

What's New

As of August 21, 2011, Pooch is updated to *version 1.8.3* for use with OS X 10.7 "Lion":



Mac OS X Lion

Pooch users can renew their subscriptions today!
Please see <http://daugerresearch.com/pooch> for more!

On November 17, 2009, Pooch was updated to *version 1.8*:

- Linux: Pooch can now cluster nodes running **64-bit Linux**, combined with Mac



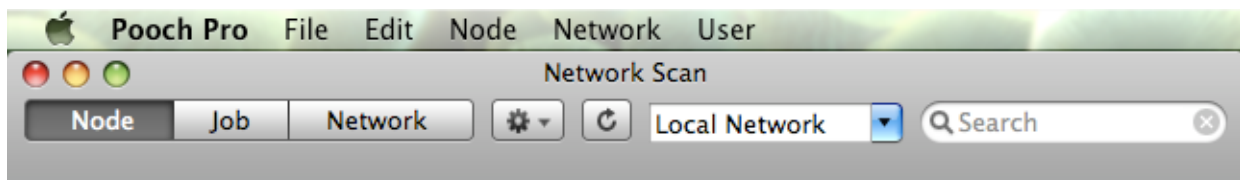
- 64-bit: Major internal revisions for 64-bit, particularly updated data types and structures, for **Mac OS X 10.6 "Snow Leopard"** and 64-bit Linux
- Sockets: Major revisions to internal networking to adapt to BSD Sockets, as recommended by Apple moving forward and required for Linux
- POSIX Paths: Major revisions to internal file specification format in favor of POSIX paths, recommended by Apple moving forward and required for Linux
- mDNS: Adapted usage of Bonjour service discovery to use Apple's Open Source mDNS library
- Pooch Binary directory: Added Pooch binary directory support, making possible launching jobs using a remotely-compiled executable
- Minor updates and fixes needed for Mac OS X 10.6 "Snow Leopard"
Current Pooch users can renew their subscriptions today!
Please see <http://daugerresearch.com/pooch> for more!

On April 16, 2008, Pooch was updated to *version 1.7.6*:

- **Mac OS X 10.5 “Leopard”** spurs updates in a variety of Pooch technologies:
 - Network Scan window
 - Preferences window
 - Keychain access
 - Launching via, detection of, and commands to the Terminal
 - Behind the Login window behavior
 - Other user interface and infrastructure adjustments
- Open MPI support:
 - Complete MPI support using libraries built into Leopard
 - Automatically installs Open MPI plis and ras Pooch modules
- MPJ Express support:
 - Launching MPI-style Java code
 - Supports .jar and .class execution types
- Long file, application, and directory name support
- AppleScript access to user-defined nodelists and remote network data
- Other features, optimizations, bug fixes, and updates

Current Pooch users can renew their subscriptions today!

Please see <http://daugerresearch.com/pooch> for more!



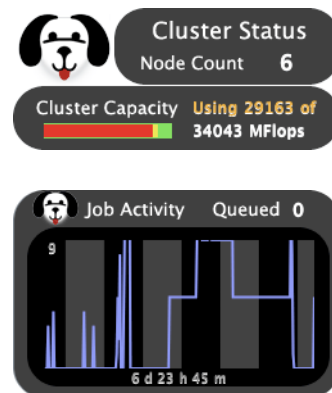
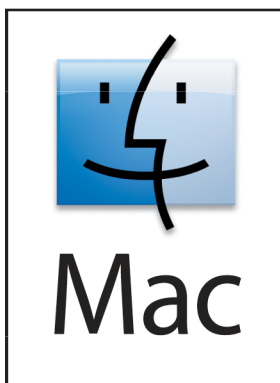
```
MPI.Init(args);
int me = MPI.COMM_WORLD.Rank();
int size = MPI.COMM_WORLD.Size();
System.out.println("Hi from <"+me+">");
MPI.Finalize();
```



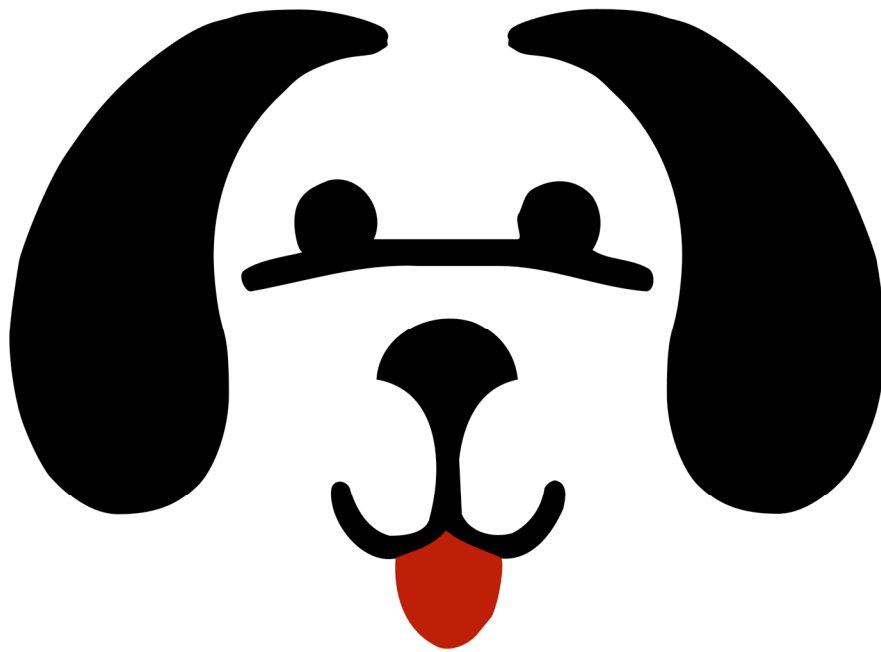
version 1.7:

- **Universal Binary** for running on mixed clusters of Intel- and PowerPC-Macs
 - manages both types of nodes, making all cluster data available
 - launches Universal binaries, enabling cross-processor clustering; Pooch is the first and only clustering solution to launch Universal Binaries in parallel
- Remote job, file, and folder retrieval support, even through proxy nodes
- New Job file format support, including saving and loading job data
 - save commonly used jobs, file and node combinations, to disk
 - supports standard file dialog and reminder sheets
- Head node setting to access and control clusters with Linux-style topologies
 - accesses nodes behind a head node acting as a firewall and NAT router
 - coordinates Pooch features to launch jobs via the head node
- Enhancements to the AppleEvent interface for job and user data
- Automatic detection and retry of preferred IP address of remote nodes
- Pooch Pro automatic user login using Keychain services
- Other features, enhancements, bug fixes, and updates

Please read on for more!



Pooch Application



Parallel OperatiOn and Control Heuristic Application
Version 1.8.3

Copyright © 2001-11 Dauger Research, Inc.

Code and manual written by Dean E. Dauger

Table of Contents

I. Installing Pooch	3
II. Your First Parallel Computation	4
III. Introduction	7
IV. Menus and Windows	
Menus	12
Job Window	24
Network Scan Window	35
Node Info Window	48
Get Files Window	49
Node Registration Schedule Dialog	50
Open MPI Module Install Window	51
V. Pooch Pro	52
VI. AppleScript	
Standard Suite	64
Pooch Suite	68
VII. Linux	72
VIII. Security	76
IX. Frequently Asked Questions	78
X. Revision History	83
XI. Credits	119
XII. Contact Information	120

I. Installing Pooch

Make sure your Macintosh running OS X 10.2¹ is connected properly to the Internet. Almost any type of network connection (10BaseT, 100BaseT, Gigabit, Airport, or a mix) is sufficient. If this Mac is on an isolated network, manually configure the TCP/IP control panel to a unique IP address from 192.168.1.1 to 192.168.1.254. There are two ways to install Pooch:

- For automatic installation, double-click the Pooch Installer:



(There is no step three. Or two.)

- For manual installation:

1. Copy the Pooch folder, containing the Pooch application, to your hard drive.
2. Open the Pooch application.

So that your Mac will be available for computation if rebooted, we recommend the following:

3. From the File menu, select “Set Pooch in Account Startup Items”.

That’s it! Your Mac is now ready for parallel computing. Repeat for additional Macs.

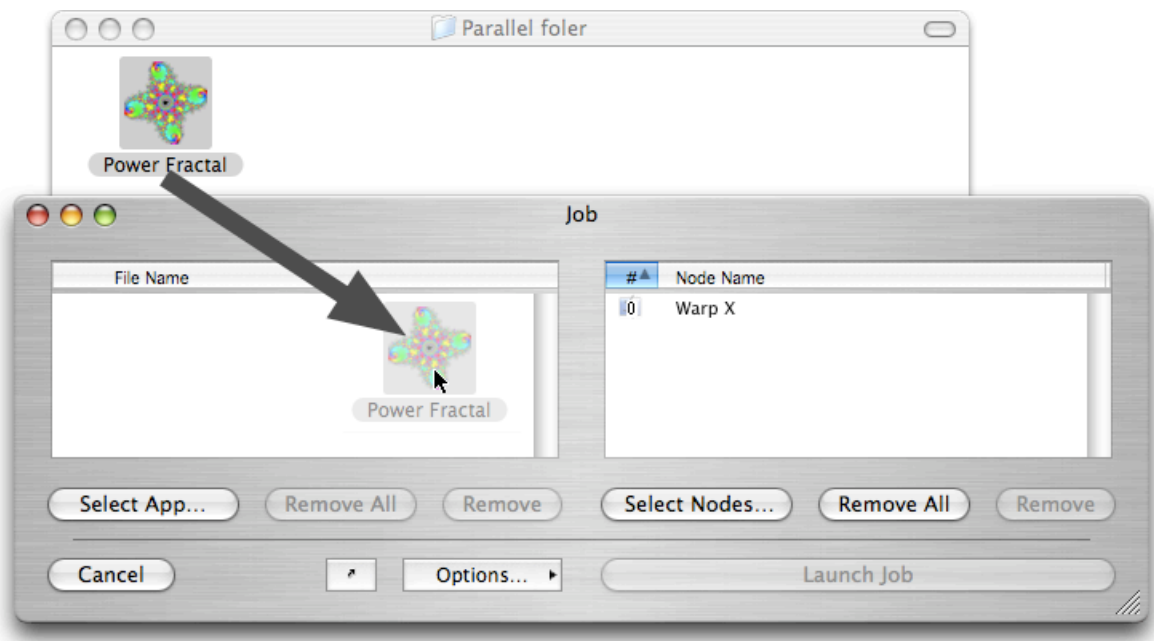
¹ Also operates on OS 9. See the Introduction for system requirements.

II. Your First Parallel Computation

Once Pooch is installed and running on each Mac of a local area network, you can use them as a parallel computer. You will need a parallel application to run. For our first test, let's use the Power Fractal app. Run it on your machine first and note its performance.

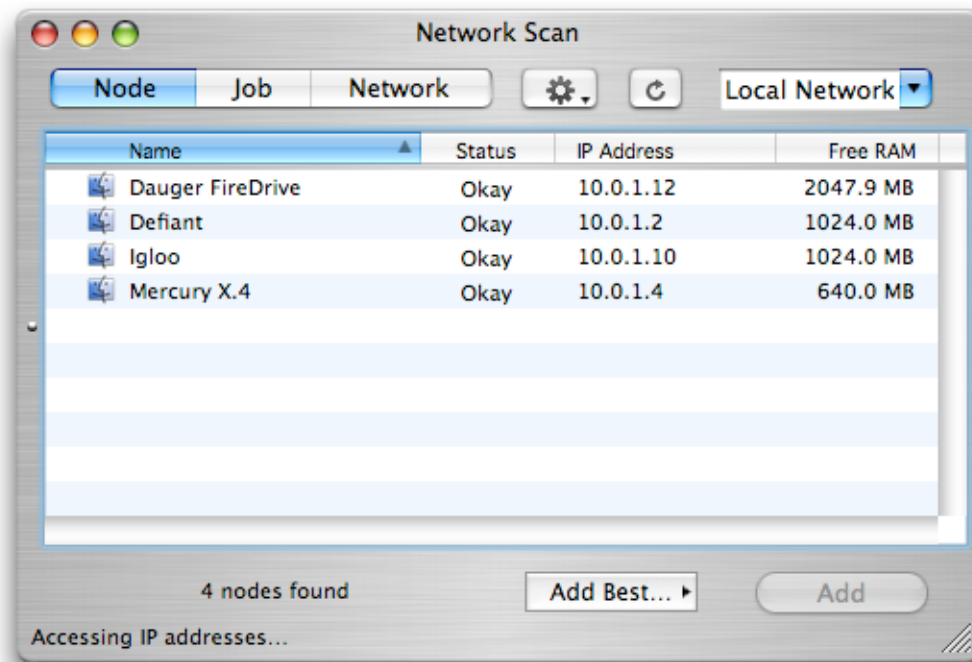
Step One - Selecting a Parallel Application

Switch to the Pooch application and select **New Job...** from the File menu, which results in a new Job Window. Drag the Power Fractal app from the Finder to Pooch's Job Window or click **Select App...**



Step Two - Selecting Nodes

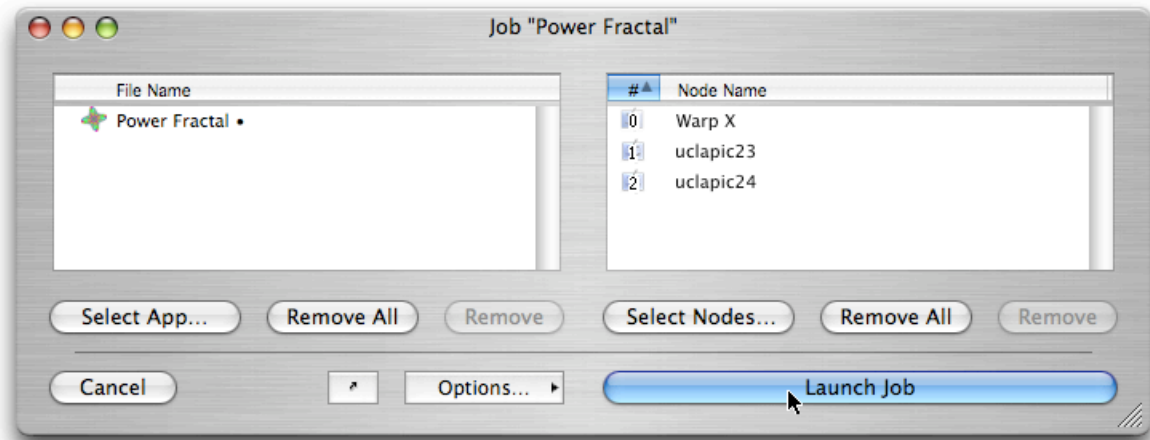
By default, Pooch selects your own node as participating in the parallel job. To add other nodes, click on **Select Nodes...** in the Job Window to invoke the Network Scan Window.



Double-clicking on a node moves it to the node list of the Job Window.

Step Three - Launching the Parallel Job

Finally, click **Launch Job** to start your parallel job.



Pooch should now be distributing the code and starting the parallel application. Once the job runs, you should see a measurable performance increase.

Congratulations! You are now operating your first parallel computer.

Optional - Your Second Parallel Computation

1. Open version 1.1 or later of Power Fractal. Note its performance.
2. From the Parallel menu, select "Automatically Launch onto Four Nodes".

The demo will now automatically ask Pooch to create a job using the four best nodes on the local network, then submit itself for parallel launch. This is an example of "Computational Grid" computing.

III. Introduction

What is Pooch?

Pooch serves as both the user interface to your parallel computer and the manager of your parallel computer. Cluster and grid computing are both examples of parallel computation. Pooch is the parallel computer management and user interface software that can prepare, launch, and monitor these types of parallel computing jobs.

The first role of Pooch is to automate the process of starting a parallel application. A parallel application is much like a normal application except that it is designed to run on a collection of intercommunicating computational nodes.² Manually starting such an application is a tedious process. Pooch automates this process, minimizing the user's effort to operate and control a parallel computation. However, Pooch provides much more beyond this fundamental function.

Pooch provides access to a great deal of information, customization, and automation regarding the life of jobs run on the cluster and the health of the cluster. Jobs can be launched immediately or queued for a later time or when specific conditions are met. Pooch can intelligently and dynamically acquire nodes using its heuristic algorithms or using custom AppleScripts prior to launch. Statistical information about nodes can be accessed, and some node behavior can be customized. Job status is monitored, and statistics about jobs are collected and retained during and after jobs have completed. Pooch serves as launcher, scheduler, queuing system, and multisystem

² This contrasts with distributed computing, such as SETI@Home, in which the pieces of its computation are all but completely isolated from one another.

cluster monitor, all presented in a friendly user interface.

The first thing that makes Pooch unique from all other software of its kind on any platform is the extension of the revered Macintosh ease-of-use to parallel computing. Since 1998, the AppleSeed project at UCLA's Department of Physics has unquestionably demonstrated the practicality of leveraging the advantages of the Macintosh platform for the goals of numerically-intensive parallel computing. While others used Linux and Windows NT with a hodgepodge of command-line only, unreliable, unstable code fragments, AppleSeed provided the only known solution for parallel computing on the Mac OS. Users of software built by the AppleSeed team are *not* required to know the inner workings of an operating system, nor are they required to hire expert assistance in such matters. Demonstrated at UCLA and around the world time and time again for three years, that advantage has enabled college students, high school and junior high school students, and even ordinary scientists on shoestring budgets to successfully build and operate their very own parallel computer.

We have continued this success in the latest incarnation of parallel computing on the Mac. Two months after the introduction of Pooch, a fellow in Hawaii used it to build and run his own five-iMac cluster. He then informed Dauger Research of his success and that he was in the sixth grade. While achieving such strides in ease of use, the same software harnesses massively parallel hardware.

Macintosh Parallel Computing

Forming a parallel computer out of a network of computers is called "cluster computing". In this case, the hardware of the parallel computer is a cluster of Macintoshes. AppleSeed provided two major software pieces essential for using such hardware for parallel computation.

The first is a communications library named MacMPI, authored by Dr. Viktor K. Decyk and introduced in 1998. MacMPI provides a means for executables on each

Macintosh to communicate with one another. It is a source code wrapper library that translates the calls by the parallel code into calls directly to the Mac OS. MPI (Message-Passing Interface) is a standard application programming interface (API) of the high-performance parallel computing industry. That API is supported on almost all large parallel computers (Cray, IBM SP, Fujitsu, SGI, etc.). Using the Mac OS, MacMPI supports a commonly used subset of those MPI calls.

The second software component satisfies a different essential need of the parallel computation: initiation. To begin, MacMPI requires information about the Macintoshes designated to participate in the computation. In addition, since, when compiled, it is mixed with the object code of the executable itself, MacMPI cannot copy and start the parallel application or start itself on all the Macintoshes that will perform the computation. These needs are filled by parallel computing launching software.

Through 2000, AppleSeed provided software called the Launch Den Mother and the Launch Puppy (LDM & LP) that filled this second need. The combination of MacMPI and LDM & LP are the key ingredients that allowed students and researchers around the world to develop and run their own parallel computing software on the Mac OS.

Meanwhile, Apple was introducing prereleases of Mac OS X in preparation for its final release in 2001. Mac OS X was fundamentally distinct from the original Mac OS while being the future of the Mac OS. Apple required Mac developers to port their code to Carbon, a new API, so that their code can run natively in OS X. Carbonizing MacMPI (renamed MacMPI_X) was straightforward, but the launching mechanism was another story. Most of LDM & LP's fundamental design made it impossible for a simple conversion to OS X. For example, LDM & LP's primary communications mechanisms and LDM's user interface were tied to pieces of the Mac OS not carried forward into Mac OS X.

Since "porting" LDM & LP to OS X would largely be a rewrite of LDM & LP, their author decided instead to simply start from scratch. Rather than a reconstruction of what came before, the new software created the opportunity to create software that vastly superseded LDM & LP's functions, capabilities, and use of the Macintosh user

interface. Utilizing three years of experience with parallel computation using Macintosh clusters at UCLA Physics, this project called for a rethinking and redesign of the user interface for a parallel computer. Hence, Pooch was born.

Using Pooch

Pooch is meant to be used in an environment where your cluster is a computational resource shared by a cooperative group, much like how a workplace printer is shared. A few scenarios are possible:

- A dedicated Macintosh cluster - Perhaps the simplest configuration, a group of eight, sixteen, or thirty-two identical Macintoshes or Xserves devoted to computation at all hours of the day. Pooch enables members of your group to submit jobs to the cluster and retrieve the output at a later time. Also, a subset of the nodes can be used for one job while a second job runs on the others. Many, including UCLA's Department of Statistics and NASA's Jet Propulsion Laboratory, have clusters like this.
- Reusing a network of individual Macs for computations - There are secretaries, scientists, artists, and professors using their own Macs for e-mail, spreadsheets, web browsing, and word processing during the day. After they leave for home, the Macs are usually idle and unused. Merely with the addition of Pooch, this network of Macs can become a computational cluster at night or on weekends.
- Combinations of dedicated and individual Macs - A few Macs, perhaps four at a time, can be added to an existing network of Macs as described in the preceding example. The groups of four devoted computational nodes can be used for debugging and diagnosing an experimental parallel code during the day, but then the entire network can be combined in the off-hours for larger (and well-tested) parallel codes. As new hardware is purchased, the compute nodes are mixed with

the rest of the population as needed. This more dynamic model is in practice in the Plasma Physics group at UCLA. This setup is well suited for a circumstance where, often, the groups of four are enough for most computations, but occasionally a member of the Plasma Physics group needs to complete some large runs to meet, for example, a conference deadline. So this researcher asks the group and gets time on the entire network. This flexibility has saved the day for more than one researcher's work.

- Be creative: You may find additional scenarios useful. For example, from a PowerBook connected to the Internet via Airport, the author has initiated and controlled, in real-time, parallel jobs computed over 40 miles away. Using a network connection at the Max-Planck-Institut für extraterrestrische Physik in Garching, Germany, the distance record has increased to 6000 miles.

Also, new in 1.8, Pooch can add compute nodes running 64-bit Linux.

System Requirements

In general, Pooch needs 4 MB of free memory on a Macintosh with a valid TCP/IP connection. Pooch requires CarbonLib version 1.2 or later and Mac OS 9 or later. The latest CarbonLib is available from <http://www.info.apple.com/support/downloads.html>.

Pooch can run on OS X too. In version 10.2 of OS X, Apple fixed significant bugs present in previous versions of OS X. Thus, we are proud to say clustering on Mac OS X 10.2 and 10.3 is fully operational and fully supported. Pooch Pro requires OS X 10.2.1 or later. Open MPI support in Pooch requires OS X 10.5 "Leopard" or later.

New in 1.8, Pooch now supports using compute nodes running 64-bit Linux in addition to Macintoshes. We recommend at least 2 GB of installed RAM on Linux. So far we have tested on 64-bit versions of Ubuntu 9, SuSE Enterprise Linux version 10, and Red hat Linux 5.3.

IV. Menus and Windows

Pooch Menus

File menu

- **New Job...** - Opens a new Job Window (see the Job Window section)

- **New Queue Job...** - Opens a new Job Window enabling the Queue Job and Automatically Acquire options.

- **Save Job As...** - Saves the job of a Job Window to a file on disk, which can later be used for another job. New in 1.7.

- **Select App and Files...** - Opens a get files dialog for selecting an application and files for the Job Window. If the Job Window is present, the file dialog will appear as a sheet. If the Job Window is not present, the dialog will be standalone, but if a file is selected it will be added to a new created Job Window.

- **Reveal Remote Files Folder in Finder** - When Pooch receives files from another node, Pooch must hold the files in a folder on the hard drive. Invoking this item directs the Finder to open that folder where those files are being held.

See **Collect Remote Files**. Also, this item is available by control-clicking the Pooch icon in the Dock.

- **Select Nodes...** - Opens the Node View of the Network Scan Window (see the Network Scan Window section)
- **Show Job View...** - Opens the Job View of the Network Scan Window (see the Network Scan Window section)
- **Recent Node Lists** - When the Job Window is open, this submenu lists the node lists of the most recently launched jobs. The number before the ":" is the number of nodes. When the node names have similarities, Pooch attempts to abbreviate the names. For example, a node list of four computers named node01, node02, node26, and node27, will be abbreviated to "4: node01, ...02, ...26, ...27".
- **Recent Apps and Files** - When the Job Window is open, this submenu lists the file lists of the most recently launched jobs. The number before the ":" is the number of files. If one of the files is an executable that was launched, it will be listed first.
- **Set Pooch in Account Startup Items** - Makes (or updates) an entry for Pooch in the Login Items system preference so that Pooch will automatically start up and make this Mac available for parallel computation after a restart. This feature is provided as a convenience.
- **Collect Remote Files In** - When Pooch receives files from another node, it must store those files in a folder on the hard drive. Pooch can hold those files either: 1. in the folder where the Pooch App itself resides; 2. inside the Temporary

Items folder, as designated by the OS; 3. in the path /tmp/pooch, which is the default case on OS X; or 4. in the path /Users/Shared/pooch/remote. Note that files in /tmp may be deleted when the machine is restarted.

- **Open Pooch Log** - Opens the Pooch Log in Console.app. Pooch logs events into a text file located at /Users/Shared/pooch/poochlog.txt. These entries detail steps in the launch process, the queuing system, and network access.

- **Quit Pooch** - Selecting this item, or pressing Command-Option-Q, will quit Pooch. Pooch is an unusual application because, while its user interface is not in use, it is meant to run at all times in the background. This mechanism will make it less likely to be quit accidentally.

- **Finder to the Foreground** - Selecting this item sends Finder to the foreground.

Edit menu

- **Paste** - Invoking this item in the Job Window pastes an item on the clipboard into its file pane. Launching such a job distributes the clipboard to the clipboards on the target nodes. In the Network Scan Window, this pastes into the remote network address field. Modified in 1.7.

Node menu

To use the items on the Node menu, you must select one node from either the node list of the Job Window or the list of nodes in the Node View of the Network Scan Window. Contextual menu-clicking on a single node in the Job Window or the Network Scan Window also invokes this menu.

- **Get Node Info** - Invokes a Node Info Window that retrieves a variety of information about a node, such as processor type, OS version, load, and free memory. It also retrieves a list of running processes (an application is one type of process) on that node.

- **Get Job Queue** - Retrieves the parallel jobs queued on that node. Typically, a job is forwarded to the Pooch at the job's node zero and held there until certain conditions have developed. For example, a job might start only when its designated start time has passed and the target nodes are no longer busy.

- **Get Files** - Opens a new window where you can retrieve files from the remote node.

- **Connect to Server...** - Opens an Apple File Sharing connection to that node, much like the menu item of the same name in the Finder. If that node has File Sharing on, you can enter your password to access files on its volumes. This feature is provided for your convenience.

Network menu

- **Register this Node** - Pooch, by default, registers its node on the network, meaning that it puts up a flag that other Pooches can see. This flag has enough information for other Pooches to identify and communicate with this particular Macintosh. Selecting "Never" removes this flag, making this Pooch invisible to other Pooches. Selecting "On a Schedule..." opens a node registration schedule dialog (see below). Pooch will then register and deregister itself according to the

specified schedule.

- **Local Scan Includes this Node** - When scanning for nodes in the local network, selecting **Always** will cause Pooch to always add itself to the network results, regardless of the Register this Node settings. Otherwise your node might not appear in the local network scan if your Pooch did not register itself. Selecting **Always** does not cause other Pooches to see your node.

- **On Node Deregistration** - When Pooch deregisters itself, Pooch will kill all jobs it is tracking on this node if **Kill All Jobs** is checked. For example, when this node deregisters itself according to the node registration dialog (see “On a Schedule...” of the “Register this Node” submenu, above), Pooch will kill any remaining jobs that it launched. If this menu item is not checked, Pooch will allow jobs to continue to their own completion.

- **Primary Address** - This submenu specifies which active IP address Pooch registers as “primary” or the default address. Use System Default has Pooch follow the precedence specified by the system, usually set via the port ordering in the Network system preference. Setting this to another IP address, if available, resets Pooch’s network registration of this node to the new setting. It will encourage, but not guarantee, that other Pooches and their jobs will use that address, perhaps because of its more optimal performance. Pooch will still listen on the other addresses, as not every IP address will be accessible from other nodes. New in 1.6, selecting the Import Addresses... option selects a whitespace-delimited text file containing additional addresses that identify this node. This is appropriate when, for example, a firewall redirects activity from a public IP address to a private one.

- **Network Scan Columns** - This submenu allows you to show and hide columns in the Network Scan Window. Check and uncheck the items to toggle them. See the Network Scan Window section for descriptions. Selecting “Show Columns...” opens a window where multiple columns can be toggled at once.

- **Automatic Rescan** - When the Network Scan Window is open, Pooch will automatically repeat its scans on the network at a particular interval. You can set this time interval, or shut off the automatic rescan, by selecting an item on this menu.

- **Job View Access Time** - This setting determines how far back in time the Network Scan Window, when in Job View, will query when compiling archived job information.

- **Discover using** - Since its birth, Pooch used Apple’s implementation of the Service Location Protocol (SLP), an IETF standard, to perform discovery of other Pooches on a TCP/IP network. In May 2002, Apple introduced Bonjour (formerly Rendezvous), also known as ZeroConfig or mDNS, to perform discovery as well. The latest version of Pooch uses both to discover on a network. For best compatibility, the default setting uses both SLP and Bonjour so that this Pooch can discover and be discovered by Pooches running on OS’s that do not support Bonjour (such as OS 9). You may select Bonjour Only if you know you will only be using Macs running OS X 10.2 or later. The benefit of using only Bonjour is faster discovery, but its drawback is not being able to locate Macs running older versions of the Mac OS.

- **Access Nodes** - Normally Pooch contacts other nodes directly to inquire about their status. A node accessed using the **Remote Node Scan** feature could

return the addresses of nodes behind a firewall or other network barriers. Selecting **via Proxy** will use the proxy connection provided by the first remote node to forward data between your local machine and those behind the firewall. New in 1.6 and available only on OS X 10.4 "Tiger", Pooch can access an Xgrid cluster. To access Xgrid, enter the host address of the Xgrid controller and its password using the **via Xgrid...** item of this submenu. If no password is required, leave that field blank. Pooch spawns jobs containing a piece of itself through Xgrid, which Pooch can then discover and access.

- **Remote Network Job Access** - Normally Pooch on other nodes contacts yours directly to forward jobs and files. For example, a node accessed using the **Remote Node Scan** feature might not be able to return files to your nodes because of firewalls or other network barriers. This feature permits the Network Scan Window to pull these files onto your machine. Typically this is used in combination with the **via Proxy** setting of the previous item. **None** turns off this feature, **Only My Jobs** accesses only jobs that resulted from a job from this node and user, typically a job triggered by **Retrieve Output** job option (see Options of the Job Window), while **All Jobs** will pull all jobs destined for this node. If the name and IP address of your node changes, then this feature might not successfully locate the jobs. New in 1.7.

Settings menu

- **New Job Automatically Reloads** - When the Job Window is created, it will remember information, such as the file list (including the application), the node list, and various job options, from the last parallel job submitted. This submenu toggles what portions of the last job it will retain and what it will discard. Note:

some of this information may become stale, such as when you move the app on your hard drive or if a node has a dynamic IP address (which could happen if your network uses DHCP). Extended in 1.7. Selecting **On Job Submit** triggers a new job to invoke as soon as the previous job is submitted to the launch queue. New in 1.7.

- **Remember Recent Node Lists** - This setting determines how many node lists will be displayed in the “Recent Node Lists” submenu of the File menu.

- **Remember Recent Apps and Files** - This setting determines how many lists of apps and files will be displayed in the “Recent Apps and Files” submenu of the File menu.

- **Job Port Number Range** - When Pooch launches a job it pseudorandomly assigns a primary TCP port number to the job for its communications. Normally Pooch chooses from 16384 up to 49152. You may specify a different range of port numbers here, say, for compatibility with a firewall. The **Use Port Number...** option in the Job Window overrides this setting.

- **Launch Unix Jobs** - By selecting **in the Background**, Unix-based executables (such as Mach-O executables and Unix scripts) are launched in the background, with output directed to a stdout.txt file. Selecting **using Terminal** directs this Pooch to launch Unix-based executables in a window of the Terminal app, allowing for user-input. This setting is local to this node: the launch on the remote nodes will use Terminal instead of background launching if **using Terminal** is checked on those nodes as well.

- **Restrict/Allow Access to Local Processes** - When this item is set to restrict,

Pooch will block other nodes from seeing process IDs of processes not launched by Pooch. This prevents someone else from using the Node Info window to kill, say, your word processor. Use this item only if you wish to restrict control of processes on this node by other Pooches.

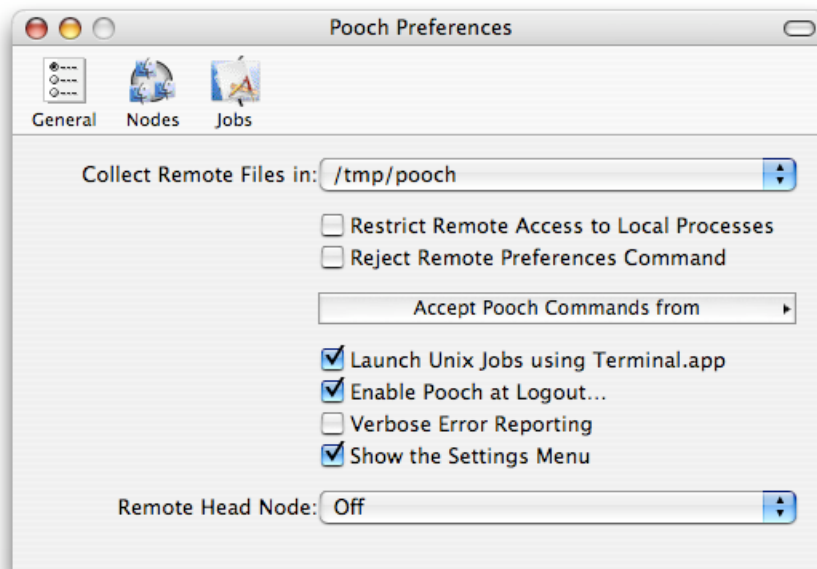
- **Accept Pooch Commands from** - When set to **Local Subnet**, Pooch will check the IP address of incoming connections against its local subnet. If they do not match, the connection will be rejected and this Pooch's listener port will reset. Selecting **Import Addresses...** allows you to select a whitespace-delimited text file containing a list of IP addresses. Pooch will compare the addresses of incoming connections against this list, including the resolution of DNS data. Setting to **Any Address** removes this restriction. Use this item only if you do not want to control this node from other subnets of the Internet.
- **Reject/Accept Remote Preferences Command** - When set to accept, other Pooches will be allowed to overwrite this Pooch's preferences and settings. For example, the node registration schedule may be modified remotely. When set to reject, other Pooches will not be allowed to modify settings of this Pooch.
- **Disable/Enable Pooch at Logout...** - Selecting this item will direct Pooch to run while nobody is logged in. This feature makes it possible to use unutilized, logged-out machines for parallel applications. Pooch automatically quits when someone logs in and will return when the user logs out or if they run Pooch. Toggling this item may require administrative authorization. The words "(To Be Enabled On Restart)" will be appended if it is necessary to restart the machine to fully enable this feature. Updated in 1.7.
- **Disable/Enable Verbose Error Reporting** - Reports internal errors, such as

network errors, in notification windows. The information reported most useful when diagnosing difficult-to-find problems during Pooch operation. It also reports a problem when Pooch encounters AppleScript objects or commands that it cannot recognize.

Pooch menu

- **About Pooch...** - displays the About Box of Pooch, which also can contain status messages. This window normally opens briefly, then closes on its own, at startup. Clicking on the window closes it.
- **Preferences...** - displays the Pooch Preferences window. This window displays all of the preferences available in Pooch. These settings are divided into three categories: General, Nodes, and Jobs. Updated in 1.7.

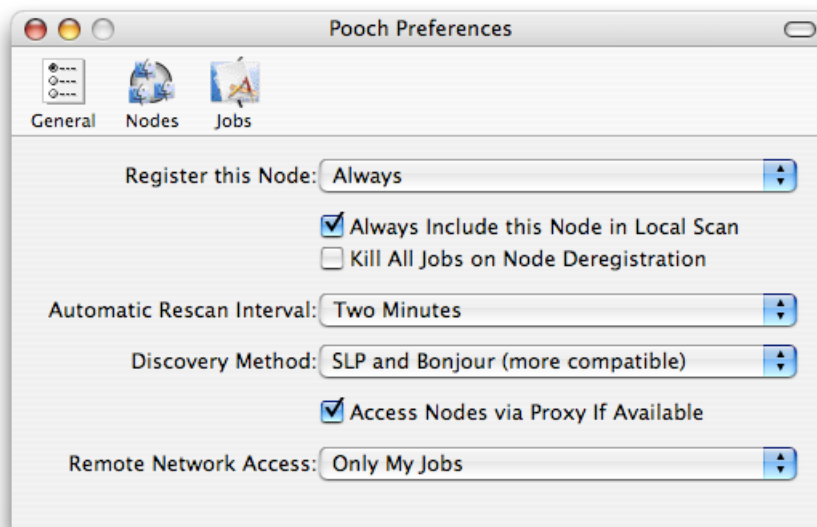
General Preference Pane



Clicking on the icons in the toolbar at the top of the window switches the Preference Window to the other panes. It is normal for this window to disappear when Pooch is in the background. For backwards compatibility, the Settings Menu is shown by default.

Remote Head Node - This preference modifies and coordinates four elements of Pooch's behavior to support launching jobs into and retrieving files from a cluster with a designated head node. If this feature is enabled, the default Network Scan window scans the head node's network instead of the Local network, the Forward via job option chooses the head node, the Retrieve Output and Queue Job options are enabled, the **Access Nodes via Proxy** setting is enabled, and the **Automatically Acquire Nodes** job option automatically selects the head node as the starting node. New head node IP addresses can be entered via the address field of the Network Scan window. New in 1.7.

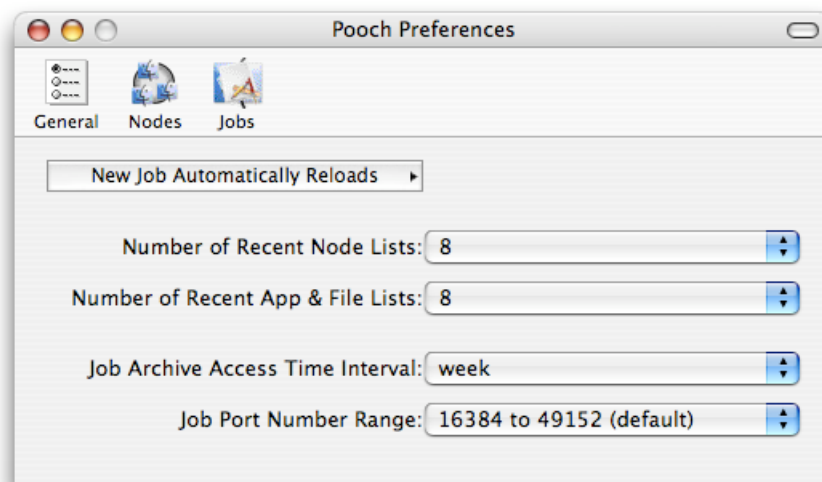
Nodes Preference Pane



Remote Network Job Access - Normally Pooch on other nodes contacts yours

directly to forward jobs and files. For example, a node accessed using the **Remote Node Scan** feature might not be able to return files to your nodes because of firewalls or other network barriers. This feature permits the Network Scan Window to pull these files onto your machine. Typically this is used in combination with the **via Proxy** setting of the previous item. **None** turns off this feature, **Only My Jobs** accesses only jobs that resulted from a job from this node and user, typically a job triggered by **Retrieve Output** job option (see Options of the Job Window), while **All Jobs** will pull all jobs destined for this node. If the name and IP address of your node changes, then this feature might not successfully locate the jobs. New in 1.7.

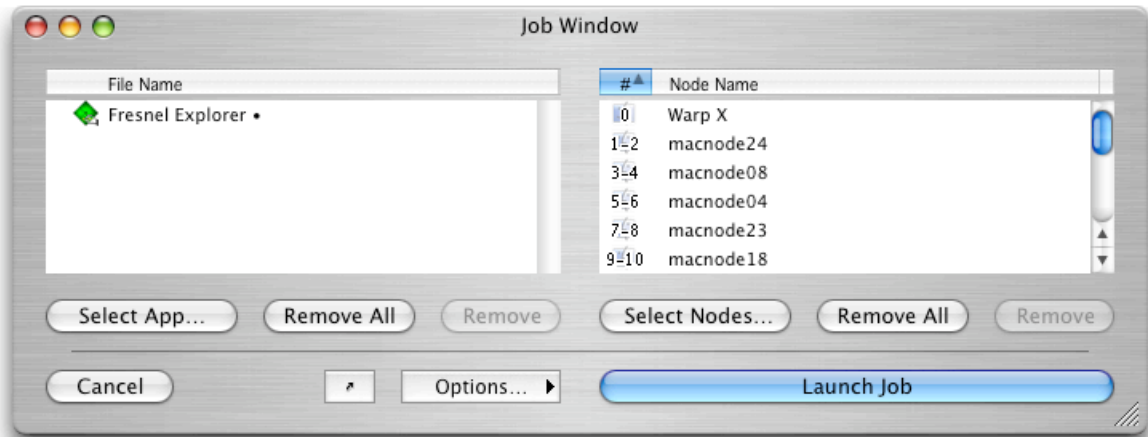
Jobs Preference Pane



Selecting **On Job Submit** of the **New Job Automatically Reloads** menu triggers a new job to invoke as soon as the previous job is submitted to the launch queue. New in 1.7.

Pooch Windows

Job Window



The Job Window is where all parallel jobs are prepared for launching, scheduling, or queuing. A job requires information about the parallel application you want executed and what nodes will run this application. This window is resizable so it can display large amounts of information.

- File List - The left pane of this window displays the list of files in this job. This list can include an application³, which, if present, is placed at the top of the list. Additional applications dragged here will be added to the end of the list, but they will not be launched by Pooch. Removing the first application in this list will cause the next application, if present, to shift to the top of the list. Additional files may be added, which some parallel apps use for input data, here. Pooch will distribute these files to the nodes on launch. Pooch is also capable of selecting

³ As of version 1.2, Pooch can recognize and launch a wide variety of applications, including Classic apps, CFM-based Carbon apps, bundle-based Carbon apps, Cocoa apps, Mach-O executables, Unix scripts, and AppleScript apps. In version 1.7, Pooch supports launching Universal Binaries. In 1.7.5, it supports .jar and .class files compiled with MPJ Express.

and copying folder structures⁴ to other nodes. The folder tree will be scanned recursively. Files inside these folders will be copied as well, but only those files present at the moment of launch. Aliases will not be resolved. You may use the **Select App...** button to invoke a dialog sheet to select an application to execute and files to distribute, or you may drag them from Finder to this pane if you prefer. Use the **Remove All** button to clear the file list, and select a file or files and click the **Remove** button to remove individual files. New in 1.8, you can also add binaries contained in the Pooch binary directory of node 0 of the Node List, below. These binaries are listed via the Job side drawer, described below. On Mac, the binaries are in `/Users/Shared/pooch/bin`, while on Linux they are in `/var/pooch/bin`.

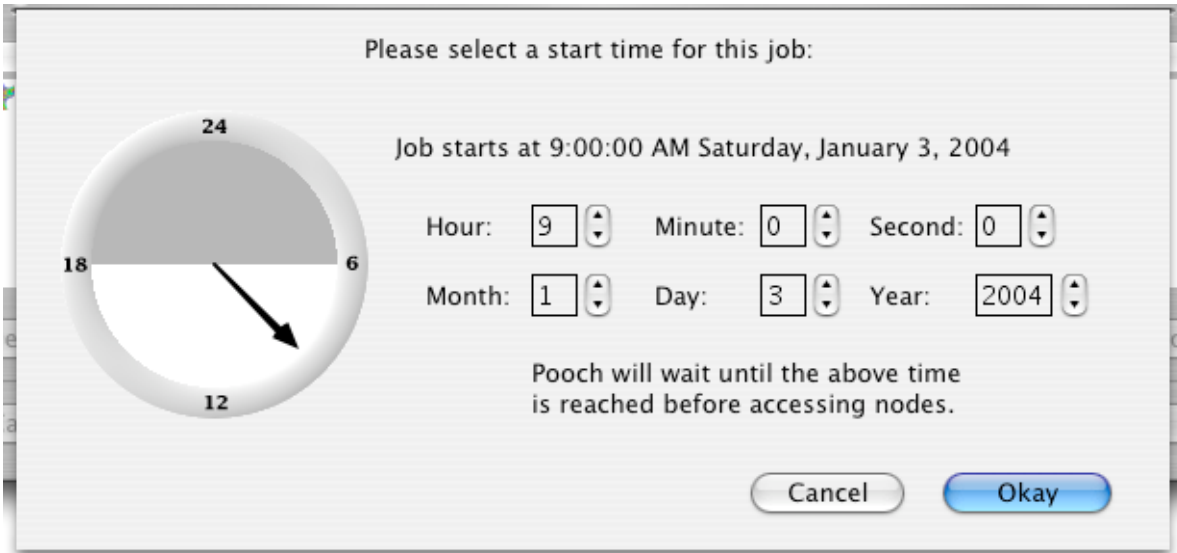
- Node List - The right pane displays the list of nodes for this job. Each node you select is assigned to run an MPI task or tasks. Each task is given a unique identification number. The number(s) on the icon to the left of the node's name indicates which task(s) that node will be assigned. (See the Tasks per Computer job option, below.) This information will be relayed to MacMPI. The parallel code can retrieve this information from MacMPI and use it to partition its problem among the processors. Experimenting with the order of these nodes or the number of tasks can be useful for debugging. Clicking on **Select Nodes...** invokes the Network Scan Window. You can add nodes to the node list by selecting them in the Network Scan Window or dragging them from that window to this one. **Remove All** clears the node list, and selecting a node or nodes and clicking **Remove** removes individual nodes from the list. The limit on the number of nodes is determined when this Pooch is registered. Note: Pooch mandates that your Mac is node zero if it is participating in the computation.

⁴ Unix permissions will be preserved only when copying between OS X machines.

□ Job Options - In OS X 10.3 and later, clicking the **Options...** button produces a drawer presenting options for this job. Holding down the **Options...** pop-up creates a menu containing the same options. Only the menu is available on previous OS versions.

◇ **Place in Subdirectory...** - When sending files to another Mac, Pooch by default places all apps and files within the same folder in which Pooch resides. Over the course of many jobs, the mixture of files can become confusing. Specifying a name here asks Pooch to create a subdirectory inside that Pooch's folder and place all apps and files in that new subdirectory. (Pooch does not write outside its folder during the launch process.) For the purposes of organization, you can name these subdirectories based on the job name, or the date, or the name of the user who submitted the job.

◇ **Start Job at Time...** - By default, when you submit a job, Pooch will start the job on the specified nodes immediately. This option, however, allows you to have Pooch place the job in the job queue and delay its start until a particular future time, any minute, any day. The job will be held in the queue on the node designated zero in the node list. (So, if your Mac is not node zero, it will be forwarded to that node.) The following shows the dialog that appears when selecting a start time:



- ◇ **Tasks per Computer** - Many of the recent Power Macintoshes have more than one processor each, so it would be appropriate to be able to allocate one parallel processing task per processor, enabling parallel computing inside boxes as well as between boxes. By default, this option is set to do just that. You may override this setting by selecting a specific number of tasks per processor. Increasing the number of tasks beyond the number of nodes will probably be less efficient, but this possibility is useful for debugging parallel code. Because multitasking is significantly better in OS X than in OS 9, we recommend launching multiple tasks per computer only on OS X. (By default, Pooch will launch only one task per computer running OS 9.) New in 1.7, control-clicking on the node icon in the node list will override this setting, allowing you to select different task counts for each node.
- ◇ **Job Type** - This option specifies the type of parallel launch needed for this parallel job. MacMPI, mpich, MPI-Pro, mpich-gm, LAM/MPI, MPJ Express, and Open MPI need information about the cluster configuration, but they

each require significantly different data formats and launch procedures. The MacMPI launch type is the default. It is generally the case that only Unix-based executables can accept information the way that the other MPIs require it, so Carbon, Cocoa, and most other types of apps will usually follow the MacMPI convention. See below for the Grid Job Type.

- ◇ **Command-Line Arguments...** - This option specifies command-line arguments to add when launching Unix-style Mach-O executables. Only some Unix-based executables require input information using such options (such as "-a -e iou"), and it is usually preferable to supply such input information via input files for portability. This item will be disabled when the file list contains other executable types.
- ◇ **Use Port Number...** - In order to use TCP/IP, an MPI needs to use a particular TCP port number to initialize its network connections. Under normal circumstances, Pooch generates a pseudorandom port number based on the job name and the time of day and supplies this number to the MPI. You can use this job option to override that random value and set a particular port number. This feature can be useful to comply with firewalls that only have specific ports open. However, the TCP convention requires that the same port number not be used again for approximately two minutes. Some TCP implementations follow this convention, and some do not, so your network could appear to be blocked if you launch a second job using the same port number too soon.
- ◇ **Forward via** - Normally Pooch attempts to contact all nodes directly when launching jobs and distributing data. In some configurations, nodes of a cluster could be protected by a firewall or a machine acting as a

firewall, such as a “head node”, which prevents this kind of access. This feature allows you to forwarding the job to an “in-between” node, which acts as a proxy. The job will be forwarded to the proxy node, which then forwards it to node 0 in your node list. Usually such proxy nodes should be chosen from the ones used for the Remote Node Scan. (New in 1.5.5)

- ◇ **Retrieve Output** - Turning this setting on directs Pooch to automatically retrieve files created and modified by a job after Pooch detects that the job has ended. Pooch checks for this condition once per minute, and the queuing system also operates on a one-minute cycle. This feature can be set to only send back the files generated at node 0 or on all the nodes. Pooch will make a “best effort” to place files in the directory on the node where the executable originated. If that node is not available for some reason, the forwarding process will be queued until it is able to complete the task. As needed, the files will be forwarded through the proxy node specified in the **Forward via** feature, above. New in 1.6, the retrieved files are accompanied by a .joblog text file containing useful information about the job’s execution.

- ◇ **Bark** - Pooch will not bark when launching a job if this item is unchecked.

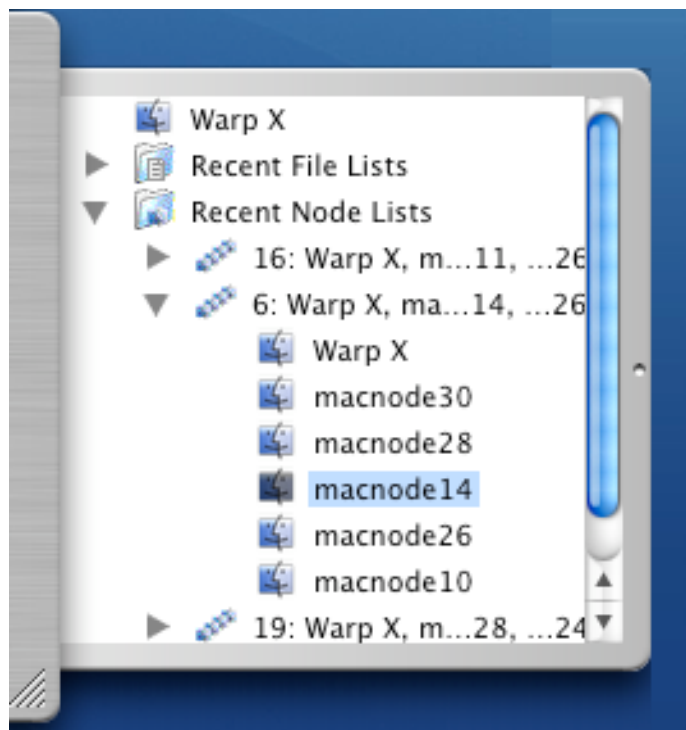
- ◇ **Don’t Use Busy Nodes** - Normally, Pooch checks if the nodes on the node list are already running a job submitted using Pooch. If any of the nodes are running such a job, Pooch will not launch the new job. This feature is useful because a parallel job usually runs best if there are no other jobs running. Unchecking this item disables this feature.

- ◇ **Queue this Job** - If this option is enabled, Pooch will save this job into a job queue and launch it as soon as possible. If the job launch process fails, which can happen if the nodes are busy or inaccessible, the job will be saved back into the job queue of the node from which the launch was attempted. Pooch will check the queue once per minute. (See the Node Info Window or the Job View of the Network Scan Window to access this queue.) Otherwise, Pooch discards the job after the first attempt. If you are using the **Start Job at Time...** feature, you may want to consider checking this flag in case a different job is already running on the same nodes.

 - ◇ **Automatically Acquire Nodes** - If this option is enabled, the Job Window will replace the node list with options to automatically acquire the nodes needed to run this job. Pooch will attempt to fulfill the request specified in the acquire controls just prior to launch. Used in conjunction with the **Start Job at Time...** option, the acquisition will be scheduled for that specified time. Used in conjunction with the **Queue this Job** option, the acquisition will be attempted again if the request was not satisfied.
- Launching, Scheduling, or Queuing a parallel job - Clicking on **Launch/Schedule/Queue Job** will ask Pooch to launch, schedule, or queue, depending on the **Start Job at Time...** and **Queue this Job** options, a parallel application onto the nodes according to the information given in this window. If your node is participating in the parallel computation, your machine's Pooch will direct the launch process or hold the job in its queue, depending on the job and the circumstances. If your machine is not participating in the computation, the

job will be forwarded to the Pooch at node zero of the computation, and that Pooch becomes responsible for the job. If the job has not yet launched, you may inquire about this job's status by accessing the job queue of the job's node zero or using Job View in the Network Scan Window. After the job successfully launches, the nodes running the parallel job will report **BUSY** and the job will report Running in the Status column of the Network Scan Window until the job ends. Getting Info about a node will return additional details about the computation's use of that node.

- Recent Items Drawer - Clicking on the recent button at the bottom of the Job Window will open a new drawer containing recent items. You can use this feature to recall recently used executables, files, or nodes.



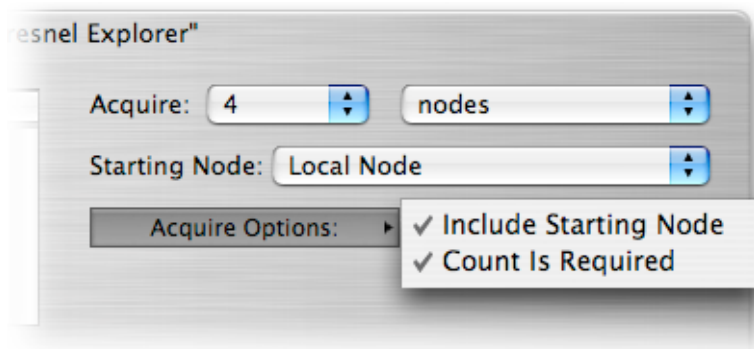
The Recent File Lists folder contains the file lists of previously submitted jobs, each of which can be opened to access specific files. The Recent Node Lists folder contains the node lists of those previous submitted jobs, which can also be opened to recall specific

nodes. The Remote Scan List displays the cached IP addresses used in the Remote Node Scan feature of the Network Scan Window. For your convenience, an icon at the top of this drawer contains the name and IP address information of your node. This drawer is available only in OS X 10.2 or later. New in 1.8, if executables are present in the Pooch binary folder on node 0 listed in the Job window, they will be listed here under "binaries". The Pooch binary folder is in /Users/Shared/pooch/bin/ on Mac and /var/pooch/bin/ on Linux.

Variations of the Job window

The Job window offers two variations from the node list. The first is the ability to acquire nodes automatically from the cluster at launch time. This feature allows the executable to be launched on nodes as they become available. New in 1.6, the second is the ability to distribute a single-processor executable on a cluster, giving them a different integer for each instance of the executable. This feature gives you the ability to explore a parameter space using an executable not otherwise designed for parallel execution.

Automatically Acquire Nodes



- Acquire nodes or processors - When the **Automatically Acquire Nodes** option is enabled, selecting from these pop-up menus specifies how many nodes or

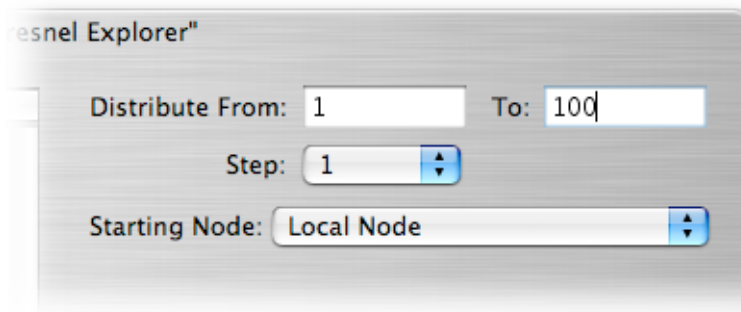
processors Pooch will attempt to acquire just before launching the job. If available, nodes of different processor counts could be combined into a parallel job. Options such as **Tasks per Computer**, do apply. Pooch will add as many available local nodes as it needs to satisfy this request.

- Starting Node of the acquisition - When the **Automatically Acquire Nodes** option is enabled, the acquisition is performed by the node selected here. By default, the local node is selected. If another node is selected, the job will be forwarded to that node prior to launch. This pop-up menu is populated with the nodes previously used for the remote network scan feature of the Network Scan window. Adding a node from the Network Scan window to the Job window also sets this acquire setting.

- Acquire Options... - When the **Automatically Acquire Nodes** option is enabled, the acquisition will include the starting node if the **Include Starting Node** item is checked. Otherwise, the starting node may or may not be included. The acquisition depends on the number, status, and rating of the nodes on the network. The starting node can be selected if it helps satisfy the acquisition request. If the **Count Is Required** item is checked, the processor or node acquisition request becomes a minimum requirement, without which the job will not be launched and, if **Queue this Job** is selected, will be queued for a later attempt. In that case, the acquisition attempt will be repeated until these requirements are satisfied. New in 1.8, **Allow Heterogeneous Platforms** will permit node acquisition to choose nodes of both Mac and Linux. Without this selected, Pooch on node 0 of the job will acquire nodes of its own platform.

Grid Job Type

New in 1.6, the Grid job type allows you distribute single-processor tasks on a cluster automatically, exercising a subset of parallel computing called “distributed computing”,⁵ well-known in brute-force key breaking and SETI@Home. This feature implicitly activates the Queue this Job, Automatically Acquire Nodes, and Retrieve Output options because its operation requires these capabilities. Launching this job creates subfolders beginning with “case” where the original executable resides and numbered with the integer assigned to that job. The output of the executables will be returned to these folders. Use the Subdirectory job option to customize these folder names.



- Distribute From: To: - When the **Grid** job type is selected, these entries specify the range of integers Pooch will use for each instance of execution. The Job in the above figure shows that it will launch cases using every integer from 1 to 100, inclusive. Pooch will create a text file named “input” in the same directory as the executable containing that integer. By default, Pooch will also append that integer to the command line of the executable. To change that setting, use the Command-Line Arguments option. The %index argument will be replaced with the integer for that case.

- Step - With the **Grid** job type is selected, this pop-up menu specifies the step

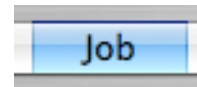
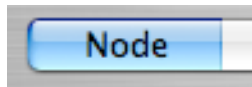
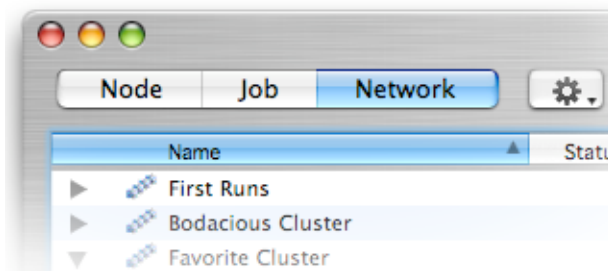
⁵ For more about the different types of parallel computing, see <http://daugerresearch.com/pooch/parallelzoology.html>

between cases. Pooch will start with the From entry and increase the integer by the Step selection. So, if 7 were selected above, Pooch will launch the executable with the integers 1, 8, 15, 22, and so on until case 99.

- Starting Node - Similar to the Starting Node selection of the the **Automatically Acquire Nodes** option, the acquisition of nodes will be performed by the node selected here.

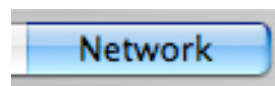
Network Scan Window

Pooch's Network Scan window features three different views, toggled via the segmented view control in its upper left corner.



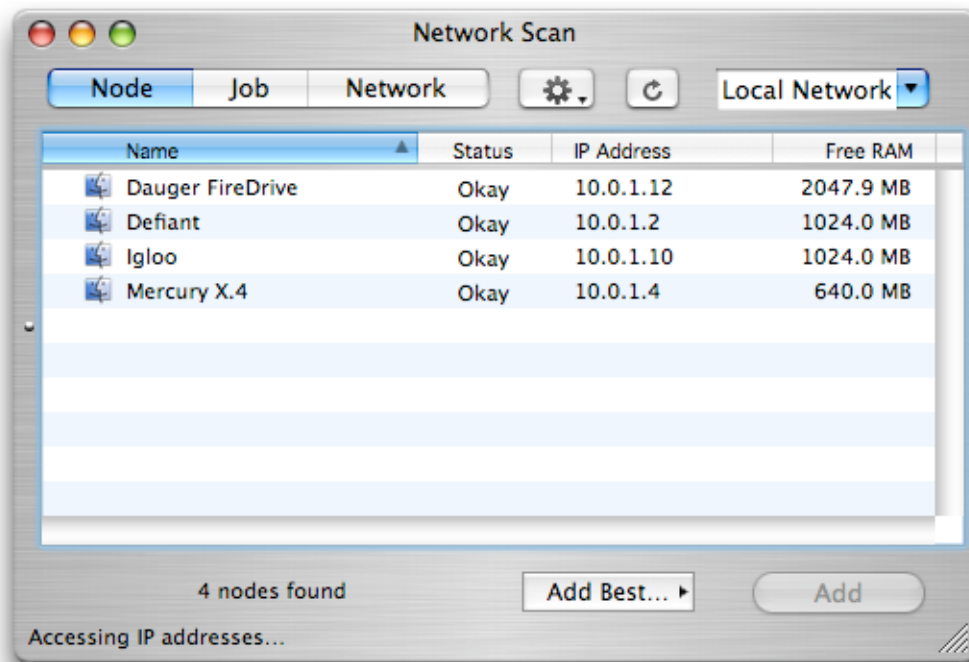
The Node View displays the nodes discovered on the network, along with diagnostics and statistics about these nodes. This view is used to access nodes, query their status and health, and select them for parallel computational jobs.

The Job View retrieves from those nodes data about jobs that are queued, launching, running, or terminated. This view compiles data from these sources to display the condition and other statistics about parallel computing jobs.



New in 1.6, the Network view displays nodes in saved node lists, as well as remotely scanned networks in one large view. Each network of nodes can be revealed by clicking on the node list's or network's disclosure triangle.

Node View



When this window is in Node View, Pooch scans the network for available nodes and displays its results here. Once Pooch has finished scanning, you can use this window to select nodes for your computation. Double-clicking on a node moves it to the node list of the Job Window, which will open if not already present. Clicking the **Add** button while a node is selected will have the same effect. Nodes also present in the Job Window will appear in gray in the Network Scan Window.

Selecting a number from the pop-up menu of **Add Best** moves that many nodes that Pooch considers “best” to the Job Window. The selection excludes nodes that are

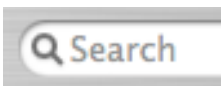
already on the Job Window's node list. The rating system Pooch uses to rank the nodes is described fully in the Network Scan Columns section below. This mechanism is designed to be more likely to select nodes that perform better than others and be less likely to select nodes that are busy or heavily loaded. Selecting the highest number from this menu selects all nodes that Pooch can communicate with and are not busy with a parallel job.

Normally, the Network Scan Window will scan the network again according to the **Automatic Rescan** specified in the Network menu. The **Refresh** button directs Pooch to rescan and refresh the list immediately. This window is resizable for large amounts of information.



- After selecting one node, you can use this Action pop-up button to

get information about that node. The **Get Node Info** and **Get Job Queue** items invoke a Node Info window. The **Get Files** items invokes a Finder-like window from which you can drag files to the Finder. **Connect to Server...** triggers the Finder to attempt to open the node as an AppleShare server.



- The Search box at the top right of the Network Scan

window accepts a search string to limit the displayed data. It compares the search string against the node and job data available in the list below and displays only those items that match. For example, if you type "G5", it will show nodes that use a G5 processor. If you type "dual", only those with two processors will appear. Other text will be compared against the names and other data to reveal appropriate matches. As with any search using text, overly specific search strings will yield little or no data, and overly general strings will produce excess. New in 1.6.



- The address box at the top middle of the window provides access

to remote clusters of nodes, something like the address box in a web browser. (Revised in 1.6.) Pooch normally can only directly scan within its own local network neighborhood. This is the default scanning process. You may always return Pooch to this mode by selecting **Local Network** from this pop-up menu or typing "Local" in the box. Technically, unless your network administrators have set up your routers to support IP multicasting, this means that Pooch can only see other nodes in the same IP subnet. (This limitation is inherited from the Internet-standard Service Location Protocol implemented in Apple's Network Services Location Manager and Apple's Bonjour implementation.)

Entering an IP address in this box allows you to circumvent this limitation. Enter the address, in DNS (e.g., mymac.mydomain.com) or "dotted quad" (e.g., 192.1.2.34) form, of a Mac you know is running Pooch. Your local Pooch will then ask the far away Pooch to do its local scan and relay the results. Essentially, the Pooch on your Mac "borrows the nose" of the other Pooch. This feature allows you to scan for nodes on any other accessible subnet. This is useful for circumstances where, say, you want to access a chem.ucla.edu Mac from a physics.ucla.edu Mac. Selecting **Local Network** returns the network search to your local network.

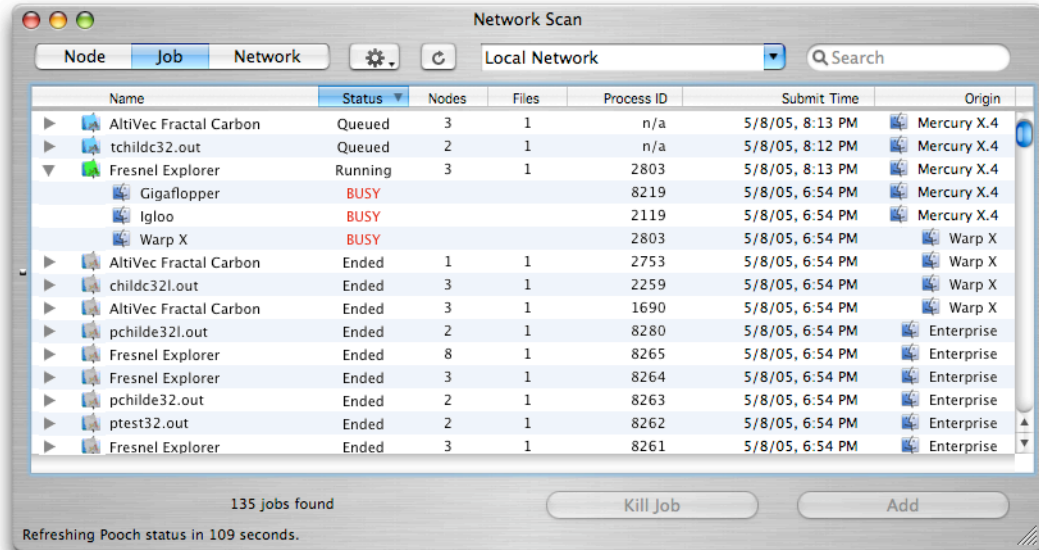
Once you enter one address, Pooch will remember that address and make it available on the menu of the pop-up button for future use. As you accumulate more addresses, Pooch will re-sort the list in order of most recently accessed first.

The functionality of this feature extends well beyond just one domain, however. From your Mac, you may use this to access nodes on any subnet *anywhere on the Internet*. This means you can even access your cluster running Pooch from your home dial-up connection.







This feature is very powerful. With this power comes responsibility. There are security features in Pooch that will make unauthorized access without Pooch

extraordinarily unlikely. These issues are discussed in the Security section.

Job View



When the Network Scan window is in Job View, it scans the network for nodes and accesses data, if present, about queued, launching, running, terminated, and aborted jobs and compiles that job data for display here. The root level of the list shows all the jobs, named according to their executable, found on the network. The job icon for each job is color coded as follows:

Icon	Color	Meaning
	white	A normal parallel computing job
	blue	A queued job ("on ice")
	yellow	The job is being launched
	green	The job is running
	gray	The job has ended
	white X on black	The job was aborted

Opening the disclosure triangle displays the nodes that will be included in, are being used for, or were included in that job, depending on the state of the job. If possible, the information about a particular node will be queried directly, otherwise the data is derived from the data associated with the job.

This view correlates and compiles this data from all the nodes on the network. A variety of otherwise independent jobs may be accessed as far back as the **Job View Access Time** preference will permit. Since nodes used in one job can be reused for another one later, nodes will naturally appear more than once in multiple jobs. Although information such as the job ID, number of nodes and files, origin, and submission time should be consistent between nodes of a job, the process ID, start time, and end time could be different. If certain nodes of a job are not available, data about a node will be derived from the job data, if possible, residing on the other nodes. The Last Access and Determined Via columns are easy ways to tell whether or not the data are fresh.

Network View

The screenshot shows the 'Network Scan' application window. The 'Network' tab is selected, and a search bar is visible. The table below lists various nodes, including those in the 'Favorite Cluster' and 'Local Network' sections. The status of all nodes is 'Okay'.

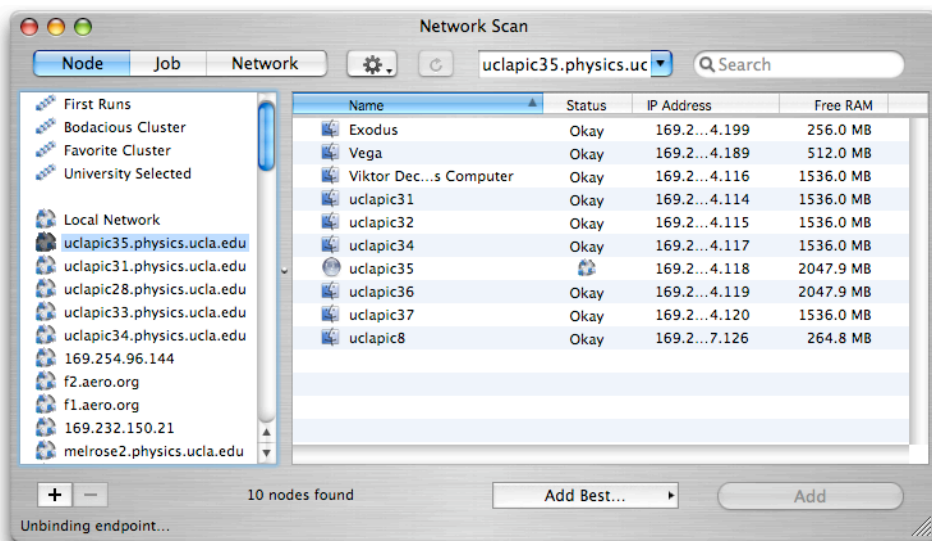
Name	Status	IP Address	Free RAM	Clock Speed	Load	Free Disk Space	Rating
First Runs							
Bodacious Cluster							
Favorite Cluster							
Mercury X.4	Okay	10.0.1.4	640.0 MB	867 MHz	97 %	12772.3 MB	0.0006232262
Defiant	Okay	10.0.1.2	1024.0 MB	1500 MHz	45 %	29235.0 MB	0.4637306
Fifth Element	Okay	10.0.1.12	2047.9 MB	2000 MHz	45 %	2597.1 MB	2.233896
Igloo	Okay	10.0.1.10	1024.0 MB	1000 MHz	37 %	17952.9 MB	2.012097
University Selected							
Local Network							
uclapic35.physics.ucla.edu							
uclapic35	Okay	169....4.118	2047.9 MB	2500 MHz	18 %	103.0 GB	3.026373
uclapic34	Okay	169....4.117	1536.0 MB	2000 MHz	41 %	132.1 GB	2.342041
uclapic36	Okay	169....4.119	2047.9 MB	2500 MHz	36 %	117.3 GB	2.799171
uclapic37	Okay	169....4.120	1536.0 MB	2500 MHz	33 %	138.6 GB	2.713188
uclapic31	Okay	169....4.114	1536.0 MB	2000 MHz	36 %	68087.0 MB	2.630926
Exodus	Okay	169....4.199	256.0 MB	800 MHz	27 %	43206.1 MB	1.9888
Viktor Decyk's Computer	Okay	169....4.116	1536.0 MB	2000 MHz	45 %	143.4 GB	2.304213
uclapic32	Okay	169....4.115	1536.0 MB	2000 MHz	12 %	127.4 GB	2.936219

14 nodes found

Refreshing Pooch status in 180 seconds.

In the Network View, the window organizes nodes into saved node lists followed by remote networks. Opening the disclosure triangle of a node lists displays the nodes it contains and includes those nodes for access. Opening a remote network triggers Pooch to access the remote node and query the nodes in its local area network, just like when using the address box as in the Node View. However, what is different is that you can have multiple networks open and accessible at once, and multiple saved node lists open as well. You may mix and match nodes from these multiple networks when selecting nodes for your parallel computing job. New in 1.6.

Network Pane



When the Network Scan window is initially opened, a pane splitter appears on the far left side of the window. Dragging that to the right opens a network pane, listing both saved node lists and remote networks. The remote networks shown are the same that was entered via the address box earlier, and they list networks in the same order, with the Local Network listed first. This pane gives another way to quickly select and scan a remote network, much like the left pane in OS X's Finder windows.

Clicking on the plus sign below the Network Pane creates a new saved node lists.

You may think of these as iTunes playlists, but for nodes. You may drag nodes from any network to these saved node lists, creating arbitrary groups of nodes useful for different kinds of parallel computing jobs. For example, you may have a group that you know is only available at night, or another set of nodes that are all G5s or dual-processor machines, but are on disparate, mixed networks or clusters. This gives you a way to group commonly used nodes together, allowing you to conveniently view, scan, and monitor these nodes as if they were together and include them in a new job in the Job window. Double-click a node list to scan them. When viewing a saved node list, select a node and press the delete key to remove them from a node list. The Network Pane and saved node lists are new in 1.6.

Network Scan Columns

The Network Scan Window displays information about each node Pooch queries. This information is shown in a spreadsheet-like format. This window is invoked using the columns button at the top of the Network Scan Window or the **Show Columns...** item of the **Network Scan Columns** submenu of the Network menu. The columns can also be added or removed using the **Network Scan Columns** submenu of the Network menu. You can drag a column at its header and resize it by clicking on its right edge. Clicking on a column header re-sorts the list according to that column.

Sometimes, if the node is running a computational intensive task, the Pooch on that node may not be able to respond immediately, delaying the appearance of this information. In that case, it probably should not be used for a new job.

<u>Column</u>	<u>Function</u>
IP Address	the “dotted quad” Internet Protocol address of this node
Machine Type	the machine type, based on its ROMs

CPU Type	the processor type (e.g., G3 or G4), including number of processors
Clock Speed	the processor clock cycle speed in millions of cycles per second
OS Version	the version of the operating system (e.g., 9.0.4 or 10.0)
Load	the estimated load on the system in %. On OS 9, this item uses the amount of CPU time, in sixtieths of a second, each application has consumed according to the Process Manager approximately every ten seconds. On OS X, this item is calculated using the results of a call to the Unix "ps" command. Since the resolution is rather low and the process of measurement can influence the measurement (like Quantum Mechanics), this number can be somewhat inaccurate. It is intended, if the number is measurably nonzero, to hint that that this node is probably in use and should be avoided.
Free Memory	the amount of contiguous free RAM available on this machine, which is the largest available space for a newly launched application. Updated in 1.7 to report more than 2 GB.
Free Disk Space	the amount of free space on the drive on which this node's Pooch resides
Achieved Performance	measured achieved single-precision floating-point performance according to a benchmark based on code from the Power Fractal app. Automatically recognizes the Velocity Engine and SSE, and updated once per day. (not multiprocessor aware)
Peak Single-Precision	

measured single-precision floating-point performance of the FPU of this processor based on an IBM benchmark. Updated once per day. (not multiprocessor aware)

Peak Double-Precision

measured double-precision floating-point performance of the FPU of this processor based on an IBM benchmark. Updated once per day. (not multiprocessor aware)

Peak Vector Float

measured single-precision vector floating-point (`vector float` or `vFloat`) performance of the vector FPU (in the Velocity Engine or SSE) of this processor based on an IBM benchmark. Updated once per day. (not multiprocessor aware)

User Idle Time

the time that has elapsed since Pooch last detected interaction from the user, such as moving the mouse or other activity

Rating

a nonnegative real number describing Pooch's rating of that node. This number is a function of the information Pooch has collected. If the node is busy, or Pooch has not successfully communicated with the node, the rating is zero. Otherwise, it is related to the above properties according to the following heuristic function:

$$f = \ln(1 + (1 - l) p) \frac{2}{\pi} \tan^{-1}(m) \frac{2}{\pi} \tan^{-1}(d) \frac{2}{\pi} \tan^{-1}(u)$$

where p is Achieved Performance divided by 300 MFlops, l is the load (from 0 to 1), m is Free Memory divided by 32 MB, d is Free Disk Space divided by 16 MB, and u is User Idle Time divided by 15 minutes. f is designed to be very low when the node is undesirable to use, such as when it is heavily loaded or is running out of memory or disk space. Note that, assuming load and memory space are non-issues, f is approximately

logarithmic with processor performance. Also, many of these variables can easily change at any moment, causing the rating to fluctuate. This rating function may change in future versions of Pooch. (To create and use your own rating, use the AppleScript interface described in Chapter V.)

Jobs Queued	the number of jobs that are queued on that particular node
Current Job Name	if the node is running a job, the name of the first parallel executable it is running
Last Access	the time and date this data was accessed, according to that node's clock
Pooch Version	the version of Pooch that node is running
Job ID	the job identification number assigned to that job
Node Count	the number of nodes that job will use, is using, or had used
File Count	the number of files supplied for that job, as submitted to Pooch
Process ID	the process identification number of a particular node's instance of the executable of a job when it is running or a job when it was running
Submission Time	the time and date this job was originally submitted
Start Time	the time and date a particular node's instance of the executable of a job was first determined to be running
End Time	the time and date a particular node's instance of the executable of a

job was first determined to be ended

Elapsed Time the amount of time a particular node's instance of the executable of a job took that node

Job Origin the node that was the origin of the job

Determined Via the node that provided the information about this job or node

Finally, the Status column has a number of different possible displays.

Status Reading

Meaning



attempting to connect to this node



a connection was established to Pooch at this node and security authorization is being approved. If the node is under heavy load, Pooch might not be getting enough CPU time to respond. On OS 9, writing your parallel app to give more background time (for example, using the checkesc function in MacMPI) would help.



access to Pooch at this node is in progress



access to Pooch at this node is almost done

Okay

the Pooch at this node was successfully accessed and is not busy

BUSY

the Pooch at this node was successfully accessed and is busy running an application launched by Pooch



("breathing") the Pooch at this node is being used for remote neighborhood access and is returning addresses of and/or accessing other nodes



retrieving job data



retrieving file and folder data



exchanging data or commands

Queued

this job was successfully queued and is awaiting launch

Launching...

Pooch is presently attempting to launch this job

Running

this job was successfully launched and is running

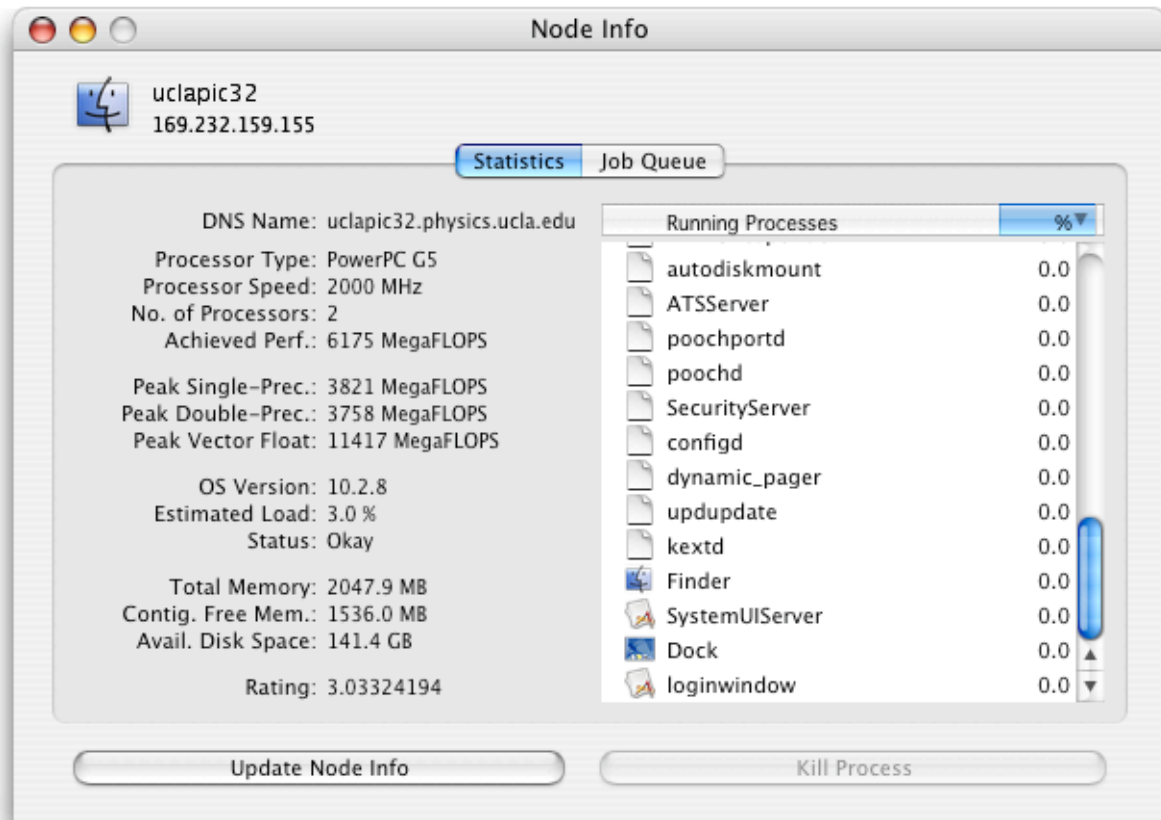
Ended

Pooch has determined that this job terminated

Aborted

Pooch was used to kill this job and it is no longer running

Node Info Window



Selecting one node in the Job window or the Network Scan window allows you to access that node's information. Much of the information on the left is also available in the columns of the Network Scan Window, with a few additions:

<u>Item</u>	<u>Function</u>
DNS Name	the IP name of this node according to the DNS network, if available
Total Memory	the total memory available, including virtual memory (Updated 1.7)

The right side shows a list of running processes each with an approximate distribution of processor load. The determination of this information is OS specific. On

OS 9, Pooch derives this information from sampling the Process Manager every ten seconds. On OS X, Pooch calls the Unix “ps” command, whose results can change at any moment and can be adversely influenced by the call to “ps”.

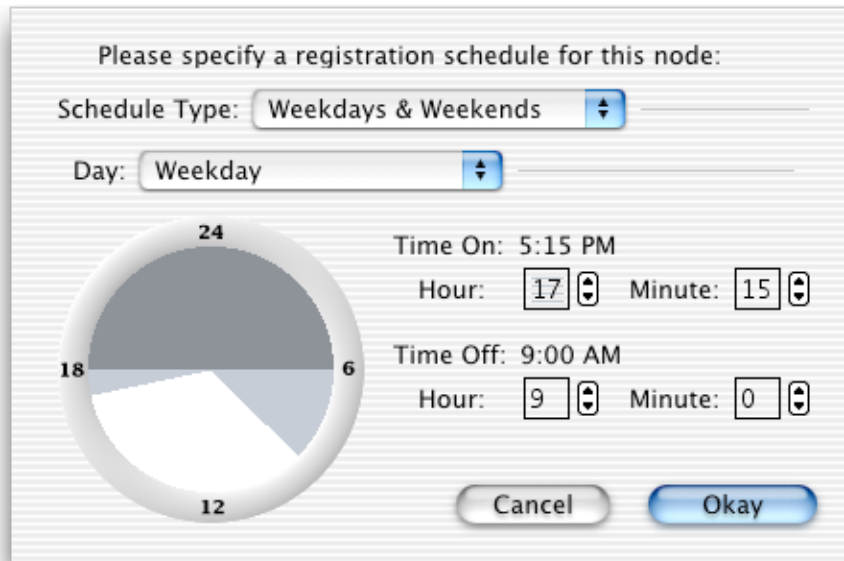
After selecting a process on this list, clicking **Kill Process** will direct the Pooch on that node, if it is running OS 9, to ask that process to quit. Applications that a user is presently using can respond to this request, and some applications ignore this request. If that node is running OS X, Pooch will use a “kill -9” command instead. However, in accordance with the Unix underpinnings of OS X, Pooch can only kill processes that Pooch has permission to kill.

Clicking on the pop-up menu will allow you to switch between **Stats & Apps** and **Job Queue**. In the **Job Queue** mode, this window will display parallel jobs in the queue on this node. The columns list the number of nodes and files selected in this job and the launch time of this job. If no launch time was selected, or if the launch time has passed, it will display Normal. Selecting one will allow you to kill that job using the **Kill Job** button.

Get Files Window

Selecting a node in the Network Scan Window or the Job Window and selecting **Get Files** from the Node menu will have your Pooch access the Pooch on that node and access files there. It displays information about the files in a window much like a Finder window. You can drag files out of it to the Finder, you can double click on a folder to access its contents, and you can Command-click on the window title to pop up a menu to back out a directory. As a security feature, you can only access the folder in which the remote Pooch resides and any of its subdirectories (folder aliases will not resolve), and you can only get files out, not drag files in. This access is completely independent of Mac OS File Sharing.

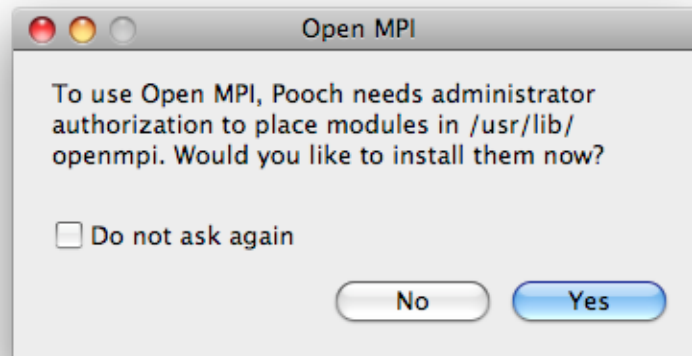
Node Registration Schedule Dialog



This dialog allows you to select the time Pooch registers and deregisters itself on the network, which determines this node's visibility by other Pooches. As seen in the above screen shot, you may set the schedule so that, on weekdays, this node is visible on the network at 5:15 pm, after you leave for home, then no longer is available for computation at 9 am when you arrive at work that morning. The Schedule Type pop-up menu can be set to Everyday, where the daily schedule will be the same every day, Weekdays & Weekends, where the schedule for weekends (Saturday and Sunday) is distinct from the schedule on Monday through Friday, and Weekly, where the schedule on each day of the week can be set separately. The Day pop-up menu selects which day you are editing, which depends on the Schedule Type setting.

For each day, the large circular dial is used to specify what portion of the day Pooch will register itself. The circle represents a 24-hour clock with midnight at the top and noon at the bottom. The highlighted sector of the circle indicates the portion of the day registration will occur. Dragging the mouse inside the circle selects times rounded to the nearest 15 minutes. Dragging outside the circle, or using the text fields, allows you to select any minute of the day.

Open MPI Module Install Window



Only OS X 10.5 “Leopard” and later has Open MPI, one of the newest open-source MPI implementations, built in. This window will appear soon after Pooch recognizes it is running on Leopard. Open MPI’s architecture requires low-level modules to work with other cluster software, but such modules need to be installed in /usr/lib/openmpi to function. Clicking Yes in this dialog triggers installation of mca_pls_pooch and mca_ras_pooch given proper authorization. Once installed, you may use the Job Type > Open MPI option in the Job window to launch Open MPI compiled executables (using Open MPI’s mpicc command, for example. Try “man mpicc” in Leopard.). Because they cannot be used for any other purpose, there is no harm in installing these modules even if you do not use Open MPI. New in 1.7.6.

V. Pooch Pro

Pooch was created to make clustering as easy to use as possible. Generally there are two ways ease-of-use can be applied: 1. minimize the effort to perform an otherwise technically challenging feat; or 2. apply the same amount of effort to achieve much more than could otherwise be done. With the Pooch Application, we have done our best, in the context of cluster computing, to pursue the former while making progress with the latter. It is clear that many users need a solution that focuses on this first goal.

However, we recognize that we can do still more addressing the latter. To do so, we require a version of Pooch that enables larger clusters to be managed for use by a larger number of users. Administrators are often responsible for managing much larger quantities of computational resources than single users usually do. They require more detailed control over how the hardware is used and shared amongst users. Administrators would like to accomplish as much as they can with the effort they employ.

Pooch Pro is designed to support that latter form of cluster computing. This “Pro” version of Pooch supplies the infrastructure of users, groups, and administrative accounts to manage the usage of cluster resources. These capabilities imply mechanisms to identify parallel computing jobs with users and track their time spent against usage policies decided by the cluster administrator as well as an account hierarchy between users and administrators.

But we should be clear that the introduction of Pooch Pro by no means is the end of Pooch. We recognize that a spectrum of cluster solutions exist, from an individual

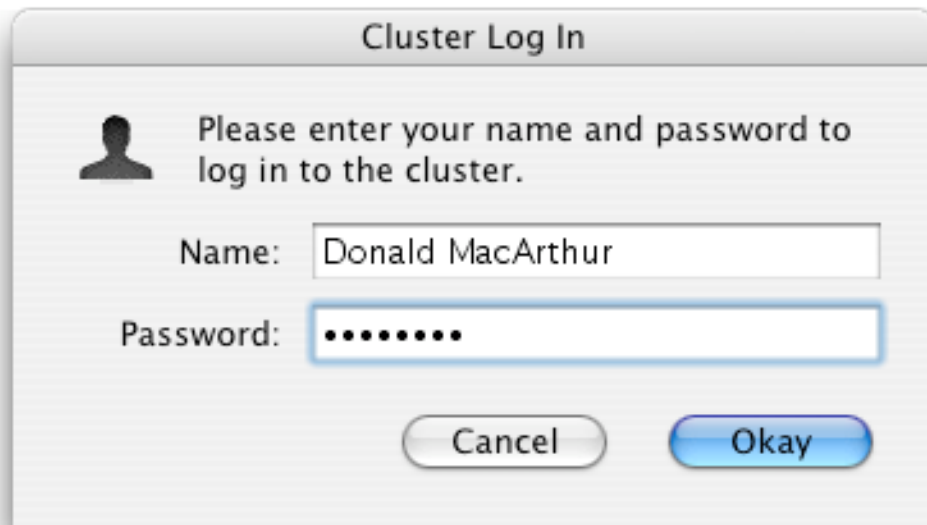
running a few nodes to many users sharing a university lab as a cluster. To better satisfy these cluster categories, Pooch has bifurcated into two incarnations: 1. Pooch continues to make it easy for a user, not necessarily skilled in computing, to keep their cluster up and running so they can maximize their time applying, rather than maintaining, the technology; while 2. Pooch Pro allows one to administer a cluster for a large number of users, all via a user interface easy-to-use for administrators and users alike. Both will evolve and improve to address their users' needs.

When Pooch Pro is first installed, it creates an initial user database file with two groups and two user accounts, user and admin. The password for the both accounts is set to a pseudorandomly generated string supplied with your Pooch CD. (For the Pooch Pro downloadable from the web site, the admin password is "pooch".) When you set up your accounts, we recommend first logging into your admin account and either set a new password or create a new administrator-level account for yourself and delete the old admin account. Then, after configuring the user accounts, use **Update User Database** to distribute your changes to all other cluster nodes. If you need to backup the encrypted user database file, it is at /Users/Shared/pooch/poochuserdata.

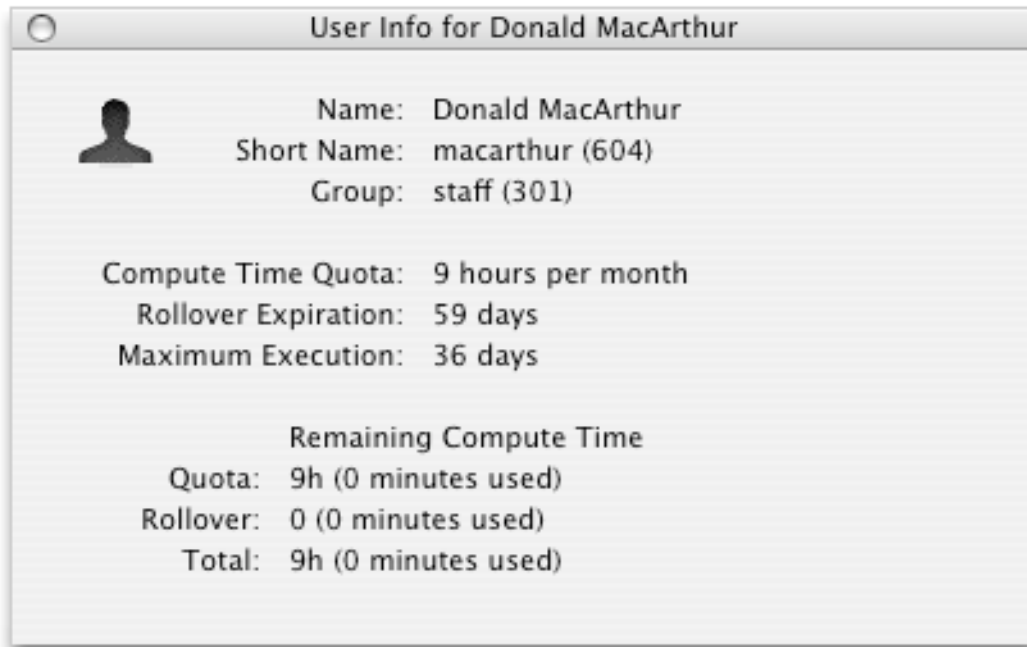
Pooch Pro Menu

User menu

- **Log In.../Log Out** - Opens a Cluster Log In window to initiate use of Pooch Pro after entering a user name and password



- **Change Password...** - Opens a Change Password Window where the user can enter and verify a new password for account log in.
- **Show/Hide User Info** - Toggles a User Info Window for the currently logged in user, showing statistics on the amount of compute time the user has consumed out of their assigned quota. See the User Administration Columns section for more detail on these fields.



- **Migrate User Access...** - Adds a reference describing this user's account information, including password data, to the Job Window. The user may then choose nodes, just as with parallel computing jobs, where a new or updated copy of their user data will reside.

- **Administrate Users...** - Opens the User Administration Window (see the User Administration Window section)

- **Update User Database** - choosing from this submenu allows the distribution of this node's user database to other nodes on the cluster. Assuming all nodes are available and running on the local subnet, the **Automatically** choice will have Pooch Pro automatically acquire those nodes for updating. The **Manually...** item adds the user database to the Job Window, making it possible to choose nodes that may not have been previously available or are outside the local subnet. Ideally, this is an operation that should be performed on all nodes of the cluster.

- **Import Users...** - Opens a file dialog to choose a text file from which Pooch Pro will interpret data to create new users and modify existing ones. This feature allows an administrator to generate a user database based on an external source or text file, then import this user data into Pooch Pro with one action. The data format must be Unix linefeed text (such as using “Make Plain Text” in TextEdit), and the first line contains tab-delimited headers followed by corresponding tab-delimited data. The import ends with the first blank line or at the end of the file. We recommend examining the output of **Export Users** for an example, although the column order does not matter to the import, nor do all columns demonstrated in export need to exist for import. “Password” is also a recognized column. New in 1.6.5.

- **Export Users...** - Opens a file dialog to choose the name and location of a new text file containing tab-delimited data about user settings, including CPU quota and rollover minutes, as well as calculated CPU time used and remaining for each user, parceled between quota versus rollover minutes. This text format should be straightforward to read by standard spreadsheet programs. These header fields are the header keywords used in **Import Users**. New in 1.6.5.

Command-line Access

When using the macmpi.tar.gz package with Pooch Pro, we recommend that you create users on the system whose user names match the user names registered in Pooch Pro’s user database and use the -m flag when launching jobs. This makes it possible correspond command-line based logins (e.g., ssh) with jobs submitted by users known to Pooch Pro. Without those consistent user names, Pooch Pro will not recognize a job submitted by users it does not know, and therefore kill those jobs because they appear to be foreign.

Pooch Pro Windows

User Administration Window

Pooch Pro's User Administration Window provides access to the user and group account database. Usually only administrators would have access to this window. This window features two different views, toggled by the view buttons in its upper left corner. A dot in the close box indicates that user or group data has recently been changed.

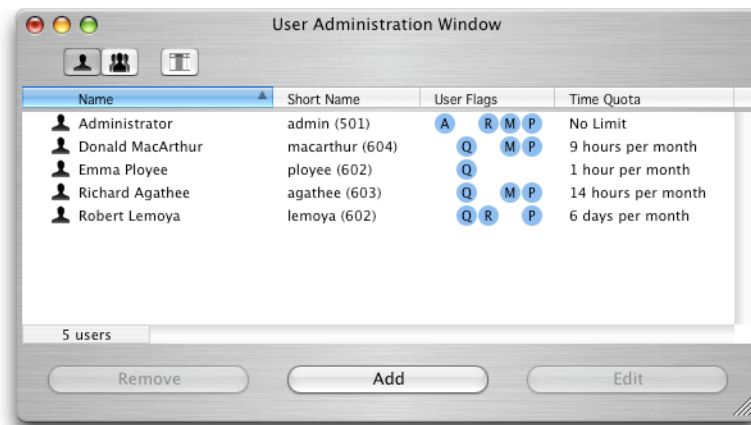


The User View displays all user accounts in the cluster, including statistical data about these users. This view is used to administer, create, edit, and delete user accounts.



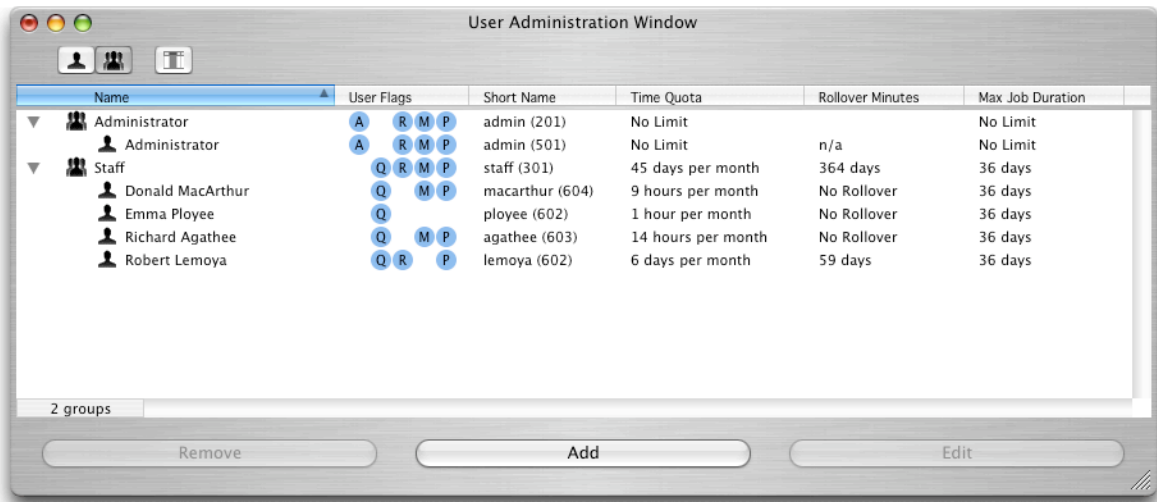
The Group View lists all groups of user accounts on the cluster and the users they contain in a hierarchal format. This view is used to administer groups and organize which users they include.

User View



The User View of the User Administration Window is where user accounts can be created, edited, and deleted by selecting the account and clicking on the buttons at the bottom of the window.

Group View



The Group View of the User Administration Window is where groups can be created, edited, and deleted. Users can be dragged between groups here.

User Administration Columns

The User Administration Window displays information users and groups in a spreadsheet-like format. This window is invoked using the columns button at the top of the User Administration Window. You can drag a column at its header and resize it by clicking on its right edge. Clicking on a column header re-sorts the list according to that column.

<u>Column</u>	<u>Function</u>
Name	the full name of a user or group
Short Name	the short name of a user (a.k.a. user name) or group. The ID number of that group or user is shown in parentheses.
Group	the short name of the user's group, with the group ID in parentheses

Compute Time Quota

the compute time allocated to a user. The quota is specified as a particular amount of time, summed over all nodes used by that user, allowed within a certain wall-clock time. For example, if a user is allocated five hours per day, then Pooch will allow that user's jobs, in aggregate, to use up to five nodes one hour each, or one node for five hours, every day

Rollover Minutes Expiration

the expiration limit of this user's rollover minutes. If a user does not use up their quota on a previous quota interval, the user may carry those minutes over to the next interval. However, those minutes will expire after the time specified here. Using the above example of a user quota set to five hours per day, if this rollover expiration is set to two days and the user did not use any time yesterday, then that user could use up to ten hours on the cluster today.

Maximum Job Duration

the wall-clock time limit of this user's jobs. Jobs submitted by the user will be limited to running for the amount of time specified here.

Total Time Remaining

the sum of the compute time available to the user if that user were to start a job now. Pooch keeps track of jobs submitted by users compares that time against the policies specified by the quota and rollover minutes policy settings. The time left is displayed in this column.

Total Time Used the sum of the compute time used by the user within the quota and rollover minute time intervals

Quota Time Remaining

the sum of the compute time available to the user according to the compute time quota rule only

Quota Time Used the sum of the compute time used by the user within the quota time intervals

Rollover Time Remaining

the sum of the compute time available to the user due to rollover minutes only

Rollover Time Used the sum of the compute time used by the user outside the quota time interval but within the rollover minute expiration

User Flags the flags of a user or group, specifying user policy features, shown graphically. Their meanings are shown in the chart below.

User Flag Icon

Meaning

A

this user or users of this group are allowed to administrate on the cluster

Q

this user or users of this group have a compute time quota limiting their allocation on the cluster

R

this user or users of this group have been allowed rollover minutes

M

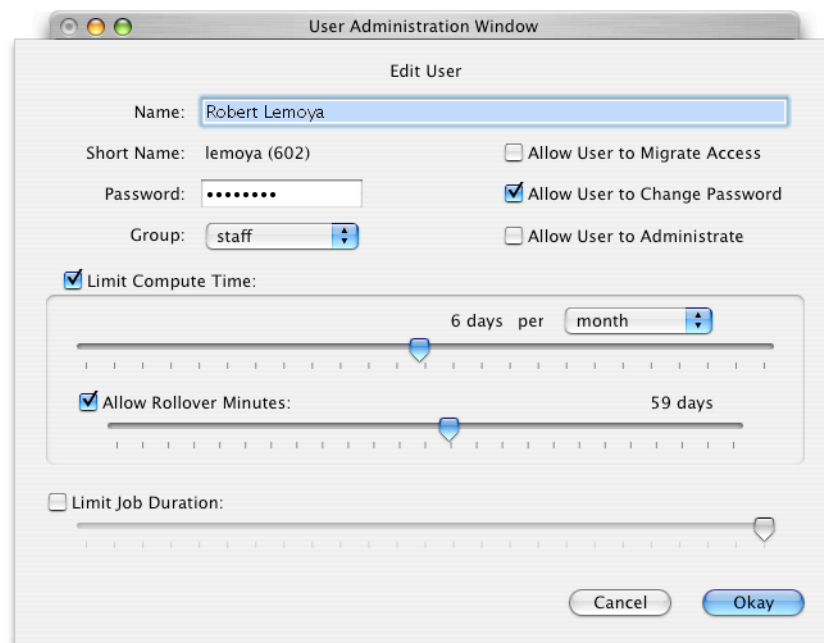
this user or users of this group have been allowed to migrate their

access to the cluster from node to node

- P this user or users of this group have been allowed to change their own password

Edit User Sheet

Editing a user via double-clicking or the Edit button invokes the Edit User Sheet.

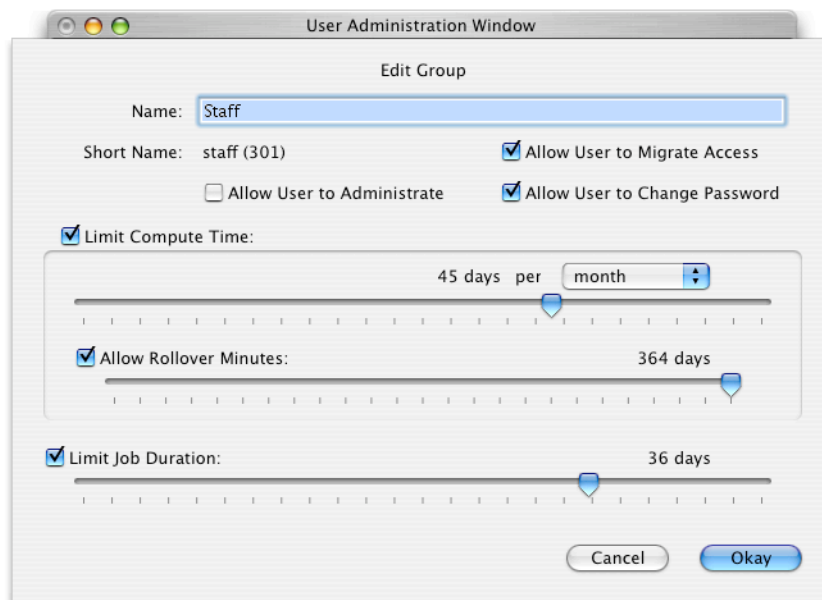


The sliders set the compute time policy for this particular user. A user's total compute quota can be limited to a certain amount of wall-clock node time per time interval, which can be a day, a week, a month, or a year. This time is the sum of all the time this user keeps any node or nodes busy, as tracked by Pooch. This calculation is compared against the quota specified in this dialog and the quota of this user's group. In addition, the user may be allowed minutes to be "rolled over" from previous quota intervals. This feature allows users to combine previously unused minutes with their current quota for their compute time. Users' individual jobs can also be limited to running for a specified

amount of time using the Limit Job Duration slider. When creating a new user, a New User Sheet appears that is very similar to the Edit User Sheet. Unless one is entered, the default password given to new users is “pooch”, all lowercase. Once the short name and user ID is set, they cannot be changed without deleting the user.

Edit Group Sheet

Editing a group via double-clicking or the Edit button while in the Group View invokes the Edit User Sheet.



Like that for an individual users, the sliders set compute time policy, but instead of just a particular user, a policy can be set for all the users in this group at once. A user’s compute time is limited to either that user’s particular quota or the group’s, whichever is more limiting. This hierarchy applies individually to the rollover minutes and job duration limits as well. When creating a new group, a New Group Sheet appears that is very similar to the Edit Group Sheet. Once the short name and group ID are set, they cannot be changed without deleting the group.

AppleScript

Pooch Pro supports three AppleScript types regarding user functions. New in 1.6.5.

```
user    noun    -- a user of the cluster  
group   noun    -- a user group of the cluster
```

The user record is data about the user, including settings specific to the user as well as calculated quota and rollover CPU time used. Similarly, a group record is data about a group. The units of the time-valued properties, such as quota or rollover time, are in minutes. Via an administrator account, these records can be accessed using the usual Standard Suite in AppleScript vocabulary. For example, to get a complete list of users, use the following command:

```
get every user
```

To create a new user, use make:

```
make new user with properties {{name:"Test User", user name:"testuser",  
has quota:yes, quota:10000, group ID:301}}
```

These capabilities are useful to set up and create a new user programmatically via AppleScript or a Unix script. If not all properties are provided, as in the make example, Pooch Pro will fill in the remaining fields with default values. Properties that return calculated time usage and remaining , of course, are ignored if fed into Pooch Pro.

```
login   noun    -- a user's login session to the cluster
```

The login record is about the login state of Pooch. Data about the currently running user can be returned via the get command, while delete can log out the user. To log in from AppleScript, use make:

```
make new login with properties {{user:{user name:"testuser"},  
password:"pooch"}}
```

VI. AppleScript

Pooch has the ability to respond to commands from a script written in AppleScript. This feature makes possible customized launch configurations and queuing systems. For example, simply by dragging an application to it, a script could launch that application automatically onto the four “best” nodes found on the network. Also, it is possible to write and launch scripts from OS X’s Unix command line; thus, launching jobs through Pooch from OS X’s Unix shell is possible. In addition, because the underlying mechanism of AppleScript are AppleEvents, the inter-application messaging system on the Mac OS, it is possible for Mac applications, including mainstream ones, to ask Pooch to query the network and launch parallel computing jobs.

You may look at the Pooch dictionary by selecting Open Dictionary... from the File menu of Script Editor (an application commonly supplied with the Mac OS for editing AppleScript scripts) and selecting Pooch App in that dialog. We also present that vocabulary here. Pooch fits its AppleScript vocabulary within the official AppleScript object-oriented approach, accompanied by only a few Pooch-specific commands and data structures.

Standard Suite

The AppleScript Standard Suite was defined by Apple Computer, Inc., to create a standard vocabulary that all AppleScript-aware applications would understand and respond to in a consistent manner. This vocabulary is meant to encompass as many

operations as possible that a user could do with an application, while allowing only a minimum of application-specific operations to fall outside this suite. The goal is, once an AppleScript user learns this suite once for one application, that user may apply that knowledge to all other applications with the greatest possible efficiency.

```
make  
new type class -- the class of the new element. Keyword 'new' is optional  
in AppleScript  
  [at reference] -- the location at which to insert the element  
  [with data anything] -- the initial data for the element, such as a  
list of alias records for the file list of a job  
  [with properties record] -- the initial values with properties of  
these items, needed for nodes in a node list of a job  
  [with last type class] -- use the data from the last submitted job for  
the new job  
Result: reference -- to the new object(s)
```

`make` is probably the second most common command you will use with Pooch. You may use it to create a new job.

```
set myjob to make new job
```

This command directs Pooch to open a new Job Window with only the node this Pooch is running on as node zero. It then sets the variable `myjob` to be a reference to the new job just created.

By default, `make new job` creates a new job with an empty file list and a node list containing the node this Pooch is running on. “`make new job with last job`” causes Pooch to derive the new job from the last job this Pooch launched according to the settings in the New Job Automatically Reloads submenu.

To add an application or files to the job, use the `make new file` form:

```
make new file at myjob with data alias "Power Fractal" of the startup  
disk
```

This command adds the Power Fractal application, which resides on the root directory of the startup disk, to the file list of `myjob`. The word `alias` typecasts the reference into an alias. Script Editor often checks to make sure the file exists before allowing you to

compile the script. Adding files is similar, and this mechanism also accepts lists of files.

To add a node explicitly to the job, use the make new node form:

```
make new node at myjob with properties node {name:"Mac number 2",  
address:"192.168.1.2"}
```

The information between the brackets is interpreted as a record containing the name and address of the node. This record is typecast as type `node`, then added to the node list of `myjob`. You may also construct a list of nodes,

```
set mynodelist to {{name:"computer01", address:"192.168.0.1"},  
{name:"computer02", address:"192.168.0.2"}}
```

and add that list to the job.

```
make new node at myjob with properties mynodelist
```

In addition, you may add the result of a node scan directly to the node list.

```
make new node at myjob with properties (node scan)
```

The `node scan` command is described in the Pooch Suite section below.

The remaining AppleScript commands of the Standard Suite that Pooch recognizes are:

```
close reference -- the object to close  
count reference -- the object whose elements are to be counted  
delete reference -- the element to delete  
exists reference -- the object in question  
get reference -- the object whose data is to be returned  
open reference -- list of objects to open  
set reference -- the object to change  
to anything -- the new value
```

These commands are general purpose because they can refer to almost any object, within reason, with the Pooch application. Some of them are used simply, e.g., `close job window` closes the job window, or `close every window` closes all windows. `count`

the node list of myjob returns an integer how many nodes are in `myjob`. Using the `open` command is the equivalent of dragging the item onto the Pooch icon, so its function overlaps with `make new file`. Commands that do not make logical sense (such as `close BUSY check of myjob`) either return an error or do nothing.

The objects that these commands act upon can vary widely. Individual nodes of a `node list` or files of a `file list` can be referred to by number, such as `node 2` or `file 4` or `item 5`. Note that, in AppleScript, all counting is 1-based, so the nodes of an `n-node node list` are `node 1` through `node n`. The `node list` or `file list` of a job can also be specified. For example, `exists file 4 of the file list of myjob` tests if that file of the job exists, while `delete the node list of myjob` clears the node list. In the former example, `file 4 of the file list of myjob` may also be simplified to `file 4 of myjob` because the reference is clear; however `item 4 of myjob` is not specific enough.

The `set` command is one of the most versatile members in this suite and will probably be the most common command you use. Once you create a job and set the variable `j` to be a reference to that job:

```
set j to make new job
```

you can set entire file or node lists:

```
set the file list of j to {alias "Enterprise:Pooch folder:pinput2d"}
```

```
set the node list of j to {{name:"computer01", address:"192.168.0.1"},  
{name:"computer02", address:"192.168.0.2"}}
```

set a particular file of a file list to another member of the same list:

```
set file 2 of j to file 1 of j
```

or set particular options of the current job:

```
set BUSY check of j to no
```

```
set launch failure queues job of j to yes
```

```
set target subfolder of j to "run001 folder"
```

```
set start time of j to date "Wednesday, October 31, 2001 12:00:00 AM"
```

```
set tasks per computer of j to use processor count
```

Setting the `target subfolder` option to `"` and `start time` to current date or an earlier

date turns off those job options. All the substructures that make up a job are listed in the job class listing in the Pooch Suite section of Pooch's dictionary.

In addition Pooch's Application Class has a two additional properties from the normal suite. You can query the `launching a parallel job` property to find out whether or not Pooch is currently launching a parallel job onto nodes. This information is useful for scripts or parallel applications to coordinate its activities with Pooch's launch process. Also, you may find out whether or not this Pooch is registered by accessing the `node registration` property.

Pooch Suite

The commands particular to Pooch are in the Pooch Suite. Expanded in 1.6.

```
node scan reference
  [for best integer] -- Search for a given number of 'best' nodes
  [for [best] processors integer] -- Search for a given number of
processors using the 'best' nodes
  [starting at string] -- scan starting at a remote node specified by an
IP address in a character string
  [BUSY boolean] -- Include the nodes labeled BUSY
  [all boolean] -- Include all nodes found
  [new scan boolean] -- Explicitly perform a new node scan
Result:  node -- list of nodes
```

```
job scan reference
  [starting at string] -- scan starting at a remote node specified by an
IP address in a character string
  [all boolean] -- Include all jobs found
  [new scan boolean] -- Explicitly perform a new node scan
  [lists boolean] -- Explicitly include node and file lists in job records
  [recalling integer] -- Recalls job records back this far in time
(e.g., days, weeks, or months)
Result:  job -- list of jobs
```

```
network scan reference
  [for best integer] -- Search for a given number of 'best' nodes
  [for [best] processors integer] -- Search for a given number of
processors using the 'best' nodes
  [starting at string] -- scan starting at a remote node specified by an
IP address in a character string
```



```

[BUSY boolean] -- Include the nodes labeled BUSY
[all boolean] -- Include all nodes found
[new scan boolean] -- Explicitly perform a new node scan
[nodes boolean] -- return found nodes
[jobs boolean] -- return found jobs
[lists boolean] -- Explicitly include node and file lists in job records
[hidden boolean] -- hide the network scan to reduce its visual impact
Result: node or job -- list of nodes or jobs

```

`network scan` directs Pooch to open the Network Scan Window and begin scanning the network. By default it returns node records, but the `with jobs` switch will have it return job records instead. `job scan` is equivalent to `network scan with jobs`, and `node scan` is equivalent to `network scan with nodes`. Pooch will wait about eight seconds before returning the results of its search. The command will only return nodes which, in that time, successfully reported their status as Okay after being discovered. Note that the results of this search may or may not overlap with the nodes already present in a job you are constructing.

When the returned object is a list of nodes, each item of this list is a record containing information about the node's name, IP address, load, free memory, OS version, achieved performance, rating, and so on. These items generally correspond to the node scan columns described in detail in Chapter IV. A complete list of the members of these records are listed and defined in the `node` class section of the Pooch Suite in the Pooch dictionary. Much of the information about a node can change at any time, so it is not recommended to cache data about other nodes. Likewise, job data can change and evolve over time, as queued jobs are launched and running jobs end.

Passing an IP address (in a string format, such as "192.168.1.7" or "farawaymac.otherdomain.edu") in the `starting at` option directs Pooch to perform a Remote Node Scan using a Pooch at that address. The success of this operation depends on whether or not a Pooch is actually running on a Mac at that address. Assuming that Pooch successfully relays information about other nodes, your Pooch will request the status of those nodes and return the results to your script as before.

Passing an integer `n` in the `for best` option of this command directs Pooch to

return only the best n nodes of the search for nodes. Nodes that are busy or cannot be contacted are never included in this list. It is possible for this command to return less than n nodes. Which nodes are the “best” nodes is determined by Pooch’s internal rating calculation, which depends upon data about that node which change from moment to moment. To determine the “best” nodes according to your own criteria, you can construct a script that sorts the results of `node scan` using a rating function of your own design.

To specify a number of processors to use, use the `for processors` option of this command. Pooch uses the same criteria as the `for best` option, except it limits the nodes it returns so that the sum of the number of processors on the return list most closely match the number of processors specified here. This is useful if you are using the Use Processor Count setting of the Tasks per Computer job option and want to launch only a limited number of tasks.

The output of this routine can be placed directly into a job’s node list:

```
make new node at j with properties node scan for best 4
```

but you may instead wish to edit the search results yourself before passing it to your job:

```
set nodesearch to node scan for best 4 starting at
"farawaymac.otherdomain.com"
-- edit the list in nodesearch here
```

...

--

```
make new node at j with properties nodesearch
```

Be creative!

```
launch job -- job to launch: required
```

The `launch` command directs Pooch to submit a job that you pass by reference. For example,

```
launch myjob
```

would launch a job that you originally made using `set myjob to make new job`.

bark

This command makes Pooch bark. Try it and listen for yourself.

Additional Reading

Documentation about AppleScript in general is available in print and on the web.

Derrick Schneider (with Hans Handsen & Tim Holmes), *The Tao of AppleScript*, 2nd edition, (BMUG and Hayden Books, Indianapolis, Indiana, USA, 1994).

Apple's official AppleScript web site <http://applescript.apple.com/>

MacScripter - an independent web site with links, information, and resources, including numerous examples, related to scripting on the Mac <http://www.macscripter.net/>

Apple's AppleScript mailing list <http://www.lists.apple.com/applescript-users>

VII. Linux

New in version 1.8, Pooch now supports using compute nodes running 64-bit Linux.⁶ This allows a user to control and direct jobs from a Mac onto a Linux cluster running Pooch. This is now possible because Pooch infrastructure code has been ported to run on Linux. Pooch's primary user interface is on the Macintosh, but now it can control a Linux-based compute cluster.

The primary scenario involving Linux is that the primary debugging and development of parallel code would run on a nearby Mac cluster running Pooch. That could be called the "development cluster". Once the parallel code is optimized and running well on the development cluster, then it can make sense to recompile the parallel job for Linux for running on the compute cluster. It is not ideal to use the Linux cluster for development since it does not have the advanced graphical user interface on the Mac present in both Pooch and MacMPI so useful for development, debugging and optimization. But it can make sense that the "production" calculations would be performed on the compute cluster composed of Macs or Linux nodes. One of the Macs in the development cluster, or one Mac near the Linux compute cluster, can serve to administrate and direct the nodes of the compute cluster.

To use this scenario effectively, we need a way for the Pooch on the Mac to specify an executable on the Linux cluster. Since the advent of Mac OS X, Pooch has saved many of its files in a common area in `/Users/Shared/pooch`. To use Linux, we extend that concept for this purpose.

⁶ As of this writing, Pooch on Linux has been tested with Ubuntu 9, SUSE Enterprise Linux 10, and Red Hat Linux 5.3.

Pooch Directory	Macintosh	Linux
Pooch directory for files common to all users on a node, such as job history and user data	/Users/Shared	/var/pooch
Pooch binary directory containing executables appropriate for launch onto a cluster by Pooch and visible to users on other nodes	/Users/Shared/bin	/var/pooch/bin
Directory for the Pooch executable	/Applications/Pooch	/var/pooch/sbin
Directory for temporary files and executables for ongoing jobs	/tmp/pooch	/tmp/pooch

The Mac part of the cluster can be used for the most user-intensive part of cluster computing: developing and debugging the parallel application. When running the code for production, both Mac and Linux nodes can be utilized.

In most other respects the Linux nodes running Pooch can be monitored and selected like any other cluster node running Pooch. The Linux nodes are visible in the Network Scan window, accessible via the Node Info window, selectable for inclusion in the Job window, and even accessed via the Get Files window. In the Job window, it is significant that one can select, from the Mac, a Linux binary on the target node, but add to that a text or other input file from the Mac. So it would make sense that the parallel application be able to accept all parameters from an “input deck” or other input data, written locally by the user. The Retrieve Output option of the Job window will also cause the Linux node to supply data back to the user’s machine.

Installation

Pooch on Linux begins with copying the `poochinstaller.tar.gz` file to the node running 64-bit Linux. The steps and commands are:

1. Copy the poochinstaller.tar.gz file to the Linux machine (e.g., using scp or sftp)
2. Log in to the Linux machine (e.g., ssh) to issue the following commands at the directory of the poochinstaller.tar.gz:
3. tar -xzf poochinstaller.tar.gz
4. cd poochinstaller
5. su
6. - enter your password for root access -
7. ./install
8. exit
9. logout

Pooch should then be installed on this Linux machine and running. You should be able to verify this via the Network Scan window of Pooch on your Mac.

Command-Line Utilities

With Pooch on Linux, we supply two command-line utilities:

poochstat - Queries Pooch to perform a network scan and report its results. It can scan for both nodes and jobs. Use "poochstat -h" to report its command-line options.

poochsub - Submits an executable and optional input files as a job to Pooch. It allows specification of nodes by IP address or automatically acquire a certain number of nodes or processors. Use "poochsub -h" to report its command-line options.

Mixing Linux and Mac Nodes

The key to cross-compiling a parallel computing between Linux and Mac is to have an MPI that communicates in a compatible way. The Cluster SDK contains

numerous examples, but here we focus on those that use LnxMPI_SUB.c. LnxMPI_SUB is a special version of MacMPI whose graphical features are absent, uses sockets, but is also compatible with the other “UB” versions of MacMPI. It has the added bonus that it can be compiled under 64-bit mode and still communicate correctly with its 32-bit counterparts. The executable that is compiled on Linux can be compiled 64-bit with the parallel application’s source code, yet the one on Macintosh can be compiled 32-bit with LnxMPI_SUB.c or MacMPI_SUB.c or MacMPI_XUB.c. We have tested these scenarios.

We have done as much as we can to make that compatible, however, the application code also needs to be flexible between 64-bit and 32-bit. For example, in C, a “long” in 32-bit is a 32-bit integer, but a “long” in 64-bit is a 64-bit integer, so when sending an array of long’s could end up double or half of what the receiver expects. It is best to replace these with explicit-sized integers or other fundamental types, such as int32_t and uint32_t in C or INTEGER*4 and INTEGER*8 in Fortran. 64-bit computing is clearly in our immediate future, as it is available on all major platforms.

Once the executables are properly compiled, the Linux executable can be placed in the /var/pooch/bin directory on the Linux node, and the Mac executable can be placed in the /Users/Shared/pooch/bin in the Mac node. New AppleScript commands can specify that the Job recognize the local versions of each of these executables in the Job window.

VIII. Security

Pooch is its own lock and key. You should keep track of your Pooch like you keep track of your keys.

Before Pooch will accept commands from another Pooch, it must receive a passcode that matches its own. Then, all subsequent commands use a 512-bit encryption key that rotates for each message in a psuedo-random manner. Only those two Pooches can predict the next encryption and decryption keys. If a mistake in the passcode or commands is made at any time, Pooch will reject the connection. Since Pooch waits a second or two before it accepts another connection, an exhaustive search for the correct encryption keys (2^{512} possibilities once per second would take over 10^{145} years) will be extraordinarily unlikely to succeed.

The first passcode and the start of the rotating key are 512-bit psuedo-random numbers derived from the registration name of that Pooch (which is set at compile time). Therefore, only Pooches of the same registration will be able to communicate with one another. Because the registration name is unique for each Pooch customer, a copy of Pooch registered to, say, MIT, will not be able to communicate with a Pooch registered to UC Berkeley. (For cross-registered Pooches or other custom configurations or encryption methods, please e-mail Dager Research.)

Security for your cluster then becomes dependent on the security of your Pooch registered with your registration name. Your Pooch can be installed on the Macs of your cluster, and, if no additional copies of Pooch exist, no one can get in. But if you make a

copy of that Pooch and bring it home to access the cluster, the security of that cluster depends on how securely you keep that extra copy of Pooch.

The nature of this security is analogous to having the ability to copy a key to a locked office. It is not uncommon to entrust a group of people with the keys to a shared resource, such as office equipment. The security of the equipment is shared by those who have copies of the key. These people understand the responsibility that comes with the privilege for that access. Access to Pooch can be shared in a similar way.

In testing by Dager Research, Pooch's ability to break through firewalls is no greater than that of a typical web browser.

If someone loses a key to an office, it is not uncommon for the office locks to be rekeyed and a new set of keys distributed to the users of that office. Likewise, if you feel the security of your Pooch has been compromised, it is possible to obtain a "rekeyed" Pooch. The Pooch code allows psuedo-random variants of the 512-bit passcode based on the same registration name. The rekeyed Pooch will not communicate with the previous Pooch of the same registration. An active Pooch subscription with Dager Research will enable you to obtain a limited number of rekeyed Pooches.

If you are using the downloadable demonstration version of Pooch, you should be aware that the same version can be downloaded by anyone else on the web. So, if they have the IP address of your Mac, they could access your Mac, to the extent that Pooch allows, over the Internet. Although guessing your Mac's IP address is unlikely, a uniquely registered Pooch makes for much better security.

IX. Frequently Asked Questions

Does Pooch work on machines running an OS before OS 9?

We're sorry, but no. Pooch uses many of the latest APIs implemented by Apple. The Data Browser interface was sufficiently implemented only in CarbonLib 1.2, and the NSLM v1.1 implementation, used by Pooch to register and search for nodes, requires OS 9. Pooch's predecessor, the Launch Den Mother and Launch Puppy, runs on OS 8.x and is available from daugerresearch.com, but is officially not supported.

Does Pooch work on OS X?

Yes. Pooch runs great on both OS X and 9. We have even run parallel apps on a mixed cluster where some Macs are running OS 9 and some running OS X 10.1 and some running OS X 10.2 through 10.5. Due to bugs in earlier OS X versions, we highly recommend using at least OS X 10.2.1 or later.

Can I use Pooch to run a node while it is logged out?

Yes. This feature is new as of version 1.3.5. You may either use the standard Pooch Installer (while holding down the option key) or the Pooch command-line installer, `poochclinstaller.tar.gz`. Due to security issues in OS X, a restart may be required to fully enable this feature. Jobs launched onto logged out nodes will run behind the login window. Because of OS X's design, that is the only way the executable can reliably run in that environment. In 1.8, this is the default behavior of Pooch on Linux.

How do I disable the Pooch's run at logout feature?

You may do so by deselecting the Enable Pooch at Logout... preference. Due to security features in OS X, administrative authorization and a restart may be required to fully disable this feature.

How do I uninstall Pooch completely?

You may remove Pooch and all its components by allowing Pooch to run normally, then holding down the Option key after launching the Pooch Installer. In the Pooch Installer, a dialog should appear that allows you to select either to upgrade or uninstall Pooch. Clicking on uninstall will delete the running Pooch and the components that allow it to run at logout. The latter process may require administrative authorization. In 1.8, the Linux version of Pooch can be uninstalled using the "uninstall" script in the poochinstaller.tar.gz.

How do I obtain a Pooch subscription, and what do I get?

With your ordered copy of Pooch, you get a one-year subscription, which can be renewed annually at 25% of the purchase price (at the time of renewal) of a new copy. As a subscriber, you will receive free updates to the software (approximately twice per year) and technical support via e-mail. The subscription also entitles you to up to six free rekeyed Pooches, if you feel your security has been compromised, per year. If you would like to upgrade your Pooch, please contact Dager Research.

When I get an update, do I have to reinstall it on every Mac myself?

On the CD you receive, Dager Research, Inc., will supply an updater program we call a Pooch Package. You can use the usual Pooch launching mechanism to deliver this Package to all your nodes, which will then unpack and update Pooch to the new one on all your nodes automatically and simultaneously. No reboot necessary.

What is different about Pooch compared to its predecessor, the Launch Den Mother and Launch Puppy?

LDM & LP were a pair of Macintosh applications that assist in starting a parallel application. I created their earliest incarnations in 1998. Neither were designed to run all the time in the background. The Launch Den Mother discovered the existence and addresses of nodes using AppleTalk calls that looked for a side-effect of the Program Linking toggle in File Sharing. It then sent an AppleEvent (of a type allowed only when Guest Access was on) to the Finder on that Mac to launch its Launch Puppy. Correct execution of this AppleEvent required that the LP be in a very particular location on a particular hard drive. Once LP was up, LDM communicated with it via the Program-to-Program Communications Toolbox (PPC Toolbox), introduced in 1990 to support AppleEvents and AppleScript, written to use AppleTalk only, and toggled via the aforementioned Program Linking button. Finally, LDM's user interface consisted of one large CustomGetFile dialog box. LDM & LP contained an innovative combination of technologies. There was nothing else out there that could do what they could do, and they became used in scores of Mac clusters worldwide.

Then, Apple announced the future of the Mac OS to be OS X. Once the specifics about X's application support became clear (early 2000), it was evident that LDM & LP were in trouble. CustomGetFile was not supported in Carbon, so its associated 3000 lines of code would be useless. Although AppleEvents would be available in another form on X, Guest Access was permanently disabled (although there were ways to use password access). The calls LDM used to discover nodes (e.g., PLookupName) did not exist in Carbon, so another method must be found. PPC Toolbox was completely unsupported in X. (Just try to find a Program Linking button in native X.) Apple itself was discouraging the use of AppleTalk in favor of TCP/IP. It was clear that a port of LDM & LP to X was completely impractical: almost everything interesting about LDM relied on APIs Apple listed as "Unsupported".

So, rather than hodgepodge LDM & LP piece by piece for X, I decided to abandon those 13000 lines of LDM & LP source code and start something new. Numerous convenience features intrinsic to AppleTalk did not exist in TCP/IP, so a completely fresh approach was necessary. Although the ideas had been rolling around in my head since early 2000, only the CarbonLib's available in late 2000 had enough functionality to provide what I needed. Then, in early 2001 (simultaneous with finishing my doctoral dissertation), I developed Pooch. Pooch:

- uses Open Transport TCP/IP networking for communications
- uses Data Browser APIs to display information about nodes
- uses three separate windows, split off from the LDM dialog some would call "cockpit-like"
- uses Service Location Protocol (SLP), through the Network Services Location Manager, to post information about itself and discover other nodes on a network

Using these APIs and experience with LDM & LP resulted in fundamental design requirements in Pooch different from LDM. For example, on OS 9, SLP *requires* a registration by a *running* application, forcing Pooch to run all the time. But, I figured I would take advantage of these requirements rather than fight them. These fundamental design differences gave the opportunity to do things that LDM & LP could never do:

- become one application. Pooch is, in a sense, schizophrenic. It has numerous different interacting programs inside that enable it to respond to many different possible requests at once.
- run anywhere on any hard drive. The AppleEvent to start the Launch Puppy was no longer necessary. Since Pooch was running anyway, it knows where it is.
- continuously monitor information about the system. Rather than rely on side-effects, it could keep track of running jobs, or load averages, or performance benchmarks, etc., itself and report such information to a querying Pooch at any time without disturbing a user, if present.

- maintain a job queue. Because Pooch runs continuously, it can retain a queue, initiate a parallel job at any specified time, and make decisions about the situation as it arises.
- be accessible over the Internet. TCP/IP, even with SLP, aren't enough to do all of what AppleTalk does. So I added the Remote Node Search feature to extend the reach of SLP to other subnets. But this has the consequence that anyone on the Internet can access the nodes, leading to the addition of security features in Pooch. And the nature of these features, relying on registration names, was to allow the user never to remember passwords, a convenience we had in LDM.
- radically improved, streamlined, and standards-compliant AppleScript support, allowing for Pooch to be directed by queuing systems, from OS X's Unix command line, or by other applications.
- the ability to launch jobs that use other MPI's for their message passing, in addition to those using MacMPI. This exemplifies how the best of the Mac is being combined with the best of Unix.
- and more. We have features in mind for the future of Pooch.

And then there were other features added to enhance the user experience, such as extending the drag-and-drop metaphor from the icon to the Job Window, becoming a repository for files and nodes and allowing the entire Finder to become a file dialog.

But the primary goal was the same: Make operating a powerful parallel computer as simple and as easy to use as possible. Easy enough for anyone to use.

X. Revision History

v1.8.3 August 21, 2011

Converted for use with **OS X 10.7 "Lion"** including:

- Pooch revised to disable use of deprecated components such as Network Services Library
- Conversion of the Pooch Installer and Pooch Package to Universal Binary
- Conversion of other Unix-level components to PowerPC and Intel fat binary
- Other minor fixes

In detail: Disabled attempts to use use Network Services Library in OS X Lion because NSL, and hence Service Location Protocol, is no longer supported. Use of Rendezvous continues. Converted Pooch Installer and Pooch Package to Universal Binary because Rosetta is removed from OS X Lion. Converted poochdaemon and poochclinstaller to PowerPC and Intel fat binary because Rosetta is removed from OS X Lion. These changes required subsequent changes the production process for producing Pooch CDs for customers. Revised use of Authorization Services in OS X allowing last operation sufficient time to complete before attempting the next, otherwise data and execution collisions would occur leaving the one or more administrator operations incomplete, for example, when installing Open MPI modules. In Linux, if Pooch cannot write to `/var/pooch/`, create a `/tmp/pooch/var/` for placement of preference and queuing resources.

v1.8.2 May 6, 2011

Several updates to Pooch include:

- Support for and corrections when using UTF-8 characters in the Job Window, Node Info window, and Network Scan window so that non-Roman characters show in computer and file names and other entities on these windows
- Corrected an Open MPI installation issue
- Updated methods on Mac OS X to acquire local IP addresses
- Other minor updates and fixes

In detail: Added support for UTF-8 characters, such as in node names, and protection from badly-formed CFStrings that could affect the Job Window, the Node Info window, and the Network Scan Window. Use of `getifaddrs`, as in Linux, to obtain local IP addresses in Mac OS X. Prevented trigger of the Finder while running at logout. Implemented greater flexibility in licensing for high numbers of processors per box. Modified installation of Open MPI modules to make use of administrator access if available so that installation is more likely to succeed. Prevented a CFString memory leak in the User Administration Window. Replaced "... " with "... " in messages and logs for Linux compatibility. On Linux, if node is not given a unique name, the name given is now includes its IP address.

v1.8.1 June 1, 2010

Several updates to Pooch include:

- Updated Open MPI support for Snow Leopard, 64-bit, and G5s
- Extended support for computer names composed of non-Roman characters
- Bundled with a new version of the Pooch QuickTime Exporter adding compatibility with iMovie '08 and '09

- Updated methods to acquire local IP addresses
- Other minor updates and fixes

In detail: Open MPI support updated for Snow Leopard, 64-bit, and G5s. Updated pls and ras pooch modules for Open MPI, and recognizes previously installed pls and ras modules. Made application path list access more fault tolerant, including when only one processor is used on a node. Fixed a bug mistaking lamd for orted. Added ability to accept computer names without MacRoman or ASCII encoding, including Japanese characters, as well as propagating these names to Bonjour registration and all fields in the Network Scan window, the Job window, the Get Node Info window, and the Get Files window. Changed Mac routine for getting IP addresses to not use DNS mechanisms, which caused long delays on Snow Leopard Server. Protected Bonjour support from premature internal data disposal. Added timeout at connection time during launch sequence. Updated Pooch Log output to handle longer strings. Corrected side effect of Get Files measuring a file size so that it begins reading at the top of the file. Added Get Files support to recognize / for directory access and long file names. Fixed crash on AppleEvent access to remote tunnel list if list is empty. Fixed get IP address list routine on Linux to better handle loopback or other local-only addresses. Confirmed that /tmp/pooch folder in Linux has full write access. Added recognition of reseller registration data for Linux. Improved command-line support on Linux for entering raw IP addresses. Updated Linux file info support. When copying folders on Linux, propagate all file metadata, not just permissions, while when source file info is not available, default to full write privileges. Updated makefile for Linux Pooch. Corrected side effect of no resource fork in Linux when accessing Get Files.

v1.8 November 17, 2009

Significant new updates include:

- Linux: Pooch can now cluster nodes running **64-bit Linux**, combined with Mac

- 64-bit: Major internal revisions for 64-bit, particularly updated data types and structures, for **Mac OS X 10.6 "Snow Leopard"** and 64-bit Linux
- Sockets: Major revisions to internal networking to adapt to BSD Sockets, as recommended by Apple moving forward and required for Linux
- POSIX Paths: Major revisions to internal file specification format in favor of POSIX paths, recommended by Apple moving forward and required for Linux
- mDNS: Adapted usage of Bonjour service discovery to use Apple's Open Source mDNS library (Thanks to Quinn "the Eskimo"!)
- Pooch Binary directory: Added Pooch binary directory support, making possible launching jobs using a remotely-compiled executable
- Minor updates and fixes needed for Mac OS X 10.6 "Snow Leopard"

In detail: Updated and replaced long, unsigned long, and int datatypes in all parts of the code. Replaced all instances where FSSpec was used to use character paths. Replaced uses of FSIterator with scandir. Replaced access to file system with Unix-compatible equivalents. Added modules for reseller registration. Replaced all Pascal strings with Linux-compatible equivalents. Constructed a Sockets-based communications layer compatible with Pooch, and connected that layer with an abstraction API for backwards compatibility, converting numerous network features in the process to the new APIs and factoring networking and Pooch background task sections. Isolated Mac-specific APIs and data types into an abstracted library for cross-compilation with Linux. Integrated the Linux icon into the Pooch GUI when reporting on Linux nodes. Converted Bonjour support to cross-compile and use Apple's Open Source mDNS library. (Thank you Quinn the Eskimo!) Added Pooch binary directory, protocol, and implementation, making possible launching jobs using a remotely-compiled executable. Added Unix-based port and protocol for local communications with command-line utilities. Added node caching for Pooch binary listing and other status data. Corrected a cosmetic error in the Network Scan window for Snow Leopard. Added "Allow Heterogeneous Platform" flag for automatic node acquisitions, allowing for mixed Linux and Mac launches. Added Apple

Event codes for POSIX path specification, remote file (Pooch binary folder) specification with options, and allow heterogeneous platform acquisition. Fixed an issue locating Terminal.app in Leopard and Snow Leopard due to changes in Terminal's creator specification. Created new command-line utilities poochstat and poochsub. Discovered and compensated for inadequate support by Linux for gethostbyname, directory conventions, malloc_size, getopt, init.d, and fork. Total code length: 115,557 lines in C.

v1.7.6 April 16, 2007

Significant new features include:

- Open MPI support on Mac clusters running OS X 10.5 "Leopard":
 - Complete MPI support using libraries built into Leopard
 - Automatically installs Open MPI pls and ras Pooch modules
- Commands to and detection of the Terminal in 10.5
- Long file, application, and directory name support
- Other features, optimizations, bug fixes, and updates

Leopard-specific Pooch changes: Open MPI support in the Apple Event library, Job window, launch engine. Added Open MPI module install window and implementation. Added support for retrieving full path of executables to be supplied to orterun of Open MPI. Target the Terminal process by bundle ID rather than signature for detection, command, and launch. Compensate for unwritable Carbon app resources. Allow very long (>256) LAM and Open MPI commands. Correctly report error status over the network from LAM and Open MPI command execution. Correctly add resulting executables from LAM and Open MPI launch to watch list. Created orterun support for Open MPI launch. Developed Open MPI pls and ras modules, compiled for PowerPC, Intel, PowerPC 64-bit, and Intel 64-bit, for launching via Pooch: mca_pls_pooch and mca_ras_pooch. Special thanks to J. M. Squyres.

Other Pooch changes in detail: Properly access and store last minor number of system version, like 10.4.11 and beyond, and report them via the Pooch user interface and over the network. Added support for long file names, long application names, and long directory names in Job structures, launch engine, Pooch puppy module, complete file-folder access, IP list file write, recursive file duplication, launch file copy display. When localizing Job handles, search for matching long file names if data present is not specific enough. Increase allowed tasks per job for 8-core Mac Pros. Correctly report status of Pooch tar ball installation. Adjust Network Scan window settle time. Log node list length if accessed via Apple Event. Check that Network Scan settled before finishing network scan Apple Event. Eliminate "www." in download version. Implemented workaround for a bug in NSLM/SLP of OS X 10.4 and later that return nonsense clientContextPtr, causing a possible crash. Prevent discovered nodes from being added when scanning for nodes via a user-defined playlist or remote node scan. Corrected internal message sent status in launch engine state after sending launch command in MacMPI, mpich, MPI/Pro, mpich-gm, LAM, and MPJ job types and job forwarding. Added two minute timeout for file allocation during launch process.

v1.7.5 November 21, 2007

Significant new features include:

- **Mac OS X 10.5 "Leopard"** spurs updates in a variety of Pooch technologies:
 - Network Scan window
 - Preferences window
 - Keychain access
 - Launching via the Terminal
 - Behind the Login window behavior
 - Other user interface and infrastructure adjustments
- MPJ Express support on Mac clusters:
 - Launching MPI-style Java code

- Supports .jar and .class execution types
- AppleScript access to user-defined nodelists and remote network data
- Other features, optimizations, bug fixes, and updates

Leopard-specific Pooch changes: Pooch Network Scan window revision to use toolbar items with segmented controls, including custom menu behavior and new callbacks. Encapsulated Action control menu response. Pooch daemon adjustments. Adjusted ps command options for poochdaemon detection, job detection, and load calculation. Keychain adjustments because interaction allowed flag now works correctly. Adjust Pooch Preferences window to use kWindowUnifiedTitleAndToolbarAttribute. Use larger General, Node, and Job icons in Preferences window. Compensated for Terminal's loss of trmx signature by accessing its bundle ID instead for launching via Terminal. poochdaemon launch into boot context compatible with Leopard.

Other Pooch changes in detail: Added MPJ Express support for launching MPI-style Java jobs on Mac clusters. Added AppleScript interface to set and access MPJ Job type. Added MPJ type to job search keywords. Added recognition of .jar and .class file types as Java executables for MPJ. Added cancel button to Search of Network Scan window. Correctly worked around a notification when fed a nil clientContextPtr from NSLM. Add Network Scan window response to Search button click in addition to return. Add tasks per node minimum. Added accessors to trigger network scan refresh. Fix Node Info window to refresh with new node rather than the old. Updated Pooch Fractal benchmark and peak vector flops to properly use Intel Core 2 chips and beyond. Generalize network access for other listeners port. Access multiple addresses via Unix interface. Properly reported when Job window has changed. Added recognition of gestaltCPUPentium as Intel Core. Added support for access to user nodelist via AppleScript. Adjusted launching diagnostics to report error sources via the Pooch log. Update Login Items preference bundle ID to loginitems. Reorganized and factored network access infrastructure between low-level, proxy, and "puppy" sections. Fixed string list access possible crash when fed zero strings.

v1.7.1 May 15, 2007

Proxy time out. Revise network access for slow connections and large files. Have faceless Pooch reacquire listener port with override. Added proxy connection diagnostics when verbose preference is set. Attempt Keychain access before Job window asks for login data. Add automatic queuing if launching via remote head node. Update and recognize nodes in Job data only when data is non-zero. Handle URLs from NSLM that might be longer than 255 bytes. Update Login Items preference bundle ID to com.apple.loginitems. Accurately report that a job is launching via AppleScript. Enable Keychain error reporting when verbose preference is set. Use Pooch AppleEvents to indicate when to attempt Keychain user access. Correctly report node via data in Network Scan window. Compensate for time zone when reporting job elapsed time across continents. Correct endian swap of job ID and PSN in Network Scan window. Prevent excess substitutions in Job window from Network Scan results. Added maximum node count and remote address list AppleScript access. Add Keychain enable retry in case stale Keychain data exists. Add never include self feature in automatic node acquisition. Address directory update issues affecting the Pooch queue. Fix issue when attempting manual rescan after ten retries have passed. Revise handing of job retrieval for large files and slow connections. Readjust security measures in case of temporary message error. Prevent double endian flips when receiving the job timeline. Readjust retry time outs with reducing retry time for remote node access. Added capability to detect and select node address according to the Unix interface.

v1.7 May 15, 2006

Significant new features include:

- **Universal Binary** for running on mixed clusters of Intel- and PowerPC-Macs
- Remote job, file, and folder retrieval support, even through proxy nodes
- New Job file format support, including saving and loading job data

- Head node setting to access and control clusters with Linux-style topologies
- Enhancements to the AppleEvent interface for job and user data
- Automatic detection and retry of preferred IP address of remote nodes
- Pooch Pro automatic user login using Keychain services
- Other features, enhancements, bug fixes, and updates

Pooch changes in detail: Universal Binary release. Implemented and added user interface for remote job access mechanism. Implemented Job file type support in user interface and Job window, including save sheet reminders and standard save sheet dialogs. Implemented Head Node setting, influencing default behavior in the Network Scan window, the Job forwarding option, via Proxy setting, automatic node acquisition, retrieve output job option, and queuing job option. Implemented more ways to acquire file icons in file pane of Job window. Added support libraries for complete reading and writing of folders and files. Modified Node Access library to locate and retrieve jobs and files on remote nodes. Implemented node access retry in Node Access library to attempt primary address of remote node. Added recognition for .app extension like that of .out. Set job name just prior to job submission. Implemented job window reinvoke after job submission preference. Eased access to port number of job in Job window. Updated use of Open Transport API for endian changes. Increased proxy time based on data transmission sizes. Log list of IP addresses on startup. Added retry for large send sizes from background mechanism. Added reporting of user name data in AppleEvent job structures. Added access delay to launch state in AppleEvent interface when job just launched. Added AppleEvent dictionary entry and implementation for TCP port of job data. Allowed for initial file name to be specified in Navigation dialogs. Fixed queuing system to launch properly if only one job queued. Added logging for launch errors, including identification of problem nodes, and reporting. Fixed endian conversion for multi-virtual node launch. Implemented additional timeout detection in launch mechanism. Implemented optimized secure encryption methods. Implemented secure connections at beginning of launch mechanism, including logging. Fixed allocation bug

on post-launch reporting to origin node. Updated Preferences window with new features and allowed for its resizing. Fixed tunnel address string detection in Network Scan library. Updated Status window with antialiased, positioned fonts. Fixed Navigation data possible crash when file dialogs are closed. Fixed possible crash if refresh in Network view of Network Scan window. Restricted Kill Job button if accessing by proxy, while triggering proxy node if Kill Job is invoked. Edited Status icon to reflect new node states and consistent display of status text. Implemented support to detect and display greater than 2 GB of RAM using sysctl. Implemented workarounds for resource handle bug in recent jobs list, preferences saving, and process watch list. Added support for Nodes per Box specific to a node in job structure and contextual menu access in node pane of Job window. Added support calculations for virtual node specification. Added support for long job names based on long file names. Added support for job identification for retrieval. Fixed endian conversion for grid and node acquire structure types. Fixed peak Vector float flops endian conversion. Fixed endian conversion backwards compatibility with old Job properties structure types as well as new extensions. Added file info parameter record endian conversions. Handled endian PinRect issues in Network Scan Columns and User Administration windows. Extended search support for possible quad- and octuple-processors. Implemented launch of Console.app to display poochlog.txt. Reorganized File menu with new items. Modified Paste behavior to the default in dialogs, the address field in the Network Scan window, and submit to the Job window. Optimized backgrounding timings. Node Info window displays node name in title bar, if available. Allowed Node Info data to display without process data. Updated Total Memory field to include virtual memory estimate, and updated Free Memory field to show greater than 2 GB. Process checking for null jobs before converting endians. Ignore /private prefix in acquired paths to processes. Added routine to tell Console to open a file. Added standardized Nodes per Box Job structuring setting. Altered code to fit within Xgrid's library restriction, fixing a bug that appeared in later versions of OS X 10.4. Updated benchmarking code to include new Power Fractal implementations in Altivec and SSE. Added option to save and retrieve Pooch Pro user login data in Keychain, implementing

time delay if user is not present. Converted endians for encrypted Pooch Pro user data to disk and over the network. Redirected Pooch Job History data into centralized, common system location from user location, with consumption of old job data. Converted endians for Job History data. Implemented Job accumulation cache and filter to optimize network and job scan for user minutes calculation. Added delay to accumulate calculation until user idle. Fixed job accumulation when launch occurred before Network scan. Log if user or group data is corrupt. Converted endians for initial password calculation. Extended Pooch user export/import to include quota and rollover flags. Fixed possible crash when refreshing on Network view of Network Scan window. Fixed Unix permissions of the queued jobs folder. Prevented crash during deletion of one-year-old archived jobs when getting such jobs. Converted endians for Get Files window and Bonjour and NSL/SLP access implementations. Signaled automatic user database update and direct Job window click as higher priority to queuing system. Added icons data for new Job, Job queue, Job history, Address list, and node list file types. Converted endians for client-side proxy node operation. Modified poochdaemon installers to conform to Unix permissions requirements in late versions of 10.4. Revamped Pooch Package and Pooch Installer to incorporate both CFM and UB versions of Pooch and install the appropriate version, from OS 9 to Intel. Implemented hidden feature to launch jobs on both OS 9 nodes and Intel nodes.

v1.6.5: January 18, 2006

Significant new features include:

- Updates to utilization of technologies in Mac OS X 10.4 "Tiger"
- Enhancements to the command-line interface
- Pooch Pro supports import and export User Data via text files
- Pooch Pro enables access to user data and login state via AppleScript
- Other feature enhancements, bug fixes, and adaptations to changes in OS X

Pooch changes in detail: Modified /tmp/pooch folder creation to set all Unix permissions to true. Fixed search to find flops and processor count correctly. Added to AppleScript interface to support new command-line features. Fixed ps parser to not read past end of file or duplicate ps output. Randomized ps temp file name to reduce file and permissions conflicts. Modified message diagnostics to print more reliably. Added ability to open Pooch job queue files to create a new Job window. Prevented pooch.unix from calling Process Manager or Apple Event Manager. Migrated Node Info window and its functions to use Node Access library. Modified the Pooch at Logout feature to use root privileges to start the poochdaemon. Removed excess debugging messages, enabling some according to user's preference. Added cmd-ctrl-Q trigger for launch queue. Fixed text import to navigate folders. Fixed CPU time tracking to exclude null data. Fixed Network Scan window's nodelist pane behavior to properly delete playlist entries. Implemented Kill All Job behavior when option-clicking on Kill Job button, also displaying "Kill All Jobs". Fixed job status field display color when it follows a **BUSY** node in Node Scan window. Implemented export and import of user data via tab-delimited text files, flexibly parsing header and data fields. Implemented new AppleScript data types for user, group, login session, and time interval selectors and connecting the handlers to the user database, making it possible to modify the login state and user database of the cluster via scripts. Added default processor per box selector. Enhanced randomness of TCP port number selection to help prevent conflicts. Added recognition of dual-core G5 CPU type.

Added endian flippers for all 40+ data structure types and implemented them at all data entrance and exit points in Pooch due to migration to Universal. Modified header dependencies, disabled bridge code, modified resource behavior and icon access, compensated for missing target ID AppleEvent coercion and API changes for migration to Xcode from Metrowerks. Added recognition of Intel CPU type for Pentium 4.

v1.6: May 10, 2005

Significant new features include:

- Utilization of new technologies in Mac OS X 10.4 "Tiger":
 - Automator - Incorporate Pooch actions into workflows
 - Dashboard - View cluster status and activity in an aesthetic environment
 - Spotlight - Locate data in Pooch jobs via the system
 - Xgrid - Pooch makes Xgrid nodes accessible to its users
- Massively renovated Network Scan window, following cues from iTunes, Finder, and Safari, including support for user-defined node lists in their own movable Network pane, a multipurpose action pop-up menu, an address box, an all-new Network view mode, and even a Search mechanism
- Extended AppleScript fidelity, including exposing job data acquired on the cluster
- Support for grid-style jobs, specifying arbitrary argument ranges for identical executables
- Revised queuing system with a centralized queue for all users on the system
- Revisions to adapt to adjustments in OS X 10.4 "Tiger", particularly the StartupItem mechanisms
- Support for a new web-based interface
- Other feature enhancements, bug fixes, and adaptations to changes in OS X

Pooch changes in detail: Major renovations to the Network Scan window, both at internal and external levels. Internally, factored the routines into separate components for node access, network scanning and compilation, and the window user interface, making it possible for each lower-level component to operate independently of the upper-level components. Externally, used the Segemented control to switch views, added a new Network View, added a playlist-inspired left-side pane, with a splitter that detects mouse-overs and saves state, for user-defined node lists and networks, replaced the remote node scan button with an address box with a pop-up menu, accessors to alter job history lookback dynamically, added an action pop-up to eliminate the node

info pop-up and network scan columns button, added a Search field to reduce both node and job data, added alternating colored backgrounds in DataBrowser for OS X 10.4, reorganized the refresh button and other controls, following cues from Finder, iTunes, and Safari. All this while still properly supporting drag-and-drop between Network Scan panes and the Job window and preserving the previous look for OS X 10.2, 10.1, and OS 9. Added ability to automatically hide the Network Scan window. Generalized job aggregate data structures. Worked around bug that corrupted QuickDraw after editing DataBrowser item. Updated CarbonEvent handlers. Added routines to translate HIRect and the traditional QuickDraw Rect. Added dozens of wrappers and constants for support routines for the new and updated controls in the Network Scan window. Added the ability to add complete node data, node acquire data, and arbitrary job data to the Job window, as well as being able to acquire a complete job record from the Job window. Created and implemented new Grid Job Type, with the ability to launch a series of jobs, stepping through a range of integers for an executable and customize how the index is entered on the command line and how the index appears in the subfolders. Added actions on the event of a job ending, including creating a job log file to include in automatically retrieved files. Fixed a Job properties access bug when no files or nodes are present. Added Job name and Node List set/get accessors for job structures. Abstracted file list updates. Inserted workaround for jumping Job window drawers, especially in 10.4. Adjusting to changes in OS X 10.4, altered the Pooch StartupItem installation to properly set unix permissions and eliminate excess files, and wrote a new mechanism based on CFPreferences to add Pooch to the Account Startup Items list. Created a new version of Pooch that operates in the Mach-O space without a user interface, adjusting all components to the new environment, and eliminating CFM wrapper routines, yet preserving the bark. Added implementations for menu items, configuration dialog, and Xgrid job launching routine. Reorganized internal Unix and utility routines. Generalized StringListHandle support routines. Altered network error reporting routines for selective reporting into poochlog file. Converted all external mentions of "Rendezvous" with "Bonjour" for 10.3 and later. Altered the

Bonjour internal interface to optionally expose multiple addresses per node. Reduced potential memory leak in resolution callback. Installed correct call sequence to stop Bonjour's services resolver, eliminating "burden on the network" warnings. Added detection and reporting of nil infoPtr returned from NSLPrepareRequest. Added new triggers to the job launch queue. Created all-new implementation of job queue internal data system to centralize the queue on the machine and update the format for OS X, implementing new job data storage and retrieval systems while automatically converting data from the old system. New queue is more efficient and reports on possible errors in poochlog. Redirected Launch mechanism messages to poochlog and factored node scan for automatic acquire code if running in Unix mode. Reduced declarations of Job properties structure. Generalized the file/folder scan structures and the user login detection routine. Reduced AppleEvent dependencies on old url data structures. Fixed scrambled name bug when importing nodes without names. Added ability to accept and expose node acquire data, file list entity, node list entity, and complete job data via AppleEvents. Added and implemented network scan and job scan, with new modifiers, AppleScript commands to acquire job data instead of node data. Detected new clock speed format for > 2GHz and fixed clock speed display. Added calculations using UTCTime. Altered 64-bit comparison calculations. Cached system version Gestalt calls. Updated FSRef routines to report descriptive errors. Created node and job data to text conversion routine for Search and other features. Generalized Node List structure manipulation routines. Added ability to minimize windows on command and detect mouse movement events. Added self identity modification address list implementation and import routines, generalizing friendly-address implementation. Fixed a potential bug when importing addresses while the Job window was open. Altered subdirectory creation to not delete existing directories. Changed Node Info window to use frames only before 10.3 and provide a horizontal scroll in the Job Queue view. Added accessors for strings in preferences file. Altered window names and hide Settings menu on 10.4 to follow OS X conventions. Added Job file type.

v1.5.5: September 15, 2004

Significant new features include:

- Support for jobs using LAM/MPI on local TCP/IP-based networks
- Support for mpich-gm, enabling Pooch to launch jobs on clusters connected using Myrinet hardware
- Forwarding a job via another node, such as a "head node", to submit jobs to nodes behind a firewall
- Automatic output retrieval system, detecting and returning new and modified files once the job that wrote them is finished
- Better optimized and more reliable proxy address access
- Limiting access to a node from a specific set of IP addresses
- Choosing a preferred primary address if there are multiple
- Other minor feature enhancements and bug fixes

In detail: Reorganized job processing and archival routines. Added user data to job record before launch. Added logging to /Users/Shared/pooch/poochlog.txt. Added data recording to launch mechanism for LAM. Added logging of launch events to poochlog. Optimized calls to update Status Bar during launch, reducing flicker and excess CPU consumption. Factored forwarding job detection. Added preparation, access, command, and launch functions for mpich-gm and LAM, connecting them to the main launch engine. Cached calls to TickCount(). Added function and detection flags to redirect file collection for auto retrieve function. Added check for nil commID before proceeding with launch. Added calls to place Terminal.app in foreground when launching using Terminal. Added archiving of forwarded job data. Report when communication retry count exceeded. Added 3 second timeout for key rotation function during launch. Added recognition of G4 7447 and 7457 and G5 970 and 970FX in Network Scan Window, Node Info Window, and AppleEvent results. Added check for nil ProcessSerialNumber in quit and AppleEvent functions as workaround to

prevent killing user space. Extended explicit collection of command-line data from ps command. Increased scope of pid search for wraparound process numbers. Prevent overflow of command string read from ps. Report reason and quantity limit in error string when user policy triggers job kill. Added Perl script routine. Added check for 0 process id to prevent logout of user space (#3805951). Added retention and processing of friendly addresses in listener functions. Redirect application file of watch list and archived job for multiple executable instance when launching multiple tasks per node. Added simultaneous send of multiple packet data to proxy controller routines. Added state requirements for mpich-gm launch command. Added state and explicit check for access to Internet Services and Open Transport before endpoint access with appropriate retry loopback. Report access from outside friendly address list. Added slave LAM command. Added AutoRetrieve directory command. Separated functions for network access and listener activity, making cluster access possible while another Pooch is running. Added primary address setting with Network menu modifications. Added pro version application AppleEvent variable. Added flags to node scan AppleEvent to report all and BUSY nodes, with a new scan performed only when requested or required. Added cancel launch AppleEvent. Added auto retrieve, forward via, and user name AppleEvent access to job data. Added mpich-gm and lam job types to AppleEvent access. Added LAM path access. Retrieve node status data even if communication is not complete. Added check to prevent user file addition to jobs. Include process path watch to BUSY check. Factored common preferences access. Cleaned up job view entry additions in Network Scan Window. Optimized and bottlenecked status text drawing in Network Scan Window, adding grayscale option. Adjusted initial order for job view. Overwrite job item entry only if node "owns" the job. Added ability to send multiple special commands to a node during network scan. Correctly kill multiple task per node jobs via Network Scan Window. Reorder node status access if special command are to be sent. Optimized accumulated job history task. Made Rendezvous and NSLM registration sensitive to listener state. Added user interface for friendly address import in menu and preferences window. Adjusted User Admin Window column dimensions. v1.5.6:

Prevented crash during deletion of one-year-old archived jobs.

v1.5: June 28, 2004

Significant new features include:

- The debut of Pooch Pro, introducing:
- User, group, and administrative account system, including secure password log in
- Quota and time limit-based management of cluster resources, including rollover minutes
- User database management with Unix-style user and group organization
- User associated job tracking and job history database
- A new modern, graphically-based user administration interface
- Other minor feature enhancements and bug fixes

In detail: New User menu and infrastructure for user log in and log out, password modification, user info display, user migration, user administration, and user database updating. New Show Job View menu item. New detection of User Idle Time, reporting mechanisms in the Node Info Window and the Node Scan Window and incorporation into the Rating function. Improved reporting of dialog status for compatibility with Carbon events. Pooch Pro requires log in for Job Window to function. Added recognition of correct user and group icons to the Job Window. Revised Job Window file acceptance for secure handling of user database files. Revised job launch rules incorporating user compute time quota calculations. Added flags to Status structure for Pooch Pro and encryption algorithm existence and User Idle Time. Extended Node Scan Window column infrastructure to allow over 32 columns while maintaining backwards compatibility. Corrected count of selection in Job View of Node Scan Window. Fixed activation of Get File Window and Get URL from Job View of Node Scan Window. Updated Pooch Version column of Node Scan Window for "Pro" label. Added Elapsed Time calculation and User Name column for jobs listed in the Node Scan Window.

Restricted users from killing others' jobs, except for administrators, via the Node Scan Window. Added standard Okay/Cancel and double-entry sheets. Modified parallel progress bar structure for additional flags for Pooch Pro. Added version number to About Box. Added links to User Administration accessors for windows routines. Made Job submission visible to Pooch Pro code. Created wrapper library for access to Apple's CDSA implementation for strong encryption and decryption. Added mechanisms to filter commands requiring correct execution of job launch and other Pooch communication sequences. Added support for encrypted communications. Cleared out job handle on new connections. Added support for incorporation of new user data. Added block for queuing system based on user compute time quota calculation. Fixed an issue where cancel was not accepted during the file info step of the launch sequence. Added Node Scan Window support for view specification on open. Added time interval to string conversion. Added mechanism to save current process watch list to disk in case Pooch is restarted. Change to leave last call to ps in place. Added comparison of job execution time with user compute time quota calculation and maximum duration limits to process watch routine. Introduced new Pooch resource structure and support infrastructure for cross-platform compatible resource files. Corrected Job Archive Limit preference for year limit. New modules for user database management and file infrastructure, including password hash and user limit resolution. New administrative windows and dialogs. New job history accumulation and management organized by user. New user log in management for reporting of currently logged in user's data. New Pooch Pro icon and About Box graphic.

v1.4.5: May 14, 2004

Significant new features include:

- Revised, renamed, and reorganized main menus for greater clarity and consistency with the OS X user-interface guidelines
- New proxy connection support for access behind a firewall when using the

Remote Node Scan feature

- New user-specified control over port number selection range for compatibility with firewalls
- Enhanced node selection the Job View and the Recent Items drawer of the Job Window
- Updated Pooch Preferences window reflecting new features and new behaviors
- Other minor feature enhancements and bug fixes

In detail: New Network menu. New hierarchal menus for Launch Unix Jobs, Access Nodes via, Local Node Scan, and Node Deregistration. New Job Port Range support and user interface. New Job Queue menu item in File menu. New Proxy Connection support for up to sixteen simultaneous bidirectional connections via the node assisting with the Remote Node Scan. New preference to prevent remote access to processes not launched by Pooch. Adjusted event callbacks to not filter key events when dialogs are open. Had Demo version remove poochdaemon at expiration. Changed Node Scan Window user interface, logic, and accessors for other windows to respond correctly to node selection in Job View, including double-clicking and contextual menus. Allowed multiple selection in Job View. Allowed network connections to nodes to continue while adding them to the Job Window. Fixed an issue which corrupted status information, particularly processor counts, when adding nodes to the Node pane of the Job Window. Added support for OS X-style, callback-supporting, single-entry dialog sheets supporting copy and paste. Report retry count in Status column of Network Scan Column when Verbose Error Reporting is on. Report proxy node in Determined Via column when connection attempt to a node via that proxy. When sorting by Status in Job View, secondary sorting occurs using Submission Time. Added proper sorting using Process ID. When new nodes are introduced to the Network Scan Window, SLP data is chosen first, letting Rendezvous data can be reaccessed later. Generalized automatic rescan mechanism to extend time for proxy connections. Allowed for very long (up to 127 characters) DNS names for IP addresses. When proxy is available, Network Scan Window

automatically alternates between proxy and direct attempts with a pseudorandom starting try, extending timeouts exponentially. Added support for simple OS X-style Cancel and Okay dialogs. Added proper support for Command- to simple modal dialogs. Added new and renamed preferences to Preferences Window, requiring its heightening. Adjusted menu disabling and hiding for new Network menu. By performing post-open triggers, worked around an issue where Job Window's drawers, while opening, would randomly drag their parent window around the screen. Added drag-out support for items in Recent Items drawer. Allowed multiple selections in Recent Items drawer. Utilized new-style sheets for Job Options requiring them. Updated plst and vers resources to allow Version data to appear correctly in Finder. Extended Automatic Rescan interval to ten minutes. Added support for use of /Users/Shared folder instead of /tmp. Disallowed Kill Process in Node Info Window when no process data given. Corrected pid data reporting when concluding pid search. Supported restriction of process ID data. Passed proxy access flag through Status structure. Fixed node structure creator routine for correct behavior with nil strings. Added automatic listener endpoint check for failure and auto-recovery. Separated small network send buffer from receive buffer. Added response to T_DATA event on OTLookErr. Corrected bug in endpoint struct open routine to set TCP_NODELAY. Extended timeout delaying reuse of endpoint structures. Added dialog indicating success of Startup Items setting. Revised splash graphic.

v1.4: January 2, 2004

Significant new features include:

- A new queuing and scheduling system for retaining jobs until specific conditions are met integrated with:
- A new job tracking system designed to track the progress and execution of jobs and retain historical data about terminated jobs
- An enhanced and updated graphical user interface that includes the new

“brushed metal” look available in OS X 10.3 and later.

- The Node Scan Window is now the Network Scan Window, divided into a Node View and a new Job View mode to scan the entire network for queued, launching, running, terminated, and aborted jobs and display statistics about these jobs
- An updated Node Info Window displaying additional data and utilizing tabbed viewing and consistency with the OS X user-interface guidelines
- New drawers for the Job Window: the Recent Items drawer displays and recalls recently used files and nodes, and nodes of the remote node scan cache, and the Options drawer accesses all the options of the job
- An “Automatically Acquire Nodes” feature so jobs can automatically acquire nodes given a minimum number of nodes or processors to acquire starting at any specified node
- An updated, antialiased Start Time dialog for the Job Window
- A new Pooch Preferences window, accessible from the Pooch menu, featuring toolbar-style access to preferences in Pooch
- Enhancements to preferences and settings
- New installation choices, triggered by the option key, in the Pooch Installer
- A variety of minor bug fixes

In detail: Added node list type to the Apple Event dictionary and enabled access to nodelist_ip information via AppleScript and interapplication communication. Added accessors to security authentication and authorization in OS X. Restructured Pooch utility code into six separate modules depending on category and type. Customized behaviors and appearance to OS X 10.2 and 10.3. Enhanced signaling and communication between Network Scan Window and the Apple Event support and launch modules. Prevented registration of 0.0.0.0 address via SLP. Prevented the automatic SLP registration retry from disturbing other code behavior. Upgraded CarbonLib headers to version 1.6. Added and implemented automatic node acquire mode of the Job Window and launching mechanism, with corresponding adjustments

for job monitoring and display. Added Job Option drawer with corresponding controls. Added Recent Items drawer to Job Window. Added clipboard distribution feature, including pasting into the Job Window and copying using remote data. Correct handling of nil AllAddresses data from Rendezvous. Searched through all addresses resolved by Rendezvous for valid addresses, rather than only use the first. Correctly tallied Rendezvous addresses with repeat accesses through the Network Scan Window. Responds to Rendezvous discovery events at all times while the Network Scan Window is open, rather than just during the SLP scan. Added and implemented zoom button for Get Files window. Allowed Node Info and Job Queue to be invoked when viewing the Get Files window. Enabled delayed trigger for Get Files and Node Info windows for compatibility with Carbon Events. Used smaller system font for consistency with Finder. Replaced old Job IDs for the queuing system with Queued Job IDs. Implemented Job Timeline data structures to hold past, present, and future job data. Implemented job archive and retrieval mechanism, establishing Job resource type. Updated Process Watch List to incorporate job as well as process data. Implemented Job Timeline accessors for Job Archive, Process Watch List, and Job Queue. Added launch mechanism to Network Scan Window subscriber list. Prevented uninitialized data from entering the Apple Event identification routine. Accessorized Pooch version data for access by multiple parts of Pooch. Fixed an issue where the Apple Event node scan command would retrieve BUSY nodes. Limited node communication retries in the launch mechanism to 10, then reporting failure. Separated submission jobs versus working jobs in the launch mechanism, including appropriate resubmission into queue. Bridged compatibility between new Job Properties data and older Job Queue accessors and kill mechanism, including automatic acquire features. Check for null address in job data. Allowed job submission to place new jobs at either end of queue. Allowed display of IP address in launch status window if no name is present. Once launched, all processes are now submitted with job and file data for full recognition and tracking. Correctly initialized LaunchApplication records in launch mechanism. Enabled preparation and retrieval of Job Timeline data on remote nodes with GetJobTimelineCommand. Attempted to

register network communication activity as user activity for screen savers. Track and retain data about aborted jobs. Implemented command-line argument Job Option on remote nodes. Performed a+w permission operation on all files created by Pooch. Prevent caching of 0.0.0.0 IP address. Returns true when checking if 127.0.0.1 is self. Created and implemented Pooch Preference window with three panes. Implemented appropriate Carbon Events to invoke and drive the Pooch Preference window. Added preference to kill the Settings menu. Conversion of Node Scan Window to Node View and Job View of Network Scan Window with addition of Carbon Events, the brushed metal look, and related control modifications. Added mechanisms to retrieve job information and coordinate the killing of jobs across multiple nodes. Added and implemented Zoom button to Network Scan Window. Generalized, and updated with antialiased graphics, the Node Registration interface for use with the Start Time Job Option. Implemented little arrows controls for arrows of the Start Time Job Option. Job Window rejects files if nil is passed for file specification. Check for possible null node names and addresses in Recent Node Lists. Appended Recent Nodes and Files, rather than replace the present node and file lists in the Job. Added Carbon Events and brushed metal look to Job Window. Separated job recall into node list and automatic acquire modes. Job Window edits keyboard focus. Accessorized self node addition to Job Window. Allowed specification of initial Job Windows node list versus automatic acquire option. Eliminated OS 9-style brushed metal when in OS X. Relabeled Launch Job button depending on Job Options. Responded to modified dragging and tracking in Job Window while using Carbon Events. Generalized IconRef access for files and special cases. Use of OS X-style display name instead of actual file name in file list pane of Job Window. Modified highlight region drawing for brushed metal look and automatic acquire option. Added Job ID and submission time initialization when Job Window submits jobs. Abstracted control updating separately for Job Window and its drawers. Embedded Node Info Window's data display in its new two tab control. Node Info window lists changed to smaller system font. Node Info Window no longer disables Node menu inappropriately. Progress pie in Node Info Window is antialiased in OS X.

Rearranged data display to hold more data and animate as retrieval completed. Allowed for scrolling in Job Queue mode. Accepted supplementary Status structure data. Display proper rounding for OS version numbers. Worked around potential race condition during fast logout to login transition. Created IconRef accessor to icon resources. Created and implemented Job Archive Time Access Interval preference. Enabled access to Preference Window through Pooch menu item. Enabled and implemented Pooch at Logout toggle with administrative authorization. Added application and other main event implementation through Carbon Events. Extended Network Scan Columns controls in three columns. Added and implemented Job Item lists, with disclosure triangles, and new Job-related display columns for Job View, compiled from retrieved data. Implemented acceptance of supplemental status data, Job Timeline retrieval, and multiple node commands. Designed new suite of Job icons, Network Neighborhood icon, Node Info icon, and others. Added recognition of PowerPC G5 CPU type to Network Scan Window and AppleEvent interface.

v1.3.5: June 20, 2003

Major new features include:

- Support for launching parallel jobs on nodes while they are logged out. This new capability is appropriate for the new Compute Node Xserve and for running on administered Macs such as in a student lab.
- Support for a new suite of command-line utilities
- Improved job support for Unix access

In detail: Added implementation to monitor login/logout state so that it will hide all its menu bar when logged out and quit if the login status changes. Added support for Carbon events to support the Dock menu items and horizontal and vertical scroll wheels in the Node Scan, Get Files, Node Info, and Job Windows. Added support for contextual menus in the Node Scan and Job Windows. Minimized the appearance of the menu bar

while running at logout. Implemented retention of additional column sort criterion and direction information in the Node Scan Window. Implemented time out reset when OT support was off when the user wanted to perform a Node Scan. Added code to prevent the IP address of a node in the Node Scan window from being overwritten while communication with that node was in progress. Added support and preferences for saving files in the Temporary Items folder and /tmp/pooch in addition to the folder where Pooch resides. Implemented Reveal in Finder AppleEvent support for sandboxes. Modified launch mechanisms and job window accessors to accept Unix command-line arguments and convey them to the executables, if applicable, for all job types. Added code to change Pooch's Unix umask so all can read and write files that Pooch creates. Optimized the Recent Node and File List menus so that the menu would not take so long to appear. Added interface and implementation for opening File Sharing from any window that specified a node. Added and implemented skip node zero, job type, tasks per node, and Unix command-line arguments properties to the job definition and Okay/BUSY status and time stamp to the node definition in AppleScript interface. Fixed a bug in the AppleScript interface and the Node Scan Window that reported negative memory if 2 GB of RAM was installed and increased the detection limit to 4 GB. Improved the Node Scan completion queue in the AppleScript interface to more intelligently allow the needed amount of time for communication with other nodes to complete, including while performing a Remote Node Scan. Added a more intelligent Pooch Preferences folder locator in case the normal Preferences folder was not available. Implemented high-level IP address caching and comparison routines with corresponding changes in the launch mechanism and the job window, for cases where Pooch's node has more than one IP address. Implemented a workaround in the Node Registration Schedule dialog so the day selection menu would redraw properly when switching schedule types. Fixed automatic benchmark starting mechanism when the node is not registered. Created a caching mechanism for ps access. Improved internal support for variable length job options, including command-line arguments. Changed default sandbox preferences. Changed launch mechanism to only put Finder in foreground if

Pooch knows it is in the foreground. Optimized mpich launch timing. Changed the behavior of the Subdirectory option of the Job Window to be more likely to create a new subdirectory name rather than reuse the old one. Created a workaround and warning for blank node names, a circumstance that Rendezvous will not accept. Eliminated extraneous diagnostic reporting in Rendezvous implementation. Created a workaround for a DataBrowser display while scrolling bug in 10.2.3. Created a workaround to maintain background colors after DataBrowser reset them to white. Added command-control-R key command to reset Pooch's network and node settings, in case OS X did not notify Pooch.

v1.3.1: December 31, 2002

A few minor bug fixes and cosmetic updates. In particular, a bug that only occurs in OS X 10.1 was fixed having to do with the Node Scan features of Pooch and the new Rendezvous implementation.

v1.3: October 4, 2002

Major new features include:

- Support for launching multiple tasks per physical node, making it possible to use the same parallel communications API for parallel computing both within a box and outside the box
- Registration, discovery, and resolution via Rendezvous, a.k.a. ZeroConfig
- Pooch Lite - a special version of Pooch with no user interface useful for running on administered Macs such as at a student lab

In detail: Created complete Rendezvous registration (with auto-retry), continuous browse, and on-request resolution implementations. Revised Pooch's OS X process acquisition code to focus its search for processes it launched. Excluded requests for null process number watches. Keep track of jobs by session ID so that multiple tasks can be

associated with a particular job. Adjusted for differences in ps on OS X 10.2 versus 10.1. Deletes previous mpich procgroupfile before writing a new one. Extended internal job data format and accessors to handle node information for Tasks per Computer and Use Port Number features. Added internal time stamp to node status information. Implemented discovery accessors and additional job option retention and added these settings to preferences file. Modified slave and master node launching mechanisms for multiple tasks per computer for all MPI types. Modified launch sequence to always query for latest status from nodes before launch and update internal job data. Launch process now forwards all job data to slave nodes before launch. Created complete recursive file and folder copy mechanism. Modified master and slave node to recursively copy executables and files on command. Implemented wait states for mpich launching due to fragility in mpich's p4. Job Window updates node 0 name, address, and status information of the current job if node 0 was this node. Fixed possible memory leak in recent node and file list constructors. Created task per computer counter for Job Window node list display. Fixed node list display error if name was zero length. Adjusted to condensed font if node numbering became too wide. Changed default subdirectory job option suffix from "f" to "folder". Implemented new Tasks per Computer and Port Number job options in Job Window. Modified job window accessors to handle extra node information. Set Running Applications list to order by descending CPU time by default, rather than ascending application name. Optimized Node Info access and disconnection speed. Created higher-level AppleEvent node information accessors and creators and extended implementations to AppleEvent and Job Window accessors. Added AppleEvent access to new Job Window features. Implemented two second minimum time for node scan command to complete to allow Rendezvous and SLP discovery time to initialize. Implemented "for best processors" clause. Fixed node scan request queue bug that prevented more than one request from being queued. Added "time stamp" property for typeNode AppleEvent data. Added information to help tags for Node Scan Window pop-ups. Fixed bug that prevented node scan columns from being reorganized in OS X. Generalized Node Scan Window node list accessors to

handle NodeStruct types from Rendezvous in addition to raw URL data from NSL/SLP. Added reset NSL/SLP list accessor. Encapsulated slaving launching mechanisms for each MPI type. Created workaround for CloseOpenTransport hang bug. Connected node scan tunnel to Rendezvous in addition to NSL/SLP. Extended slave node launching mechanisms to handle multiple Tasks per Computer feature. Separated MPI/Pro launch into its own command. Created and implemented retain job information and duplicate files commands.

v1.2.1: June 30, 2002

When launching jobs via the Terminal.app while Terminal is not yet running, added a search by the name "Terminal.app" in addition to its signature code. Improved the error code reporting in case Terminal.app launching fails. If two copies of Pooch are launched, and the first has already set up its network connections, the second copy of Pooch will remain inert, retrying once per minute, rather than interfere with the first. If NSLStandardDeregister fails, Pooch will report the error, rather than simply beep. Changed the Node Scan Window code so the browser list sort and movability default settings are more consistent across columns. Changed the Rating column margin setting to 5 pixels on OS 9. Corrected a problem in the Job Window where the Launch Job button would remain disabled after using the Recent Node Lists menu. Updated this manual's mpich description to recommend installation in the home directory.

v1.2: March 27, 2002

Major new features include:

- Support for jobs compiled with mpich and MPI-Pro
- Support for Cocoa apps and other bundle-based applications
- Complete support for Unix-based executables, including Mach-O executables and Unix scripts
- Complete support for Unix-style permissions, processes monitoring, process kills

- Registration of nodes on the network can now be scheduled
- Recently selected nodes and files can be recalled

In detail: Added the ability to recognize folders, Unix scripts, bundles, and bundle icons to the Job Window. Added the flexibility of adding more than one executable to the Job Window, while maintaining that the first file is the first executable, even when files are later removed, and labeling that executable with a •. Altered the style specification of columns of the Job Window, Node Scan Window, Node Info Window, and Get Files Window for greater consistency between OS X and OS 9. Altered the internal control and window spacing specification for greater consistency throughout Pooch. Altered the Job Options pop-up to accept submenus. Added and implemented the Job Type submenu to the Job Window. Altered Job Window so its node list displays IP address when there is no name. Altered the Node Info Window text display, Node Scan Window node count placard, Get Files placard, and about box to use the correct antialiased fonts when running on OS X. Added calls to the Remote Node Scan cache of IP addresses to “warm up” the DNS search when the Node Scan Window is opened rather than only when the pop-up is first clicked. Fixed a bug preventing help tags from appearing under some circumstances. Corrected the status string region of the Node Scan Window to enable itself more quickly when its window is made active. Added the Clock Speed column to the Node Scan Window. Prevented the remote node scan node information from being overwritten prematurely. Prevented the Node Scan Window from displaying too many nodes. Optimized the parallel node query retry mechanism. Changed the ThemeWindowBackground of the Start Time dialog and single entry dialogs to the default on OS X. Added code to save and retrieve information about a series of recently launched jobs. Added the recent node and file list submenus with abbreviated names and the ability to restore them to the current Job Window. Changed the Register this Node preference so that it could accept Always, Never, and Scheduled modes instead of just on and off. Added Node Registration Schedule dialog, switchable between Everyday, Weekend/Weekday, and Weekly modes, with corresponding implementation in the node

registration code, while attempting to follow Apple's HI guidelines. Added implementation and interface to save, recall, and display recently used nodes, files, and applications for new jobs, including on-the-fly abbreviation of node names in menus. Correctly retrieve and recognize Unix permissions of all files and folders on OS X, then use that information to recognize Unix-based applications. All permissions of files and folders are preserved when copying between OS X machines. Added code to correctly recognize, copy, and launch application bundles, especially Cocoa apps. Added the option to launch Unix-based executables via a Unix tcsh script rather than through a call to system. Added code to recognize Process Manager processes by signature. Added the option to launch Unix-based executables through the Terminal app. Terminal is called by sending AppleEvents. If Terminal is not running, it is launched via Launch Services. Added the ability to "kill -9" processes on OS X. Added code for writing procgroup files and specifying Unix command-line options to pass cluster information to the mpich master and slaves. Added environment variable specification and launch code for MPI-Pro jobs. Added the ability to kill all running processes tracked by Pooch at once. Added utilities to extract and search for arbitrary information, including searching for particular POSIX paths or particular process ID numbers, from "ps" output. Organize process tracking by process ID on OS X. Tracks all Unix processes, rather than only those reported by the Carbon Process Manager. Added tracking of Unix process creation by POSIX path, even if the process takes up to a minute to appear in "ps". Extended data structures and added implementations to supply Unix process information to other nodes in addition to Process Manager information while retaining backwards compatibility. Added "node registration" information and implementation to AppleScript interface. Added "job type" to AppleScript job class type and corresponding implementation to specify MacMPI, mpich, and MPI-Pro job types. Correctly clear the job options structure when there are no options present in a job data structure. Added job type, subdirectory, and start time accessors for job data structures. Added and implemented Kill All Jobs on Deregister, Recent Node Lists Length, Recent Apps and Files Lists Length, and Use Terminal For Unix Jobs preferences. Added and implemented the

Accept Remote Preferences Command preference in a private resource that cannot be remotely overwritten. Added and implemented Job Type automatic reloading preference. Schedule more time to launching process during a parallel launch, improving launch speeds. Encapsulated and optimized the Parallel Launch Status window so that it updates itself much more smoothly and cleanly and uses internal timers to leave only a minimal burden on the launch process. Corrected an error on OS X where garbage might appear in the about box when NSL/SLP attempted to register. When NSL/SLP fails to successfully register (e.g., when the network interface is unavailable), Pooch retries with decreasing, random frequency, rather than at regular intervals. Changed the Select Apps and Files dialog to accept folders, bundles, and multiple selections. Correctly report network, hard drive, and system activity to the Power Manager when remote requests and commands occur or when launching jobs. Created new commands to launch bundle-based applications, mpich executables, and MPI-Pro executables, to create a new subdirectory and specify a bundle application simultaneously, and to retrieve OS X file information. Changed the "change directory" command to accept a parent directory specification. Extended the set file information command to accept OS X file information. Fixed a bug where the Parallel Launch Status window progress bars would jump. Altered send file sequence of the launch process to recursively explore a folder structure using FSOpenIterator and FSGetCatalogInfoBulk. Altered the TCP port number information. Provided correct rounding of processor clock speed in Node Info Window, Node Scan Window, and AppleScript interface. Recognizes the PowerPC G4 7455. 36,326 lines of code.

v1.1.2: January 11, 2002

Added Node Scan Columns "Show Columns..." window. Optimized and made more fault tolerant, because of particular behaviors in Open Transport, the initial connection algorithms in the launching mechanism for greater speed and reliability when launching onto large numbers of nodes. Reorganized and rearranged the Job Window and Node Info Window for almost complete compliance with Apple's Aqua user interface

guidelines. Made the job window horizontally resizable. Added recognition of the PowerPC 7450. Under OS X, made the "Remote Node Scan...", "Open Apps and Files...", "Place in Subdirectory...", and "Start Job at Time..." dialogs appear as sheets. Modified the connection algorithms in the Node Scan Window to attempt access to all found nodes once before giving other nodes a second chance to connect. Modified timeouts and other timings for the Node Scan Window, Node Info Window, and the launch process. Greater use of DrawStringUsingThemeTextBox, with the correct font and size selections, in the Node Info Window, the Job Window, the Node Scan Window, the Get Files window, and the About Box for smoother text under OS X. Added Always Include this Node in Local Scan setting with corresponding implementation. Added Accept Commands from Local Subnet Only setting with corresponding implementation. Gave the launch process code and network scan engine code greater encapsulation. Instructed Pooch to wake up the Macs upon successful launch on all nodes. Modified to recognize window classes correctly when determining the frontmost window. Corrected certain behaviors in the launch progress bar. Corrected a cosmetic error when the network scan code reports its registration. Corrected internal network structure allocation code for the case when a part of Pooch requests to connect to more than 64 other Pooches. Correctly notifies the system of network activity when Pooch receives commands. Acquires the correct IP address when multiple network interfaces are active.

v1.1: October 2, 2001

Fully operational on OS 9.x with CarbonLib 1.2.x and today's OS X 10.1. AppleScript interface implemented and supported, making customized parallel launches, customized job queues, Unix command-line calls to Pooch, and fully automated parallel application launching possible.

Worked with Apple to fix NSL, Open Transport, and CoreServices bugs in 10.0.x, leading to the fixes in 10.1 release. On X, now uses the Unix "ps" command to determine load distribution on a node. On X 10.1, uses CSCopyMachineName to determine the

computer name. Heuristic node rating added and exposed to the Node Scan Window and AppleScript interfaces. Add Best... pop-up replaces Add All button, leading to alteration of corresponding selection list calculation routines. Revised Node Scan Window code so that, when automatic rescan occurs, the scan results no longer disappear and reappear when being updated. Nodes selected for launch in the Job Window now appear in the Node Scan Window in gray, rather than disappearing. Revised Network Neighborhood Node Scan pop-up menu and behavior. Added the ability to recognize and launch Mach-O, Unix-based executables. Altered Open Transport endpoint settings and optimized background time calculation and internal task code to improve network performance, file transfer speed, and launch speed. When remembering the last job and node 0 was itself, updates the node 0 IP address if it has changed. Added Verbose Error Reporting setting to reduce "nuisance calls". Added the job option to skip the launch of node zero. Internal window code structure reorganized and factored to a greater degree. In X, ATSUI text methods now used for custom content Data Browser columns to match font type and size of the other columns.

v1.0.2: May 2, 2001

Fully operational on OS 9.x with CarbonLib 1.2.x and today's OS X.

Prevented a possible circumstance where Pooch's connector endpoint would continuously reject an incoming password. Added code to the Set Pooch in Login Items to operate correctly when there is no loginwindow.plist file. Fixed a cosmetic bug in the Job Launch status window where the text could overlap. Fixed a cosmetic bug so that the node list font looks the same as the file list font. Corrected a Launch Job button behavior. The DNS translation routines correctly report a dotted quad.

Remaining known issue on X: The problem of the Process manager returning zero is still present.

v1.0.1: April 25, 2001

Fully operational on OS 9.x with CarbonLib 1.2.x. This Pooch contains workarounds for known bugs in OS X, so this Pooch should be fully operational with today's X.

Known issues in OS X 10.0.1:

- Pooch uses the Process manager to monitor the CPU time taken by other apps and estimate load. However, Apple's documentation says the Process Manager on X returns zero CPU time for all processes, making these calculations impossible directly in Carbon. A workaround through Unix may eventually be devised.
- During the search for nodes, NSLM does not respect maxSearchTime parameter, resulting in infinite searches. The current workaround is to set maxSearchTime to zero and use NSLCancelRequest.
- The official way to access the user-specified computer name entered in the Sharing system preference, CSCopyMachineName, returns the wrong name. Another way is being used.
- Setting Pooch in Login Items automatically. There is no documented way to do that, so I edit the loginwindow.plist file myself.
- A few cosmetic errors remain, such as custom content Data Browser columns not quite matching the font size of standard text Data Browser items.

v1.0: April 17, 2001

First public release - Fully functional on OS 9.x with CarbonLib 1.2.x. Due to known bugs in OS X, Pooch on OS X cannot be officially supported. Over 23,000 lines of code in C.

Known issues in OS X Release Candidate:

- During search for nodes, NSLM does not respect maxSearchTime parameter, resulting in infinite searches. Attempts to cancel the search result in a hang. However, this problem does not restrict launching a job onto OS X machines from an OS 9 machine.

- There is no reliable way to access the user-specified computer name entered in the Sharing system preference. The official way, `CSCopyMachineName`, returns the wrong name.
- An assortment of cosmetic errors, such as Aqua buttons not being compatible with texture backgrounds.

XI. Credits

Pooch Concept, Code, Engineering, and User-Interface Design

Dean E. Dauger, Ph. D.

Pooch Icon Art

Jean-Marie Venturini

Advice and Brainstorming

Viktor K. Decyk, Ph. D., Alan B. Dauger, Ph. D.

Kevin T. Sinclair, EE & MBA, Robert M. Zirpoli, III, M. E., Catherine C. Venturini, M. S.

Beta Testing

Viktor K. Decyk, Ph. D., Frank S. Tsung, Ph. D., John W. Tonge, Ph. D.

Compiler

Metrowerks CodeWarrior Pro 6

We also wish to thank employees of Apple Computer for their assistance, technical and otherwise, who helped demystify the future of the Mac OS and saved a lot of headaches:

Tim Parker, Warner Yuen, Robert Kehrer,
John Tucker, Kevin Arnold, Steve Zimmer, Ernest Prabhakar, Ph. D.

XII. Contact Information

Pooch is released by Dauger Research, Inc. The latest version of this document is available from the Pooch web site. We are open to suggestions and wish lists regarding parallel computing issues. Depending on your needs and the nature of the request, custom configurations of Pooch can be arranged.

Dauger Research, Inc.	http://daugerresearch.com/
Pooch web site	http://daugerresearch.com/pooch/
The Dauger Research Vault, providing tutorials, sample code, and other documentation	http://daugerresearch.com/vault/
Pooch technical support, questions, and feedback	pooch@daugerresearch.com
Dauger Research mailing list info	http://daugerresearch.com/ mailing-list/

The latest MacMPI libraries are available from the AppleSeed web site.

AppleSeed	http://exodus.physics.ucla.edu/appleseed/
UCLA Plasma Physics Group	http://exodus.physics.ucla.edu/

Pooch and this document are Copyright © 2001-11 Dauger Research, Inc.
Macintosh, Mac OS, and Mac OS X are trademarks of Apple Computer, Inc.

Last Update: August 21, 2011