1994].
- [3] V. K. Decyk, D. Dauger, and P. Kokelaar, "How to Build An AppleSeed: A Parallel Macintosh Cluster for Numerically Intensive Computing," *Physica Scripta* T84, 85, 2000.
- [4] <http://asu.info.apple.com/>
- [5] See <http://daugerresearch.com/>
- [6] <http://developer.apple.com/techpubs/macosx/Carbon/networkcomm/OpenTransport/opentransport.html>
- [7] V. K. Decyk, "Benchmark Timings with Particle Plasma Simulation Codes," *Supercomputer* 27, vol V-5, p. 33 (1988).
- [8] V. K. Decyk, "Skeleton PIC Codes for Parallel Computers," *Computer Physics Communications* 87, 87 (1995).
- [9] <http://polymath-usa.com/>
- [10] 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).
- [11] K.-C. Tzeng, W. B. Mori, and T. Katsouleas, "Electron Beam Characteristics from Laser-Driven Wave Breaking," *Phys. Rev. Lett.* 79, 5258 (1997).
- [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] <http://dauger.com/DaugerDissertation.pdf>
- [14] Merav Opher, Luis O. Silva, Dean E. Dauger, Viktor K. Decyk and John M. Dawson, "Nuclear reaction rates and energy in stellar plasmas : The effect of highly damped modes", accepted and to be published in *Physics of Plasmas*, 2001.
- [15] <http://www.stat.ucla.edu/research/gSCAD/>
- [16] Chengkin Huang, John Tonge, Jean-Noel Leboeuf, and John M. Dawson, "Particle Simulations of Plasma Confinement in a Levitated Dipole", *International Sherwood Fusion Theory Conference, Paper 1C46*, April 2001.
- [17] <http://www.oac.ucla.edu/portal/>
- [18] http://ad-www.larc.nasa.gov/~cah/NASA_G4_Study.pdf
- [19] <http://www.globus.org/>
- [20] <http://www.apple.com/macosx/>