Accepted for publication in the IEEE Cluster 2003 Proceedings

“Plug-and-Play” Cluster Computing using Mac OS X

Dean E. Dauger
Dauger Research, Inc.
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, to achieve accessible computational power for our research goals, we developed the tools to build numerically-intensive parallel computing clusters on the Macintosh platform. Our technology maximizes productivity because it is designed to allow the user, without expertise in the operating system, to most efficiently develop and run parallel code, enabling the most effective advancement of scientific research. Collaborating with USC and NASA’s JPL, our team has demonstrated the performance and scalability potential of Mac clusters by achieving over 217 Gigaflops on 33 XServes and over 233 Gigaflops on 76 Power Mac G4s. But we find that the usability and reliability of the technology is as important as its performance. The ongoing dissemination of OS X, a Unix-based Mac OS, is providing the best tools of the Mac and Unix in one computing solution. With this development, Mac clustering is becoming the technology that will move parallel computing into the mainstream. See: http://exodus.physics.ucla.edu/appleseed/ and http://daugerresearch.com/

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 smaller machines can be combined to provide sufficient access to computational power. In this article, we describe the approach to cluster computing technology that we find best achieves these goals for scientific users and, ultimately, for the mainstream end user.

One approach, introduced in the mid-1990’s, used a parallel computing message passing library with the Linux operating system and became known as “Beowulf”-style cluster computing. [1] The Message-Passing Interface (MPI) [2] has become a dominant industry standard [3], and many MPI implementations are available under open source license. Proponents of the Beowulf approach often quote only the cost of open source code (free) with commodity Intel hardware (commodity off the shelf (COTS)) in their price to performance ratios.

Even though cost appears to be their greatest motivation, proponents of Beowulf-style Linux/Intel clusters fail to address the costs of construction, maintenance, and repair incurred due to the fundamental nature of Beowulf hardware and software.

Since no one controls the complete environment, subtle inconsistencies between hardware manufacturers and in the operating system often become sufficient to cause failure. Because they must sell at commodity prices, COTS hardware manufacturers cannot afford to bear the responsibility to make sure their hardware reliably operates in a user’s permutation of hardware and software. Because they are open source, the components of Linux and Linux-based software have numerous authors of varying quality, easily resulting in software incompatibilities, while no one is obligated to fix bugs or give official support.

In the end, users learn that Beowulfs can be fragile: users often resist making any adjustments, even to the *kernel version*, for fear of breaking the application. Tracking down such problems is time consuming and difficult and requires paid, difficult-to-find, and therefore expensive, expertise. The users of Beowulf hardware and software are forced to assume the responsibilities that COTS manufacturers and open-source authors do not, significantly increasing the end-user’s cost.

The Beowulf community does not appear to recognize those problems and other practical issues such as accessibility. Since the graphical user-interface (GUI) was introduced to the consumer almost twenty years ago, the mainstream user has come to expect a GUI. Meanwhile, the Beowulf user-interface has remained at a command-line level for years and shows no indication of improving. This disparity between the computer science community and the mainstream indicates 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, enabling parallel computing to have a greater impact for end users. Beowulf has taught us that the solution must be productive and cost-effective by requiring only a minimum of time and expertise to build and operate the parallel computer. Specifically, *the time needed to assemble and run a working cluster should be minimized.* 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 total price to performance ratio and a higher commitment to the original purpose of such systems: provide the user with large amounts of *accessible* computing power.

At UCLA's Plasma Physics Group, we have been using a solution that meets those criteria since 1998. It is based on the Macintosh Operating System using PowerPC-based Macintosh (Power Mac) hardware; we call it a Mac cluster. [4] The simplicity of using Mac cluster technology makes it the most cost-effective solution for all but the largest calculations. In our ongoing effort to improve the user experience, we continue to streamline the software and add numerous new features. With OS X, the latest, Unix-based version of the Mac OS, we are seeing the convergence of the best of Unix with the best of the Mac.

Our goal is to maximize the benefits of parallel computing for the end user. Our approach is unique because, while other solutions seem to direct little, if any, attention to usability and reliability, we find such issues to be as important as raw performance. We believe the ultimate vision of parallel computing is (rather than merely raw processor power) when the technology is so reliable and trivial to install, configure, and use that the user will barely be aware that computations are occurring in parallel. This article presents our progress in building the "plug-and-play" technology to make that vision come true.

We have extended the Macintosh's famed ease-of-use to parallel computing. While extensively applying those technologies for research in physics, our efforts have been focused on *both* performance and streamlining the user experience. In the following, we describe how we build an Mac cluster and demonstrate what we use to operate it. By describing the experience of using the cluster in application to our group's 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 have seen in the evolution of this type of cluster computing, in light of ongoing transitions in the platform.

II. The Cluster

A. Building a Mac Cluster

The following paragraphs completely define the components and procedures for setting up a Mac cluster:

Building an Mac cluster begins by collecting the hardware: Power Mac 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.

System software is a simple matter: Macs come preinstalled with Mac OS X. Configuring the Macs generally involves making sure each Mac has an working Internet or IP connection and a unique name, specified in the Network and Sharing System Preferences, respectively.

Finally, a software package called Pooch is used to operate the cluster. A download version is available. [5] Running the installer on a hard drive of each Mac completes the parallel computer. Software installation on a node takes only a few seconds, a brevity unheard of among other cluster types.

The reader should deduce two major points from its simplicity and efficiency of the above description. First, the time spent by the end user, at less than a few minutes, is short. (By comparison, specialists, except for the top experts in the field, spend weeks or months assembling and installing a Beowulf cluster.)

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)). Computing hardware can be different (G3s of any speed, G4s of any speed, multiple processors, desktops, portables, rack-mount XServes). There is no permanent head node; a node is designated node zero for the duration of the job only. 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.

This design is very robust. When we demonstrate the technology to others, we often ask the audience to volunteer their computers to add to the Mac cluster. The cluster runs despite the wide configuration variety of these volunteer machines. The reader should note that the following capability is unique in clustering: Not only can the operating system on different nodes be different versions of the Unix-based OS X, but the operating system on some cluster machines can be any variant of OS 9, a classic Mac OS descendent. The Mac cluster design has great implications for the mainstream because end users need not be concerned with such details.

B. Running a Mac Cluster

For the purpose of testing a Mac clusters, the AltiVec Fractal Carbon demo, a demonstration parallel application, is available for free download. [5] This demonstration of high-performance computing also runs on a single node.

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.

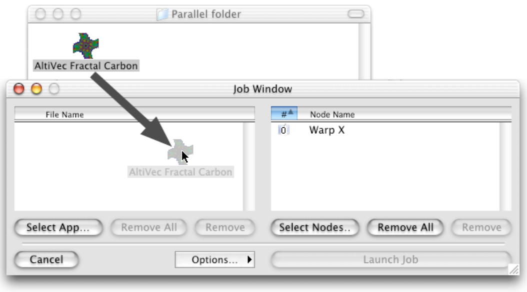

Figure 1. To set up a parallel computing job, the user drags a parallel application, in this case the AltiVec Fractal Carbon demo, and drops it in the Job Window of Pooch.

Next, the user chooses nodes to run in parallel. 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. Double-clicking on a node moves it to the node list of the Job Window. If a machine running OS X has two processors, Pooch can use them as if they were separate nodes.

Finally, the parallel job must be started by clicking on Launch Job. 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.

III. Middleware

A. Pooch

Pooch is a parallel computing and cluster management tool designed to provide users maximum accessibility to parallel computing. The latest version was released in June 2003. Pooch can organize the job's files into subdirectories on the other nodes and retrieve files on those nodes containing output from completed jobs. It can queue jobs and launch them only when certain conditions have been met. It also has the ability to kill running jobs, launching jobs, and queued jobs.

A fundamental difference between Beowulf-style tools and Pooch is: All Pooch operations use dynamically-determined information. Pooch, therefore, does not require an administrator to maintain any static data files about the cluster. In fact, Pooch makes as few assumptions as possible about the cluster configuration, a unique design decision that is key to its flexibility and tolerance. Pooch uses TCP/IP-based services to discover the existence and addresses of other nodes on the network on any subnet of the Internet. On OS X 10.2 and later, Pooch's node discovery implementation uses Service Location Protocol and Apple's new Rendezvous (a.k.a. ZeroConf) simultaneously. [6] Pooch uses encrypted connections to determine up-to-the-minute information about nodes, including their availability and capability. Pooch has even been used to combine nodes at UCLA in Los Angeles, California, with machines in Munich, Germany, 10 000 km apart. Further details are in the documentation available with the distribution.

Pooch supports the widest variety of parallel programming environments, enabled by the convergence of technologies in OS X: Carbon, Cocoa, Mach-O, Unix shell scripts, or AppleScripts. [7] Pooch supports three different Message-Passing Interfaces (MPIs): MacMPI, mpich, and MPI/Pro. [8] Because of OS X, MPIs of such varied histories are all now supported in the one environment.

Pooch features four user interfaces. In addition to its drag-and-drop GUI illustrated above, Pooch's AppleScript interface makes it possible to write automatic and interactive scripts that perform customized job queuing and other cluster operations. A new suite of command-line utilities was introduced in June 2003 to provide a complete command-line interface to Pooch, making it easy for users to log in from other platforms to control cluster operations.

In addition, by sending commands through interapplication messages called AppleEvents, other applications can directly control Pooch to perform Grid-like behavior. While the present incarnation of Globus

[9] and related technologies combine resources on a supercomputer level, our technology combines desktop machines. Unlike Globus and Condor [10], these features are installed, configured, and run using an accessible, easy-to-use interface. The AltiVec Fractal carbon demo and the Fresnel Diffraction Explorer, an optics parallel application, can initiate their own utilization of a local cluster. With only a menu selection, these desktop applications can automatically take advantage of resources elsewhere on the cluster. Such powerful yet easy to use features are the prerequisites for parallel computing to become mainstream.

B. MacMPI

MacMPI, freely available from the AppleSeed site at UCLA Physics, is Decyk's 45 routine subset of MPI implemented using the Mac OS networking APIs. MacMPI_X, the current version of MacMPI, uses Apple's latest Open Transport implementation of TCP/IP. [11]

Using MacMPI, we achieve excellent network performance comparable to other networking implementations. We achieve near peak speed of 100BaseT for large messages. Apple's most recent versions of their Power Mac G4 hardware also come with built-in Gigabit Ethernet ports. Via a crossover Ethernet cable, we see over three times the performance of 100BaseT. On OS X, we are able to compare these results with the performance of open-source MPIs. Further details are on our web site. [12] A new version of MacMPI we call MacMPI_S is being developed using Unix sockets. At present, the Open Transport implementation, at over 50 MB/s, outperforms MacMPI_S's 40 MB/s bandwidth.

IV. Real-World Experience

A. 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.

In 2002, Apple introduced the XServe, a rack-mounted version of a Power Mac G4 meant for server solutions. In collaboration with the Applied Cluster Computing Group at NASA's Jet Propulsion Laboratory, the AltiVec Fractal Carbon demo has achieved over 217 Gigaflops on their 33-XServe dual-processor G4/1000 cluster. [13]

University of Southern California gave our team the opportunity to run the Fractal demo and Pooch on 56 of their dual-processor Power Mac G4/533's plus 20 of their dual-processor Power Mac G4/450's. We achieved over 233 Gigaflops on this cluster. [14] The reader should note that these machines were not meant for cluster work. They were part of a Language Arts undergraduate computer lab, yet they have achieved supercomputer-level results. This result has implications for other university computer labs in the world. The latest version of our software operates on unused, logged-out machines of such computer labs.

In addition, a recent milestone was set with AppleSeed software. We were able to run a 127 million particle 3D electrostatic PIC simulation [15,16] on an four-node Macintosh G4/1000 dual processor cluster. The total time was 17.6 seconds per time step, with a grid of 128x128x256. As of this writing, the cost of these machines is less than $10 000. Very interesting physics can now be done with limited resources. It was only eight years ago that such calculations required the world's largest supercomputers!

B. 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.

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 Mac cluster is unique in this regard, made possible by how tolerant the software is of variations in configuration.

In addition, the flexibility of the Mac 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, 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. The cluster also encourages a more interactive style of parallel programming, in contrast to the batch-oriented processing encouraged by most other cluster types. 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. Checking for mistakes early allows one to save a great deal of computation time that might otherwise be wasted.

C. Parallel Code Development

So that the Plasma group's physics researchers can maximize their time studying physics, we have added enhancements, beyond basic message-passing, to MacMPI that make it easier for them to develop parallel programs.

One of these is the monitoring of MPI messages, controlled by a monitor flag in MacMPI, which can log every message sent or received. In its default setting, a small monitor window appears, shown in Figure 2. In this window, status lights indicate whether the node whose screen is being examined is sending and/or receiving messages from any other node. Green indicates sending, red indicates receiving, and yellow means both. 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.

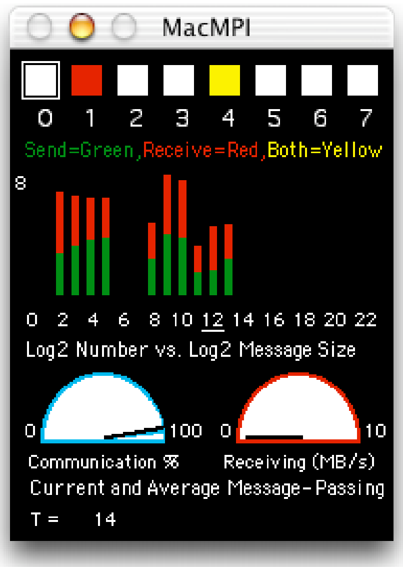
MacMPI
0 1 2 3 4 5 6 7
Send=Green,Receive=Red,Both=Yellow
8
0 2 4 6 8 10 12 14 16 18 20 22
Log2 Number vs. Log2 Message Size
0 100 0 10
Communication %
Receiving (MB/s)
Current and Average Message-Passing
T = 14

Figure 2. The monitor window of MacMPI_X, which keeps track of statistics about the execution of the running parallel application.

The monitor window also shows a similarly color-coded histogram of the size of messages being sent or received. The purpose of this histogram is to draw the user's attention to the length of the messages the code is sending. The two dials in MacMPI_X's monitor window show the approximate percent of time spent in communication and the average and instantaneous speeds achieved during communication. While approximate, those indicators have been invaluable in revealing problems in the code and the network.

D. Plasma Physics and Additional Applications

The PIC codes at the UCLA Plasma Physics Group are used in a number of High-Performance Computing projects, such as modeling fusion reactors [17] and advanced accelerators [18]. 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 and user-friendly parallel machines such as the Macintosh clusters.

Simplifying the problem of building, operating, and maintaining a parallel cluster allows our group to use its cluster to focus on physics research. The Mac cluster at UCLA Physics is primarily used for plasma physics projects. One of those is the Plasma Microturbulence Project. The goal of that project is to predict plasma and heat transport in fusion energy devices. Recent calculations by James Kniep and Jean-Noel Leboeuf have concentrated on studying various mechanisms of turbulence suppression in devices such as the Electric Tokamak at UCLA [19] and the DIII-D tokamak at General Atomics [20]. The researchers involved use the Mac cluster for smaller problems when they need fast turnaround for fast scoping, as well as for production calculations which can be accommodated on our eight-node clusters. These results have been favorably compared to experimental observations of plasma microturbulence characteristics in DIII-D discharges [21].

Outside of UCLA Physics, Professor John Huelsenbeck of UCSD Biology and Professor Fredrik Ronquist of Uppsala University wrote and distributed pMrBayes, a parallel application that performs Bayesian estimates of phylogeny for biology. Their program uses a Markov chain Monte Carlo simulation technique to approximate the probability distribution of trees. While discussing other approaches to run their parallel code, they describe "the simplest method is to use Dauger's program Pooch to control the jobs." [22]

For further details on other success stories using the Mac cluster continue on our web site. [5]

V. Conclusion

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. Mac clusters' feature set is on par with other cluster types and is expanding to grid-like features. However, unlike other approaches, the Mac cluster 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. Our and others' work shows that Mac clusters, relative to other cluster types, enable its users to most effectively utilize their computational resources.

The Macintosh platform is in the midst of a change. The Mac cluster software runs on Mac OS X, which is based on Unix. [7] Many Unix applications already have been ported to the new operating system. Those Unix applications could be combined with parallel computing on OS X or they could be made into parallel applications themselves. On OS X, the Mac cluster software can utilize mpich or other libraries written for Beowulf's message passing, such as MPI/Pro. The best of Unix is being combined with the best of the Mac. Because of their ability to make computational power accessible, Macintosh clusters are uniquely capable of maximizing the impact of parallel computing for scientific and mainstream users.

VI. Acknowledgements

Many people have provided us useful advice over the last two years. We acknowledge help given by Bedros Afeyan from Polymath Research, Inc., Ricardo Fonseca from IST, Lisbon, Portugal, Frank Tsung and John Tonge from UCLA, and the Applied Cluster Computing Group at NASA's Jet Propulsion Laboratory.

VII. 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] Most major supercomputing centers only use MPI for distributed-memory parallel computing. The absence of other message-passing schemes on new hardware is evident at NERSC: http://hpcf.nersc.gov/software/libs/ and at NPACI: http://www.npaci.edu/BlueHorizon/guide/ref.html
[4] 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.
[5] See http://daugerresearch.com/
[6] http://developer.apple.com/macosx/rendezvous/ and http://www.zeroconf.org/
[7] http://www.apple.com/macosx/
[8] See links at: http://daugerresearch.com/pooch/mpi.html
[9] http://www.globus.org/
[10] http://www.cs.wisc.edu/condor/
[11] http://developer.apple.com/techpubs/macosx/Carbon/networkcomm/OpenTransport/opentransport.html
[12] See http://exodus.physics.ucla.edu/appleseed/
[13] For further details, see: http://daugerresearch.com/fractaldemos/JPLXServes/JPLXServeClusterBenchmark.html
[14] For further details, see: http://daugerresearch.com/fractaldemos/USCCluster/USCMacClusterBenchmark.html
[15] V. K. Decyk, "Benchmark Timings with Particle Plasma Simulation Codes," Supercomputer 27, vol V-5, p. 33 (1988).
[16] V. K. Decyk, "Skeleton PIC Codes for Parallel Computers," Computer Physics Communications 87, 87 (1995).
[17] 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).
[18] K.-C. Tzeng, W. B. Mori, and T. Katsouleas, "Electron Beam Characteristics from Laser-Driven Wave Breaking," Phys. Rev. Lett. 79, 5258 (1997).
[19] 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).
[20] James C. Kniep, Jean-Noel Leboeuf, and Viktor K. Decyk, "Gyrokinetic Particle-In-Cell Calculations of Ion Temperature Gradient Drriven Turbulence with Parallel Nonlinearity and Strong Flow Corrections", Poster 1E25, April 28, 2003 International Sherwood Fusion Theory Conference, Corpus Christi, Texas. http://www.sherwoodtheory.org/sherwood03/agenda.html
[21] T. L. Rhodes, J.-N. Leboeuf, R. D. Sydora, R. J. Groebner, E. J. Doyle, G. R. McKee, W. A. Peeble, C. L. Rettig, L. Zeng, and G. Wang, "Comparison of turbulence measurements from DIII-D L-mode and high-performance plasmas to turbulence simulations and models", Phys. Plasmas 9, 2141-2148 (2002).
[22] http://morphbank.ebc.uu.se/mrbayes3/