EFFICIENT CLUSTER MANAGEMENT USING THE OPEN SOURCE CLUSTER APPLICATION RESOURCE

BY

DEEPAK PRASANNA

B.S., University of Illinois Urbana-Champaign, 2002

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2004

Urbana, Illinois

TABLE OF CONTENTS

1. INTRODUCTION

A cluster is a set of independent multiple computers connected together through software and networking. The most basic definition is when two or more computers are used together to solve a problem. Clusters are typically used for high availability (HA) for greater reliability or high-performance computing (HPC) to provide greater computational power than a single computer can provide [1].

Beowulf clusters are scalable performance clusters based on consumer hardware, on a private network, with an open source software (i.e., Linux) infrastructure. The builder can improve performance proportionally by adding machines. The commodity hardware can be as simple as two networked computers each running Linux and sharing a file system or as complex as 512 nodes in a high-speed, low-latency network.

A Beowulf cluster is best suited for parallel tasks. These are tasks that require very little communication between nodes to complete, so that adding more nodes results in a near-linear increase in computation with little extra synchronization work. These types of parallel programs are also known as linearly scalable [2].

Genetic algorithms are one example of an application of parallel computation, since each added node can simply act as another trial to evaluate the fitness of a population. Rendering an animation sequence is another application because each node can be given an individual image, or even sections of an image to render quite easily even using standard programs like the Persistence of Vision Raytracer (POV-Ray) [3].

1.1. Existing Methods of Cluster Management

Current methods of cluster building and setup are tedious and time consuming. Although there are many books available, only a skilled Linux system administrator would find it trivial to build a cluster. Each cluster is an experiment, and cluster builders have to download, compile, buy, and test a large variety of software and hardware components. Innovations in technology in both hardware and software also happen at a very rapid pace, and current cluster-building methods cannot adjust to this pace.

The most common method of cluster building is doing it manually. This method involves the most amount of time since there is virtually little or no automation in the process. There are many commercial and free software tools available for clustering. Commercial tools such as ClusterWorx and Cluster Systems Management are good, but they are expensive. Open source tools such as MOSIX and Blacklab are free, but they lack the reliability and support necessary for good cluster management. To balance this need for reliability and support at no cost, the Open Cluster Group (OCG) created the Open Source Cluster Application Resource (OSCAR) in April 2000 [4].

1.2. Overview of Thesis

The remainder of this paper will discuss the setup of an OSCAR managed cluster. Chapter 2 discusses some common hardware features in Beowulf clusters as well as the test cluster that OSCAR was implemented on. Chapter 3 discusses the actual installation of OSCAR. Chapter 4 discusses the performance enhancements of Local Area Multiplier/Message Passing Interface (LAM/MPI) 7.0, and Chapter 5 discusses future areas for cluster improvements.

2. CLUSTER HARDWARE

2.1. Overview

As stated previously, Beowulf clusters do not have to be made with expensive high-end hardware. Typically, each computer is made up with the bare minimum where the CPU and RAM contribute the most to the cost. Other essential items for each computer are a video card, a hard drive, and a network card. Most modern motherboards now offer the video and network cards integrated onto the actual board itself, further reducing the cost.

Another important component is the network switch. A switch is preferred over an Ethernet hub or router because it has the least amount of packet overhead. Network bandwidth can be a bottleneck in hub-based clusters due to collisions. Switches have also become relatively inexpensive. A 100-Mb based switch is sufficient, but 1000-Mb-based switches are also available.

A keyboard/video/mouse switch box is useful during the installation process, but it is not necessary. It also adds significant cost to the cluster because additional cabling between all the computers and the switch are required.

2.2. EMtel Hardware

The EMtel cluster, which OSCAR was tested on, consists of 16 computers. Each of these computers is a Pentium 4 2.0 GHz machine with 256-MB RAM, a 10-GB hard disk, an ATI Raedon 7000 video adapter, and a CD-ROM. The motherboard is an Intel Desktop 850 with a built-in 10/100 BaseT chipset. The switch is from D-link with 24-ports each capable of a 100-Mb connection.

3. OSCAR

3.1. Project History

The OCG decided to optimize OSCAR for HPC clusters. The open-source solutions for each component in the proposed OSCAR system were identified and work began on creating a unified package. The primary objective of the OCG was to make the installation, configuration and management of modest sized clusters easier [4].

OSCAR was the first project started by the OCG. This group is an informal group primarily made up of commercial and government organizations as well as universities. While membership to the OCG is open to anyone, it is directed by a steering committee with committee positions up for election every two years [4]. Currently, the steering committee is made up of representatives from IBM, Indiana University, Intel, MSC.Software, the National Center for Supercomputing Applications (NCSA), and Oak Ridge National Laboratory (ORNL) [5].

3.2. OSCAR Terminology

Each computer in a cluster is referred to as a *node*. In an OSCAR cluster, there are two types of nodes, a *server* and a *client*. A server node handles all of the requests of the client nodes. It may also be referred to as the *master node*. A client node only performs computations. It may also be referred to as a *slave node*.

An OSCAR cluster is made up of one server node and one or more client nodes, where all the client nodes must have the same hardware configuration (i.e., same hardware components).

An OSCAR package is a set of files that are installed on the cluster. Most packages in OSCAR and in Red-Hat are stored as Red-Hat Package Manager (RPM) files. An OSCAR package can be as simple as a single RPM file, or it can be more complex, including a mixture of

RPM and other configuration installation scripts. The OSCAR packages provide the most of the features in OSCAR clusters.

OSCAR packages can be divided into three categories: core packages, included packages, and third party packages [6]. Core packages are mandatory for installing and using OSCAR. Included packages are provided in the official OSCAR distribution. These are created by OSCAR developers. Third party packages are not included in the official OSCAR distribution. They can be added during the installation process.

3.3. OSCAR Installation

3.3.1 Supported distributions

Red-Hat 8.0, 9.0, and Mandrake 9.0 are all fully supported by OSCAR 3.0. There are other Linux distributions that could work with OSCAR, but they have not been fully tested.

3.3.2 Minimum system requirements

Tables 3.1 and 3.2 list the minimum system requirements of the server and client nodes, respectively. They are primarily based on the system requirements of Red-Hat 9.0.

Table 3.1 - Minimum System Requirements for the OSCAR Server Node

CPU of i586 or above
A network interface card that supports a TCP/IP stack
A second network interface card that will act as a router to a public network
At least 4 GB total free space – 2 GB under / and 2 GB under /var
An installed version of Linux, preferably a fully supported distribution

Table 3.2- Minimum System Requirements for the OSCAR Client Nodes

CPU of i586 or above
A disk on each client node, at least 2 GB in size (OSCAR will format the disks during the installation)
A network interface card that supports a TCP/IP stack
Same Linux distribution and version as the server node
All clients must have the same architecture (i.e., ia32 vs. ia64)
PXE enabled BIOS

3.3.3 Downloading OSCAR distribution packages

OSCAR 3.0 can be downloaded at *http://oscar.sourceforge.net/*. There is a choice of three different kinds of distributions to download. The *Regular* distribution contains all of the OSCAR packages required to install and operate an OSCAR cluster. This is the typical installation distribution for most users. The *Extra Crispy* distribution also includes the source code packages in addition to the *Regular* distribution packages. The *Secret Sauce* distribution only contains the source code packages of OSCAR.

3.3.4 Installing Linux on the server node

This paper assumes that the user knows how to clean install a Linux operating system and that the user has root access to the server node.

Red-Hat 9.0 is the preferred operating system to install for OSCAR and will be referred to for the remainder of this paper. A full install of the operating system is recommended. A full install ensures that any dependencies for packages that OSCAR requires are resolved. The main Linux installation requirement is that some X windowing environment such as GNOME or KDE must be installed. It is best to not install operating system updates after installing Red-Hat. This may disrupt some of OSCAR's RPM dependencies. Instead, install OSCAR first, and then install the distribution updates.

3.4. Setting up the OSCAR Installation Wizard

3.4.1 Disk space requirements

OSCAR has certain requirements for server disk space. Space will be needed to store the Linux RPMs and to store the images. The RPMs will be stored in /tftpboot/rpm. Two Gigabytes is usually enough to store the RPMs. The images are stored in /var/lib/systemimager and will need approximately 2 GB per image. Although only one image is required for OSCAR, more

images can be created in the future.

When installing a new server, it is recommended that 4 GB are allocated in both the root (/) directory (which contains /tftpboot) and /var filesystems when partitioning the disk on the server node.

The amount of free space on the drive's partitions can be checked by issuing the command `df –h` in a terminal. The result for each file system is located below the available column heading.

If the root (/) partition has enough free space, enter the following command in a terminal: `mkdir -p /tftpboot/rpm`. If the root partition does not have enough free space, create the directories on a different partition that does have enough free space and create symbolic links to them from the root directory. For example, if the partition containing /usr contains enough space, use the following commands, `mkdir -p /usr/tftpboot/rpm` and then `ln -s /usr/tftpboot /tftpboot`. The same procedure should be repeated for the /var/lib/systemimager subdirectory.

3.4.2 Unpacking the OSCAR package

Unpack the OSCAR package in the /root directory. Do not unpack it on a windows based machine and then copy it to Linux. This action will cause the files to be unreadable. The `tar zxf` command is used to uncompress and unpack the compressed package file. After this action is done, all of the installation files will be placed in the directory oscar-3.0, which will be located within the current directory when the tar command was executed. For example, if /root was the current where the OSCAR package was unpacked, then all of the OSCAR install files will be located in /root/oscar-3.0.

3.4.3 Configure the OSCAR installation script

A configuration script needs to be run in order to create all of the directories that OSCAR will finally use. This configure script is located in the oscar-3.0 directory. The command `./configure` will run the script from that directory.

Once the configure script successfully completes, OSCAR is ready to be installed on the server. No changes have been made to the system beyond the oscar-3.0 directory itself. Running `make install` from the terminal will copy the files to the system. The files copied will include the base OSCAR files as well as startup scripts for the profile.d/ area. The startup scripts add OSCAR to the path and set environment variables like `$OSCAR_HOME`.

After the install command is completed, the rest of the OSCAR setup can be done from /opt/oscar, the default directory of the OSCAR installation.

3.4.4 Configuring the Ethernet adapter for the cluster

Assuming that the server node is to be connected to both a public network and the private cluster subnet, two Ethernet adapters are required to be installed on the server. This is the preferred OSCAR configuration because exposing the entire cluster (all of the client nodes) could be a security risk. The server acts as a firewall, and is the only node that requires a public IP address on the public network. Also, certain software used in OSCAR (such as DHCP) may conflict with the external public network when tries to find the client nodes.

Once both adapters have been physically installed on the server node, they need to be configured. Any network configuration is sufficient; popular applications include neat, netcfg, or manually editing the configuration files in a text editor.

There are some requirements that need to be satisfied before continuing with the installation. Most Linux distributions default to the *hostname* localhost (or

localhost.localdomain). The hostname needs to be changed from this default name. The *public adapter* is the adapter that connects the server node to a public network. Configure this node according to the guidelines of the public network that the cluster is connected to. The *private adapter* is the adapter connected to the TCP/IP network with the rest of the cluster nodes. This adapter needs to use a private IP address, an appropriate subnet netmask, and must be activated at boot time. There are three private IP address ranges: 10.0.0.0 to 10.255.255.255; 172.16.0.0 to 172.32.255.255; and 192.168.0.0 to 192.168.255.255. Additional information on private intranets is available in [7]. Do not use the IP addresses 10.0.0.0 or 172.16.0.0 or 192.168.0.0 for the server. One of these addresses will cause the network installs of the client nodes to fail.

Reboot the server node to make sure that all if the changes have been saved. To confirm that all Ethernet adapters are in the *UP* state, enter the command `ifconfig –a` from a terminal. For this thesis, the private adapter is referred to as eth0, and the public adapter is referred to as eth1.

3.4.5 Copy Linux distribution RPMs to /tftpboot/rpm

These packages are the RPMs included with Red-Hat 9.0 on the Red-Hat installation CDs. They need to be copied into the /tftpboot/rpm directory. When each CD is inserted, Linux automatically makes the contents of the CD be available in the /mnt/cdrom directory. For each CD, locate the directory that contains the RPMs. In Red-Hat 9.0, the RPMs are located in the RedHat/RPMS directory. Copy the RPMS into the /tftpboot/rpm directory with a command. This process should be repeated for all of the installation CDs.

3.5. Running the OSCAR Installation Wizard

Change directory to the top-level OSCAR directory and start the OSCAR install wizard by typing `./install_cluster <device>`, where `<device>` should be replaced with the

private Ethernet adapter (eth0) configured previously. This script installs prerequisite packages on the server, copies OSCAR RPMs to /tftpboot/rpm, installs all OSCAR server RPMs, updates /etc/hosts with OSCAR aliases, updates /etc/exports, adds OSCAR paths to /etc/profile, and updates system startup (/etc/rc.d/init.d) scripts restarts affected services.

When the script is completed, the OSCAR graphical user interface (GUI) will be launched. The wizard, as shown in Figure 3.1, will continue through the rest of the OSCAR installation. To use the wizard, the user completes a series of steps, each step being started by pressing one of the buttons on the wizard. The steps must be completed sequentially. For each step, there is also a *Help* button located directly to the right of the step button. When pressed, the Help button displays a message box describing the purpose of the step.

Figure 3.1 – OSCAR Installation Wizard

3.5.1 Step 0: Downloading additional OSCAR packages

This step is optional. Extra packages not required for the regular OSCAR installation can be downloaded via the OSCAR Package Downloader (OPD) tool. This tool automatically inserts these packages into the OSCAR hierarchy. Some additional packages are Ganglia and Globus. Figure 3.2 shows the OPD GUI.

3.5.2 Step 1: Select packages to install

This step is only necessary if the user wants to change the default OSCAR packages to be installed. Again, this is not necessary for the typical OSCAR installation. However, if the default package selection should be changed, then select this option. This step is necessary if any additional packages were downloaded via the OPD.

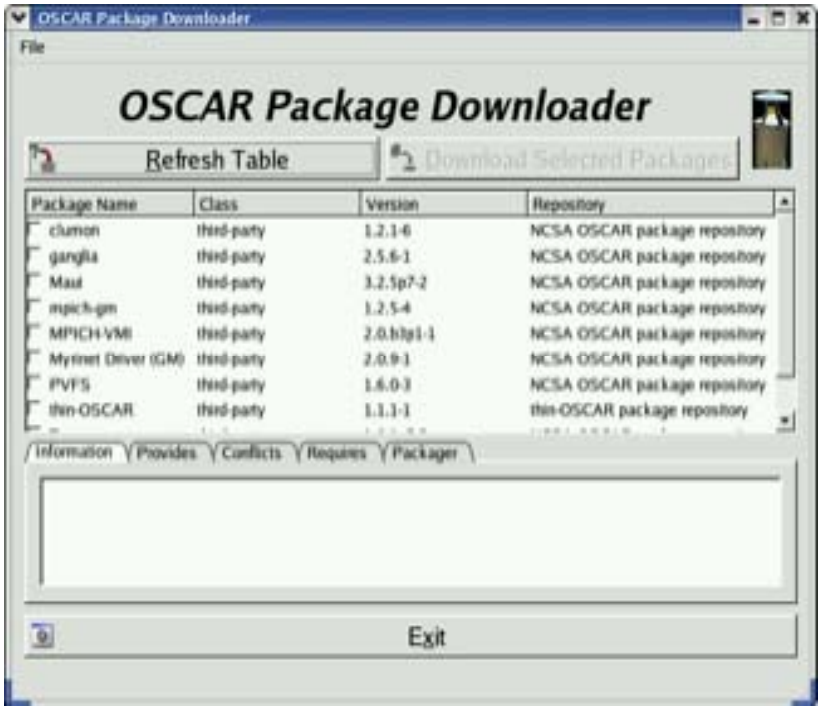

Figure 3.2 – OPD Tool

When the button is clicked, a window such as that in Figure 3.3 should appear. Each of the packages that the OSCAR installer has found are listed in the main frame. Core packages must be installed and cannot be unselected. Included packages can be unselected if desired. One package that should not be installed is the Pfilter package. This package filters network traffic so that it only allows ports that SSH and MPI uses to communicate the client nodes. It is not necessary and in fact reduces performance since additional rules have to be checked for incoming and outgoing packets.

This window only shows OSCAR packages. It does not show individual RPMs. Once a set of OSCAR packages has been selected to install, click on the *Exit* button to save the selections and return to the main OSCAR window. Closing the window yields the same result as clicking Exit and there is no way of defaulting to the original settings, so the package list should be finalized before proceeding to the next step.

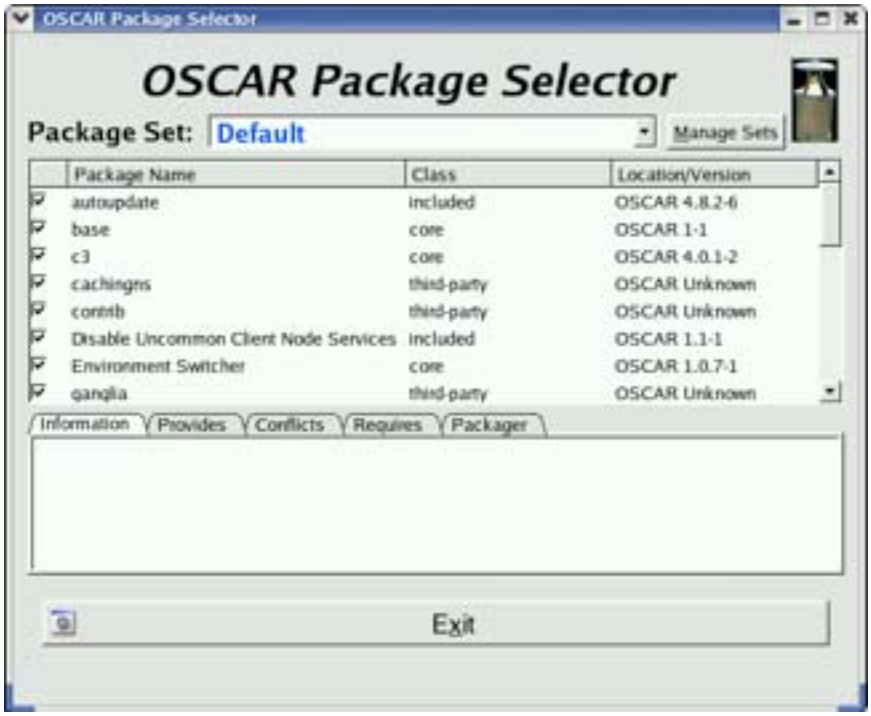

Figure 3.3 – The OSCAR Package Selector

3.5.3 Step 2: Configuring OSCAR packages

Some OSCAR packages have configuration options. Clicking on the *Configure Selected OSCAR Packages* button will bring up a window listing all the packages that can be configured. Figure 3.4 shows a sample of the configuration window. Clicking on any of the packages' *Config…* button will bring up a panel for configuring that package. Select whatever options are appropriate for that package, and then click on the *Save* button to save the selections, or the *Cancel* button to cancel all of the selections and leave the original settings. To revert any changes that were made, simply click on the *Default Configuration* button and then the Save button. This step is optional. If this step is skipped, then the default configuration values will be used. The default MPI package is LAM/MPI.

Figure 3.4 – Configuration GUI

3.5.4 Step 3: Install OSCAR server packages

Pressing the *Install OSCAR Server Packages* button will install various RPMs and configurations on the server node. This step may take several minutes. Status messages will appear in a shell window. A popup will appear as shown in Figure 3.5 indicating the success or failure of this step. Click on the *Close* button to dismiss it.

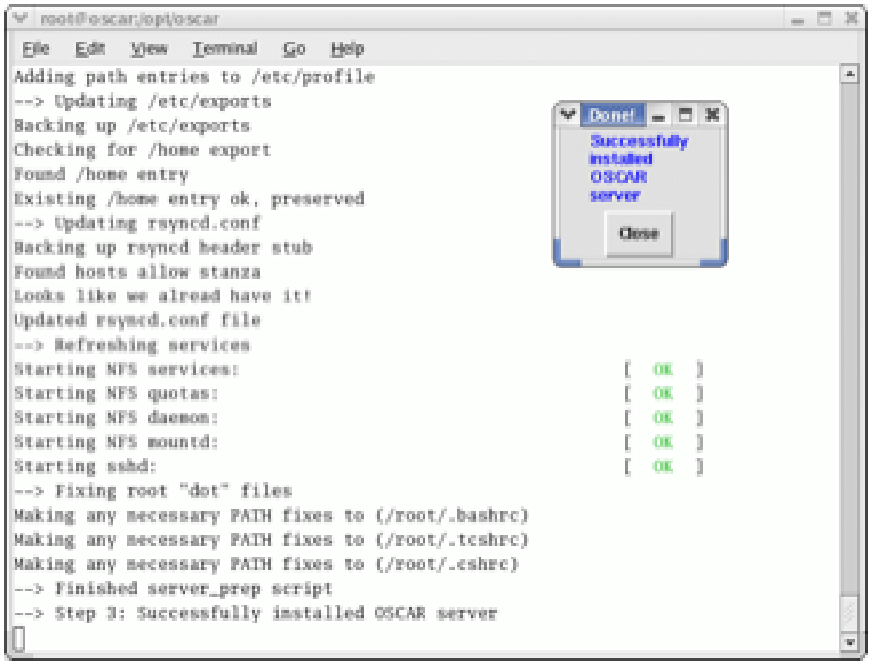

Figure 3.5 – Popup Showing Status of Installation

3.5.5 Step 4: Build OSCAR client image

This step will build the image that will be installed on all of the client nodes. Certain precautions should be taken before actually pressing the *Build OSCAR client image* button. Ensure that the SSH daemon's configuration file (/etc/ssh/sshd_config) on the server node has the `PermitRootLogin` variable set to *yes*. After the OSCAR installation, this variable can be set back to *no*, but it needs to be yes during the install because the configuration file is copied to the client nodes. Root must be able to login to the client nodes remotely. Also, ensure that TCP wrapper settings are not too strict. The /etc/hosts.allow and /etc/hosts.deny files should allow all traffic from the entire private subnet. Figure 3.6 shows the GUI of the build image window. All of the default values should work properly. When all of the above criteria are met, click the *Build Image* button.

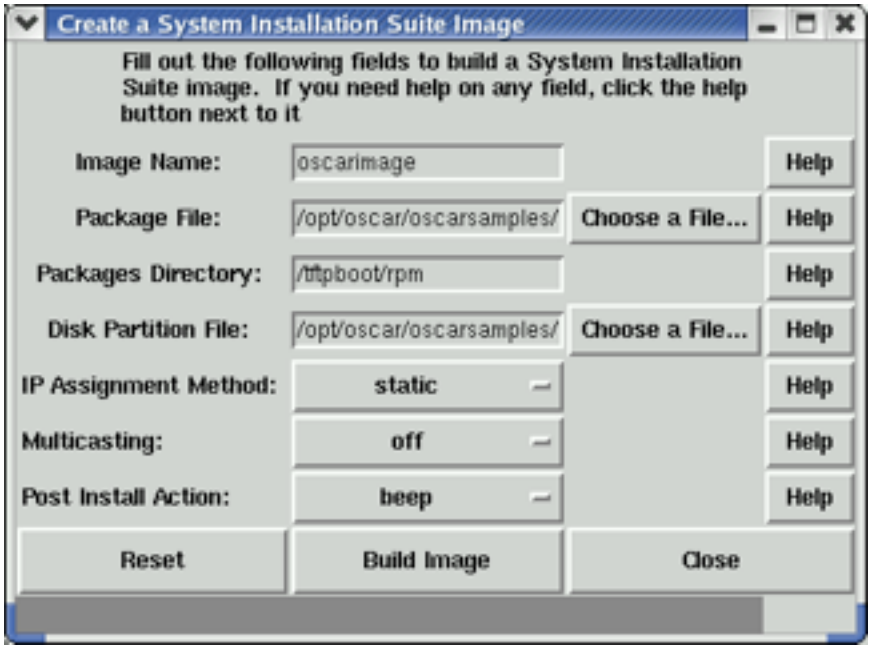

Figure 3.6 – GUI for building a client image

The image name is the image about to be built. It is only necessary to modify this if multiple

boot images are going to be created. The package list is simply a list of RPM file names (one per line). The default file contains all of the required packages that the client node will need. The disk partition file contains information on how the image will structure the client hard drives. It is not necessary to modify these files.

The post install action field tells the client what to do after the image is done copying to a client node. The options are do nothing, beep, or reboot. If automatic reboot is used, make sure that the basic input output system (BIOS) on each client is set to boot from the local hard drive before attempting a network boot. If a client's boot order has to be changed to do a network boot before a disk boot to do an install, do not use automatic reboot. Building the image may take several minutes; the red progress bar on the bottom of the window will show the status of the image creation.

There will be a lot of output in the shell window during the image creation. It is normal to see some warning messages. These messages can be ignored. A success or failure popup window will appear upon completion of the build.

3.5.6 Step 5: Defining the OSCAR clients

In this step, information about the clients is entered. This information is used by the server node to communicate with the clients. Figure 3.7 shows the OSCAR client window. The image name is the name of the image that was created in the previous step. If multiple images were created, select the appropriate image to use. The domain name should contain the server's domain name if it has one. If not, the default domain or any other domain name will suffice. This entry should not be left blank. The base name field is used to specify the first part of the client name and hostname. It will have an index appended to the end of it. This name cannot contain an underscore character.

The number of hosts field specifies how many nodes to create. If the cluster has 16 nodes, then this value should be 15 (excluding the server node). This value has to be greater than zero. The starting number field is number appended to the first client to be built. It will be incremented for each subsequent client. The padding field adds zeros to the client number based on the value specified. If the value is one, then the client node name might look like `oscarnode01`.

Figure 3.7 – OSCAR Client GUI

The starting IP specifies the IP address of the first client. It will be incremented for each subsequent client. This IP address will be incremented for subsequent clients. Make sure that this value is not the same is the private IP of the server node. The subnet mask specifies the netmask for the cluster. If the cluster has less than 254 nodes, then the netmask value should be 255.255.255.0. The default gateway value should be the private IP address of the server node since all packets should be routed through it [6].

The *AddClients* button will create the necessary information to build all the clients. Once this step is completed, press the Close button.

3.5.7 Step 6: Set up networking

The clients will be identified by their MAC addresses. The MAC addresses will verify that the correct IP information goes to the correct client node. The DHCP protocol will use the MAC addresses to assign the appropriate IP address. Therefore, the OSCAR installation wizard must know the MAC addresses of each of the client nodes. To collect the MAC addresses, press the *Setup Networking* button. The OSCAR network utility dialog box will be displayed. Before proceeding further, make sure that the client nodes are setup to network boot as specified earlier.

The *Collect MAC Address* button will set the server to listen for MAC addresses being broadcast as each client node boots up. The MAC addresses appearing in the left hand column of Figure 3.8 are of each client as they boot up.

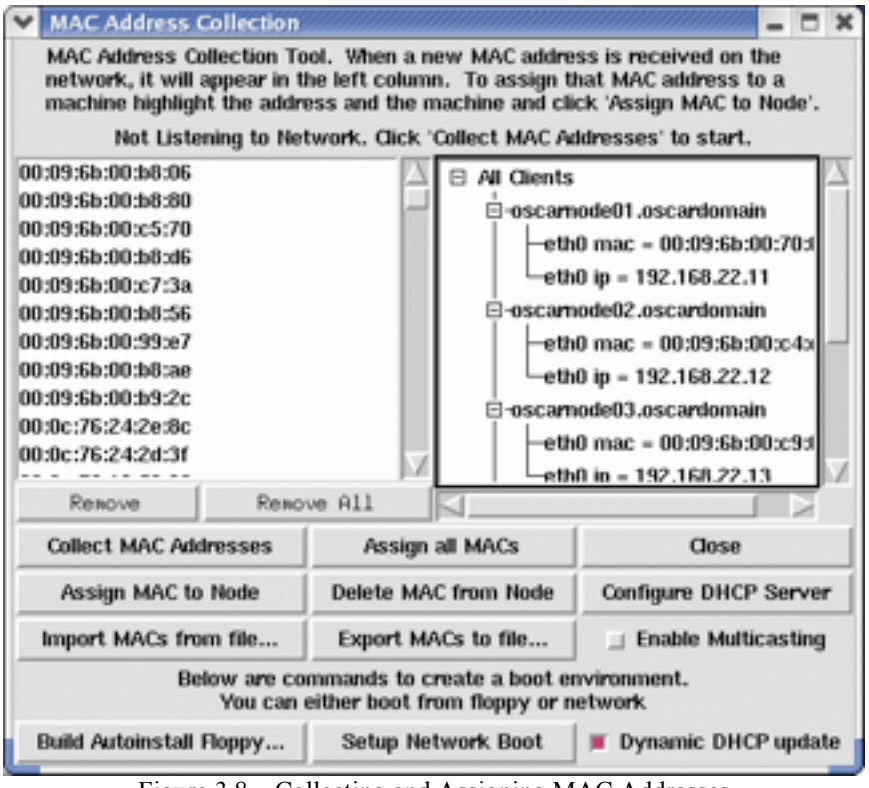

Figure 3.8 – Collecting and Assigning MAC Addresses

The clients should be boot up in sequential order as they are arranged physically. Then the *Assign all MACs* button can be used to assign the respective IP addresses in the order that the

clients are arranged. The *Dynamic DHCP update* box should be enabled. Once all of the MAC addresses have been collected, press the *Stop Collecting MACs* button. Before closing the MAC Address Collection window, press the *Setup Network Boot* button. This command will tell the server to accept preboot execution environment (PXE) requests from each client node. Network booting can also be done via floppy disk. The *Build Autoinstall Floppy* button will create a floppy disk that will enable a network boot. In this case, the BIOS in each client should be set to boot from a floppy disk first. This paper uses the PXE network boot option. Once the network setup parameters have been established, press the *Close* button.

3.5.8 Client installations

The server node will install the image created in step four to each of the clients. Simply boot up each client to start the transfer. Again, Each client should be set to network boot after trying the hard disk in the BIOS. The network install can be done on one or more clients at a time. If more clients are network booting simultaneously, then the overall client install process will be slower. When each client completes the network boot, the client will halt, beep, or reboot based on the post install action value given in step four. Each client must be rebooted either automatically or manually before completing the cluster setup.

3.5.9 Step 7: Complete cluster setup

This step will run the final installation configurations scripts from each OSCAR software package, and perform various cleanup and re-initialization functions. A popup window will indicate the success or failure of this step as shown in Figure 3.9. Press the *Close* button to dismiss it.

3.5.10 Step 8: Test cluster setup

A simplistic test suite is provided in OSCAR to ensure that the key cluster components

such as MPI, Open Secure Shell (OpenSSH), Portable Batch System (PBS) are functioning properly. Press the *Test Cluster Setup* button. This will open a separate shell window to run the tests in. The cluster's basic services are checked and then a set of root and user level tests is run. A sample dialog is shown in Figure 3.10.

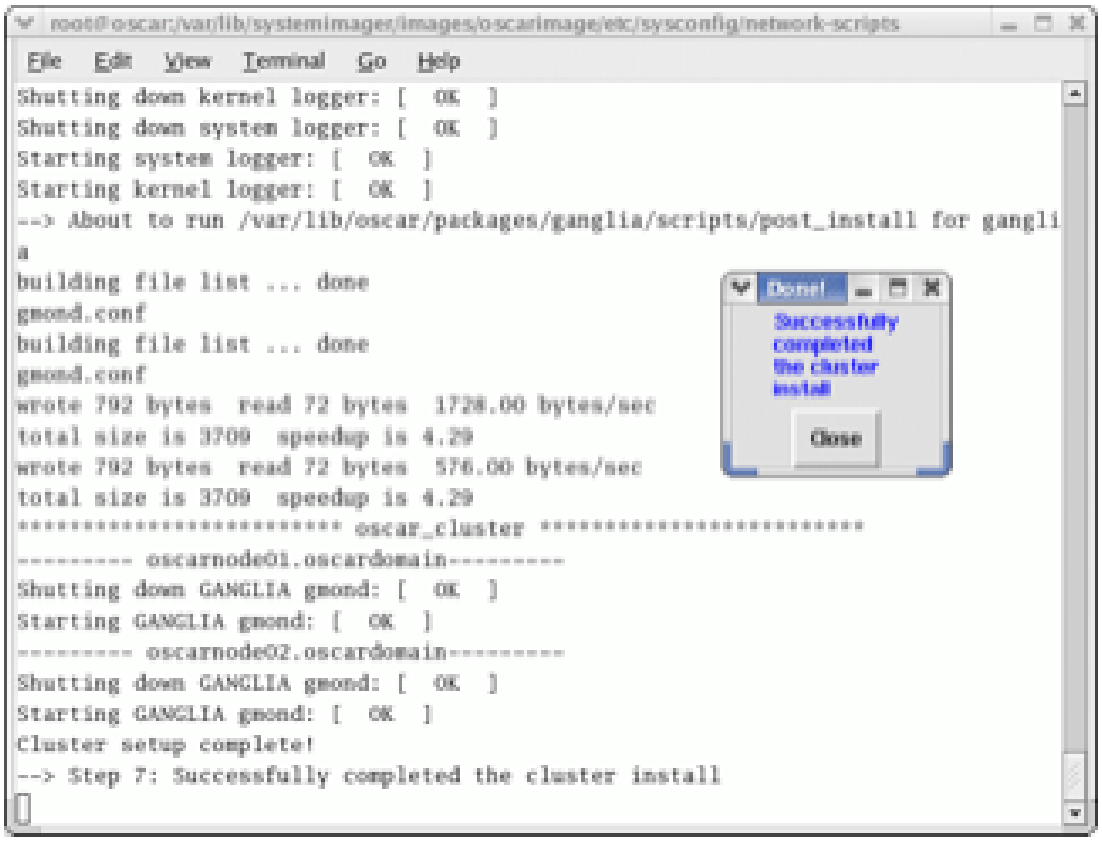

Figure 3.9 – Completing Cluster Shell Output

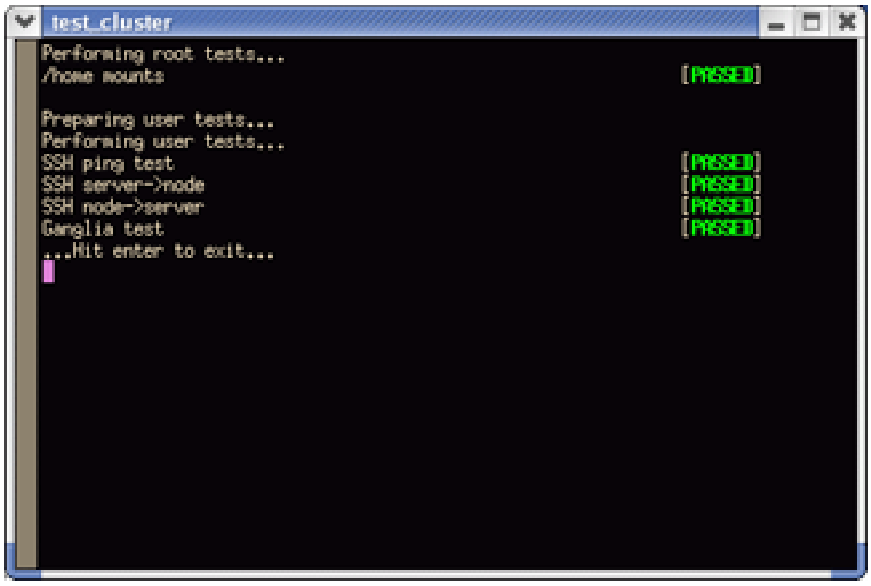

Figure 3.10 – Sample Test Window

If all the tests passed, then the cluster is officially setup and ready to use. A more advanced test suite can be found at [8].

3.5.11 Adding/Deleting client nodes

Client nodes can be added or deleted using the OSCAR Installation Wizard. Figure 3.11 shows the GUI for adding additional nodes. The process is exactly the same as if adding the node during the initial cluster setup. The fields from Figure 3.8 (Section 3.5.7) that will typically change are the number of hosts, starting number and IP address. The number of hosts should reflect the number of new client nodes being added. The starting number should start after highest numbered client in the existing cluster. The same is true for the starting IP address.

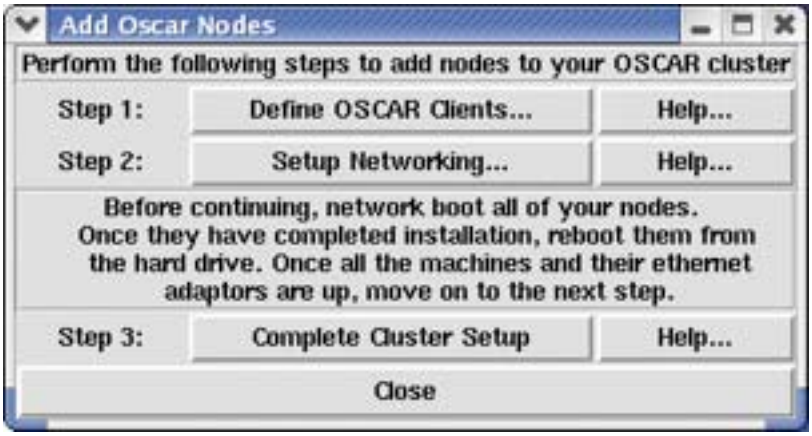

Figure 3.11 – Adding additional nodes

Deleting a node simply removes the node information from the server node, and then redistributes the new host file to all of the remaining client nodes. After a node has been deleted, it can safely be disconnected from the cluster network. Figure 3.12 shows a sample list of nodes to delete when the *Delete OSCAR Clients* is pressed.

3.5.12 Adding user accounts

User accounts can be added on the server node using any traditional Linux method. Red-Hat 9.0 provides an easy tool to create accounts in X11. This tool can be accessed by typing `redhat-config-users` in a terminal in X11.

Figure 3.12 – Deleting a Node

3.6. OSCAR Features

3.6.1 OpenSSH

OSCAR uses the secure shell (SSH) protocol for communication between the server and client nodes. SSH uses 128-bit Rivest-Shamir-Adleman (RSA) encryption for password authentication [9]. Legacy communication protocols such as telnet, rsync, and ftp are strongly discouraged and disabled during the OSCAR installation.

3.6.2 OSCAR Password Installer and User Management (OPIUM)

OPIUM is a useful OSCAR tool that synchronizes accounts and configures SSH on the client nodes. OPIUM is periodically run by root on the server node to make sure that the `passwd` and `shadow` files are up to date on client nodes. It also manages the SSH public key authentication on the client nodes when a user is logged on to the server node [10].

3.6.3 System Installation Suite (SIS)

SIS is a tool for installing Linux systems over a network. It is used in OSCAR to install images on client nodes. SIS also manages the database that OSCAR obtains its cluster configuration information. It is an image-based tool. The image created in step 4 of the OSCAR install process is exactly this image. It can also be used to update clients when software updates are done on the server node. More information and documentation can be found at [11].

3.6.4 Cluster Command Cluster Tools (C3)

The Cluster Command Control (C3) tools are a set of cluster tools developed at ORNL that are useful for both administration and application support. The suite includes tools for command execution over the entire cluster, file distribution and gathering, process termination, remote shutdown and restart, and system image updates [12].

4. LOCAL AREA MULTIPLIER/MESSAGE PASSING INTERFACE (LAM/MPI) 7.0

4.1. Overview

MPI is a set of functions and libraries enabling programmers to write parallel programs that pass messages between processes to make up an overall parallel job. MPI is suitable for

supercomputing parallel machines such as the IBM SP, SGI Origin, etc., but it also works well on Beowulf clusters. The MPI standard was designed to support portability and platform independence. As a result, users can achieve cross-platform development [13].

LAM/MPI is a high-performance and open source implementation of the MPI standard that is maintained at the Open Systems Lab at Indiana University. LAM/MPI supports MPI-1 and MPI-2 standards. More information about LAM/MPI, including all the source code and documentation, is available at [14]. Version 7.0 supports C, C++, and Fortran.

LAM/MPI is also a library that implements the standard the LAM run-time environment. LAM is a user-level, daemon-based run-time environment that provides many of the system level services required by MPI programs. Both major components of the LAM/MPI package are designed as component frameworks with small modules that are selectable (and configurable) at run-time. This component framework is known as the System Services Interface (SSI) [15].

4.2. System Services Interface (SSI)

The System Services Interface (SSI) makes up the core of LAM/MPI. It influences how many commands and MPI processes are executed. Run-time decisions can be made about which interface instance to use, and to enable passing of tunable parameters to target instances. LAM may export one or more instances of any given system interface. At run time, the decision is made as to which instance will be selected. Optionally, parameters may also be passed to the interface instances at run time.

The overall intent for SSI is to allow a modular plug-n-play approach to the system services that LAM and MPI must use. Specifically, each component instance in is a self-contained set of source code. It has its own directory structure and can configure, build, and install itself. The LAM frameworks will automatically detect each instance and invoke the

corresponding hooks during the outer-level configuration, building, and installation phases.

This type of architecture will make it easy to add drop-in modules to the LAM framework and have them be seamlessly integrated into the run-time environment of LAM/MPI. From a source code perspective, this architecture creates a logical abstraction barrier between each instance and the rest of LAM. Not only does such a modular approach make is easier to add new interface instances; it can also facilitate development of interface instances by third parties [16].

SSI provides a component framework for the LAM run-time environment (RTE) and the MPI communications layer. Components are selected from each type at run-time and used to affect the LAM RTE and MPI library. There are currently four types of components used by LAM/MPI. They are `boot`, `coll`, `rpi`, and `cr`. The `boot` module controls how the LAM environment is started most often through lamboot. The `coll` module controls the MPI collective communication functions. These functions can be optimized for basic, symmetric multi-processors (SMPs), and shared algorithms. The `rpi` module controls the MPI point-to-point communications in MPI programs. For example, these communications can be Transmission Control Protocol (TCP) communication (the default option) or Myrinet communication. The `cr` module controls the ability to checkpoint and restart MPI jobs at the thread level. More information can be found in [17].

5. CONCLUSION

The Beowulf cluster can be easily setup for MPI programming via OSCAR. The OSCAR configuration wizard does all of the micromanagement of the nodes for the user. This micromanagement refers to the host files, authentication scripts, installation images, security, and synchronization required for proper cluster maintenance. Nodes can be added or deleted quickly, thus creating a scalable computing environment. The different modules offered in LAM/MPI 7.0 provide further optimization of how MPI programs are run on the cluster.

Additional performance improvements can be done to the actual design of the cluster. RAM can be used to entirely replace the hard drive on each node. This is known as a *diskless cluster*. The main advantage of a diskless cluster is the speed increase in the access time to data on each node. RAM access time is several times faster than a hard disk's access time. Also, nodes are much easier to manage since all nodal information is stored on the server node. The client nodes then require a fraction of the data that their disk-based counterparts require. All information is transferred at boot time for each client. When combined with OSCAR, this diskless cluster is known as Thin-OSCAR, which is currently in development by the OCG [18].

REFERENCES

[1] D. H.M. Spector, *Building Linux Clusters*. Sebastopol, CA: O'Reilly & Associates, 2000, pp. 10-11.

[2] R. Buyya, *High Performance Cluster Computing Volume 2*. Upper Saddle River: Prentice Hall, 1999, p. 651.

[3] Persistence of Vision and Raytracer Pty. Limited, "POV-Ray: Documentation," August 2004, http://www.povray.org/documentation/.

[4] The Open Cluster Group, "Bylaws," January 2002, http://www.openclustergroup.org/bylaws.html.

[5] The Open Cluster Group, "Steering committee members," 2001, http://www.openclustergroup.org/people.html.

[6] J. Brechin and M. E. Michael, "NCSA OSCAR Installation Guide," University of Illinois at Urbana-Champaign, March 2002, http://oscar.openclustergroup.org/tiki-download_file.php?field=12.

[7] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. de Groot, and E. Lear, "Address allocation for private internets," BCP 5, RFC 1918, February 1996.

[8] Open Systems Lab, "LAM test suite version 7.1.1", University of Indiana, September 2004, http://www.lam-mpi.org/7.1/test-suite.php.

[9] The OpenBSD Group, "OpenSSH manual pages," September 2004, http://www.openssh.org/manual.html.

[10] The Open Cluster Group, "OSCAR 3.0 user's guide," November 2003, http://oscar.opencluster.org/tiki-download_file.php?field=34.

[11] The SIS Team, "The application stack," March 2002, http://www.sisuite.org/stachk.shtml.

[12] M. Brim, R. Flanery, G.A. Geist, and B. Luethke, "C3 Documentation," 2001, Oak Ridge National Laboratory, http://www.csm.ornl.gov/torc/C3/C3documentation.shtml.

[13] W. Gropp, S. Huss-Lederman, A. Lumsdaine, E. Lusk, B. Nitzberg, W. Saphir, and M. Snir. *MPI - The Complete Reference: Volume 2,* the MPI-2 Extensions. Cambridge, MA: MIT Press, 1998.

[14] Open Systems Lab, "LAM/MPI documentation," University of Indiana, March 2004, http://www.lam-mpi.org/using/docs/.

[15] G. Burns, R. Daoud, J. Vaigl, “LAM: An open cluster environment for MPI,” in *Proceedings of Supercomputing Symposium*, 1994, pp. 379-386.

[16] J. M. Squyres, and A. Lumsdaine, “A component architecture for LAM/MPI,” in *Proceedings, 10th European PVM/MPI Users' Group Meeting*, 2003, pp. 379-387.

[17] J. M. Squyres, B. Barrett, and A. Lumsdaine, “Boot system services interface (SSI) modules for LAM/MPI,” Computer Science Department, Indiana University, TR575, 2003.

[18] Center for Scientific Calculation, Sherbrooke Univerity, “Development, installation and maintenance of Elix-II, a 180 node diskless cluster running thin-OSCAR,” May 2003, http://oscar.openclustergroup.org/tiki-download_file.php?fileId=30.