2008

Debian Clusters for Education and Research: The Missing Manual

Kristina Wanous

DEBIAN CLUSTERS FOR EDUCATION AND RESEARCH
THE MISSING MANUAL

A project
Submitted
in Partial Fulfillment
of the Requirements for the Designation
University Honors

Kristina Wanous
University of Northern Iowa
May 2008

This Study by: Kristina Wanous

Entitled: Debian Clusters for Education and Research: The Missing Manual

has been approved as meeting the thesis or project requirement for the Designation University Honors.

5/5/08
Date

Dr. Paul Gray, Honors Thesis/Project Advisor

5/7/08
Date

Jessica Moon, Director, University Honors Program

A Reflection on the Debian Clusters Project

Kristina Wanous, Project Lead

Moore's Law states that the number of transistors that can fit on a single computer CPU doubles every two years [1]. As a corollary to this, the potential processing speed of computers doubles every two years as well. For the last several years, this has meant that computing speeds for processors has continued to grow, climbing from hertz to megahertz and now gigahertz without needing to have a change in the underlying structure of chips other than the addition of more transistors. Unfortunately, this paradigm has hit an insurmountable problem: the maximum number of transistors on a chip is reaching a wall because once the transistors become too close, they begin to overheat. This problem has encouraged the hardware industry to take on a new approach for faster speeds: adding more processors to a single motherboard, and adding multiple CPUs to a single processor chip ("cores").

Even as the number of processors and cores continues to increase, the demand for computing power has skyrocketed [2]. Extending the paradigm one step further, if multiple cores are better than one core, and multiple processors are better than a single processor, why not have multiple computers? This is the idea behind a cluster. A cluster is group of computers networked and working together to solve a problem faster than any single computer working alone. Clusters are used across a wide variety of disciplines, from weather forecasting and molecular dynamics, to simulating the effects of earthquakes on building structures and animating films like *Finding Nemo* and *Shrek* [3]. In order to meet the growing demand for this high processing power, more individuals need to be trained in the nontrivial setup and administration of clusters.

Debian Clusters for Education and Research: The Missing Manual [4], or just "Debian

Clusters" for short, meets the needs of this growing group of people. The Debian Clusters project is an online resource for anyone wishing to set up a cluster, particularly for those using Debian as the operating system. It features detailed step-by-step tutorials for every part of cluster setup and administration. In order to be accessible to people new to clusters, the walkthroughs assume no previous knowledge of clustering or networking and explain the principles and vocabulary behind the steps, yet are complex and detailed enough to be used as a reference by experienced cluster administrators. The operating system used in the project and tutorials is Debian [5], a Linux distribution, and the justification for this will be further explained.

The Debian Clusters project also has a counterpart at the University of Northern Iowa. Slightly over a year ago, Intel donated forty barebones servers to the Computer Science department. I used eight of these to set up a real Debian cluster from scratch while writing the online tutorials. Walking through all the steps myself ensured I would address all the issues and problems faced by someone setting up a cluster for the first time. When I started the project in fall 2006, I had little to no Linux or cluster experience, and so I was an ideal candidate for knowing what others without cluster experience would need to learn about. In a sense, the documentation online at the Debian Clusters website is also a transcript of everything that I have installed and configured on my cluster here at UNI.

The tutorials at the Debian Clusters site start with basic networking services. These include setting up a firewall, filesystem, Internet addresses, domain name, and authentication. Each one of these required extensive research on my part, since I had no previous experience with Linux administration. Many options exist for each of these services, and after reviewing them I went with the most common paradigms. The firewall pages detail how to set up a transparent firewall that will forward access to users to the cluster while preventing many

different kinds of initial attack attempts. NFS, the network file system, is fully explained as a way to mount users' home directories throughout the cluster, allowing them to be accessed from each one of the worker nodes. Assigning IP addresses using DHPC, the dynamic host configuration protocol, is detailed as well as setting up domain names for the cluster and for individual machines. Finally, LDAP is used as a way to allow authentication to be distributed across the cluster but managed in only one location.

With these basic networking details out of the way, the documentation continues on to explain a common cluster paradigm: users interact with the cluster through one machine, called the head node, which takes care of orchestrating job scheduling and other details on the worker nodes, which do the raw computing power. Rather than installing all the software on each one of the worker nodes, a worker node "master image" can be created on one node and then copied out to the hard drives of each of the other nodes. This cloning process can be done several different ways; Debian Clusters walks through three common methods after expanding upon the advantages and disadvantages of each.

Once all of the worker nodes have been cloned, configuration of the cluster to begin act in a synchronized method can be started. This involving installing a scheduler and queue; in this case, Torque and Maui are used. The scheduler and queue are responsible for dividing up the clusters' resources in an organized manner to maximize usage, often called resource allocation. Users submit programs to be run, called "jobs", and the scheduler and queuer schedule when the jobs will be run and which worker nodes will do the processing power. The scheduler and queue also make sure that no user has an unfair advantage to the resources over the other users, and that the users cannot interfere with each others' jobs (resource isolation).

Special software is also required in order to take advantage of multiple machines working

together. The basis for much of this software is MPI, the message passing interface. The Debian Clusters site covers how to install one implementation of MPI called MPICH. MPICH includes a set of libraries for parallel programming that users can utilize to write C or Fortran code that can run over multiple machines at once. These libraries are also used in many implementations of scientific software. Tremendous numbers of scientific software packages exist, but due to limitations on time and resources, only a few could be installed and configured as part of the Debian Clusters project. In the end, ones that were common and utilized at the University of Northern Iowa or its associated clustering partner, Earlham College, were picked. These included two packages for doing molecular dynamics, NAMD and Gromacs. These are often used in chemistry and physics. To represent biology, mpiBLAST was installed, a parallel implementation of the BLAST algorithm that does searches over databases of proteins or nucleotides. The installation and configuration of each of these (which was quite complicated in some cases) is detailed at the project site.

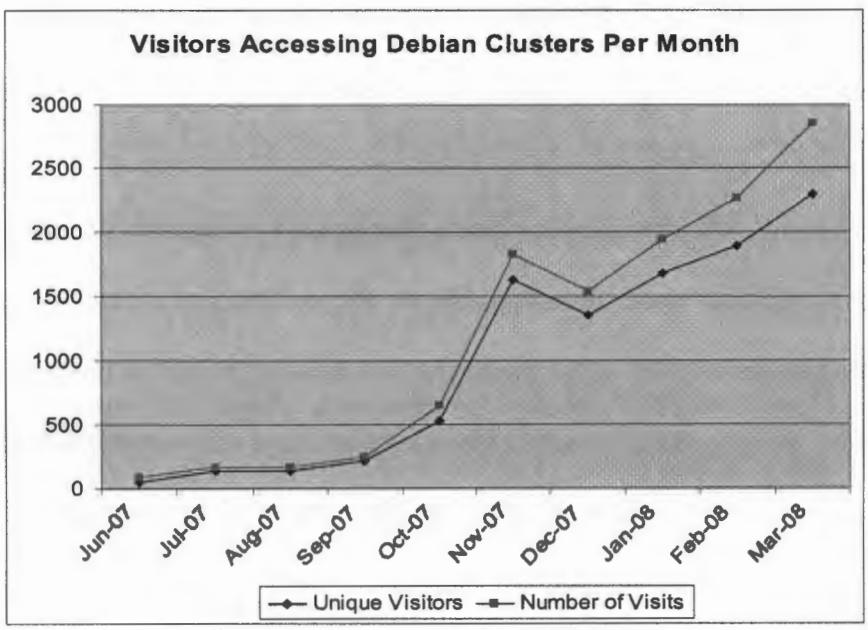
Finally, monitoring the load and services of a cluster is an important part of cluster maintenance. As part of addressing these, the process of installing and securing a web server was documented. A web server is necessary in order to display results from a monitoring program such as Nagios or Ganglia, both of which are covered in detail. While writing the Nagios tutorials section of the website, I happened to need to do some Nagios tweaking for a different cluster here at UNI as part of my job, and so the pages about Nagios also cover how to create custom plugins for Nagios in addition to using the ones that come with a basic installation.

Watching the use of the website grow has been rewarding. Since the website debuted in 2006, the number of unique visitors accessing the project has grown to over 2000 a month. This is displayed below in Figure 1. The website has also been referenced by others in a number of

different contexts, including the Debian user forums and several cluster forums run by private individuals. Feedback from users on the project has also been positive; comments written to me by e-mail have included "your pages are of big help" [6], "great howto on debian clusters...great resource" [7], "I am setting up a cluster on my university, and your website has been a GREAT help" [8], and "it really is the missing manual!" [9].

Figure 1: Usage of the Debian Clusters project.

These statistics were collected by the web log monitoring software Awstats [10].



I believe this demand is in part due to the unique nature of the project. While many resources exist on specific cluster components, few attempt to tie the use of these components into a single reference for those learning about clusters. When I first started working clusters two years ago, I met a steep learning curve. Many existing online and published resources assume a basic knowledge about networking and clustering software, making it difficult to use them as a beginner.

Further, many of these assume that individuals will be using a Linux distribution based

off Red Hat as the operating system of the cluster, creating another difficulty. Red Hat has traditionally taken a de factor role in clusters, and a lot of installation manuals as well as software packages are created specifically for Red Hat. As my project shows, Debian is a viable alternative, though Debian users are currently wanting for documentation. Debian is toted as one of the most stable Linux distributions; new packages undergo many rounds of testing before being incorporated into the main repository. It also offers many advantages to someone new to clusters or to Linux. The Debian Clusters project, in addition to providing a resource, attempts to encourage the use of Debian in clusters instead of the default Red Hat, though the use of Debian in supercomputing technologies has already become more widespread in the past couple years [11, 12].

For me, this experience has been exactly what I hoped it would be. When I first began the project, I had limited skills in Linux and cluster administration. Since then, I have built my own cluster and understand all the components that go into the setup and customization of one. Further, all of my experience has been captured in the Debian Clusters website, which will continue to exist as a resource for others. Judging by the usage of the website, demand for cluster references exists, and I am proud to have contributed to filling that need.

Completing this project has been a motivating factor in my life for the past year. As it draws to a close, I am both saddened and thrilled to have successfully completed such a large undertaking. It has much professional and personal meaning for me, and it serves as a snapshot of my studies during my undergraduate career. I am also glad to have contributed to the Debian and cluster communities by creating a resource that others will hopefully continue to use for years to come.

References

[1] Gordon E. Moore. "Cramming more components onto integrated circuits." Electronics, vol. 38, no. 8, 1965. Available: http://download.intel.com/research/silicon/moorespaper.pdf (URL)

[2] Alexandru Pancescu. (2007, Sept.). *Computer Industry to See Continued Growth.* Softpedia [Online]. Available: http://news.softpedia.com/news/Computer-Industry-to-See-Continued-Growth-65678.shtml (URL)

[3] Matt Hurzwitz. (2004, Nov. 9). *Incredible Animation: Pixar's New Technologies.* [Online]. Available: http://www.uemedia.net/CPC/vfxpro/printer_10806.shtml (URL)

[4] Kristina Wanous, Paul Gray, Charlie Peck. (2008, April). *Debian Clusters for Education and Research: The Missing Manual.* University of Northern Iowa. [Online] http://debianclusters.org/ (URL)

[5] Ian Murdock. (1996, Jan. 6). *A Brief History of Debian: The Debian Manifesto.* [Online]. Available: http://www.debian.org/doc/manuals/project-history/ap-manifesto.en.html (URL)

[6] Pau. (2007, Oct. 19). [Online]. Personal e-mail: typo

[7] Jan Dettmer. (2008, Feb. 11). [Online]. Personal e-mail: cluster

[8] Tim Russel. (2008, March 1). [Online]. Personal e-mail: NFS mounting problem on cluster

[9] Claudius Buerger. (2008, March 26). [Online]. Personal e-mail: Debian onto nodes

[10] Laurent Destailleur. (2008, Apr. 6). Awstats. [Online]. Available: http://awstats.sourceforge.net/ (URL)

[11] Timothy Morgan. (2006, Aug. 14). *HP Gives Debian Linux Equal Billing to Red Hat and SUSE.* IT Jungle [Online]. Available: http://www.itjungle.com/breaking/bn081406-story01.html (URL)

[12] Andrew Hendry. (2008, March 13). *Debian Linux cluster beats supercomputer in tsunami warnings.* PC World News [Online]. Available: http://www.pcworld.idg.com.au/index.php/id;1611547855;pp;1 (URL)

Main Page

Debian Clusters for Education and Research

This site is a series of step-by-step tutorials for anyone interested in building a Debian cluster. It is geared particularly towards those interested in a Debian cluster for education or research purposes. This site is being written alongside the construction of a real Debian cluster at my institute, starting from the ground (basic networking, DHCP, DNS, authentication) up through installation a queue/scheduler and scientific software.

However, these tutorials and guides take some assumptions about users for granted, including basic Linux and Debian knowledge (see the **Basic Linux Skills** page for clarification and pointers to resources). This is not meant as an introduction to Debian or on how to cluster in general - thankfully, there are plenty of helpful tutorials and guides already in existence on those topics. A basic understanding of Debian and cluster technology will aid in following the tutorials.

Comments, criticisms, and suggestions are welcome and appreciated. Please send them to the project lead, Kristina Wanous, at kwanous <at> debianclusters <dot> org.

About the Debian Clusters Project

Project Overview

- **Network Topology**

Setting up Debian

- **Base Installation of Debian**

Setting up Firewall Services

- **NAT with IPTables**
- **Fail2Ban: Preventing Brute Force SSH**

Setting up Services

- **Mounted File System: NFS**
- **Handing out IPs: DHCP**
- **Name Service: DNS and BIND**
- **User Authentication: LDAP**
    - **Using LDAP**

Creating Head Node and Worker Node Images

- **Bash Profile Modifications**
- **Password-less SSH for Users**
- **Installing Compilers**, including **Intel Compilers**
- **Udev: Renaming Network Interfaces**
- **Setting a Dynamic Hostname by DNS**

Cloning Worker Nodes

- First, make sure the above services are set up for the head node and worker.

Cluster Time-saving Tricks

Setting up Scientific Software on Debian

- **Source Installation Paradigm**
- **MPICH: Parallel Programming** - an MPI implementation is necessary for many parallel applications
- **Gromacs: Molecular Dynamics**
- **NAMD: Molecular Dynamics**
- **mpiBLAST: Nucleotide/Protein Searching**

Using a Scheduler and Queue

- **Resource Manager: Torque**
- **Scheduler: Maui**
- **Troubleshooting Torque and Maui**
    - **Torque and Maui Sanity Check: Submitting a Job**
    - **Torque and Maui: Submitting an MPI Job**

Using Torque and Maui:

- **Torque Qsub Scripts**
- **Torque Queue Configuration**

Cluster Monitoring on the Web

- **Setting up a Web Server**
- **Securing the Web Server**
- Monitoring Services: **Nagios**
- Monitoring Load: **Ganglia**

About the Debian Clusters Project

Note about the Debian Clusters Project from the Project Lead, Kristina Wanous

When I first started working with clusters, I had to climb a technical learning curve before I could really start utilizing and understanding the hardware and software I was putting together. Looking for written and online resources, I mostly found that those who needed to know already did, and that many resources are written for those already "in the know," at least partially. There's a lot of information to be gleaned from old newsgroup and mailing list archives, for those with the patience to look.

However, one of the problems with putting together knowledge this way is that, investigating installing MPI or LDAP, it's often distribution specific. Debian has its own, non-Red Hat way of getting things done, a lot of the resources out there are for Red Hat or at least specific to something other than Debian. These are great for general background knowledge but don't help in the specifics of installing and configuring for Debian.

This begs the question, why use Debian? Simply put, it's accessible. As a free (as in speech *and* beer, in this case) open source project, it offers an inexpensive alternative to other Linux distributions that come packaged with support fees. Debian supports all the functions and features necessary to build and maintain a working cluster, and this project means to demonstrate that. Further, Debian is a great operating system in general for people who'd like to learn more about Linux or clustering utilities. Its flexibility allows it to be used by everyone, from people who want to build a cluster in their basement to those who want production clusters.

This project is particularly geared toward those who would like to build a cluster for education or scientific research. However, it takes some assumptions about users for granted, including basic Linux and Debian knowledge (see the **Basic Linux Skills** page for clarification and pointers to resources). This is not meant as an introduction to Linux or Debian - thankfully, there are plenty of helpful tutorials and guides already in existence on those topics.

The assembly here is a compendium of the knowledge that I was able to gain from books, websites, and those who have helped me with this project.

I would especially like to thank a few individuals who have helped me. Most of the basis of the information here, as well as a lot of the specifics, have come directly from one professor's experience – Dr. Paul Gray, my mentor here at the **University of Northern Iowa (http://www.uni.edu/)** – and I could not have taken on this project without him. He's also provided me with all the hardware, software, and technical support necessary for this endeavor, as well as being a staunch advocate. (Check out his project, the **Bootable Cluster CD (http://bccd.cs.uni.edu/)** , which I also help with.) I would also like to thank Professors Charlie Peck from Earlham College and Tom Murphy from Contra Costa College for being my "mentors away from home". Their experience and support have added to this project in countless ways. Thanks also to my fellow students Jessica Puls, Alex Lemann, and Kevin Hunter for their help with various subjects. Finally, for all of his interest in the project and constant encouragement, thank you to my good friend and employer Todd Thomas.

Comments, criticisms, and suggestions are always welcome and appreciated. Please send them to kwanous <at> debianclusters <dot> com

-- Kristina Wanous

Project Overview

This project will walk through installing and configuring all the software necessary for a cluster. It starts with the assumption that you already have the hardware that you're going to be using, and that you have already **installed Debian** – preferably the testing version used at the time of this writing, **Lenny (http://www.debian.org/releases/testing/)** – on the machines you'll be using. (If you want some background on this, see the **Basic Linux Skills** page for pointers to resources.)

The information on the website is laid out in approximately the same order I followed in configuring my own cluster. I'll make reference to my cluster throughout the project and use examples from it (more information in **Network Topology**). Not all the information may be applicable to you, or you may need to tweak it to fit your needs.

The first steps in setting up a cluster are to configure the firewall, services, head node, and worker node image. In this particular example, I'll first be building the firewall and services on the same machine, then working on another machine to turn into the head node plus file services. Then I'll be working on the worker node image, and then sending that image out to the rest of the worker nodes.

Before getting started, there a few things to be decided on:

- First and foremost, the IP range and subnet mask for the internal network. (192.168/16, 172.16/12, and 10/8 are set aside for private networks and are non-routable. These are very good choices.)
- Second, the hostnames for the systems will come into play very early on. It's a good idea to have a plan about these.
- Third, the IPs for the head node, firewall, and any services machines also need to be known early on. (See the first point.)

For more information, continue on to **Network Topology**.

Basic Linux Skills

As mentioned on the front page, this is not an introduction to Linux or Debian. However, the assumptions used in this project are fairly small (with the exception of the first one!).

These include the abilities to:

- install Debian
- become root (have root access)
- navigate directories (cd)
- apt-get install something from the Internet
- view the items currently in the directory (ls)
- find the IP and MAC address of machines (ipconfig)

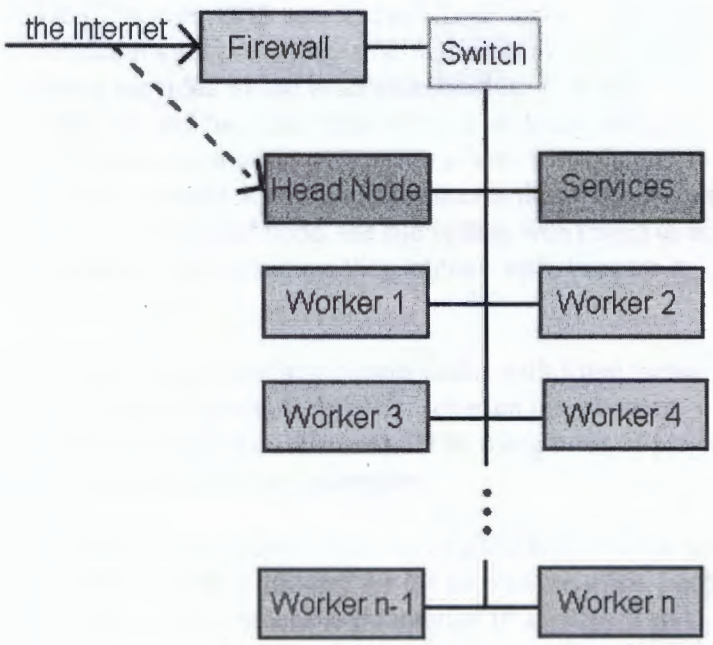
For those needing to get caught up to speed, I have found several sites to be helpful in learning Linux and Debian. These include

- **Linux Online (http://www.linux.org/lessons/)**
- **About Debian Linux (http://www.aboutdebian.com/)**
- **Debian GNU/Linux System Administration (http://www.debian-administration.org/)**
- **DebianHelp (http://www.debianhelp.co.uk/)**

Network Topology

A Typical Layout

A typical cluster setup has a few common components. First, it typically has one machine acting as the firewall and the gateway to the Internet. The firewall and all of the rest of the machines are then connected to an internal network with a switch. (There may be multiple internal networks – one for communication amongst the nodes, one for gathering runtime data, and such.) Users log into the head node, which is responsible for submitting jobs to all the other work nodes. Essential services like DHCP, DNS, and LDAP can be run on the firewall, head node, or on other dedicated services machines.



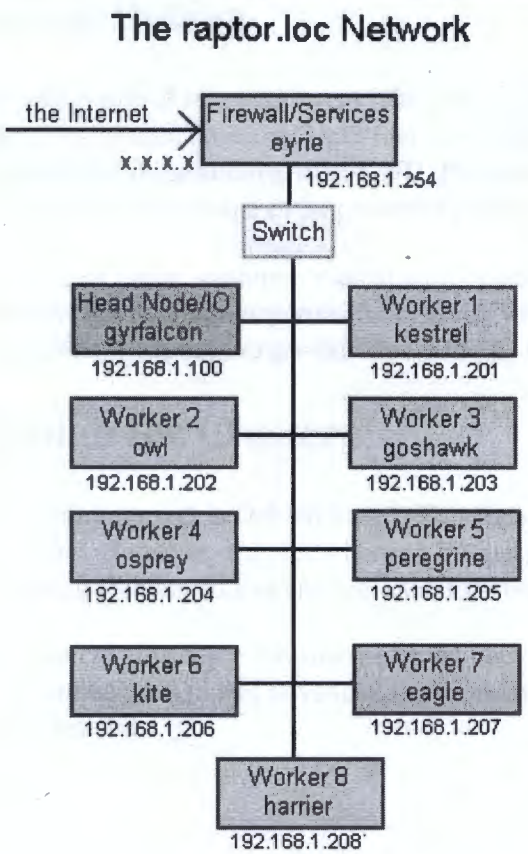
A typical cluster layout.

To access a service on the cluster - for instance, SSH or a web server – a user contacts the firewall. However, the firewall takes all requests to itself and forwards them on to an appropriate node (in other words, it takes requests and turns it around towards a pre-specified node on the inside network without the user needing to know what's going on behind the scenes). In this way, the firewall is transparent and only accessible from the inside of the network, making it much more secure. The firewall also does SNAT and DNAT (source network address translation and destination network address translation) to allow the machines on the internal network to send network traffic out through the firewall and also receive it.

To the right is network diagram typical for a cluster setup. Users SSH into the firewall's address but are validated through and connected to the head node. All of the machines are connected through the switch. The firewall has two IP addresses – one on the outside and one on the internal network. One machine is dedicated to running services for the cluster.

The dashed line indicates that, although the connection to the Internet runs through the firewall, from a user's perspective the firewall does not exist.

A Specific Layout



The cluster layout being used in the making of the Debian Clusters project.

To the left is the network layout I'll be using for my cluster, the one I'm building as I write the Debian Clusters documentation. The firewall will be providing most of the services in addition to doing SNAT/DNAT. It will be the DHCP server, DNS server, and LDAP server. The only service it's not providing is NFS, the file system. NFS is being provided by the head node because it is I/O intensive and the head node will not be doing anything other than interacting with users, which is less intensive than the worker nodes' job. This means that when people SSH into the head node, the file system won't need to be mounted on the machine they interact with, because it will be local.

People often name their cluster nodes with some theme, like types of penguins or coffee, or even just numbers (node100, node101, etcetera). I'll be using birds of prey in this cluster and in my examples.

Notice that the firewall has two IP addresses, one for the Internet (the X's), and one for the internal network. Each of the other machines has an internal IP address. The ranges for non-routable (private) IP addresses safe to use on the internal network are `192.168/16`, `172.16/12`, and `10/8`. I'll be using `192.168.1/24`.

Even before the cluster has been fully built, the IP addresses and host/domain names have been decided on, as well as the overall domain name for the internal network. This will be essential in configuring services. If you can get the MAC addresses of the worker nodes before setting them up, even better.

Here's a quick cheat sheet that might be helpful in following the examples:

eyrie	firewall/services	X.X.X.X 192.168.1.254
gyrfalcon	head node/IO	192.168.1.200
kestrel	worker node	192.168.1.201
owl	worker node	192.168.1.202
goshawk	worker node	192.168.1.203
osprey	worker node	192.168.1.204
peregrine	worker node	192.168.1.205
kite	worker node	192.168.1.206
eagle	worker node	192.168.1.207
harrier	worker node	192.168.1.208

Setting up Debian

About Debian

Debian is one of many different "flavors" of **Linux (http://en.wikipedia.org/wiki/Linux)** , an open source operating system based on Unix that uses **GNU software conglomeration (http://en.wikipedia.org/wiki/GNU_Project)** . Several other Linux flavors have been based on Debian in order to take advantage of its powerful package system, including Ubuntu and Knoppix.

Other than Linux, common mainstream operating systems include the **Microsoft Windows (http://en.wikipedia.org/wiki/Microsoft_windows)** family and the **Macintosh OS X (http://en.wikipedia.org/wiki/Mac_OS_X)** family.

Debian for Clusters

This entire project is Debian-based. This means that my cluster uses Debian as the operating system on all of its nodes. However, it's pretty hard to jump right into my cluster documentation if you're unfamiliar with installing Debian, so I've created a **walkthrough on installing Debian**.

I'm also creating more advanced Debian documentation on how to **tune the Linux kernel** within Debian. This can be used to add advanced functionality to the operating system, as well as to remove functionality not really needed.

Base Installation of Debian

Debian is of course the operating system of choice for this project. For people who work with Linux on a regular basis, installing Debian is a piece of cake, but the process can be a little intimidating for first time users. Follow the guide below to get a base installation of Debian up and running.

Getting Debian

The first step is obtaining a copy of the Debian installer. Since Debian is open source, this isn't too difficult. Visiting the Debian project's website, **http://www.debian.org/**, you'll see a few different options under the "Getting Debian" link on the left sidebar. You can download a small network installer image, download the full installer, download the entire current repository (something like 14 different CDs), order CDs... whew. Lots of options. Fortunately the first two are both pretty easy.

I prefer using the network installer, netinst. Since it's network based, however, you'll need to make sure you have a connection to the Internet on the machine you're installing it onto. Next you'll have to decide which of the three distribution lines you want. Debian comes in three flavors - stable, testing, and unstable. Stable Debian consists of packages that are proven to work, and a lot of effort goes into this validation. Because of this, it's also updated less frequently. Testing are packages that have been used for a short amount of time consistently in unstable. Testing is updated frequently. Unstable is bleeding edge current. For a balance between stability and usability, go with testing.

The testing installer images are **available here (http://www.debian.org/devel/debian-installer/)** from the Debian.org site. Under the first header, "netinst (generally 135-175 MB) and businesscard (generally 20-50 MB) CD images", choose your architecture. If you don't know what it is, you're probably running i386. Click on the architecture and you'll see a list of files. You want the one called `debian-testing-<your architecture>-netinst.iso`. Download this file and save it to your desktop.

Next, using your favorite CD burning software, burn the image to a CD. Your software may call this something like "Burn an Image" or "Bootable CD". If you don't have any burning software that supports this option, you can download a **14-day trial of BlindWrite (http://www.vso-software.fr/download.php)** from VSO Software. (I'm not endorsing them, but I've used the trial and it worked fine in my experience.)

Congratulations, you now have a Debian installer CD!

Installing Debian

The Basics

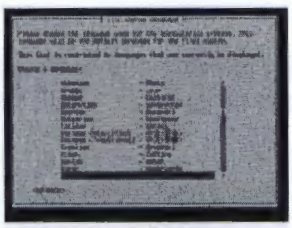
Warning: Installing Debian on the hard drive will erase any operating system and any data you currently have on it. Make sure you've copied off everything you want before you install Debian!

The Debian network installer CD splash screen.

Note to Virtualization Users: If you're trying out Debian as a virtual machine within a host operating system rather than installing it as the base operating system, you may have problems with it detecting the drive. In VMware, you must create a custom build rather than a typical build a choose an IDE hard drive, *even though SCSI is recommended.* If you're just

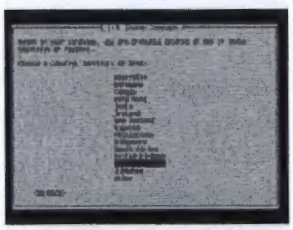
installing Debian as the base operating system (a "normal" install), this doesn't apply to you, and go ahead and continue below.

Take the CD to the machine you're about to install Debian on. Next, boot up the machine from the CD. You may have to hit a key to boot from CD, or enter the BIOS and enable booting from CD there. The magic keystroke for this varies by distribution - it could be F2, F8, Del, Esc, or something else. Or, you may not have to do anything at all, and you'll automatically boot up into the splash screen shown on the right. Go ahead and hit Enter to begin the installation process.



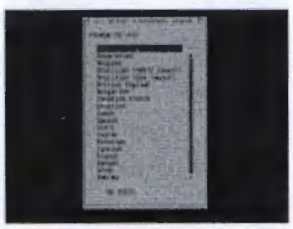
Choosing the language for the installation

First, you'll be prompted for the language to be used during the installation process. Unless you prefer something other than English, hit enter and continue. If you want to choose something else, use the up and down arrows until the one you want is selected, then hit enter.



Choosing the country

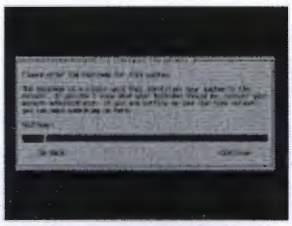
Next, choose your country. If you chose English in the previous step, the United States will automatically be selected for you already. Hit Enter if that's correct, otherwise use the up and down arrow keys until the one you want is selected, then hit enter.



Choosing the keyboard mapping

Now the installer's asking for the keyboard mapping. This is basically asking, when you hit a certain key on the keyboard, what letter should it be? If you live in the United States, yours is already selected. If that's not correct for your language, scroll down and choose the one that is. Then hit enter.

A few screens will flash by as the installer scans the CD and loads additional modules from it, then detects your hardware and attempts to connect to the network. If it cannot connect to the network, make sure your cable is plugged in correct and connected. If retrying fails, you may have to configure it manually.

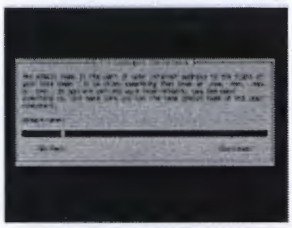


Setting the hostname for the system

The next prompt is for the hostname of the system. The hostname of the system is kind of like the first and last name of a person. It's used to identify the system. For instance, when you SSH into the system, if you're using BASH, you'll see a prompt like this:

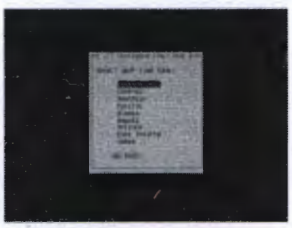
```
kwanous@gyrfalcon:~$
```

In the above example, my username is `kwanous` and the hostname of the machine is `gyrfalcon`. For this project, I'm going to be naming my nodes with a birds of prey theme, but you don't have to be as whimsical. You can use `headnode` or `nodeX` if you want, but it must be all lowercase. Delete the default of `debian`, enter yours, and hit enter. This can be changed later by editing `/etc/hostname`.



Setting the domain name for the system

The domain name should be the one this machine will be on. For instance, you might have an internal domain for your cluster (more about this in **Network Topology**. You may have gotten one already from your DHCP server. Enter one if you know it, otherwise leave it. This can be changed later by editing `/etc/resolv.conf`.



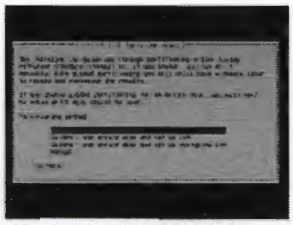
Selecting the correct timezone

Move the selection up and down with the arrow keys until it's on the correct timezone for your location, then hit enter.

Partitioning

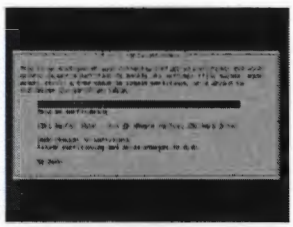
So far, installation has been pretty easy. The most difficult thing you had to choose was your keyboard mapping. You might be wondering if I just wrote this guide because I had too much free time on my hands. :) This is the part where it begins to get a bit tricker. Now we partition the drive, meaning we split it up into different sections that will be used differently. Most typical Windows installations only have one partition on the hard drive. Everything is kept in one file system. Linux is different in that it needs a minimum of two partitions: one for the file system, and one for the swap space.

Swap space is like virtual RAM in Windows. It's an area of the drive that Linux sets aside specifically to use when RAM gets full. The size of it will never shrink or grow; we're setting it right now. There are lots of varying opinions on how much swap space to set; I typically use two times the amount of RAM I have. For instance, in a machine with 1 GB of RAM, I will set 2 GB of swap space.



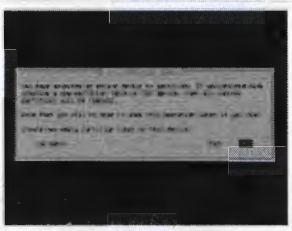
Choosing how to partition the disk

To get into this process, select Manual and hit enter.



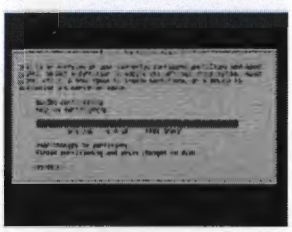
Select the correct hard drive

Even if you only have one hard drive, Debian will create a list of hard drives for you and ask you to select which one you want to set up. You can see in the screenshot that I'm using VMWare for this installation and so my hard drive is called VMWare Virtual IDE Hard Drive. Select your hard drive and hit enter.



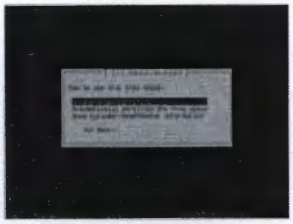
Creating a new partition table

If you're installing a hard drive with an existing operating system, you'll need to delete the previous partitions first. Unfortunately, I don't have screen shots on that. However, once you've deleted them, you'll get back to this same point. The installer asks whether it should create a new partition table. Even though the default is no, hit Tab twice to get to Yes and hit enter.



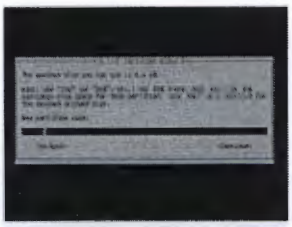
Returned to selecting a hard drive, but with a FREE SPACE section

You'll be returned to a previous screen, but this time it has a nice FREE SPACE section under the hard drive. Select with the arrow keys, then hit enter.



Choosing how to use the free space

You'll be prompted as to how you'd like to use this space. Keep the default entry, "Create a new partition", and hit enter.

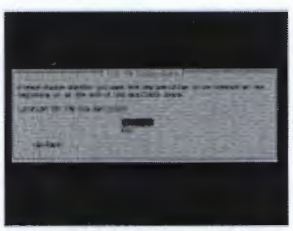


Setting the partition size

Now you need to set the size of the partition. By default, the Debian installer enters the full size of your hard drive. You don't want to use the full size, because you need to same some for swap. Enter the full size of your drive (you'll see it listed at the top as well, mine is 6.4 GB in the picture) minus the size the swap space will be. (See the **beginning of this section** for a discussion on how much swap space to use.) Then hit enter.

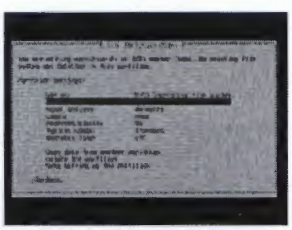
Choosing the type of partition

You'll be prompted for the type of partition. There can only be four primary partitions, but you can have as many logical partitions as you want. We're only using two partitions, so we won't need any logical partitions. Keep the default of "Primary" and hit enter.



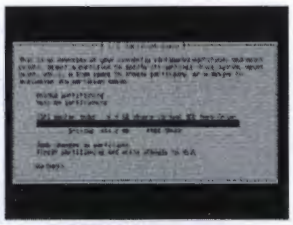
Choosing the location for the partition

You can put the partition at the beginning of the hard drive or the end. Keep the default of "Beginning" and hit enter.



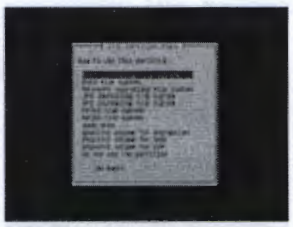
Viewing the partition

Whew, now the partition has been set up. You'll notice that the installer has set this up as a file system partition by default, which is what we want. (That's the "Use as: Ext3 journaling file system" part.) The "Mount Point: /" means that this will be the root file system, where all of the files will go. This is what we want. There's one small addition to this we need to make, not because it's necessary, but because it's better in principle: use the arrow keys to select "Bootable flag: off" and hit enter. It will set it to bootable, then return you to the same screen. Key down to "Done setting up the partition and hit enter.



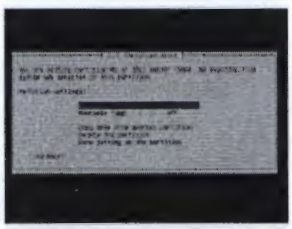
Viewing the new state of the hard drive

You'll be returned to the familiar partitioning window that shows you the state of your hard drive. Now you'll see the partition you just created, followed by the FREE SPACE line again. Select the FREE SPACE line and hit enter. Again, choose to keep the "create a new partition" default. This time, when it enters the rest of the space on the hard drive for you, hit enter. (Unless you messed up on the first partition and need to go back and "Delete a partition" to start partitioning over again.) Keep the default of a "primary" partition type, too. You won't be prompted for where to put the partition since you're using the rest of the free space on the hard drive. When see the partition settings window, we'll need to set this as swap space. Key down to "Use as: Ext3 journaling file system" and hit enter.

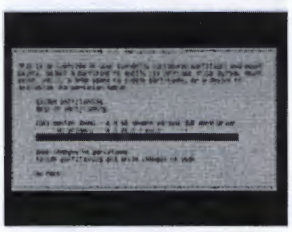


Selecting the partition format

You'll see a list of different formats this partition could be used for. Key down to "swap area" and hit enter.

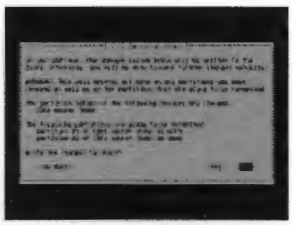


Once it returns you to the previous window, you'll see you have quite a few options fewer now that it's swap not a file system partition. That's ok, we're done with it anyway. Select "Done setting up the partition" and hit enter.



Viewing the finished state of the partition table

You'll be return once again to viewing the state of the partition table. By now, it should have two proud partitions like mine on the left does: a file system partition, and a significantly smaller swap area partition. Key down to select "Finish partitioning and write changes to disk" and hit enter.



Writing out the changes to the partition table

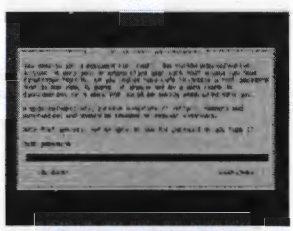
It will ask you to be sure that you're really ready to write out the changes to the disk. This is the step where it becomes permanent. Hit tab twice to select "Yes" and hit enter. Depending on the machine, it may take a while for this to finish. Then it will go straight into installing the base system. Yay! Done partitioning!

Installing the System

Now the installer gets to go to town and install Debian.

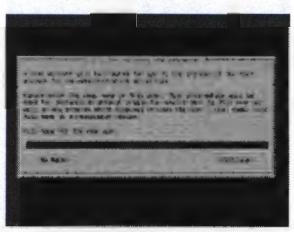
The Debian installer chugging away installing the base system

Depending on your Internet connection and computer speed, it may take a while for it to install everything. Expect at least five minutes and possibly a lot longer.



Choosing the root password

On most flavors of Linux, with Ubuntu being a little strange about this, the root account is the administrative account. The root account has absolute power. There can be lots of user accounts, but only one root... "one root to rule them all." Pick a nice, secure password. You can change it later by signing into the root account and issuing `passwd`. Once you enter it and hit enter, you'll be prompted to type it again, just to make sure you typed what you really thought you did...



Enter the full name of the person you're creating the first user account for

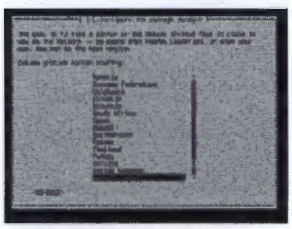
Next you'll create a normal user account. First, it asks you for the full name of the person who will use this account. This information isn't used as much anymore as it used to be. You can leave it blank or enter a full name, which ever you want. **This is not the name of the account they will use to log in as.**

Entering the account name for the first user account

This is the step where you choose the important part - the user/account name. This will be used as the identifier for the user all of over the system, and the username they will use to log in as. You'll see mine as `kwanous` in all of my examples.

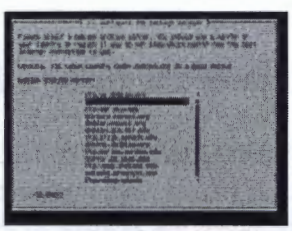
Entering the password for the user account

Like the root account, this account also needs a password. As a security measure, **don't use the same password as the root account!** After you enter it and hit enter, you'll be prompted for it again.



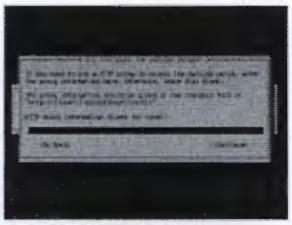
Selecting the correct country for the apt repository

Next, the installer will prompt you to help it set up `apt`, the packaging system used to update and install software ("packages"). Select your country and hit enter. (Or, if you're lucky enough to know someone who manages a repository, or you run one yourself, scroll up to the top of the list to choose "enter information manually" and enter your own.)



Choosing an apt mirror

Now you'll select the specific mirror, or source, that you want to use. Your choice may be hit or miss - I'd go with their recommendation for `ftp.<your country>.debian.org`.

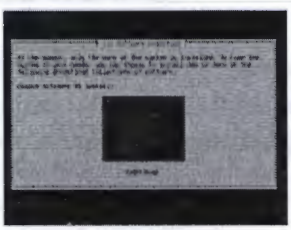


Entering a proxy for the apt mirror

Unless you know for your mirror that you specifically need a proxy, you probably don't. Leave it blank and hit enter. It will attempt to connect to the mirror just to make sure everything's in order.

Popularity contests aren't just for people

You'll be asked if you want to allow your statistics to be counted for popularity contests among the packages. You can if you want. I usually keep the default of "No" and hit enter. You'll notice the next screen flashes by that it at least says that it installs and configures the survey regardless of what you choose. :P



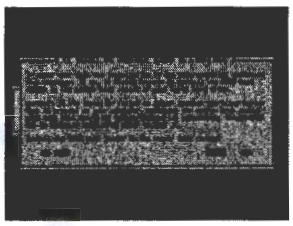
Choosing the software to install

This next step is very important. For the cluster, you'll want as little software as possible, because you want it streamlined rather than bulky. That said, my tutorials will walk you through everything you need to install, so you **don't need to install a bunch of preselected software** now. You don't need a print server, a windowing manager, or anything else along those lines. In fact, they'll just take up space and slow your system down, and maybe even create some security holes. That would be a "bad thing".

It's unfortunately very easy to accidentally hit the wrong thing in this step. Use the arrow keys like normal, highlight the two options that have asterisks by default ("Desktop environment" and "Standard system"), and hit **space** to deselect them. Once you have no asterisks in any of the entries, hit enter to continue.

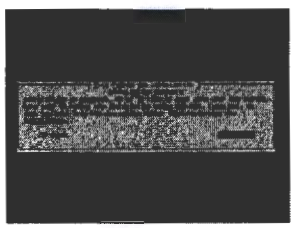
Installing the Boot Loader, GRUB

GRUB is the boot loader of choice for this installer. In fact, it's the only choice for this installer. (The other common boot loader is LILO. There are other Debian installer CD images that contain both or just LILO.) Boot loaders are responsible for jump starting your operating system when you boot the computer up. You definitely want one.



Prompt for installing grub

When you're asked whether you want to install grub, keep the default of "Yes" and hit enter.



Done!

After it finishes installing grub, you're done! It will give you a prompt telling you that it's done installing. After you hit enter, it will reboot the system. Like it says, be sure to take out the CD so it doesn't boot back into the installer.

Updating

The first thing you'll want to do is update your software. Log in as root using the password you supplied earlier. You won't be greeted with a congratulatory installation message, unfortunately, just the prompt. To update all your installed packages, first run

```
apt-get update
```

This updates the cache local on your computer about what versions packages are as well as what packages are available. Then, to automatically upgrade everything you have installed to the newest version, run

```
apt-get dist-upgrade
```

Most packages will take care of upgrading without any input from you. However, there will be an occasional one that prompts. Usually, if you don't understand what the prompt is asking, go with the default option.

My "Base" Software to Install

Finally, since (if you followed my recommendation), you installed the barest bones necessary to get the system up and running, there are three helpful packages you'll want to install that weren't installed as part of the installation.

First, you'll want the ability to **SSH (http://en.wikipedia.org/wiki/Ssh)** into the system, as well as out of it. You'll get both abilities with

```
apt-get install ssh
```

Second, at some point, you'll want `rsync`. It's used to intelligently copy files around. Get it with

```
apt-get install rsync
```

Third, I always pipe long files to `less` to make them easier to read. It's installed with

```
apt-get install less
```

Then you're done!

Setting up Firewall Services

Because of the setup with the firewall forwarding SSH and other requests, security measures need to be set up in two places, both the firewall and the head node. This the barebones minimum for setting up a firewall; a more strenuous approach is highly recommended. Here, we will set up

- **NAT with IPTables**
- **Fail2Ban to prevent brute force SSH attacks**

IPTables: DNAT/SNAT

Iptables is built into Debian and doesn't need to be installed. It needs to be configured on the firewall to forward requests between the Internet and the internal network (DNAT and SNAT). See **NAT with IPTables** for directions.

Fail2Ban

Fail2ban utilizes IPTables to stop brute force SSH attacks. After a certain number of SSH attempts with a failed username or password, the IP address of the attacker is temporarily blocked from SSH for a certain amount of time. Because SSH requests are forwarded by the firewall to the head node, the head node needs to be set up with Fail2ban. See **Fail2Ban: Preventing Brute Force SSH** for directions.

NAT with IPTables

Intro

The firewall can use IPTables to forward packets between the Internet and the internal network. IPTables is the interface to changing the built in netfilter firewall built into the Linux kernel. We'll also use IPTables to forward SSH requests for the firewall to the head node, making the firewall transparent. (Users of the cluster should interact with the head node, not the firewall.)

For my example (see **Network Topology**), this means that users will see something like the following. They specify the address of my firewall, eyrie, but then are deposited onto gyrfalcon, my head node.

```
kwanous@cassowary:~$ ssh eyrie.X.X.edu
kwanous@eyrie.X.X.edu's password:

Linux gyrfalcon 2.6.18-4-486 #1 Wed May 9 22:23:40 UTC 2007 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
kwanous@gyrfalcon:~$
```

Most newer version of Debian (etch, sid, lenny) come with iptables installed. However, to make sure iptables installed with the latest version, run

```
apt-get install iptables
```

Configuring IPTables

IPTables is generally configured from the command line; it isn't read from a file. To make changes that "stick" after a reboot, a bash script run at startup can be used to enter all the configuration commands. We've already created a script for you - **IPTables Script**. Download this, save it as `local`, place it in `/etc/init.d/`, then change its status to be executable (`chmod +x local/code>`).

Next the file needs to be symlinked to a place where it will be loaded as the firewall is started up. Debian stores the files loaded during runtime in <code>/etc/rc*.d, where * is the runlevel: 0, 1, 2, 3, 4, 5, 6, or S. In Debian, run level 2 is the normal run level. `/etc/rcS.d` stores the scripts that are run regardless of the run level. Within a given run level directory, the scripts are run in order of lowest to highest numbered. Symlinks that start with `S` are executed with the argument `start`, `K` (for kill) ones are executed with `stop`.

To be the most secure and not allow any slips while the server is started up, the `local` script needs to be executed before the networking script. Networking is `S40networking`, so `S39` will do. Symlink the file with

```
ln -s /etc/init.d/local /etc/rcS.d/S39local
```

or to `/etc/rc2.d/S39local` if you prefer.

Understanding the Iptables Commands

This iptables commands in the script use bash variables set earlier in the script, but they could just as easily be specified in plain within the script. Be sure to change the values to fit your network. `$LOCALNET` should be the IP range of the local network specified as a CIDR mask, and $EXTERNIP should be the IP address for the firewall's external interface. $SSHHOST should be the head node's IP address.

Source Network Address Translation (SNAT)

The first iptables command does the SNAT - translating packets generated by computers on the internal interface to go out to the Internet. Any machines seeing packets on the Internet from the cluster will see them as coming from the firewall, and they will respond to the firewall. Rather than accepting the packets themselves, the firewall them forwards them to the inside of the network.

```
iptables -t nat -A POSTROUTING -d ! ${LOCALNET} -j SNAT --to ${EXTERNIP}
```

- `-t nat` specifies that this rule's type is network address translation (NAT), aka IP masquerading
- `-A POSTROUTING` appends a rule to the `POSTROUTING` chain, meaning it will be processed after all the other possible processing has been done
- `-d ! ${LOCALNET}` means any packets destined for an IP not within `${LOCALNET}`
- `-j SNAT` means to jump to the SNAT rule
- `--to ${EXTERNIP}` specifies that any packets leaving will assume the IP `${EXTERNIP}`

All together in English, this rule says to take any packets not destined for sources within the internal network and send them out to their destination on the outside network after changing the source destination IP address to the firewall's IP.

Destination Network Address Translation (DNAT)

This rule does just the opposite. It takes SSH packets coming in from the Internet and sends them along to the head node.

```
iptables -t nat -A PREROUTING --dst ${EXTERNIP} -p tcp --dport 22 -j DNAT --to-destination ${SSHHOST}
```

- `-t nat` specifies network address translation (NAT), aka IP masquerading
- `-A PREROUTING` appends a rule to the `PREROUTING` chain, meaning that this will take affect before any rules have a chance to work on the packet
- `--dst ${EXTERNIP}` specifies original the destination of the packet (to the firewall)
- `-p tcp` specifies `tcp` as the protocol
- `--dport 22` specifies the destination port of the packet
- `-j DNAT` means to jump to the rule DNAT to do the destination network address translation

In English, this means to take any packets addressed to the firewall that are TCP packets on port 22 (SSH), change their destination to the head node, and forward them on to the internal network. This could be changed to forward all tcp packets inside by removing `--dport 22`, but a firewall that forwards everything inside the internal network wouldn't be much of a firewall.

Starting and Stopping NAT

Since the `local` script has been symlinked into the `/etc/rcS.d/` or `/etc/rc2.d/` directory, it will automatically load at boot. To start and stop the script from the command line, use

```
/etc/init.d/local start
```

or

```
/etc/init.d/local stop
```

Since this script is responsible for allowing the nodes inside the firewall their Internet access, be advised that stopping this script will kill the Internet connection to any machines behind the firewall.

References

- Kirch, Olaf, and Terry Dawson. Linux Network Administrator's Guide. 2nd ed. Sebastopol, CA: O'Reilly Media, Inc., 2000.
- Brockmeier, Joe "Zonker", Dee-Ann LeBlanc, and Ronald W. McCarty, Jr.. Linux Routing. 1st ed. Indianapolis: New Riders, 2001.
- **http://www.5dollarwhitebox.org/wiki/index.php/Howtos_Basic_IPTables**
- **http://www.linux-mag.com/id/282/**

IPTables Script

This script is used to configure iptables for DNAT and SNAT (destination/source network address translation). It's part of the **NAT with IPTables** page and explained there. This file should be saved to `/etc/init.d/local` and symlinked to `/etc/rcS.d/S39local`.

```
#!/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
NAME=local
DESC="local services"

# Replace this with your firewall's static IP
EXTERNIP="X.X.X.X"

# The IP address of the internal machine that will respond to SSH requests
SSHHOST="192.168.1.200"

# The IP range of the internal network
LOCALNET="192.168.1.0/24"

case "$1" in
  start)
        iptables -t nat -A POSTROUTING -d ! ${LOCALNET} -j SNAT --to ${EXTERNIP}
        iptables -t nat -A PREROUTING --dst ${EXTERNIP} -p tcp --dport 22 -j DNAT --to-destination ${SSHHOST}
        echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
        ;;
  stop)
        echo 0 > /proc/sys/net/ipv4/conf/all/forwarding
        iptables -t nat -F
        ;;
    *)
        N=/etc/init.d/$NAME
        echo "Usage: $N {start|stop}" >&2
        exit 1
        ;;
esac

exit 0
```

Fail2Ban: Preventing Brute Force SSH

Fail2ban is used to combat brute force SSH attacks. It does this by watching the log files for invalid logins by specific IPs under a certain amount of time and then using iptables to ban them. If you've set up **NAT with IPTables**, your users will be specifying the firewall's name when they SSH in, but they will actually be SSHing into the head node. Since the head node handles all SSH requests, Fail2Ban should be installed on the head node.

To install fail2ban, run

```
apt-get install fail2ban
```

Running `iptables -L` after this should now show a chain for fail2ban.

Configuring Fail2ban

Fail2ban is automatically configured for the most part. However, little items need to be tweaked. `/etc/fail2ban/fail2ban.conf` is responsible for general settings for fail2ban, such as what log to append to. More specific settings can be changed in `/etc/fail2ban/jail.conf`. However, it's recommended that this file not be directly changed. Instead, make a copy to jail.local (`cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local`) and the local file with override the .conf one.

First, find `ignoreip`. It's always important for you to have a way in! These are IPs are fail2ban will ignore - IPs listed here can always have invalid login attempts and still not be blocked. In my file, I'm putting down the network ranges for my internal network (192.168.1.0/24) as well as one other trusted IP address of a machine that I will be able to SSH into if need be. **These need to be space separated!** If they are not, fail2ban won't block anyone.

```
# "ignoreip" can be an IP address, a CIDR mask or a DNS host
ignoreip = 192.168.1.0/24 X.X.X.X
```

Default options are listed somewhere near the top of the file. Although there are specific filters specified further down, these default options will take effect unless override in the specific filters. `bantime` specifies how long an IP address sits in "time out" before it is allowed to attempt to log in again. The default of 600 seconds (10 minutes) is probably fine.

Individual filters are specified with brackets surrounding the filter's name. By default, only `[ssh]` is active. Notice this filter has a higher `maxretry` than specified in the default above. `Maxretry` specifies the number of times an IP address can attempt to log in before being banned. I changed mine down to 3.

```
[ssh]

enabled = true
port    = ssh
filter  = sshd
logpath  = /var/log/auth.log
maxretry = 3
```

After making changes to any files, restart fail2ban with `/etc/init.d/fail2ban restart`.

Oops!... Unblocking Blocked IPs

Fail2ban timesout are only temporary. Still, it's important to know how to unblock an IP address once fail2ban has started blocking it. If you do `iptables -L`, you'll see all the IPs currently blocked:

```
gyrfalcon:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
fail2ban-ssh  tcp  --  anywhere             anywhere            tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain fail2ban-ssh (1 references)
target     prot opt source               destination
DROP       0    --  lucient.rootmypc.net  anywhere
RETURN     0    --  anywhere              anywhere
```

Notice `lucient.rootmypc.net` is currently being blocked here. Rather than waiting ten minutes for it to be unblocked, you can tell iptables to drop that rule. The syntax is `iptables -D <rulename> <rule line>`. To unblock `lucient.rootmypc.net`, I issued

```
gyrfalcon:/etc/fail2ban# iptables -D fail2ban-ssh 1
```

Entering `iptables -L` again should show that that rule is now gone, and that IP address is again allowed to SSH in (at least until they try to log in incorrectly the magic number of times again).

Links

- **http://www.the-art-of-web.com/system/fail2ban/**
- **http://www.fail2ban.org/**

Setting up Services

There are four essential services for a cluster I'm going to walk through setting up: a mounted file system (NFS), a DHCP server, a domain name server (BIND), and authentication services. I'll set them up first on the services machines as well as the head node and the worker node image I'm creating. This is done before all the worker nodes are imaged; this way the configuration only needs to be done once.

Shared Filesystem

NFS is used to share a filesystem amongst the nodes. This way users' directories can show up on all of the machines without having to copy the files over each time something is changed. It is also going to be used to install shared software one; this way the software only has to be installed and configured once.

- **Mounted File System: NFS**
  - **NFS Server**
  - **NFS Client**

Handing out IP Addresses

DHCP is used to hand out IP addresses on the local network. Since my cluster is going to have its own non-routable IP range (192.168.1.X), all of the machines won't be accessible to the outside world. I need an internal DHCP server to assign the IP addresses to the individual machines if I don't want to have to individually set the IP address on every single machine.

- **Handing out IPs: DHCP**
  - **DHCP Server**
  - **Troubleshooting DHCP**

Domain Names

DNS - and specifically, we'll use the implementation BIND - is used to hand out domain names to each of the machines. Again, I could set them up individually on every single machine, and then hand out a list to each machine (called `/etc/hosts`) that contains the name of each machine and which IP address it goes to, but it's much less time consuming and easier to set up the domain names once using DNS and to have all the configuration in one location.

- **Name Service: DNS and BIND**
  - **Installing and Configuring BIND**
  - **Creating Forward DNS Records**
  - **Creating Reverse DNS Records**
  - **Testing and Troubleshooting BIND**

User Authentication

LDAP is used for setting up an authentication service across the nodes. This will allow users to have the same user login and password across all the nodes, and again, allow configuration in one place rather than on each individual machine.

- **User Authentication: LDAP**
  - **LDAP Server**
  - **LDAP Client**
  - **Using LDAP**
  - **Troubleshooting LDAP**

Mounted File System: NFS

The Network File System (NFS) allows the entire cluster of computers to share part of their file system - very important for running **parallel code like MPI**. In this paradigm, one (or more) machines hold the actual files on their physical hard drive and act as an NFS server while the other nodes "mount" the file system locally. To the user, it looks like their files exist on all of the machines at once.

With NFS, though multiple servers can exist, they must be mounted in different places. (Two machines cannot both share their `/home` directory and have everyone else mount them, but one machine can share `/home/students` and another can share `/home/faculty`.)

I'll use an NFS share for all of my users' home directories, as well as shared software.

NFS setup is relatively easy, though it varies by what the machine's role will be.

- **NFS Server**
- **NFS Client**

Troubleshooting

Sometimes an error like the following occurs:

```
peregrine:~# mount /shared
mount: RPC: Timed out
```

Restarting the portmap service has worked in these situations for me:

```
peregrine:~# /etc/init.d/portmap stop
Stopping portmap daemon....
peregrine:~# /etc/init.d/portmap start
Starting portmap daemon....
peregrine:~# mount /shared
```

Links

- **http://www.debianhelp.co.uk/nfs.htm**
- **http://uwsg.indiana.edu/usail/network/nfs/tips.html**

- man exports

NFS Server

This is part two of a three page tutorial on setting up NFS. The full tutorial includes

- **Mounted File System: NFS**
- **NFS Server**
- **NFS Client**

Supporting Packages for the NFS Server

Any machines that will act as NFS servers need to

```
apt-get install nfs-common nfs-kernel-server
```

Included in the dependencies for these is `portmap`, which is responsible for communicating on the proper ports and passing connections over.

/etc/exports

Next, `/etc/exports` needs to be configured. This file, automatically installed, controls which part of the server's file system will be shared with the other machines. Comments can be added with the `#` sign. The format of the file is

```
<directory to share> <allowed machines>(options)
```

(Notice that there's no space between the allow client machine specification and the opening parentheses for the options.)

For mounting users' home directories (more about this with **LDAP**), it's often wise to mount the directory some place other than `/home` because all of the Debian machines will already have a `/home` directory. For instance, I'm using `/shared` because it doesn't already exist on all of my NFS clients, and there won't be confusion with mounting over files already in that directory.

The clients can be specified a variety of ways: by IP address, domain name, or CIDR mask. Below in my example, I'm allowing all machines with an IP in my network range (192.168.1.1 - 192.168.1.254) to mount this file. Since I'll also be configuring them with **DNS** to be `X.raptor.loc`, I could also use `*.raptor.loc` for my clients. (However, using IP addresses doesn't require DNS to be up and running, taking out one possible point of failure, and so is generally more stable.) Multiple specifications for a given mount point may be space-separated with parentheses immediately following each client specification, like the following.

```
<directory to share> <allowed machines #1>(options for #1) <allowed machines
#2>(options for #2)
```

Options for /etc/exports include

- `rw/ro` - `rw` allows both reads and writes to the filesystem. NFS acts read only (`ro`) by default.
- `async/sync` - The asynchronous option allows the NFS server to respond to requests before committing changes. According to `man`, this can improve performance but can corrupt data in the event of a crash.

Synchronous is the default.

- `wdelay/no_wdelay` - Write delay allows the NFS server to put off committing changes to disk if it suspects that another write is coming shortly.
- `subtree_check/no_subtree_check` - Subtree checking improves security by checking not just that the client has rights to mount a directory, but all of the directory's subdirectories as well. Subtree checking is enabled by default, but the NFS server will complain if you don't specifically indicate it.
- `root_squash/no_root_squash` - Root squashing prevents a root user on a machine using the filesystem to act as it if is the root user on the actual filesystem; this is more secure. It is on by default.

My `/etc/exports` file looks like this:

```
/shared   192.168.1.0/24(sync,no_wdelay,subtree_check,rw,root_squash)
```

I am mounting gyrfalcon's `/shared` directory for any machines within my internal IP range. Sync is enabled, so the NFS server finishes writing changes to disk before responding to new requests. I do not have a write delay, so changes will be immediately written. Any subdirectories will also be checked for proper permissions before being mounted. The directory will be readable and writable, and root on any NFS clients will not have the same rights as root on gyrfalcon itself.

Restarting the NFS Server (Learning to Share)

Go ahead and restart the nfs-kernel with

```
/etc/init.d/nfs-kernel-server restart
```

or with

```
exportfs -var
```

Both methods restart the NFS server. `exportfs` can also be used to restart the behavior just towards a specific client. Just to eyrie, for example, would be

```
gyrfalcon:~# exportfs 192.168.1.254:/shared
```

To make sure that the mount is being shared, use `showmount -e`. You should see the shared directories listed:

```
gyrfalcon:/user# showmount -e
Export list for gyrfalcon:
/shared 192.168.1/17
```

Next...

Continue on to setting up the **NFS clients**.

NFS Client

This is part three of a three page tutorial on NFS. The full tutorial includes

- **Mounted File System: NFS**
- **NFS Server**
- **NFS Client**

NFS Client Packages

Any machines that will act as NFS clients and will be using the shared filesystem need to

```
apt-get install nfs-common
```

Again, `portmap` is included with `nfs-common`, so it doesn't need to be installed separately. Then `/etc/fstab` needs to be configured.

/etc/fstab

`/etc/fstab` provides the opposite functionality as `/etc/exports` - rather than telling what to export, this file tells the machine what to import and where to mount it. In other words, this includes eventhing it needs to mount, even its own hard drives, floppy drives, cdrom drives, and such. The format of this file is as follows:

```
<source to mount from> <local mount point> <type of filesystem> <options> <dump> <pass>
```

You'll want to add the NFS line under any existing lines, so that it gets mounted after your local drives. When you specify an NFS mount, use

```
<NFS server>:<remote location>
```

You should be able to use the `defaults` option, which uses the options you set up in `/etc/exports` on the NFS server, and `dump` and `pass` don't need to enabled, so they can have `0's`.

My `/etc/fstab` looks like this.

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/sda1       /               ext3    defaults,errors=remount-ro 0       1
/dev/sda2       none            swap    sw              0       0
/dev/hdb        /media/cdrom0   udf,iso9660 user,noauto     0       0
/dev/fd0        /media/floppy0  auto    rw,user,noauto  0       0
192.168.1.200:/shared     /shared   nfs     defaults        0       0
```

After specifying it in `/etc/fstab`, it will automatically be mounted when the machine starts up, if the mount point exists. (If you're using `/shared` or another directory that isn't automatically created as part of Debian, you'll need to create the directory.) To mount it without having to reboot, use

```
mount <mount point>
```

For instance, mine would be `mount /shared`. Similarly, you can also do `umount <mount point>` to unmount a filesystem.

Troubleshooting: NFS Mounts not Loading at Boot

I had a problem with my firewall not automatically mounting the NFS systems at boot, for whatever reason. I could issue `mount -a` as root as soon as the system booted up, but it wouldn't boot at load time, despite the `/etc/fstab` file. To "hack fix" it, I added my own script at `/etc/rcS.d/S46mymount`. (46 runs right after S45mountnfs.sh and S46mountnfs-bootclean.sh.) It needs to be executable (`chmod +x nfshack`), but the file itself is simply:

```
#!/bin/bash

mount -a
```

If anyone knows of a better fix for this, please contact me at kwanous <at> debianclusters <dot> org.

Handing out IPs: DHCP

The DHCP server is responsible for handing out dynamic IPs to all the machines on the network who don't have static IPs explicitly set. This allows much easier configuration of IP addresses - rather than having to go in and edit the IPs on different machines, all the settings for IP addresses are in one location.

The machine that will act as the DHCP server needs to have new packages installed and configured. The DHCP clients only need to be set to receive IP addresses dynamically (see below).

- **DHCP Server**
- **Troubleshooting DHCP**

DHCP Tips

Determining Eth Names

To determine which port corresponds to `eth0` or `eth1`, plug an Ethernet cable into one of the ports but not both, then run

```
mii-tool
```

You'll be able to tell which interface that port belongs to based on which one is negotiating a link. If you'd like to change which is which, or give them completely new names, see **Udev: Renaming Network Interfaces**.

Setting a Dynamic IP

Setting a machine to receive a dynamic IP is easy. Unless you specifically specified this when you first installed Debian (and most people don't), then it's already set up for you. In `/etc/network/interfaces`, a dynamic IP set up by the Debian installer looks like this:

```
# The primary network interface
auto eth0
iface eth0 inet dhcp
```

The important part here is the `dhcp` (versus static). In this example, eth0 will send a DHCP request out to the network. The response from a DHCP server includes what eth0 should set its IP address to, the netmask, the default gateway (router), and any domain name servers. Of course, this means that you need to have a DHCP server set up to do this!

Setting a Static IP

Generally, the only machine in a cluster that should have a static IP is possibly the outside interface of the firewall, and the DHCP server. Setting up a static IP requires editing two flies: `/etc/network/interfaces` again, and `/etc/resolv.conf`. Because a machine with a static IP won't contact a DHCP server, it needs much more background information about the world.

/etc/network/interfaces

Instead of using `dhcp` like a machine with a dynamic IP does, `static` needs to be specified in `/etc/network/interfaces`, and then the `address`, `netmask`, and `gateway` all need to be specified. In my setup, my firewall needs to have a static IP on its outside interface (eth0), which looks like this:

```
# The primary network interface
auto eth0
iface eth0 inet static
        address X.X.X.X
        gateway X.X.X.X
        netmask 255.255.128.0
```

- `address` is the IP address it should have.
- `netmask` is the netmask for this network
- `gateway` is the router it uses to talk to the outside Internet

Because it acts as a DHCP server on the internal interface (eth1), it also has a static IP address there. Notice this time that it doesn't have a gateway specified - that's because it already has a gateway specified in the above statements and doesn't need another (it will route through it's own eth0 to eth0's default gateway).

```
# The secondary network interface (goes to the switch)
auto eth1
iface eth1 inet static
        address 192.168.1.254
        netmask 255.255.255.0
```

/etc/resolv.conf

Machines with static IP addresses also need to be told who their nameservers are, since they won't receive that information from their DHCP server. (This is another reason why it's nice to have dynamic IPs - if your nameserver changes and your entire cluster used static IPs, you'd have to change this file for every machine.) This is specified in `/etc/resolv.conf` with

```
nameserver X.X.X.X
```

where `X.X.X.X` is the IP address of the nameserver. Multiple nameservers can be specified on multiple lines.

Links

- **http://www.isc.org/index.pl?/sw/dhcp/**
- **http://www.linux-mag.com/id/473**

DHCP Server

This is the second page of a three-part tutorial on setting up DHCP. The full tutorial includes

- **Handing out IPs: DHCP**
- **DHCP Server**
- **Troubleshooting DHCP**

The DHCP server is responsible for handing IP addresses for all the machines with dynamic IPs on the internal network (which should be all of them, preferably).

Installation

On the machine to be the DHCP server, first,

```
apt-get install dhcp3-server
```

It should install successfully, then end with an error like this:

```
Starting DHCP server: dhcpd3 failed to start - check syslog for diagnostics.
invoke-rc.d: initscript dhcp3-server, action "start" failed.
```

Checking out `/var/log/syslog` will yield the following:

```
eyrie:~# tail /var/log/syslog
Mar  8 12:52:01 eyrie dhcpd: Wrote 0 leases to leases file.
Mar  8 12:52:01 eyrie dhcpd:
Mar  8 12:52:01 eyrie dhcpd: No subnet declaration for eth0 (192.168.10.203).
Mar  8 12:52:01 eyrie dhcpd: ** Ignoring requests on eth0.  If this is not what
Mar  8 12:52:01 eyrie dhcpd:    you want, please write a subnet declaration
Mar  8 12:52:01 eyrie dhcpd:    in your dhcpd.conf file for the network segment
Mar  8 12:52:01 eyrie dhcpd:    to which interface eth0 is attached. **
Mar  8 12:52:01 eyrie dhcpd:
Mar  8 12:52:01 eyrie dhcpd:
Mar  8 12:52:01 eyrie dhcpd: Not configured to listen on any interfaces!
```

In other words, the DHCP service won't start because it has not been configured to hand out IP addresses on any of the network interfaces. The file responsible for this is `/etc/dhcp3/dhcpd.conf`.

/etc/dhcp3/dhcpd.conf

This file has a lot of examples to look through, but that also adds a lot of kruft to the file. I'd recommend making a copy of this file (`cp dhcpd.conf backup-dhcpd.conf`) and then removing most of the example comments to make the file easier to read.

The first lines to add/change are right below the comment

```
# option definitions common to all supported networks...
```

The options specified here will take affect by default for all subnets below this line. (You might have multiple subnets if you're running multiple switches with different IP ranges on different ones. If you just have one switch, you'll probably just have one.) If you don't specify these options here, you'll want to specify these

within the individual subnet declarations. Options include

- `option routers` are the default gateways. You'll want the internal IP address of your firewall here.
- `option domain-name-servers` are the nameservers. You should put the IP address of your cluster's own nameserver (**Name Service: DNS and BIND**) as well as possibly the nameserver your firewall uses outside of the network. They should be comma separated if you specify multiple nameservers.
- `default-lease-time` is the amount of time a DHCP lease (which includes the IP address of the machine requesting plus all the other stuff just specified above) should be given out if a specific time isn't requested. The value is in seconds by default. 14400, or four hours, is a typical value.
- `max-lease-time` is just what it says - the maximum amount of time the DHCP server will issue a DHCP lease for.

Eyrie will be running the DHCP server for me on its internal interface. Here's the corresponding section on eyrie:

```
# option definitions common to all supported networks...
option routers 192.168.1.254;
option domain-name-servers 192.168.1.254, X.X.X.X;
default-lease-time 14400;
max-lease-time 14400;
```

Next the individual networks need to be declared. These begin with the IP address and netmask with any directives in { brackets }. The IP range and netmask the subnets are determine which interfaces the DHCP server will give service to. The declaration follows this syntax:

```
subnet <subnet IP> netmask <subnet mask> {
        <any options>
}
```

The declaration for my internal network is shown below. There's also an example with **DHCP Server for Three Subnets**.

```
subnet 192.168.1.0 netmask 255.255.255.0 {
        option domain-name "raptor.loc";
        deny unknown-clients;
}
```

- `option domain-name` indicates that this network should be called "raptor.loc"; this will be important later on with **Name Service: DNS and BIND**
- `deny unknown-clients` means that IPs and leases will not be handed out to computers not listed in this file; not just anyone should be plugging in and receiving an IP address from my DHCP server!

That of course leads to the next step: declaring the clients that the DHCP knows and should supply DHCP leases to. Each computer that should be issued a DHCP lease (IP address) needs to be listed. The interfaces listed in this file will be the only ones that the DHCP server responds to; any DHCP requests from other computers will be ignored. Further, because it's by interface (MAC address), if different interfaces need IP addresses, each MAC address for each machine needs to be specified individually. Here is a typical one:

```
host gyrfalcon.raptor.loc {
        hardware ethernet 00:e0:81:75:94:30;
        fixed-address 192.168.1.200;
        option host-name "gyrfalcon";
}
```

The declaration begins with the host name of the computer with the domain name of this particular internal network appended to it. The MAC address is specified with `hardware ethernet`. `Fixed-address` is the IP address that will be given to gyrfalcon when it requests on. `Option host-name` specified what this particular client should be called. The client will later use this to set its own host name (preventing the host name from having to be set in multiple places). If you're setting up multiple networks for the cluster, again see the dhcp.conf for the **DHCP Server for Three Subnets**.

Restarting the DHCP Server

After adding information for all of the clients, the DHCP server should be ready to start. Do this with

```
/etc/init.d/dhcp3-server restart
```

Errors like the following indicate a syntax problem in `/etc/dhcp3/dhcp.conf`

```
eyrie:~# /etc/init.d/dhcp3-server restart
dhcpd self-test failed. Please fix the config file.
The error was:
Internet Systems Consortium DHCP Server V3.0.4
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/
/etc/dhcp3/dhcpd.conf line 25: unexpected end of file

^
Configuration file errors encountered -- exiting
```

whereas a successful restart looks like this

```
eyrie:~# /etc/init.d/dhcp3-server restart
Stopping DHCP server: dhcpd3.
Starting DHCP server: dhcpd3.
```

You can make sure it's really running with

```
ps aux | grep dhcp
```

Then, double check for any errors with `tail /var/log/syslog`. While you're looking here, make sure that the DHCP server is only handing out leases on the proper interface. You want to have a message like this

```
Jul  2 16:32:39 eyrie dhcpd: No subnet declaration for eth0 (X.X.X.X).
Jul  2 16:32:39 eyrie dhcpd: ** Ignoring requests on eth0.  If this is not what
Jul  2 16:32:39 eyrie dhcpd:    you want, please write a subnet declaration
Jul  2 16:32:39 eyrie dhcpd:    in your dhcpd.conf file for the network segment
Jul  2 16:32:39 eyrie dhcpd:    to which interface eth0 is attached. **
```

This is a good thing, because you don't want to be handing out IP addresses to the outside network, just the internal network for the cluster.

Trouble?

For help with other errors than the ones listed here, see **Troubleshooting DHCP**.

DHCP Server for Three Subnets

This is part of the **DHCP Server** tutorial.

The DHCP server for this cluster is 172.16.0.254, 192.168.10.254, and 10.0.0.254 on its three interfaces. It serves the same machines on each of their three interfaces. Each subnet needs to be declarated separately. Each interface on each machine also needs its own separate declaration on each subnet. The declaration for one host, latte, has been left for an example.

```
# option definitions common to all supported networks...
default-lease-time 14400;
max-lease-time 14400;
option routers 172.16.0.254;
option domain-name-servers 172.16.0.254, 10.0.0.254;

subnet 172.16.0.0 netmask 255.255.255.0 {
  range 172.16.0.100 172.16.0.150 ;
  option domain-name "gig1.loc";
  deny unknown-clients;
}

subnet 192.168.10.0 netmask 255.255.255.0 {
  range 192.168.10.100 192.168.10.150;
  option domain-name "fe0.loc";
  deny unknown-clients;
}

subnet 10.0.0.0 netmask 255.255.255.0 {
  range 10.0.0.100 10.0.0.150;
  option domain-name "gig2.loc";
  deny unknown-clients;
}

host latte.gig1.loc {
        hardware ethernet 00:E0:81:44:AD:9E;
        fixed-address 172.16.0.199;
        option host-name "latte";
}

host latte.gig2.loc {
        hardware ethernet 00:E0:81:44:AD:9F;
        fixed-address 10.0.0.199;
        option host-name "latte";
}

host latte.fe0.loc {
        hardware ethernet 00:E0:81:44:AD:D1;
        fixed-address 192.168.10.199;
        option host-name "latte";
}
```

Troubleshooting DHCP

This is the third page of a three-part tutorial on setting up DHCP. The full tutorial includes

- **Handing out IPs: DHCP**
- **DHCP Server**
- **Troubleshooting DHCP**

Changing dhcp.conf

After changing dhcp.conf, make sure to restart the dhcp server. `/etc/init.d/dhcp3-server restart` should work, as should `/etc/init.d/dhcp3-server stop` and then `/etc/init.d/dhcp3-server start`.

/var/lib/dhcp3/dhcpd.leases

The DHCP log file can be a handy tool for diagnosing whether leases have been handed out and for how long. Unfortunately, only leases handed out to unknown hosts (hosts not specifically mentioned in `/var/lib/dhcp3/dhcpd.leases`) will be logged in this file.

Verifying DHCP

The first step is making sure that the dhcp server is running. Issuing a `ps aux` command and grepping it for dhcp is the easiest way to do this. If the dhcp server is running, something like the following should be returned.

```
eyrie:~# ps aux | grep dhcp
root      4033  0.0  0.1   2488  1044 ?        Ss   00:24   0:00 /usr/sbin/dhcpd3 -q
root      4090  0.0  0.0   2176   768 ?        S<s  00:25   0:00 dhclient3 -pf /var/run/dhclient.eth0.pid
                                                                 -lf /var/lib/dhcp3/dhclient.eth0.leases eth0
```

If the server isn't running, try starting it with `/etc/init.d/dhcp3-server start`. If that fails, check `/var/log/syslog` for information as to why it failed to start.

On the other hand, if the server is running, sometimes it's difficult to tell whether it actually is handing out DHCP leases. There are two ways to double check if it is by running `tcpdump` on the dhcp server or on the client asking for an IP address. Both will require physical access to a dhcp client.

Client-side Testing: Dhclient

First check whether the client can request a dhcp lease. Do this by sitting in front of the client and executing `dhclient`. Ideally, it should return the following

Client Success

```
peregrine:~# dhclient
Internet Systems Consortium DHCP Client V3.0.4
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/00:50:04:b2:eb:a2
Sending on   LPF/eth0/00:50:04:b2:eb:a2
Sending on   Socket/fallback
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.254
bound to 192.168.1.100 -- renewal in 6425 seconds.
```

- The IP after `DHCPACK` should be the IP address of the DHCP server.
- The IP after `bound to` should be the IP address specified for this client in `/etc/dhcp3/dhcpd.conf`.

Client Failure

Alternatively, you might receive something like this:

```
peregrine:~# dhclient
Internet Systems Consortium DHCP Client V3.0.4
Copyright 2004-2006 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/eth0/00:50:04:b2:eb:a2
Sending on   LPF/eth0/00:50:04:b2:eb:a2
Sending on   Socket/fallback
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 7
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 9
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 16
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 12
No DHCPOFFERS received.
```

In this case, the dhcp server either isn't receiving the request or may be failing to respond. Double-check that the MAC address in /etc/dhcp3/dhcpd.conf matches the MAC shown by a `dhclient`.

Server-side Testing: Tcpdump

Tcpdump can give an idea of what's going on from the dhcp server's perspective. It should be easy to install with `apt-get install tcpdump`. Once installed, it can be used to display all the traffic coming in and out of a given network interface. You'll either want to do this sitting in front of the computer (if you're going to listen on all interfaces) or by listening only on the interface handing out DHCP leases. SSHing into the computer and then running tcpdump is a bad idea because the ssh connection will constantly be generating traffic, unless you specify which ports to listen on.

Tcpdump can be run on just one interface using

```
tcpdump -i <interface> -n port 67 or port 68
```

Ports 67 and 68 are specifically for DHCP traffic - 67 for requests and 68 for responses.

Server Success

Running `dhclient` from the client computer should result in traffic like this on the dhcp server if successful: notice in the last line, the IP address for the dhcp server is shown responding back (BOOTP/DHCP, Reply) to the client with its new IP address.

```
eyrie:~# tcpdump -i eth1 -n port 67 or port 68
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
15:08:32.100367 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP, Request from 00:50:04:b2:eb:a2 (oui Un
15:08:32.104525 IP 192.168.1.254.bootps > 192.168.1.100.bootpc: BOOTP/DHCP, Reply, length 300
```

Server Failure

If no traffic occurs, then the request isn't going through the server, which means something may be wrong with the network in between. This can be tested with pings.

Alternatively, traffic like the following indicates that the server sees the request but is ignoring it. Make sure that /etc/dhcp3/dhcpd.conf is set up correctly to hand out leases and then the MAC address for this client is correct.

```
eyrie:~# tcpdump -i <interface> -n port 67 or port 68
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
15:04:03.105460 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP, Request from 00:50:04:b2:eb:a2 (oui Un
15:04:07.105566 IP 0.0.0.0.bootpc > 255.255.255.255.bootps: BOOTP/DHCP, Request from 00:50:04:b2:eb:a2 (oui Un
```

Name Service: DNS and BIND

Domain Name Service (DNS) is responsible for resolving IP addresses to domain names and vice versa. On the Internet, it's used exactly for that. When you type a URL in a web browser, like `www.debianclusters.com`, your machine asks a nameserver for the IP address of the web server responsible for that URL. This is called forward DNS. Reverse DNS refers to the opposite process - finding a domain name from a URL.

A cluster uses DNS for the same purpose, but none of the domain names are available to the public Internet. If you've been following through these tutorials, you've seen that I've been using the internal network `raptor.loc`, though `cluster.loc` or `debian.cluster` or anything else that doesn't map to an external domain on the Internet would have worked just as well. DNS is then used to give the machines their host names and to simplify services that run over SSH. Some software, like **MPICH**, require both forward and reverse DNS to be set up.

BIND, the Berkeley Internet Name Domain, is one implementation of DNS, and it's commonly used in Linux. As of the time of this writing, **Debian lenny (http://www.debian.org/releases/testing/)** is running version 9.4.1.

The step involved in setting up the DNS server (with BIND) are:

- **Installing and Configuring BIND**
- **Creating Forward DNS Records**
- **Creating Reverse DNS Records**
- **Testing and Troubleshooting BIND**

Master versus Slave Nameservers

There are primarily two types of nameservers: master nameservers and slave nameservers.

Depending on the number of nodes, most smaller clusters only need one nameserver running to serve DNS for the entire cluster. However, for larger setups (and for larger zones on the Internet), multiple nameservers need to be set up to handle traffic as well as provide fallbacks if the primary nameserver(s) fail. Having to update multiple nameservers with new information seems contrary to the idea of gathering all the domain information on one computer (rather than manually updating `/etc/hosts` files), and it is!

This is where the distinction of the nameservers comes into play. Master nameservers hold original information - they're the places where the zones are configured. Slave nameservers, at regularly scheduled times, ask the master nameservers for all of the DNS information (called a zone transfer) and then cache that information. They can also store the information in a temporary files in case the primary nameserver goes down. How long the slave holds this for, as well as how often it asks the master nameserver for information, is part of the configuration in the **forward DNS** and **reverse DNS** records.

Changes in the master nameserver are then propagated through the slave nameservers when the they query the master for new information. Slaves can tell if the information has been updated based on the serial numbers in the records. In addition to that, by default Bind9 (the version on Debian Lenny, as of the date of writing this) automatically notifies any slave nameservers of changes whenever bind on the master is restarted with changes.

When setting up only one nameserver for a cluster, that cluster is the "master" nameserver, though the naming distinction isn't very important when there's only one server.

References

- **Berkeley Internet Name Domain (BIND) (http://www.isc.org/index.pl?/sw/bind/)**
- **http://www.debian.org/doc/manuals/network-administrator/ch-bind.html**
- Liu, Cricket, and Paul Albitz. DNS and BIND. 5th ed. Sebastopol, CA: O'Reilly Media, Inc., 2006.

Installing and Configuring BIND

This is the second page of a five part tutorial on setting up **Name Service: DNS and BIND**. The full tutorial includes

- **Name Service: DNS and BIND**
- **Installing and Configuring BIND**
- **Creating Forward DNS Records**
- **Creating Reverse DNS Records**
- **Testing and Troubleshooting BIND**

Installing Bind 9

The first step in setting up the DNS server is to

```
apt-get install bind9
```

Then go ahead and run `ps aux | grep named` to make sure that the service started successfully (`named` is the name of the Bind daemon). If not, check `/var/log/syslog` for errors.

/etc/bind/named.conf.options

This is the file that specifies how the DNS server works. Even though you should be running this on the internal network, behind a firewall, there are a couple of "good practice" habits to add to this file. Within the `options` brackets of this file, below any existing entries, add the following:

```
version none;
allow-query { address1; address2; etcetera; };
allow-transfer { none; };
```

- `version-none` tells the server not to respond with the Bind version when queried (such as when being queried by `dig`). The version can also be specified in quotes. Since there are version-specific Bind exploits, it's just a generally good idea not to give out the version for free.
- `allow-query` specifies which machines are allowed to use this nameserver for lookups. Again, this probably won't be an issue because the server should be behind a firewall and not advertising on the general Internet, but it's still a good practice. Add your machines instead of "address1; address2; address3"; I'm specifying mine with `192.168.1.0/24`. (On the other hand, if you were running this nameserver for a domain name on the internet, you'd want to allow queries from anyone trying to get to your site, so you wouldn't want this option!)
- `allow-transfer` specifies the machines that are allowed to download all of the zone information (all of DNS information, basically, about which IP addresses go to which domain names, and vice versa - more about zones in the next section). If you're running any slave nameservers, you'll want to add their information here, because they need to be able to download all the information. Without this option, any old server would be able to download all of the information, which is a security risk.

Here's what mine looks like, after I took out all the developer comments (be sure to make a backup if you do this!). The first three lines in options were already automatically added.

```
options {
        directory "/var/cache/bind";

        auth-nxdomain no;    # conform to RFC1035
        listen-on-v6 { any; };

        # Added - KW
        version none;
        allow-query { 192.168.1.0/24; };
        allow-transfer { none; };
};
```

There are a few important points about syntax here - and Bind is very particular about syntax! It will fail if you have any syntax errors. (`tail /var/log/syslog` if refuses to start for pointers to which lines on which files are the problem.) Semicolons are used to end a statement or option. This means everything from options within brackets (notice mine has a semicolon after `192.168.1.0/24`) to ending brackets. Bind is pretty good about printing readable errors about missing semicolons:

```
eyrie:/etc/bind# /etc/init.d/bind9 restart
Stopping domain name service...: bind.
Starting domain name service...: bind failed!
eyrie:/etc/bind# tail /var/log/syslog
Jun 29 00:38:42 eyrie named[2870]: starting BIND 9.4.1 -u bind
Jun 29 00:38:42 eyrie named[2870]: found 1 CPU, using 1 worker thread
Jun 29 00:38:42 eyrie named[2870]: loading configuration from '/etc/bind/named.conf'
Jun 29 00:38:42 eyrie named[2870]: /etc/bind/named.conf.options:9: missing ';' before '}'
Jun 29 00:38:42 eyrie named[2870]: loading configuration: failure
Jun 29 00:38:42 eyrie named[2870]: exiting (due to fatal error)
```

As shown above, to test your changes to the file, restart bind with

```
/etc/init.d/bind9 restart
```

/etc/bind/named.conf.local

This is where you declare your "zones". On the Internet, zones usually refer to groups or single domains or subdomains. (For instance, `debianclusters.com` might be a zone, including `subdomain1.debianclusters.com` and `subdomain2.debianclusters.com`). That being said, unless you're running multiple networks (through multiple switches), you'll probably only have two - one for forward DNS and one for reverse DNS.

There's more about that in the next section. For now, know that you'll need two files (for each domain/IP address).

First, you'll need one for the internal domain name. I've been using `raptor.loc` in my examples. This is the name of the zone. When creating the file for that zone, the convention for file naming is to typically put `db.<name>` (so I'm naming mine `db.raptor.loc`), but technically the name doesn't matter except for readability.

The second file is for the IP addresses for reverse lookup. Because of the way reverse lookup is conducted, the way this domain is declared is by taking the IP address, reversing the order of the sections, and then appending `.in-addr.arpa`. For example, the zone name for my 192.168.1/24 becomes `1.168.192.in-addr.arpa`. I'll call that file `db.1.168.192`.

My `/etc/bind/named.conf.local` is below. There's also an example of a `named.conf.local` file for a **cluster that's running three IP ranges on different switches**.

```
zone "raptor.loc" {
        type master;
        file "/etc/bind/db.raptor.loc";
};

zone "1.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/db.1.168.192";
};
```

The next step is to begin creating these files. Continue on to **Forward DNS - Name to IP Address (db.yourdomain)**!

3 IP Ranges Cluster named.conf.local

This is an example of `/etc/bind/named.conf.local` file from a nameserver on a cluster running over three different internal networks whose domain names are `gig1.loc`, `gig2.loc`, and `fe0.loc`.

This is part of the **Name Service: DNS and BIND** page.

```
zone "gig1.loc" {
        type master;
        file "/etc/bind/db.gig1.loc";
};


zone "gig2.loc" {
        type master;
        file "/etc/bind/db.gig2.loc";
};

zone "fe0.loc" {
        type master;
        file "/etc/bind/db.fe0.loc";
};

zone "0.16.172.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0.16.172";
};

zone "0.0.10.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0.0.10";
};

zone "10.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/db.10.168.192";
};
```

Forward DNS - Name to IP Address (db.yourdomain)

This is the third page of a five part tutorial on setting up **Name Service: DNS and BIND**. The full tutorial includes

- **Name Service: DNS and BIND**
- **Installing and Configuring BIND**
- **Creating Forward DNS Records**
- **Creating Reverse DNS Records**
- **Testing and Troubleshooting BIND**

/etc/bind/db.yourdomain

`/etc/bind/db.yourdomain` is the file responsible for resolving a host's IP address from its domain name. This is the `db.yourdomain` file we referred to in the `/etc/bind/named.conf.local` file in the last section. It needs to be manually created.

Time to Live

The first line in this file should specify the TTL (the time to live) for this zone. This is how the nameserver runs before it rechecks the local files to see if any changes have been made. It is specified with `$TTL` and then the time. The default value, without any units, is in seconds, but minutes and hours can be specified with `m` and `h` respectively, like this:

```
$TTL 24h
```

Start of Authority Record

The next line needs to be the start of authority (SOA) line. The format of this record is very specific, as shown below.

```
<domain name>.   IN  SOA  <primary nameserver>. <email address of admin>. (
                     <serial number>
                     <time to refresh>
                     <time to retry>
                     <time to expire>
                     <negative caching ttl>
)
```

- `domain name` - The domain name needs to be specified followed immediately by a period.
- `IN` and `SOA` refer to this being an Internet and start of authority record.
- `primary nameserver` - This is domain name of the server you're currently setting up to be the master nameserver.
- `email address of admin` - This field is meant be helpful for people rather than the computers. It is the e-mail address for people to contact if they have problems with the domain. Of course, it's less important on an internal network on a cluster. Something like root@yourdomain should be fine. Replace any `@` characters with periods.

The rest of the fields here are much more important when running multiple nameservers, one as the master and one or more as slaves that draw their information from the master. It isn't a bad idea to have a backup DNS server running in case the first one goes down.

- The `serial number` can be any whole number, as long as it increments as this file changes. The serial number is what slave nameservers use to determine whether any changes have occurred since the last time they contacted the master nameserver. A common convention is to use the date, yyyymmdd, with the number of changes that have occurred that day appended to it. Then it's easy for an administrator to tell if the serial number has been incremented when making changes.
- `refresh` is the amount of time a slave nameserver runs before contacting the master nameserver to see if any changes have occurred in the zone data.
- `retry` is the amount of time a slave nameserver waits before contacting the master nameserver again after having attempted to reach it once and failing.
- `expire` is how long the slave nameserver will continue to run and provide data after having lost contact with the master nameserver.
- `negative caching ttl` is the amount of time the slave nameserver will give "negative" information about a domain, saying that it doesn't exist or the type of data requested doesn't exist.

My file so far looks like this. Semi-colons are comments.

```
$TTL    24h

raptor.loc.   IN  SOA   eyrie.raptor.loc.   root.raptor.loc (
        2007062800 ; serial
        3h         ; refresh
        30m        ; retry
        7d         ; expire
        3h         ; negative caching ttl
)
```

Nameserver Records

Although we've already specified the primary nameserver above in the SOA record, we need to specify any nameservers here, with the format

```
<domain name>. IN NS <nameserver1>.
<domain name>. IN NS <nameserver2>.
```

and so forth. `NS` here stands for Name Server. They can be specified by domain name or by IP address, but specifying it by IP address (you guessed it) takes out a potential point of failure. Make sure to specify the master name server here, as well as any slaves.

I only have my single nameserver and no slaves:

```
raptor.loc. IN NS 192.168.1.254.
```

Address Records

Address (A) records are where we finally get to specify the domain names for each of the machines. Here, the format is

```
<full domain name>. IN A <IP address>
```

Make sure to specify each of the machines on the internal network. The IP addresses here need to match those specified with the **DHCP** server. Notice mine follow the layout in my game plan from the **Network Topology**. Here's my full `db.raptor.loc` file. Semicolons are comments.

```
$TTL    24h

raptor.loc.   IN  SOA   eyrie.raptor.loc.   root.raptor.loc (
        2007062800 ; serial number
        3h         ; refresh time
        30m        ; retry time
        7d         ; expire time
        3h         ; negative caching ttl
)

; Nameservers
raptor.loc.  IN  NS  192.168.1.254.

; Hosts
eyrie.raptor.loc.        IN  A   192.168.1.254
gyrfalcon.raptor.loc.    IN  A   192.168.1.200
kestrel.raptor.loc.      IN  A   192.168.1.201
owl.raptor.loc.          IN  A   192.168.1.202
goshawk.raptor.loc.      IN  A   192.168.1.203
osprey.raptor.loc.       IN  A   192.168.1.204
peregrine.raptor.loc.    IN  A   192.168.1.205
kite.raptor.loc.         IN  A   192.168.1.206
eagle.raptor.loc.        IN  A   192.168.1.207
harrier.raptor.loc.      IN  A   192.168.1.208
```

Restarting Bind

Bind can be restarted at this time

```
/etc/init.d/bind9 restart
```

and it should successfully reload. It will complain about its missing file(s), though.

```
eyrie:/etc/bind# /etc/init.d/bind9 restart
Stopping domain name service...: bind.
Starting domain name service...: bind.
eyrie:/etc/bind# tail /var/log/syslog
Jul  3 19:22:42 eyrie named[2847]: command channel listening on ::1#953
Jul  3 19:22:42 eyrie named[2847]: zone 1.168.192.in-addr.arp/IN: loading from master file /etc/bind/db.1.168.
```

Continue on to the next section, **Reverse DNS - IP Address to Name (db.yourIPreversed)**, to fix this problem.

Reverse DNS - IP Address to Name (db.yourIPreversed)

This is the fourth page of a five part tutorial on setting up **Name Service: DNS and BIND**. The full tutorial includes

- **Name Service: DNS and BIND**
- **Installing and Configuring BIND**
- **Creating Forward DNS Records**
- **Creating Reverse DNS Records**
- **Testing and Troubleshooting BIND**

db.yourIPreversed

`/etc/bind/db.yourIPreversed` is the file responsible for reverse DNS, finding a host's domain name from its IP address. This is the `db.yourIPreversed` file we referred to in the /etc/bind/named.conf.local file two sections ago. It needs to be manually created.

The first parts, the TTL, SOA record, and the nameserver should be set up as in **Forward DNS - Name to IP Address (db.yourdomain)**. Look back to that page for a refresher. You should be able to copy it, and then change the places that specify the domain. In mine, references to `raptor.loc.` needed to be changed to `1.168.192.in-addr.arpa.`

```
$TTL    24h

1.168.192.in-addr.arpa.   IN  SOA   eyrie.raptor.loc.   root.raptor.loc (
        2007062800 ; serial number
        3h         ; refresh time
        30m        ; retry time
        7d         ; expire time
        3h         ; negative caching ttl
)

# Nameservers
1.168.192.in-addr.arpa.  IN  NS  192.168.1.254
```

PTR Records

Instead of address records, however, the information for getting hostnames from IP addresses needs to be specified. When DNS looks up a host name by IP address, it reverses the order of the octets and appends `in-addr.arp` as the IP "hostname". This record needs to store those records and point to what their real domain names are. These are called the `PTR` records. The format for these is

```
<IP with order reversed>.in-addr.arpa. IN PTR <full domain name>.
```

My full `/etc/bind/db.1.168.192` file is below.

```
$TTL    24h

1.168.192.in-addr.arpa.    IN  SOA    eyrie.raptor.loc.    root.raptor.loc (
        2007062800 ; serial number
        3h         ; refresh time
        30m        ; retry time
        7d         ; expire time
        3h         ; negative caching ttl
)

; Nameservers
1.168.192.in-addr.arpa.   IN  NS   192.168.1.254.

; Hosts
254.1.168.192.in-addr.arpa.      IN PTR  eyrie.raptor.loc.
200.1.168.192.in-addr.arpa.      IN PTR  gyrfalcon.raptor.loc.
201.1.168.192.in-addr.arpa.      IN PTR  kestrel.raptor.loc.
202.1.168.192.in-addr.arpa.      IN PTR  owl.raptor.loc.
203.1.168.192.in-addr.arpa.      IN PTR  goshawk.raptor.loc.
204.1.168.192.in-addr.arpa.      IN PTR  osprey.raptor.loc.
205.1.168.192.in-addr.arpa.      IN PTR  peregrine.raptor.loc.
206.1.168.192.in-addr.arpa.      IN PTR  kite.raptor.loc.
207.1.168.192.in-addr.arpa.      IN PTR  eagle.raptor.loc.
208.1.168.192.in-addr.arpa.      IN PTR  harrier.raptor.loc.
```

Restarting Bind

Bind is now ready to be restarted and tested! Issue

```
/etc/init.d/bind9 restart
```

and continue on to **Testing and Troubleshooting BIND**.

Testing and Troubleshooting BIND

This is the last page of a five part tutorial on setting up **Name Service: DNS and BIND**. The full tutorial includes

- **Name Service: DNS and BIND**
- **Installing and Configuring BIND**
- **Creating Forward DNS Records**
- **Creating Reverse DNS Records**
- **Testing and Troubleshooting BIND**

Bind Won't Restart

I've only see this happen once. Bind was just installed in the usual manner but wouldn't restart. I was using `rncd reload`, which just tells the daemon to reload the configuration files.

```
metaserver:/etc/bind# rndc reload
rndc: connection to remote host closed
This may indicate that
* the remote server is using an older version of the command protocol,
* this host is not authorized to connect,
* the clocks are not syncronized, or
* the key is invalid.
```

This happened many times, even after uninstalling and reinstalling Bind. Eventually I just restarted the machine, and after that, it worked fine. I'm not sure what caused it, but rebooting the machine solved the problem.

/var/log/syslog Errors

Bind should have just been restarted with `/etc/init.d/bind9 restart`. The next step is to check `/var/log/syslog` for errors that Bind will continue in the presence of. Below are a few easy slip ups that can stop the set up from functioning correctly. After making any changes, be sure to restart Bind again.

Missing Period in a Zone File

There are lots of shortcuts I didn't have the space to cover earlier. One of these is that any IP address on the right side of a line or any full domain name that doesn't end with a period has the zone added to it. Below is what happens when using "192.168.1.254" instead of "192.168.1.254." in part of the nameserver declaration. Because the first one doesn't have a period at the end, the zone `raptor.loc` is appended to it, which can't resolve correctly.

```
Jul  3 19:43:18 eyrie named[2961]: zone raptor.loc/IN: NS '192.168.1.254.raptor.loc' has no address records (A
```

Filename Typo

The names of the zone files you create in `/etc/bind` need to match those you specified in `/etc/bind/named.conf.local`. If they don't, you'll get an error like this.

```
Jul  3 19:22:42 eyrie named[2847]: zone 1.168.192.in-addr.arp/IN: loading from master file
  /etc/bind/db.1.169.192 failed: file not found
```

Ignoring out-of-zone-data and 0 SOA/NS Records for Reverse DNS?

This one's a little more cryptic than the other errors.

```
Jul  3 19:49:28 eyrie named[3028]: /etc/bind/db.1.168.192:3: ignoring out-of-zone data (raptor.loc)
Jul  3 19:49:28 eyrie named[3028]: /etc/bind/db.1.168.192:12: ignoring out-of-zone data (raptor.loc)
Jul  3 19:49:28 eyrie named[3028]: zone 1.168.192.in-addr.arp/IN: has 0 SOA records
Jul  3 19:49:28 eyrie named[3028]: zone 1.168.192.in-addr.arp/IN: has no NS records
```

The clue comes in with the two different zones mentioned in the error - `1.168.192` and `raptor.loc`. Neither one should know anything about the other. If one references the other, it's probably because the important parts (the domain name specification) weren't specified after **copying the forward DNS records to the reverse DNS records**. Oops!

Turning Logging On/Off

For hunting down some of the problems when turning DNS loose for the first time, it's often handy to have logging turned on. To have DNS log all queries to `/var/log/syslog`, use

```
rndc querylog
```

Having `querylog` on will result in lines like the following being written. To turn it off, issue the same command again.

```
Jul  3 21:25:40 eyrie named[3189]: client 192.168.1.200#32793: query: eyrie.raptor.loc IN A +
Jul  3 21:25:41 eyrie named[3189]: client 192.168.1.200#32793: query: gyrfalcon.raptor.loc IN A +
```

Testing

All right, finally time to test it out! The easiest way to test out Bind is with the `host` command, followed by an IP address or name. Using an IP tests out reverse DNS, while using a name tests out forward DNS. It's a good habit to test it out on both the server and one of the clients as well. Below are a few common problem areas.

Record not found, server failure

```
eyrie:~# host eyrie
eyrie A record not found, server failure
```

This indicates that the correct nameserver isn't being used. For machines with static IPs, this means lines needed to be added to `/etc/resolv.conf` to tell it to query the right nameserver. Adding these next two lines does the trick. The first tells it to append your domain name to partial queries (in my case, so querying for `harrier` becomes a query for `harrier.raptor.loc`), and the second tells it to ask the proper nameserver. This should go before any other lines in your file.

```
search <your domain>
nameserver <nameserver IP>
```

For dynamic IPs, the domain and the nameserver IP address need to be specified in **/etc/dhcp3/dhcpd.conf options on the DHCP server**.

Does not exist, try again

```
eyrie:~# host eyrie
eyrie does not exist, try again
```

This is part of the same problem as above, specific to the `search <your domain>` line in `/etc/resolv.conf`. Edit that file manually for machines with static IP addresses, or see **DHCP Server** to change it for machines being handed IPs dynamically.

```
eyrie:~# host eyrie
eyrie.raptor.loc does not exist, try again
```

Other half of the same problem from above. This time, since `search` has been specified, the domain is appended as it should be, but the correct nameserver isn't being contacted.

Record query refused

```
eyrie:~# host eyrie
eyrie.raptor.loc A record query refused
```

After getting this error, you'll see an entry in `/var/log/syslog`:

```
eyrie:~# tail /var/log/daemon.log
Jul  3 21:02:22 eyrie named[3095]: client X.X.X.X#32790: query 'eyrie.raptor.loc/A/IN' denied
```

This indicates a problem with an `allow-query { }` option in `/etc/bind/named.conf.options`. The field can take arguments in various forms, including IP addresses and CIDR masks. (See `man named.conf` for options.) These need to be separated with semi-colons and the list ended with a semi-colon as well.

Does not exist (Authoritative answer)

```
gyrfalcon:~# host eyrie
eyrie.raptor.loc does not exist (Authoritative answer)
```

At least it's sure about it! Actually the `authoritative answer` bit means that the correct nameserver is being asked, which means that the correct nameserver is being asked. Asking for a hostname and receiving this answer indicates a problem with forward DNS; asking for an IP and receiving it means a problem with reverse DNS. If the zone file is set up correctly, the serial number has been incremented when making changes, and Bind has been restarted after any changes, then this might be a missing period issue in the forward/reverse DNS file. If you can do something crazy like this

```
eyrie:/etc/bind# host gyrfalcon.raptor.loc.raptor.loc
gyrfalcon.raptor.loc.raptor.loc A       192.168.1.200
```

then it's definitely a missing period in `/etc/bind/db.yourdomain` (for forward DNS) or `db.yourIPreversed` (for reverse DNS).

Success!

```
kwanous@gyrfalcon:~$ host peregrine
peregrine.raptor.loc    A       192.168.1.205

kwanous@gyrfalcon:~$ host 192.168.1.205
Name: peregrine.raptor.loc
Address: 192.168.1.205
```

User Authentication: LDAP

About LDAP

Lightweight Directory Access Protocol (LDAP) is a network-based authentication system, similar to Active Directory or Kerberos. It is used in order to manage users in one centralized place rather than having to create a user account for each person on every single computer in a network or cluster. When installing and configuring LDAP, at least one computer is set up as the LDAP server. This is the computer that controls the configuration of LDAP for all the LDAP clients. (It is possible to have multiple LDAP servers assisting with the load, in which case one server acts as the master and the other as secondaries using slurpd. However, that is beyond the scope of this tutorial.) Clients are computers that use the authentication provided by LDAP. The server itself can also be a client, though it does not need to be. (In cases where LDAP users should not be logging into the LDAP server, the server should not be set up to authenticate using LDAP. This is generally the case.)

There are three values that need to be decided ahead of setting up LDAP. These are the Uniform Resource Identifier (URI) for the network, the administrative account, and the administrative password.

The base URI will be used in several prompts, including asking for the DNS domain name. If you have a domain name, that can be used. Otherwise, something .loc or .local may be preferred. In the examples given, raptor.loc will be used. In accordance with the X509 specification format, this will be specified with dc's (domain components). Raptor.loc would be given as `dc=raptor,dc=loc`. When specifying a user, cn (for common name) is appended to the front. kwanous@raptor.loc would be `cn=kwanous,dc=raptor,dc=loc`. kristina.wanous@raptor.loc would be `cn=kristina,cn=wanous,dc=raptor,dc=loc`.)

The administrative (or LDAP root) account needs to be decided on ahead of time because different defaults will be used in different packages. When installing the LDAP configuration package, migrationtools, the default for the account will be admin, but other packages will use manager as the default. Pay careful attention to this! Not matching the accounts will result in odd, non-functioning behavior. The adminstrative password is the password to this account. This also needs to match, of course.

As a side note, one alternative to LDAP is NIS (**Network Information Service (http://en.wikipedia.org/wiki/Network_Information_Service)**). However, unlike LDAP, NIS passes passwords as clear text over the network. This is far less secure than LDAP.

Setting up LDAP

The LDAP tutorial is broken into multiple parts. First, an LDAP server must be installed and populated with data. Then clients need to be configured to communicate with the LDAP server.

- **LDAP Server**
- **LDAP Client**
- **Using LDAP**
- **Troubleshooting LDAP**

References

- Chapter 2 LDAP Concepts & Overview (http://www.zytrax.com/books/ldap/ch2/)
- LDAP Overview (http://www.directory-applications.com/ldap3_files/frame.htm)
- Using OpenLDAP on Debian to serve System Users (http://aqua.subnet.at/~max/ldap)
- LDAP Linux HOWTO (http://en.tldp.org/HOWTO/LDAP-HOWTO/)

LDAP Server

This is the second page of a five-part tutorial on LDAP. The full tutorial includes

- **User Authentication: LDAP**
- **LDAP Server**
- **LDAP Client**
- **Using LDAP**
- **Troubleshooting LDAP**

LDAP Server Overview

The steps involved in setting up an LDAP server consist of

- installing slapd
- installing migrationtools, configuring the migrationtools script, and running the script

slapd

Slapd is the LDAP server daemon - it's what runs in the background and answers when other computers ask for LDAP authentication. Install slapd with apt-get. The command should just be `apt-get install slapd`. The only prompt for this package will be to enter the adminstrative password.

Once it's installed, verify that slapd is running with the command `ps aux | grep slapd`. You should see something like this:

```
openldap  9741  0.0  1.0  14456  2720 ?        Ssl  08:21   0:00 /usr/sbin/slapd -g openldap -u openldap
```

However, the initial configuration does not set up everything necessarily, and it will need to be reconfigured. Use `dpkg-reconfigure slapd`.

Omit OpenLDAP server configuration?

- Choose no.

DNS domain name:

- *You'll use this value again later on. I entered `raptor.loc` for dc=raptor,dc=loc.*

Name of your organization:

- Mine is the University of Northern Iowa.

Admin password:

- This is resetting the password you entered earlier. If you changed your mind about it, now's a good time to switch passwords.

Database backend to use:

- Choose BDB, for Berkley Database. This is the preferred method.

Do you want your database to be removed when slapd is purged?

- Keep the default of no.

Move old database?

- Keep the default of yes.

Allow LDAPv2 protocol?

- Since I'm reconfiguring this from scratch, I will not be using any of the old protocols. I kept the default of no.

Now it's time to start up slapd, if it isn't already running. Try `ps aux | grep slapd`, and if you only see your command running, it isn't running...

```
eyrie:~# ps aux | grep slapd
root      4792  0.0  0.0   1784   524 pts/0    R+   13:56   0:00 grep slapd
```

Start it with

```
/etc/init.d/slapd start
```

and then issue the above command again.

migrationtools

Slapd should be running at this point to continue. Next, the migrationtools package will be used to take care of moving everything over from the default Unix password-based authentication to LDAP-based. After issuing the command `apt-get install migrationtools`, everything will be dumped into the directory `/usr/share/migrationtools`. Installing migrationtools will also install ldap-utils.

Moving into that directory with the command `cd /usr/share/migrationtools` and viewing the code will show all the files commonly used by the Linux/Unix environment that LDAP will need to integrate with. These include items like group, hosts, passwd, fstab, et cetera. The .pl files are Perl scripts; the .ph is a Perl header. The files should look something like this:

```
peregrine:/usr/share/migrationtools# ls
migrate_aliases.pl              migrate_group.pl
migrate_all_netinfo_offline.sh  migrate_hosts.pl
migrate_all_netinfo_online.sh   migrate_netgroup_byhost.pl
migrate_all_nis_offline.sh      migrate_netgroup_byuser.pl
migrate_all_nis_online.sh       migrate_netgroup.pl
migrate_all_nisplus_offline.sh  migrate_networks.pl
migrate_all_nisplus_online.sh   migrate_passwd.pl
migrate_all_offline.sh          migrate_profile.pl
migrate_all_online.sh           migrate_protocols.pl
migrate_automount.pl            migrate_rpc.pl
migrate_base.pl                 migrate_services.pl
migrate_common.ph               migrate_slapd_conf.pl
migrate_fstab.pl
```

The migrate_all files will run all the scripts, ensuring that they don't need to be run individually. migrate_all_online should be used if the LDAP system is running (which it should be, if slapd is installed), or migrate_all_offline if it is not. However, before running this script, the migrate_common.ph file should be modified.

migrate_common.ph

Migrate_common has all the flags and settings important for configuring the scripts. Most of the defaults should be fine, with a few exceptions. PADL, the implementer of LDAP, by default inserts its own domain name in some areas if the script is not changed. A quick search in a text editor like vi or nano should help locate any instances of "padl" that need to be changed. These include (but may not be limited to):

```
# Default DNS domain
$DEFAULT_MAIL_DOMAIN = "padl.com";

# Default base
$DEFAULT_BASE = "dc=padl,dc=com";
```

Use the URI name for the cluster, in DNS form for $DEFAULT_MAIL_DOMAIN and in X509 form for $DEFAULT_BASE. I'll use the name `dc=raptor,dc=loc`. These are the values I used:

- $DEFAULT_MAIL_DOMAIN = "raptor.loc";
- $DEFAULT_BASE = "dc=raptor,dc=loc";

migrate_all_online.sh

Now we're ready to run the setup script. Since slapd is already running, we'll use migrate_all_online.sh (otherwise, it would be migrate_all_offline.sh). Run this with the command `./migrate_all_online.sh` and follow along with the prompts below:

Enter the X.500 naming context you wish to import into: [dc=raptor,dc=loc]

- The value you entered for $DEFAULT_BASE in migrate_common.ph should be here. If not, specify it.

Enter the hostname of your LDAP server [ldap]:

- This should be the hostname of the machine you're currently setting up as the LDAP server. I'm on the machine eyrie, so I typed `eyrie`.

Enter the manager DN: [cn=admin,dc=raptor,dc=loc]:

- This is the adminstrative user name for LDAP. The default is fine, otherwise you can change it. Even if you keep the default, you'll need to remember the name of this account!

Enter the credentials to bind with:

- This is the adminstrative account password, as entered for slapd earlier.

Do you wish to generate a DUAConfigProfile [yes|no]?

- Type no.

After this prompt, the script should begin to go to town and finish with

```
/usr/bin/ldapadd: succeeded
```

Authentication Failure?

If you get a message about authentication failing, you need to run `dpkg-configure slapd` (see above). If you have to cancel the script, or in other special circumstances, the script will exit with a message like this:

```
ldap_add: Already exists (68)
/usr/bin/ldapadd: returned non-zero exit status: saving failed LDIF to /tmp/nis.10513.ldif
```

To get around this, the configuration needs to be set to continue in the presence of errors. Take the temporary file specific to your system and continue running the rest of the script with this command:

```
ldapadd -x -c -D "<adminstrative account>" -f <temp file> -W
```

- -x specifies to use simple binding credentials
- -c specifies to continue the script in the presence of errors
- -D specifies to use the following domain
- administrative account - Mine was `cn=admin,dc=raptor,dc=loc` - yours should be specific to your domain, as set up earlier
- -f means using the following file:
- temp file - This the file specified in the error. /tmp/nis.10513.ldif is the file specific to the system I'm running on - yours will probably be something else.
- -W tells it to prompt for the password

After you enter this, you'll be prompted for the LDAP password again. Enter it and continue. If the script exits without an error message, you're ready to continue.

Sanity Check

At this point, everything *should* be up and running on the LDAP server. To do a sanity check, check if you can find the user information for a user already on your system. Use the line

- ldapsearch -x uid=<an existing ID on the system>

or, for more information, do it as the root LDAP user, specifiying your root user name and domain after the -D. `-W` specifies that you'll be prompted for the LDAP administrative password.

- ldapsearch -x uid=<an existing ID on the system> -D "<your administrative account>" -W

I searched for `kwanous`. You should get results like this:

```
eyrie:/usr/share/migrationtools# ldapsearch -x uid=kwanous -D "cn=admin,dc=raptor,dc=loc" -W
Enter LDAP Password:
# extended LDIF
#
# LDAPv3
# base <> with scope subtree
# filter: uid=kwanous
# requesting: ALL
#

# kwanous, People, raptor.loc
dn: uid=kwanous,ou=People,dc=raptor,dc=loc
uid: kwanous
cn: KWanous
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword:: e2NyeXB0fTdVN1cxRWxCL0FtY0E=
shadowLastChange: 13689
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 1000
gidNumber: 1000
homeDirectory: /home/kwanous
gecos: KWanous,,,

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

The reason you're able to search for an existing user is because the migration script put all of the current users into LDAP. You're still able to become that user (`su - kwanous`) like normal.

LDAP Client

This is the third page of a five-part tutorial on LDAP. The full tutorial includes

- **User Authentication: LDAP**
- **LDAP Server**
- **LDAP Client**
- **Using LDAP**
- **Troubleshooting LDAP**

LDAP Client Overview

The steps involved in setting up an LDAP client consist of

- configuring NSS
- configuring PAM

NSS

libnss-ldap

Libnss-ldap is a Name Switch Service (NSS) module that allows LDAP to authenticate users. First, `apt-get install libnss-ldap`.

LDAP server Uniform Resource Identifier:

- You'll want to enter the IP address of the computer you set up to act as the LDAP server. If you're setting up a client on the same machine as the server, that be the default, `ldap://127.0.0.1`, for the localhost. In my case, the IP is 192.168.1.254, so I entered `ldap://192.168.1.254`.

Distinguished name of the search base:

- This is the same URI as specified when setting up the client. Mine was `dc=raptor,dc=loc`.

LDAP version to use:

- Keep the default of 3.

LDAP account for root:

- This is the account set up earlier. If you used the default earlier, cn was equal to admin, not manager. Mine is `cn=admin,dc=raptor,dc=loc`.

LDAP root account password:

- Enter the same password as setup for LDAP root.

If you later run `dpkg-reconfigure libnss-ldap`, you'll get a few more options, but all of these should keep their default values. The extra options are

Does the LDAP database require login?

- Keep the default of No.

Special LDAP privileges for root?

- Keep the default of Yes.

Make the configuration file readable/writeable by its owner only?

- Keep the default of No.

ldap.conf

Finally, the file `/etc/ldap/ldap.conf` needs to be configured. This file specifies where the computer running the LDAP services can be reached at. It should currently contain the lines

```
#BASE   dc=example, dc=com
#URI    ldap://ldap.example.com ldap://ldap-master.example.com:666
```

The BASE needs to be replaced with the values for your set up that you specified when configuring the **LDAP Server**. URI needs to be changed to point to that specific computer via its IP or hostname. IP is preferable since if there are problems with DNS, it will still function. Both these lines need to be uncommented. For instance, my values are

```
BASE    dc=raptor, dc=loc
URI     ldap://192.168.1.254
```

nsswitch.conf

/etc/nsswitch.conf is the file responsible for the order of where files should be checked to authenticate a user. Right now it should have values something like these, as well as many others:

```
passwd:         compat
group:          compat
shadow:         compat
```

The `compat` specifies that the system should first and only check the compat files - the default Unix files. So, for passwd check /etc/password, for group check /etc/group, and for shadow check /etc/shadow. We need to change these so that LDAP is checked as well. We'll also change `compat` to `files`, which is another way of saying the default files. Change the nsswitch.conf values for the following:

```
passwd:         files ldap
group:          files ldap
shadow:         files ldap
```

Notice that `files` is first. This gives precendence to the local users on the machine before checking LDAP. This is especially important for when the LDAP server may be down.

PAM

PAM, or Pluggable Authentication Module, provides the backend for nsswitch.conf to communicate with other authentication implementations such as LDAP. This isn't necessary on the LDAP server (unless the LDAP server is also an LDAP client), because the LDAP server isn't authenticating LDAP users to that machine. PAM is responsible for accepting the password for a user when they log in as well as changing the password. Without PAM configured correctly, you'll see errors like the following.

```
gyrfalcon:~# passwd mycooluser
passwd: User not known to the underlying authentication module
passwd: password unchanged
```

Your users also won't be able to log in. Kind of a problem!

libpam-ldap

These are the modules to allow PAM to talk to LDAP. Install with `apt-get install libpam-ldap`. For most prompts, keep the default settings.

Make local root Database admin.

- Keep the default of Yes.

Database requires logging in.

- Keep the default of No.

Root login account

- Change this to the root user at your domain. Mine is `cn=admin,dc=raptor,dc=loc`.

Root login password

- Put in the password for the account you used in the previous step (the "LDAP root password").

PAM Files

Now that the files for PAM to talk to LDAP are in place, you'll need to update the PAM files themselves. All four are located in `/etc/pam.d/`. These next couple changes are important - if done incorrectly, they can make your system unbootable. It's a good idea to make a backup at this point, such as running `rsync -plarv /etc/pam.d/ /etc/pam.d.orig/`.

common-account

This is responsible for accounts - who is and who is not allowed on the system. The file should currently consist of a line like this:

```
account required        pam_unix.so
```

This line specifies that the system should check for an account with the default UNIX files. You'll want to change it to these two lines:

```
account sufficient      pam_ldap.so
account required        pam_unix.so try_first_pass
```

With this configuration, the system will first try to verify an account with ldap (using `pam_ldap.so`). If it finds one, that is sufficient for the account to be verified - it doesn't need to have an entry on the local machine as well. However, if that fails, the account must exist on the local machine (`pam_unix.so`). `try_first_pass` specifies that the password originally entered by the user should be checked against pam_unix.so after it fails against pam_ldap.so - this prevents the user from having to enter his/her password twice.

common-auth

The file should currently consist of a line like this:

```
auth    required        pam_unix.so nullok_secure
```

We want to change this file also to look at LDAP first. `nullok_secure` specifies that logging in without a password is all right if authentication is accomplished another way, such as with ssh keys. Add the LDAP line before the existing line, and add `try_first_pass` to the second line, like this:

```
auth    sufficient      pam_ldap.so
auth    required        pam_unix.so nullok_secure try_first_pass
```

common-password

These are the files involved with changing and manipulating password tokens. The only uncommented line in the file should look like this:

```
password   required   pam_unix.so nullok obscure min=4 max=8 md5
```

We want to add the LDAP line before it:

```
password   sufficient   pam_ldap.so
password   required   pam_unix.so nullok obscure min=4 max=8 md5
```

This file doesn't need the tag `try_first_pass`.

common-session

`common-session` refers to the files responsible for what a user can/cannot do - it controls the limits of the

environment. For instance, it can be used to limit how many processes a user can create. It is generally used to diminish users' capability on the system. The file should look like this:

```
session required        pam_unix.so
```

Again, we need to add the LDAP line, so the file looks like this:

```
session sufficient      pam_ldap.so
session required        pam_unix.so
```

Sanity Check

At this point, everything *should* be up and connected to the LDAP server. Without having installed `ldap-utils`, you won't have `ldapsearch`, but that's fine unless you specifically want it. (If you do want it, `apt-get install ldap-utils`.)

You should be able to change users passwords as root, as the user, and also be able to become

```
su - mycooluser
```

and id

```
id mycooluser
```

your LDAP users. Congratulations!

Using LDAP

This is the fourth page of a five-part tutorial on LDAP. The full tutorial includes

- **User Authentication: LDAP**
- **LDAP Server**
- **LDAP Client**
- **Using LDAP**
- **Troubleshooting LDAP**

Using LDAP

For any of these tutorials, you'll need to have ldap-utils installed on whatever machine you're trying to adminster LDAP with. This is done with an easy `apt-get install ldap-utils`.

There are a few common tasks you'll probably become quite familiar with while using LDAP:

- **Changing an entry in the LDAP database**
- **Removing an entry from the LDAP database**
- **Adding users to the LDAP database**
- **Searching the LDAP database**

Changing an Entry

To update one of the entries, use the utility `ldapmodify`. Ldapmodify can be used a number of different ways: from the command line, interactively, or taking data from a file. In most small changes, it's easiest to use the interactive mode.

If I do an `ldapsearch -x uid=kwanous`, I get back these results:

```
# kwanous, People, raptor.loc
dn: uid=kwanous,ou=People,dc=raptor,dc=loc
uid: kwanous
cn: KWanous
...snipped...
homeDirectory: /home/kwanous
gecos: KWanous,,,
```

Unfortunately, the homeDirectory is incorrect. It should use my **NFS mount**, not the local hard drive. I'm going to walk through correcting this. This same process of starting up LDAP and modifying a field will work for other fields, too.

Ldapmodify

First, start up ldapmodify with the credentials to bind with. In my case, this is

```
ldapmodify -x -D cn=admin,dc=raptor,dc=loc -W
```

- `-x` specifies to use simple authentication
- `-D` is used to specify the LDAP administrative user's credentials

- -W will cause you to be prompted for the password for the LDAP administrative user

If you enter the password correctly, you'll see an error message like below.

```
gyrfalcon:~# ldapmodify -x -D cn=admin,dc=raptor,dc=loc -W
Enter LDAP Password:
ldap_bind: Invalid credentials (49)
```

If you enter it correctly, though, you'll be greeted with a blank line. This confused me for quite a while the first time, but it just means that ldapmodify is in interactive mode and ready to have you update an entry.

First, specify which entry will be modified. This is taken from the second line, after the comments, of the LDAP search. In this case, I specified

```
dn: uid=kwanous,ou=People,dc=raptor,dc=loc
```

Once entering it, hit enter to get to a new line. Next, LDAP needs to be told what to do with this entry. I want to modify it, so I entered

```
changetype: modify
```

After that, LDAP needs to know which field to modify. I need to change the home directory, so on a new line, I entered

```
replace: homeDirectory
```

Finally, LDAP takes in the new value for the field. This is specified with the field name and then the new value (again on a new line). In my case, that means entering

```
homeDirectory: /shared/home/kwanous
```

Once you've entered the changes, it's time to exit ldapmodify and flush them. On the other hand, you can make more changes to that same entry by adding a hyphen on a new line and then entering the replace field and new values again. To exit ldapmodify and make the changes, press the keys CTRL and D at the same time.

Below is the full script for my changes:

```
gyrfalcon:~# ldapmodify -x -D cn=admin,dc=raptor,dc=loc -W
Enter LDAP Password:
dn: uid=kwanous,ou=People,dc=raptor,dc=loc
changetype: modify
replace: homeDirectory
homeDirectory: /shared/home/kwanous
-
modifying entry "uid=kwanous,ou=People,dc=raptor,dc=loc"
```

If I do another ldapsearch for kwanous, I see that the home directory has been changed:

```
dn: uid=kwanous,ou=People,dc=raptor,dc=loc
uid: kwanous
cn: KWanous
...snipped...
gecos: KWanous,,,
homeDirectory: /shared/home/kwanous
```

References

- **Chapter 4 The ldapmodify Tool (http://docs.sun.com/source/816-6400-10/lmodify.html)**

Removing an Entry

`ldapmodify` can also be used to delete an entry by specifying `changetype: delete` instead of `modify` or `add`.

A shorter way uses the utility `ldapdelete`. Here, the LDAP entry to delete is specified on the command line. If you ran the `migrate_all_online.sh` script in the **LDAP Server** tutorial, all of the users from `/etc/passwd` now have corresponding entries in the LDAP system, including root. It's a good idea to take the root account out.

To delete the root account user from LDAP, run

```
ldapdelete -x -D "cn=admin,dc=your,dc=cluster" -W
"uid=root,ou=People,dc=your,dc=cluster"
```

- `-x` specifies to use simple credentials
- `-D` is the LDAP administrative account. Be sure to change `dc=your,dc=cluster` to your actual domain name.
- `-W` will cause it to prompt for the administrative password
- `uid=root,ou=People,dc=your,dc=cluster` is the account to delete. Again, be sure to change this value to your actual domain name.

By replacing "root" in the above command to the one you want, you can delete any user from LDAP.

Adding Users to LDAP

Most of the time, many users need to be added at once. This is described below.

To add just one user, the utility `ldapmodify` is often more convenient to use. This uses the same process as changing an entry with `ldapmodify`, except the `changetype` should be `add`. See the two above sections for more information.

Adding Users in Bulk

First, create a list of new user names with one on each line in `/tmp/users`.

To add a user, you'll need to create the stats for them. The easiest way to do this is to pull off the information from an existing user and then change it for the new user. To get all the basic information and output it to a file, use the command

```
ldapsearch -LLL -x uid=<existing username> >> /tmp/template
```

Go through `/etc/template` and change all out instances of the old user with variables that will be replaced (USERID, UIDNUM, and GIDNUM). After editing, my file is shown below. Your `dn` should be specific to your domain, as should your homeDirectory!

```
dn: uid=USERID,ou=People,dc=raptor,dc=loc
uid: USERID
cn: USERID
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: UIDNUM
gidNumber: GIDNUM
homeDirectory: /shared/home/USERID
gecos: USERID,,,
```

I'm going to be adding all my users to GIDNUM=100, which is the group `users`, and I'll be starting with the userid (UID) 1001. You can modify the following script to use something different. Paste the following script into a file and make it executable (chmod o+x <filename>), then run it.

```
#!/bin/bash
# Short little LDAP-file creating script

GIDNUM=100
UIDNUM=1001

for x in `cat /tmp/users`
  do
    sed "s/USERID/$x/g" /tmp/template | sed "s/UIDNUM/$UIDNUM/g" | sed "s/GIDNUM/$GIDNUM/g" > /tmp/$x.ldif
    UIDNUM=`expr $UIDNUM + 1`
  done
```

It will generate one file for each new user name in `/tmp/users`, and replace the variables with correct values from the file with the list of user names. To see the files after they've been made, issue

```
ls /tmp/*.ldif
```

It's always a good idea to check a few one of them out and make sure the variables were replaced as you were expecting them to. Next, all of these files need to be added to the LDAP database. Again, copy and paste this script and make it executable. Change the part that says "<your credentials here"> to your fully qualified administrator user name (such as `"cn=admin,dc=raptor,dc=loc"`). With the current configuration, you'll be prompted for your password each time. If you'd rather, you can change

```
-W
```

to

```
-w "<your password here>"
```

Finally, run it.

```
#!/bin/bash
# Short little ldapadd script

for x in `ls /tmp/*.ldif`
  do
    echo "Adding user file $x"
    ldapadd -x -D "<your credentials here>" -W -f $x
  done
```

Sanity Check

You should see an output like the following for each user as the script is running:

```
Adding the user file /tmp/mycoolnewuser.ldif
Enter LDAP Password:
adding new entry "uid=mycoolnewuser,ou=People,dc=raptor,dc=loc"
```

After the script finishes, make sure you can search for the user -

```
ldapsearch -x uid=<new username here>
```

and then from a machine acting as an LDAP client, id the user -

```
id <new username>
```

and become the user -

```
su - <new username>
```

Home Directories and Passwords

At this point, the users have been created, but none of them have home directories or passwords. (Since they don't have passwords, they won't be able to log in.)

First, you'll need to set a password for each user - from a machine that's an **LDAP Client**! - with

```
passwd <username>
```

If all goes well, you'll see

```
gyrfalcon:/etc/pam.d# passwd mycooluser
New password:
Re-enter new password:
LDAP password information changed for mycooluser
passwd: password updated successfully
```

Next, to add the home directories... there are two options. First, one line can be added to PAM, the authentication mechanisms, to take care of this for you. On the other hand, you can manually add them yourself.

Permanent Solution

To have home directories automatically created for users the first time they log in, edit `/etc/pam.d/common-session`. Above the existing lines, add this

```
session required pam_mkhomedir.so skel=/etc/skel/ umask=0022
```

Replace `/etc/skel/` with the location of your skeleton files, if you have them somewhere else. After making this change, users will have their home directories created for them from now on.

One-time Solution

To create the home directories manually, but in one fell swoop, use the following script. You'll need to create the file, make it executable, and again change it for your values before running it. Right now, I have the home directories being created on my **NFS mount**.

```
#!/bin/bash
# Short little script to add home directories

for x in `cat /tmp/users`
  do
    rsync -plarv /etc/skel/ /shared/home/$x/
    chown -R $x:100 /shared/home/$x/
  done
```

Again, keep in mind that this is a one-time fix. If you want a more permanent solution, see the section above.

Searching

`Ldapsearch` is the utility for searching the LDAP database. (It comes with the `ldap-utils` package.) It's a powerful and flexible interface to the LDAP database.

Ldapsearch supports a few different filters, but before we get into that, it's important to recognize there are two different ways of accessing the LDAP database:

- anonymously: `ldapsearch -x`
- with admin credentials, which will prompt for the password: `ldapsearch -x -D "<adminstrative account>" -W`
    - the administrative account should be something like `"cn=admin,dc=raptor,dc=loc"`

Administrative credentials will give slightly more information, such as showing the encrypted password field. I'll just be using anonymous authentication, but administrative authentication can be used for any of these examples, too.

Presence Filters

Presence filters use a wildcard character (*) to just see if something exists in the database. For instance,

```
ldapsearch -x "objectClass=*"
```

will search for every entry with any corresponding `objectClass`. To see everything that has an `ipServicePort` entry, use

```
ldapsearch -x ipServicePort=*
```

Exact Filers

Exact filters check to see if a value matches exactly. This can be used to search for a specific user,

```
ldapsearch -x "uid=kwanous"
```

to see the top level entry (the organization),

```
ldapsearch -x "objectClass=organization"
```

or even see a list of every account in the system, such as this line

```
ldapsearch -x "objectClass=account" | grep "dn: "
```

Substring Matching

Substrings can also be searched for with `ldapsearch`. For instance,

```
ldapsearch -x "uid=t*"
```

returns all the entries whose `uid` field starts with `t`. These might include `tommy`, `tammy`, `tmitchel`, etcetera. Using

```
ldapsearch -x "uid=*t*"
```

would find any entries with a t somewhere in the `uid` field. (A few you'll find for sure with this search are `root`, `gnats`, and `statd`.)

Approximate Filers

Approximate filters are in the implementation, but the specification of these are very vague. Your mileage may vary! Use `~=` for an approximate filter. On my system, searching with

```
ldapsearch -x "uid~=kwanos"
```

did indeed return the same results as searching for `"uid=kwanous"`.

References

- **http://www.novell.com/documentation/nas4nw/usnas4nw/nasnwenu/ldapsrch.html**

Troubleshooting LDAP

This is the last page of a five-part tutorial on LDAP. The full tutorial includes

- **User Authentication: LDAP**
- **LDAP Server**
- **LDAP Client**
- **Using LDAP**
- **Troubleshooting LDAP**

Updating after Changes

After making any change in an LDAP configuration, like the location of the LDAP server, multiple files need to be updated -

- /etc/ldap/ldap.conf
- /etc/pam_ldap.conf
- /etc/libnss-ldap.conf

Changes to the administrative password for LDAP database need to be changed at

- /etc/libnss-ldap.secret
- /etc/pam_ldap.secret

Alternatively, dpkg-reconfigure for libnss-ldap and libpam-ldap will also work. Don't run dpkg-reconfigure for slapd, though, as this will erase the previous database (**see below** if this accidentally happens).

LDAP Checks

It's often helpful to have a non-LDAP user and an LDAP user account on a client system to test out.

- First, can you log in at all as the non-LDAP user? If not, there's most likely a problem with the basic configuration of nsswitch.conf or the pam files.
- Can you do an ldap search for the LDAP user? Use `ldapsearch -x uid=<ldap user's ID>`. This will determine whether the client can connect to the ldap server at all.
  - If not, check /etc/ldap/ldap.conf
- Can you `su - <ldapuser>`, `id <ldapuser>`, and change the user's password (`passwd <ldapuser>`)? If not, then there's probably a problem with the nsswitch and pam talking to the ldap server. **Double check** the values for
  - /etc/nsswitch.conf
  - /etc/pam.d/common-account, /etc/pam.d/common-auth, /etc/pam.d/common-password, and /etc/pam.d/common-session
  - /etc/pam_ldap.conf

How to Fix an Accidental Dpkg-Reconfigure Slapd

If you run `dpkg-reconfigure slapd` on the LDAP server and go into the database configuration options, a new LDAP database will automatically be created for you and your old one purged. Yikes! Fortunately, dpkg

creates a backup of your old database in `/var/backups/`. For instance, the directory automatically created for me after a dpkg-reconfigure was `dc=raptor,dc=loc-2.3.38-1+lenny1.ldapdb`. (My internal network is raptor.loc.)

To restore your previous slapd (LDAP server) database, first stop slapd with

```
/etc/init.d/slapd stop
```

Next, move the current LDAP server database - the one that was newly created for you, that you probably didn't really want - to a different location. You'll want to keep it in case something goes wrong. The current database is stored in `/var/lib/ldap/`. Make a folder in root's home directory and copy it to root's home directory with

```
mkdir ~/ldapempty
cp /var/lib/ldap/* ~/ldapempty/
```

Once it's been copied over, you can delete the contents of `/var/lib/ldap` using

```
rm /var/lib/ldap/*
```

Then copy over the files from the backup directory to the directory you just emptied. Yours will look similar to this, but of course, replace the directory name with your actual backup directory.

```
cp /var/backups/dc\=raptor\,dc\=loc-2.3.38-1+lenny1.ldapdb/* /var/lib/ldap/
```

If you do an `ls` on `/var/lib/ldap`, you'll see that all the files are currently owned by root. Failing to change then to openldap will cause an error like this in `/var/log/syslog` when you try to start the system back up:

```
Feb  6 12:44:39 eyrie slapd[19570]: bdb_db_open: database "dc=raptor,dc=loc": alock package is unstable.
Feb  6 12:44:39 eyrie slapd[19570]: backend_startup_one: bi_db_open failed! (-1)
```

To prevent that, change all of the files to be owned by openldap using this command:

```
for x in `ls /var/lib/ldap`; do chown openldap:openldap $x; done
```

Finally, you'll want to copy your `slapd.conf` configuration file back. Again, make a copy of the new one in case something should go wrong:

```
cp /etc/ldap/slapd.conf ~
```

and then replace it with your backed up version, which will look something like this:

```
cp /var/backups/slapd-2.3.38-1+lenny1/slapd.conf /etc/ldap/slapd.conf
```

After you've done that, you should be able to start slapd back up again:

```
cp /var/backups/slapd-2.3.38-1+lenny1/slapd.conf /etc/ldap/slapd.conf

/etc/init.d/slapd start
```

It's wise to make sure the changes took affect as you expected by doing an **LDAP search** for a user account. If you can't connect to the LDAP server, double check everything started ok by tailing `/var/log/syslog`.

Creating Head Node and Worker Node Images

Setting up one worker node and then cloning the other nodes with that one's image can save a lot of time. Rather than having to set up each one for NFS, DNS, and LDAP, one machine can be set up and all the configurations are copied over when the others are imaged.

There are also a few optional steps.

Renaming Network Interfaces

The network interfaces (normally eth0, eth1, etcetera) can be renamed if you so desire. This is done through **Udev**.

Hostnames through DNS

Rather than manually editing /etc/hostname on each worker, nodes can be set to automatically assign their names based on DNS. See **Setting a Dynamic Hostname by DNS**.

Cloning

After the one machine has been set up correctly, the other worker nodes can then be cloned using that machine's image. See **Cloning Worker Nodes**.

Bash Profile Modifications

Group Modifications

By default, when Debian creates a new user, it creates a new group for that user (the group is named the same as the group name). Rather than having each user in their own group, it's smart to add them to a common `users` or similar group. To change this default, edit `/etc/adduser.conf` and change

```
USERGROUPS=yes
```

to

```
USERGROUPS=no
```

Home Directory Modifications

If users' home directories will be on the **NFS mount**, then current users' home directories need to be moved here, and new users directories also needed to be created here.

New Users

To ensure that new users have home directories created in the appropriate place on the NFS mount, edit `/etc/adduser.conf`. In this file, the `DHOME` variable needs to be set to the new location. For instance, if your mount is called `/shared` like mine is, you would change it to

```
DHOME=/shared/home
```

Current Users

The move over for current users can be accomplished by moving their old directory over to the mount (`mv /home/user /shared/` in my case) and then editing `/etc/passwd`. The second to last entry on each line for a user is their home directory. This needs to be changed to the home directory's new location. For instance, (in my case again, since my mount is called `/shared`),

```
kwanous:x:1000:1000:KWanous,,,:/home/kwanous:/bin/bash
```

becomes

```
kwanous:x:1000:1000:KWanous,,,:/shared/kwanous:/bin/bash
```

PATH Additions

If you're using part of the **NFS mount** to share software as well as home directories, then the PATH variable for everyone needs to be modified so that it looks in `/yourmountname/bin` for executable code. (I'm calling my mount `shared`, so mine is `/shared/bin`.) To do this, edit `/etc/profile`. This will take affect for all users, current and new.

Initially, the path you're concerned with looks like this:

```
if [ "`id -u`" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/games"
fi
```

Append `:/<yourmountname>/bin` for both root accounts (the first part of the if statement) and all other accounts (the else part). Also append `:/<yourmountname>/sbin` just to the root account part. My modified portion of the file is the below.

```
if [ "`id -u`" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/shared/bin:/shared/sbin"
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/games:/shared/bin"
fi
```

History

Normally the bash shell history, executed by `history` and stored in `~/.bash_history`, only stores a limited number of commands entered (500 by default on Debian). The number of commands stored can be increased, as well as the format of the history file.

There are a couple of different places to store these changes. You could put them individually in each user's `~/.bash_profile` and then also edit the one in `/etc/skel/.bash_rc` so that changes appear in each new user's `.bash_profile`. This way, users will be able to edit this part out of their bash profile if they don't like the changes. (You'll also have to add it to `/root/.bashrc` for the root user.) On the other hand, if you always want these changes to be on for all users and to not let them be able to turn them off, edit `/etc/bash.bashrc`.

More Commands

To change the number of commands that `history` keeps track off, add the following lines:

```
HISTSIZE=10000
HISTFILESIZE=''
```

`HISTSIZE` specifies the number of commands to store. `HISTFILESIZE` being set to nothing means that the history file can grow indefinitely. Since you're going to be adding far more commands now, it's also a good idea to add

```
HISTCONTROL=ignoreboth
```

This line prevents duplicate commands from being entered in history next to each other. For instance, if you typed `ls` three times in a row with no other commands in between, it would only store `ls` once. On the other hand, if you executed `ls`, `cd mystuff`, `ls`, `cd`, `ls`, then this whole set of instructions would be stored in history. None of the commands next to each other in that sequence are the same.

Timestamped History

Adding this line

```
HISTTIMEFORMAT='%a, %d %b %Y %l:%M:%S%p %z '
```

will make `history` displayed like this:

```
kwanous@gyrfalcon:~$ history | tail -n 5
  245  Fri, 20 Jul 2007 12:39:26PM -0500 history
  246  Fri, 20 Jul 2007 12:39:52PM -0500 history | tail
  247  Fri, 20 Jul 2007 12:40:03PM -0500 nano .bash_profile
  248  Fri, 20 Jul 2007 12:41:56PM -0500 exit
  249  Fri, 20 Jul 2007 12:42:06PM -0500 history | tail -n 5
```

- `%a` is the day of the week
- `%d` is the date
- `%b` is the month
- `%Y` is the year
- `%l` is the hour
- `%M` is the minutes
- `%S` is the seconds
- `%p` is AM/PM
- `%z` is the adjustment from Greenwich Mean Time

For entries in `.bash_history` prior to making this change, the current date/time will be added to them.

Password-less SSH for Users

Background

Any applications run with **MPICH: Parallel Programming** over multiple machines will need to be able to communicate amongst the machines while running on behalf of a user. This means that users need to be able to SSH into and amongst the worker nodes without being prompted for a password. This can be set up after the nodes are imaged, but it's much easier to do this ahead of time.

Setting up Password-less SSH for Current Users

This is easy to do on an **NFS-mounted file system**. Become the user and run

```
ssh-keygen
```

When prompted for a location, keep the default location. Similarly, hit Enter without entering a passphrase twice when prompted. This will create two new files in the directory `.ssh` in that user's home directory: their private key (id_rsa) and their public key (id_rsa.pub). Cd into the new `.ssh` directory.

Next, a file named `authorized_keys2` needs to be created. This file is responsible for who can SSH into this machine as this user without a password. The user's public key needs to be added to this file. Do this with

```
cat id_rsa.pub >> authorized_keys2
```

Finally, the permissions on authorized_keys2 need to be modified so that only the user can read the file. This is done with

```
chmod 600 authorized_keys2
```

At this point, as the user, you should be able to

```
ssh localhost
```

and not be prompted for a password. If you're prompted for a password, make sure the right file was copied into authorized_keys and that the permissions are correct.

Script for all Current Users

Rather than becoming each user one at a time and doing this by hand, the process can be scripted. As root, copy the below into a text file, and change the value of `homeDirs` to be correct for your setup. (`homeDirs` is the directory on your mounted file system where the user directories are stored.)

```
#!/bin/bash
# This script will create an SSH key for each existing user and create
#  an authorized_keys file with their public key.

# Directory containing user home directories
homeDirs=/shared/home

for x in `ls $homeDirs`; do
  echo Creating SSH key for $x...

  if [[ -e $homeDirs/$x/.ssh/id_rsa.pub ]]; then
    echo "$x already has a public key"
  else
    su $x -c "ssh-keygen -N \"\""
  fi

  cat $homeDirs/$x/.ssh/id_rsa.pub >> $homeDirs/$x/.ssh/authorized_keys
  chown $x:$x $homeDirs/$x/.ssh/authorized_keys
  chmod 600 $homeDirs/$x/.ssh/authorized_keys

done
```

Change the script to be executable with

```
chmod u+x <whatever you named it
```

Then, run the script with

```
./<whatever you named it
```

You'll be prompted as to where to put the files for each user. (I didn't invest the time to fully figure that out and script it, sorry!) Just hit enter and keep the default each time.

Password-Less SSH for Future Users

Ideally, this script could be set to run every time a new user account is created. I haven't yet figured out how to do that (if you have an idea, please e-mail me at kwanous <at> debianclusters <dot> org). Still, the above can be changed to an interactive script that takes a username to create an SSH key for, and you can run it whenever you create a new user.

Again, you'll need to change it to be executable and also correct the value of `homeDirs`.

```
#!/bin/bash
# Creates an SSH key and an authorized_keys file for
#  a username given as an argument.

# Directory containing user home directorys
homeDirs=/home/shared

if ! [[ "$1" ]]
then
  echo "Usage: ./sshauthhostkeygen username"
  exit 1
fi

x=`echo $1`

echo Creating SSH key for $x...

if ! id $x > /dev/null 2>&1
then
  echo $x is not a valid user.
  exit 1
fi

if ! [[ -e $homeDirs/$x ]]; then
  echo $x does not have a home directory.
  exit 1
fi

if [[ -e $homeDirs/$x/.ssh/id_rsa.pub ]]; then
  echo "$x already has a public key"
else
  su $x -c "ssh-keygen -N \"\""
fi

cat $homeDirs/$x/.ssh/id_rsa.pub >> $homeDirs/$x/.ssh/authorized_keys
chown $x:$x $homeDirs/$x/.ssh/authorized_keys
chmod 600 $homeDirs/$x/.ssh/authorized_keys
```

Preventing Logins

While this is necessary for processes running on behalf of the user, users probably shouldn't be able to do code development and other tasks on the worker nodes themselves. To prevent shell logins (other than root), just run

```
touch /etc/nologin
```

You'll still be able to SSH in as root and then `su` to that user.

Installing Compilers

The compilers typically used for scientific research and parallel programming include C, C++, and Fortran. Other possible languages to install include Python, Perl, and Ruby. I recommend installing the compilers on the head node as well as the worker nodes. Normally users won't SSH into a node other than the head node, but this way, if you ever want to take one worker out of the cluster for a while and let someone develop on it, you have that option.

C and C++ Compilers

The GNU C compiler, **gcc (http://gcc.gnu.org/)** , is installed by

```
apt-get install gcc
```

The GNU C++ compiler, g++, is installed by

```
apt-get install g++
```

Fortran

Different Fortran compilers were released different years. They are usually referred to as the `f<year>` compiler. F77 and F90 (called Fortran 77 and Fortran 90, respectively) are currently the most common Fortran compilers.

The GNU Fortran 77 compiler, **f77 (http://www.gnu.org/software/fortran/fortran.html)** , is installed with

```
apt-get install g77
```

Install the GNU Fortran 90 compiler, **gfortran (http://gcc.gnu.org/fortran/)** , is installed with

```
apt-get install gfortran
```

Python

Python 2.4 is necessary for **MPICH**. Python is installed with

```
apt-get install python2.4
```

Intel Compilers

If you have Intel processors, you might be interested in downloading Intel's optimized compilers. They're free for non-commercial use. See the **Intel Compilers** page for more information.

Intel Compilers

Intel provides free compilers for non-commercial use. These compilers are optimized for the Intel architecture. The list of Intel compilers can be found at **Intel's site (http://www.intel.com/cd/software/products/asmo-na/eng/219771.htm)** . You'll need to agree to the requirements that your project is for non-commercial purposes, and that you probably won't get full support from Intel (not like you would with a license, anyway). Once you click "Accept", you'll see a list of compilers and performance libraries.

The "Compiler Suite Professional Edition" will give you the C++ and Fortran compilers. To obtain them, you'll need to enter your e-mail, and they'll let you into the download page and send you the URL and serial number for future reference. You'll need the serial number for later, so make sure to copy this down.

Select the C++ and/or Fortran compiler and right-click on your architecture to copy the download location. Then, from the directory on the cluster where you save your source files (personally, I like `/usr/local/src`)

```
wget <location>
```

Untar the file with

```
tar xvzf <your file>
```

and then cd into the new directory. From here, you'll need to run `./install.sh`, which will give you a series of interactive menus. Enter your serial number when prompted, and continue through. When you see the following menu -

```
Which of the following would you like to do?
1.    Typical Install (Recommended - Installs All Components).
2.    Custom Install (Advanced Users Only).
x.    Exit.
```

your selection should depend on whether you've already installed on compiler. If you have already installed one, then you've automatically installed the Intel debugger once, and you don't need a second installation, so choose "2". Otherwise, choose "1".

You'll be prompted for the install location. The compilers don't necessarily need to be on every node, so you don't necessary need to put them on an **NFS**. You will, however, want to put the compilers somewhere in the **users' path**, or update the path afterward. Until your users can do `which icc` or `which ifort`, the PATH hasn't been successfully updated, but once they can, your users can use the Intel compilers.

Udev: Renaming Network Interfaces

Network Interfaces (NICs)

When you type

```
ifconfig
```

each network interface has a name. These are usually `lo`, the internal loopback device that isn't really an interface, and `ethX` where X starts at 0 and continues up depending on how many network interfaces (sometimes called NICs) the machine has. These can actually be renamed to anything else, as long as the name is short.

In several other places, I've given examples of a setup for another cluster at my school that runs over three interfaces. It keeps three domain names - gig0, gig1, and fe0. Gig0 and gig1 are over Gigabit Ethernet, while fe0 is over FastEthernet. To help remember which domain is over which interface, the three interfaces on each machine have been renamed from eth0, eth1, and eth2 to gig0, gig1, and fe0.

Udev

Udev is responsible for managing devices on a Linux system. This includes everything from hard drives and RAM to, you guessed it, network cards (NICs). As part of this, udev assigns names and IDs to these devices, including the name that users will use to refer to the device.

If you've ever moved a hard drive from one computer to another, you might notice that the network interface used to be eth0 and it's now become eth1. What's up with that? The reason this happens is that udev stores its information about the network cards, even though this is automatically populated. When the new computer boots up for the first time, udev sees that the current NIC doesn't match the one it previously had, and so it creates a new entry and increments the number after `eth`. This is why the udev entries should also be cleared before **cloning a hard drive image**.

Old NICs can be deleted from udev and current ones can be renamed.

/etc/udev/rules.d/z25_persistent-net.rules

`z25_persistent-net.rules` is the file responsible for the names of the network interfaces. A typical line looks like this.

```
# PCI device 0x10de:0x0373 (forcedeth)
SUBSYSTEM=="net", DRIVERS=="?*", ATTRS{address}=="00:e0:81:75:94:31", NAME="eth1"
```

Changing the value of NAME is all that's necessary to change a NIC's name. Similarly, to delete an old NIC, just delete the line. Then either reboot or restart udev. To restart udev, run

```
/etc/init.d/udev restart
```

References

- Greg Kroah-Hartman. Kernel Korner - udev—Persistent Device Naming in User Space. Linux Journal, June 1 2004. Available online at **http://www.linuxjournal.com/article/7316**

Setting a Dynamic Hostname by DNS

Often, rather than having to set `/etc/hostname` for each worker node, it's more helpful to have it assign the hostname to itself automatically at boot based on the information handed to it by DNS (see **Name Service: DNS and BIND**). This is particularly true since it's most expedient to create one worker node image and then broadcast that image onto the other nodes, and it's pain to go through and change each one's hostname. Further, since all the information for which machine is which is located in DNS, why make it necessary to change that in two places?

Dotquad.c

There's a small script that set this for you, if DNS has been set up. It requires the binary from `dotquad.c`. First, save the following, `dotquad.c`, on the machine.

```
/* This file is in the public domain. */

#include <stdlib.h>
#include <stdio.h>

void range(int a){
        /* die if not valid octet */
        if (a<0) exit(1); if (a>255) exit(1);
}

int main(int argc, char* argv[]){
        int a, b, c, d;
        unsigned int ip = 0;

        if (argc != 2) exit(1);
        if (sscanf(argv[1], "%i.%i.%i.%i", &a, &b, &c, &d)<4) exit(1);
        range(a); range(b); range(c); range(d);
        ip = ip * 256 + a;
        ip = ip * 256 + b;
        ip = ip * 256 + c;
        ip = ip * 256 + d;
        printf("%u\n", ip);
        exit(0);
}
```

The binary of this file needs to be compiled and placed at `/bin/dotquad`, because the custom hostnames script will reference it there. If you haven't installed g++ yet, do that now (`apt-get install g++`). Then,

```
g++ dotquad.c -o /bin/dotquad
```

Custom Hostname Script

Below is the script that will automatically set the hostname based on DNS for the machine. I recommend setting `/etc/hostname` to some default value common to all of the machines, like amnesia. Then you can tell at a glance if DNS is working. Before this script will work, `host` must be installed on the machine. Do this with

```
apt-get install host
```

Then save the file below in `/etc/init.d/`, make it executable (`chmod +x <filename>`), and then symlink it into `/etc/rc2.d`, something like this:

```
ln -s /etc/init.d/hostnamecustom /etc/rc2.d/S21hostnamecustom
```

You can also start or stop it at the command line, after it's moved to `/etc/init.d/`. This won't take affect for any existing bash sessions; you'll have to log out and log back in to see the change.

```
/etc/init.d/hostnamecustom start
```

Hostnamecustom

```
#!/bin/sh
DOTQUAD=/bin/dotquad
case "$1" in
    start)
    echo -n 'Setting system hostname: '
        sleep 2
        for x in `ls /proc/sys/net/ipv4/conf/ | grep -v lo | grep -v all | grep -v default` ; do
        interface=$x
        echo Trying interface $x
        ip=`ifconfig $interface | grep "inet addr" | cut -d":" -f 2 | awk '{print $1}'`
        echo "IP resolved to $ip"
        if $DOTQUAD "$ip" >/dev/null ;
        then
                host=`host $ip 2>/dev/null | head -n 1 | awk '{print $2}' 2>/dev/null`
                echo "$host"
                if [ "x$host" != "x" ] ; then
                        hname=`echo $host | cut -d"." -f1`
                        echo "Host name is $hname"
                        sysctl -w kernel.hostname=$hname
                        dname=`echo $host | cut -d"." -f2-`
                        echo "Domain name is $dname"
                        sysctl -w kernel.domainname=$dname || true

                        # save the current /etc/hosts file
                        cp /etc/hosts /etc/hosts.orig || true

                        # strip out any existing entry for this host
                        cat /etc/hosts | grep -v $hname | cat > /etc/hosts.new

                        # add the new IP fqdn hname to the /etc/hosts.new file
                        echo $ip $hname $host >> /etc/hosts.new

                        # mv the new file into place
                        mv /etc/hosts.new /etc/hosts

                        # save the current /etc/hostname file
                        cp /etc/hostname /etc/hostname.orig || true

                        # add the new hostname to /etc/hostname.new
                        echo $hname >> /etc/hostname.new

                        # move hostname.new into place
                        mv /etc/hostname.new /etc/hostname

                        exit 0
                fi
        fi
        done
        if [ "x$interface" = "x" ] ; then
                echo "Assigning hostname failed."
        else
                echo `ifconfig eth0`
                echo done.
        fi
        ;;
```

```
    stop)
    echo -n 'Restoring original hostname '

        if [ -f /etc/hostname ] ; then
            hostname --file /etc/hostname
        fi
        if [ -f /etc/hosts.orig ] ; then
            mv /etc/hosts.orig /etc/hosts || true
        fi
        ;;

    restart|reload|force-reload)
            $0 stop
            sleep 2
            $0 start
            ;;
esac
```

Cloning Worker Nodes

If you've been walking through the tutorials with me, you're at the point where you have the head node and one worker node set up. Rather than having to install everything again on all the other worker nodes, we can turn all the others into copies of the one that's already set up. All the other worker nodes will receive an exact copy (a clone) of the entire hard drive of the worker node that's already set up. The data to be cloned across the different nodes is called the image. This is called cloning or imaging.

There are many different ways to clone the worker nodes, but they all follow the same basic principle. We have one machine's hard drive completely set up as a worker node image, and then the image on this hard drive will be copied onto all of the hard drives. Cloning a image prevents having to install the same software and configure the same files multiple times, possibly allowing inconsistencies and errors to slip in. Instead of doing something x times for x number of worker nodes, it can be configured once and tweaked until it's perfect, and then all the other nodes become exactly the same.

Because of that, however, the original worker node needs some tweaking before the image is ready to be compatible for all of the nodes.

Preparing the Original Worker Node Image

The Basics

First, the Debian/Linux basics: for one, make sure you're running an up-to-date copy of Debian. Do an `apt-get update` and `apt-get upgrade` (and even `apt-get dist-upgrade` if you need to) before you clone the other machines. It's a lot easier to do it once now than having to do it on each machine separately after cloning.

The machine providing the image should also have all of its services properly set up - it should be **mounting the NFS share**, receiving an IP address from the **DHCP server**, resolving **DNS** properly, and be set up to authenticate correctly using **LDAP** or another paradigm. It's much easier to fix this problem on one machine before cloning than having to fix it on x number of machines afterward. Make sure the paths, profiles, histories, and **anything else for Bash** is set up.

You'll also want to have the **compilers installed** on the worker node. If you're going to have a **dynamic domain name based on DNS**, that should also be set up. Make sure the `/etc/hostname` and `/etc/hosts` are set to something that you want all the machines to initially have, (I use "amnesia"), for prior to the first time they start up and set it with DNS. In other words, I'm about to clone `kestrel`, so I need to take out the lines referring to kestrel put in `/etc/hosts` and `/etc/hostname` by the custom host name script. When the new machines boot up for the first time, they'll have these files properly set up for them.

SSH Keys

If you want the root account to be able to SSH into each of the machines from each of the machines without needing to supply a password, now is also the time to set this up. Do this by generating a key on the machine that will be donating the image. As root, enter

```
ssh-keygen
```

and hit enter when prompted for the file (keep the default), and also hit enter (no password) twice when prompted for the password. It should look like the following.

```
waxwing:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
**********************************************
```

Next, move into the directory where the key was saved: `/root/.ssh`. We'll create a file called `authorized_keys2`. This file will contain the list of SSH fingerprints allowed to SSH into this machine without being prompted for a password. Since this image will be cloned, all of the machines will have the same SSH fingerprint (`id_rsa` and `id_rsa.pub`), so putting this key into `authorized_keys2` will allow the root account to SSH amongst all the machines without a password. Copy the fingerprint from `id_rsa.pub` to `authorized_keys` with

```
cat id_rsa.pub >> authorized_keys2
```

If `authorized_keys2` already exists, it will append this entry at the bottom. (Make sure the file only contains entries you want applicable for all of the machines that will be imaged.) If it doesn't exist, the file will be created.

To include the head node as one of the machines that root can SSH amongst freely, the contents of the `/root/.ssh/` directory need to be sync'd over. If you'd like to include the head node, run the following command from the worker node:

```
rsync -plarv /root/.ssh/ root@<your head node>:/root/.ssh/
```

SSH StrictHostKeyChecking

You'll also want to disable `StrictHostKeyChecking` for SSH. The default for this setting is `ask`, meaning that when a user (or a program acting on behalf of a user) tries to SSH from one node to another node that it hasn't encountered before, the user will be prompted as to whether they would like to accept the identification the node gives and continue SSHing in. Unfortunately, this can be a show stopper for **MPICH**, a widely used parallel programming implementation, and **scheduler and queue software** if there's no user to enter `yes` when prompted.

To prevent this from being a problem, open up `/etc/ssh/ssh_config` and find the line that looks like this:

```
#   StrictHostKeyChecking ask
```

Take out the hash (`#`) to uncomment this line, and change the value from `ask` to `no`.

Clearing out Current Udev MAC Information

One last important tip: everything will be exactly copied from the worker node that will be sending out its image. This includes the file that contains the MAC addresses of the network interfaces (NICs) and binds them to their names. This file will automatically be added to when the machine boots, but if it has existing entries, new ones will be appended, and you'll end up with eth2 and eth3 instead of eth0 and eth1. To fix this, edit `/etc/udev/rules.d/z25_persistent-net.rules` and take out any lines below these comments:

```
# This file was automatically generated by the /lib/udev/write_net_rules
# program, probably run by the persistent-net-generator.rules rules file.
#
# You can modify it, as long as you keep each rule on a single line.
```

(For more tips on udev, see **Udev: Renaming Network Interfaces**.)

Cloning the Worker Nodes

I'll show three different paradigms for cloning the worker nodes, **udpcast**, **data dumping**, and **using rsync**.

Udpcast involves setting each of the nodes to PXE boot (network boot), and then the machine donating its image sends it out across the network to the other machines. Each of the machines needs to be configured to PXE boot, and the DHCP server needs to be configured to enable PXE booting. Udpcast also has to be tweaked for your particular set up. This is a more permanent solution and allows you to re-image the machines whenever you need to.

Data dumping, or **dd**, requires the hard drives to be taken out of the cases and placed in another machine, a Linux machine or one running a bootable Linux distribution. From the console, the data is then copied from the original hard drive to each of the other hard drives. This may tend to take longer, but the setup time required before you're actually imaging is less. Of course, if you want to re-image the drives again at a later date, you need to remove all of them from the worker nodes again.

Rsyncing also requires to the hard drives to be taken out and put into another machine. It's similar to data dumping, but requires slightly more command line setup and takes slightly less time to copy over. Again, if you want to re-image the drives again at a later date, you need to remove all of them from the worker nodes again.

The tutorial will split into three sections from here, so pick your paradigm:

- **Udpcast Cloning: Preparing the Udp-sender**
- **Cloning Worker Nodes with dd**
- **Cloning Worker Nodes with Rsync**

Udpcast Cloning: Preparing the Udp-sender

This is the second page of a five part tutorial on cloning worker nodes with Udpcast. The full tutorial includes

- **Cloning Worker Nodes**
- **Udpcast Cloning: Preparing the Udp-sender**
- **Udpcast Cloning: Preparing the DHCP Server for PXE Booting**
- **Udpcast Cloning: Troubleshooting PXE Booting**
- **Udpcast Cloning: Imaging using Udpcast**

The Udpcast Paradigm

We'll use **udpcast (http://udpcast.linux.lu/)** to send out the image of one node onto all the other nodes. Udpcast is a way of sending out data to multiple locations on a network at the same time. The worker node already set up will send out its image. I'll call it the udp-sender or just sender. All the other nodes, the receiving an image and being turned into clones, will be udp-receivers or receivers. The udp-receivers require a slightly more complicated setup than the sender, because the receivers can't boot off their hard drives if they haven't had an operating system installed. Instead, they'll boot off the network before running udp-receive.

Updating the Udp-sender's Image

The udp-sender is the machine that will be cloned. It's the worker node currently set up that all the other worker nodes will be imaged from. Because each of the udp-receivers will be cloned into exactly copies of the sender, it's important to make sure the sender's hard drive is in exactly the state you want it to be in prior to cloning!

Installing Udpcast

The quickest way to install udpcast on Debian is to `apt-get install udpcast`. However, the version of udpcast obtained with apt-get is significantly older than the current version. The worker nodes will use a bootable image with a current version of udpcast, and installing the older version on the udp-sender causes problems. In other words, udpcast needs to be installed from source. The source can be found on the **Udpcast site (http://udpcast.linux.lu/source.html)** .

You'll want to keep your all your source files in one area. I'd recommend keeping them on an NFS mount, particularly since all the nodes don't need a copy of this. From `/shared/source/` on my machine, I executed

```
wget http://udpcast.linux.lu/download/udpcast-20070602.tar.gz
```

Next the tarball needs to be untarred:

```
tar xvzf udpcast*.tar.gz
```

It's a pretty standard `./configure`, `make`, `make install` process. From within the directory, execute

```
./configure -prefix=<install location>
```

If you don't have a **C compiler**, you'll need to get that before `./configure` can complete successfully. After it does, run

```
make
```

and then

```
make install
```

Type `which udp-sender` afterward to make sure that the binary was installed someplace within your path. (If it wasn't, do an `updatedb` and then `udp-sender`, and add the location to your **Bash PATH**.)

Udpcast Cloning: Preparing the DHCP Server for PXE Booting

This is the third page of a five part tutorial on cloning worker nodes with Udpcast. The full tutorial includes

- **Cloning Worker Nodes**
- **Udpcast Cloning: Preparing the Udp-sender**
- **Udpcast Cloning: Preparing the DHCP Server for PXE Booting**
- **Udpcast Cloning: Troubleshooting PXE Booting**
- **Udpcast Cloning: Imaging using Udpcast**

Bootloader and TFTP

By now, the **DHCP Server** should be set to serve up IP addresses to all of the nodes. We need to set it up so that it not only serves out DHCP leases (which includes the IP address), but also sends out an image for all of the worker nodes to boot up from. (Specifically, we'll send out an image that includes udpcast - more about that later.) This requires two new packages. First, we need a boot loader. The boot loader allows DHCP to send out an image to the nodes. The nodes will store this image in their NIC ROM and then boot from it. (This is called **PXE (Preboot eXecution Environment) (http://en.wikipedia.org/wiki/Preboot_Execution_Environment)** booting. To install the **Syslinux (http://syslinux.zytor.com/index.php)** boot loader,

```
apt-get install syslinux
```

When the nodes connect to the dhcp server to obtain a bootable image, they'll use the **trivial file transfer protocol (tftp) (http://en.wikipedia.org/wiki/Trivial_File_Transfer_Protocol)** to transfer the image. This means the DHCP server also needs to be running a TFTP server to answer these requests and send the bootable image over TFTP. `tftpd-hpa` is one of these servers, so next,

```
apt-get install tftpd-hpa
```

The configuration file for tftpd-hpa is stored at `/etc/default/tftpd-hpa`. If you open the file, you should see this:

```
#Defaults for tftpd-hpa
RUN_DAEMON="no"
OPTIONS="-l -s /var/lib/tftpboot"
```

First, to turn the server on, `RUN_DAEMON`'s value needs to be changed to `yes`. Under the `OPTIONS` field, the default directory for TFTP is shown (`/var/lib/tftpboot`). This is the directory that the TFTP server will serve up files from, so this is where the bootable images need to be placed.

Configuring DHCP for PXE Booting

Now DHCP needs to be set up to send out a PXE image. These changes need to be made in `/etc/dhcp3/dhcpd.conf`. First, the lines

```
allow booting;
allow bootp;
```

need to be added near the top of the file. These can be put before any of the "option definitions common to all supported networks" bit. Next, we'll move all the worker nodes within a group in the field. Before the start of the worker nodes, add

```
group {
        next-server 192.168.1.254;
        filename "pxelinux.0";
```

`next-server` should be the IP address of your DHCP server (mine is 192.168.1.254). We'll move the `pxelinux.0` file to its proper place shortly. Add a closing bracket (one of these: }) after the last worker node entry. My `dhcpd.conf` file now looks **like this**. For a more detailed explanation, see the **PXELINUX - SYSLINUX for network boot (http://syslinux.zytor.com/pxe.php)** page on the **Syslinux (http://syslinux.zytor.com/)** site.

After making changes to the `dhcpd.conf`, the DHCP server needs to be restarted. Do this with

```
/etc/init.d/dhcp3-server restart
```

Setting up the Udpcast Image

Since we're trying to send out the file `pxelinux.0` (specified in `/etc/dhcp3/dhcpd.conf`, it had better be in the location the TFTP server will look for it in (specified in `/etc/default/tftpd-hpa`). When installing Syslinux, it automatically puts the file in `/usr/lib/syslinux/pxelinux.0`. Copy it to the proper place:

```
cp /usr/lib/syslinux/pxelinux.0 /var/lib/tftpboot/
```

Next, we need to get the configuration file, the kernel, and the ram image from the **udpcast site (http://udpcast.linux.lu/)** (specifically, from **http://udpcast.linux.lu/bootloader.html#pxe**). From the `/var/lib/tftpboot/` directory, use

```
wget http://udpcast.linux.lu/20070602/linux
```

and

```
wget http://udpcast.linux.lu/20070602/initrd
```

The first one is the kernel (the operating system), and the second is the image that will be loaded. The image contains almost nothing except udpcast, which will be used to send out and receive the image that's being cloned.

Next, create the directory `/var/lib/tftpboot/pxelinux.cfg` and move into it. The configuration file for this image needs to be placed here. Do this with

```
wget http://udpcast.linux.lu/default
```

This file specifies where the kernel and the image are (right where we already put them), as well as some additional parameters.

PXE /etc/dhcp3/dhcpd.conf Example

This is part of the **Udpcast** tutorial. This is my `/etc/dhcp3/dhcpd.conf` file after setting it up to allow PXE booting.

```
#
# Sample configuration file for ISC dhcpd for Debian
#
# $Id: dhcpd.conf,v 1.1.1.1 2002/05/21 00:07:44 peloy Exp $
#

# PXE stuff
allow booting;
allow bootp;

ddns-update-style none;

# option definitions common to all supported networks...
option routers 192.168.1.254;
option domain-name-servers 192.168.1.254;
default-lease-time 14400;
max-lease-time 14400;

log-facility local7;

subnet 192.168.1.0 netmask 255.255.255.0 {
        option domain-name "raptor.loc";
        deny unknown-clients;
}

host gyrfalcon.raptor.loc {
        hardware ethernet X:X:X:X:X:X;
        fixed-address 192.168.1.200;
}

group {
        next-server 192.168.1.254;
        filename "pxelinux.0";

host kestrel.raptor.loc {
        hardware ethernet X:X:X:X:X:X;
        fixed-address 192.168.1.201;
}

host owl.raptor.loc {
        hardware ethernet X:X:X:X:X:X;
        fixed-address 192.168.1.202;
}

  # more hosts snipped...

}
```

Udpcast Cloning: Troubleshooting PXE Booting

This is the fourth page of a five part tutorial on cloning worker nodes with Udpcast. The full tutorial includes

- **Cloning Worker Nodes**
- **Udpcast Cloning: Preparing the Udp-sender**
- **Udpcast Cloning: Preparing the DHCP Server for PXE Booting**
- **Udpcast Cloning: Troubleshooting PXE Booting**
- **Udpcast Cloning: Imaging using Udpcast**

Troubleshooting

If the file isn't in the proper place, the TFTP server won't be able to find it, and you'll receive an error from the nodes trying to boot up -

```
PXE-T01: File Not Found
PXE-E3B: TFTP Error - File Not Found
PXE-M0F: Exiting Intel Boot Agent
```

Then doublecheck that `pxelinux.0` is in the location specified by the file `/etc/default/tftpd-hpa`.

Sender

```
Udp-sender 2007-06-02
Using full duplex mode
Using mcast address 232.168.1.205
UDP sender for /dev/sda at 192.168.1.205 on eth0
Broadcasting control to 192.168.1.255
New connection from 192.168.1.202  (#0) 00000009
Ready. Press any key to start sending data.
New connection from 192.168.1.203  (#1) 00000009
Ready. Press any key to start sending data.
New connection from 192.168.1.206  (#2) 00000009
Ready. Press any key to start sending data.
New connection from 192.168.1.207  (#3) 00000009
Ready. Press any key to start sending data.
Starting transfer: 00000009
bytes= 80 028 803 072 re-xmits=0000012 (  0.0%) slice=0122  80 026 361 856
Transfer complete.
Disconnecting #0 (192.168.1.202)
Disconnecting #1 (192.168.1.203)
Disconnecting #2 (192.168.1.206)
Disconnecting #3 (192.168.1.207)
```

Receiver

```
Udp-receiver 2007-06-02
UDP receiver for /dev/sda at 192.168.1.202 on eth0
received message, cap=00000009
Connected as #0 to 192.168.1.205
Listening to multicast on 232.168.1.205
Press any key to start receiving data!
bytes= 80 028 803 072  (385.52 Mbps) 80 028 803 072
Transfer complete.
```

Udpcast Cloning: Imaging using Udpcast

This is the third page of a five part tutorial on cloning worker nodes with Udpcast. The full tutorial includes

- **Cloning Worker Nodes**
- **Udpcast Cloning: Preparing the Udp-sender**
- **Udpcast Cloning: Preparing the DHCP Server for PXE Booting**
- **Udpcast Cloning: Troubleshooting PXE Booting**
- **Udpcast Cloning: Imaging using Udpcast**

```
kestrel:~# udp-sender -f /dev/sda
Udp-sender 2004-05-31
Using mcast address 232.168.1.201
UDP sender for /dev/sda at 192.168.1.201 on eth0
Broadcasting control to 192.168.1.255
New connection from 192.168.1.202  (#0) 00000009
Ready. Press any key to start sending data.
New connection from 192.168.1.205  (#1) 00000009
Ready. Press any key to start sending data.
New connection from 192.168.1.203  (#2) 00000009
Ready. Press any key to start sending data.
New connection from 192.168.1.207  (#3) 00000009
Ready. Press any key to start sending data.
```

Setting the Nodes to PXE Boot

All of the nodes that will be receiving an image, plus the node that will be sending its image, need to be configured to PXE boot. This should be a setting in the BIOS. It might be listed under boot options, PCI options, advanced options, or something else. Generally you're looking for something along the lines of "Onboard NIC ROM", "PXE Boot", or "Network Boot", and you want that setting to be enabled.

Configuration for the Nodes Receiving an Image

The next time that node is restarted, it should first look to the network for a DHCP server with a PXE image to boot from before booting from its own hard drive. If you start up one of the nodes, the udpcast image will load and then you'll be prompted to answer a series of questions. Rather than having to answer these questions again for each node, the `pxelinux.cfg/default` can be edited to automatically supply them.

You can do this by writing down your answers for each of the questions or manually editing the file using the documentation on the **Udpcast site (http://udpcast.linux.lu/bootloader.html#pxe)** (see "The configuration file" header). Then update the file `/var/lib/tftpboot/pxelinux.cfg/default`. This will be the file that all of the receivers use for configuration (we'll create the sender one in a moment).

Commands need to be added to the end of the line beginning with

```
append load_ramdisk=1 initrd=initrd root=01:00
```

Mine ended up being this:

```
default linux
label linux
  kernel linux
  append load_ramdisk=1 initrd=initrd root=01:00 auto=yes lang=US kbmap=US netmodule=AUTO netmodparm= dhcp=yes
  ipappend 1
```

- `auto=yes` means that the user should not be prompted for any parameters already specified.
- `lang` is the language
- `kbmp` is the keyboard map
- `netmodule=AUTO` means the network interface should automatically be detected
- `netmodparm` are parameters for the network interface. Even though there aren't any, the parameter needs to be entered or the computer booting with this configuration will prompt the user for them.
- `dhcp=yes` means that the nodes should use their information on IP, gateway, etcetera from the DHCP server (you can specify your own instead if you need to).
- `port=9000` specifies that the transfer will occur on port 9000 (the default, but again it needs to be entered so it doesn't prompt)
- `enableDiskModule` needs to be yes, since it specifies if the drivers for the hard drive should be loaded
- `diskmodule` is the specific driver for the disk. AUTO causes it to automatically detect them.
- `disk` specifies what is to be imaged. In my case, the hard drive to be imaged is `/dev/sda`. (To determine yours, use `fdisk -l`.)
- `compr` is the compression; it's fine not to use any.
- `umode` can be `snd` (send) or `rcv` (receive). This is the file all the receivers will take, so use `rcv`.
- `udpcparam` are any additional parameters. Again, there are none, but unless it is specified, the prompt will come up.

Configuration for the Sending Node

The sending node needs a different configuration file to tell it to send the image of its hard drive rather than receiving one. The `default` file is used by any PXE booting machines without their own individual configuration files. Rather than having the udp-sender use the default file, we'll specify its own for it. The first place a machine looks for a configuration file is a file that starts with 01- followed by its MAC address specified with dashes. For instance, a machine with a MAC address of 00:0e:0c:4d:a4:b2 looks for a file named `01-00-0e-0c-4d-a4-b2`.

Create this file by copying the default one:

```
cp default 01-<your sender's MAC address>
```

Then we need to update the file with the correct parameters for the sender. Specifically, the `umode` parameter needs to be changed from `rcv` to `snd`. `udpcparam=` also needs to be changed to `udpcparam=broadcast`.

Links

- **The SYSLINUX Project site (http://syslinux.zytor.com/)**
- **The UDPCast site (http://udpcast.linux.lu/)**

Cloning Worker Nodes with dd

This is the second page of a two part tutorial on cloning worker nodes with `dd`. The full tutorial includes

- **Cloning Worker Nodes**
- **Cloning Worker Nodes with dd**

Data Dumping: dd

Performing a data dump, or "dd", is a quick and dirty way to image a hard drive. Unlike using **Updcast**, where the udp-sender broadcasts its image out to the network and all of the other nodes receive the image at the same time, a data dump essentially works point-to-point. The other main difference is that dd's occur on one machine with the sending and receiving hard drives hooked up to it. Multiple data dumps can be occurring at the same time, but each one is a separate process and having too many run at once will bog down the system.

Preparing for the dd

The first step is to take the hard drives out of the worker nodes (unless they're already out), and put them all into another machine. This machine can be the machine holding the hard drive that's already set up, but it doesn't need to be. Next, the operating system of the machine that's going to be cloned shouldn't be running while this operation takes place. The easiest way to get around this is to use a bootable CD, like **Ubuntu's (http://www.ubuntu.com/)** live CD or **Knoppix (http://www.knoppix.org/)** . After all of the drives are hooked up, turn the machine on and boot from the CD.

Next, you'll need to become root. If you're using an Ubuntu CD, do this by issuing `sudo su -`. Otherwise, become root as you would normally. Then run

```
fdisk -l
```

If you don't run it as root, it will run without returning any drives. What you want to see is something like this:

```
root@ubuntu:~$ fdisk -l

Disk /dev/sda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1        9483    76172166   83  Linux
/dev/sda2            9484        9729     1975995   82  Linux swap / Solaris

Disk /dev/sdb: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1        9484    76180198+  83  Linux
/dev/sdb2            9485        9728     1959930   83  Linux

Disk /dev/sdc: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1               1        9483    76172166   83  Linux
/dev/sdc2            9484        9729     1975995   82  Linux swap / Solaris
```

`Fdisk -l` shows all of the hard drives currently plugged into the system. In the example above, I have three hard drives plugged in: `/dev/sda/`, `/dev/sdb`, and `/dev/sdc`. The `sd` part refers to them being SATA drives; IDE drives will show up as `hd`-something.

Next, we need to figure out which hard drive is the one that's already set up. If you don't already have an operating system installed on the others, you'll be able to see the difference when using `fdisk`. But if you already do, and you can't tell which hard drive should be the source for the data dump, you can start mounting them one at a time and checking them. To mount a drive, first create a directory

```
mkdir /mnt
```

and then mount one of the hard drives

```
mount <your hard drive>1 /mnt
```

The `1` is important. Rather than mounting an entire hard drive (including the swap space), you want to only mount the part that contains a filesystem. For instance, mounting one of mine would be

```
mount /dev/sda1 /mnt
```

instead of `mount /dev/sda /mnt`. To unmount, use

```
umount <your drive>1 /mnt
```

Running dd

Make sure you know which hard drive will be the source before you run `dd`. It's very easy to accidentally overwrite the drive that you've put effort into setting up with another drive by accident. Once know you which drive it is and you're ready to continue, unmount all of the drives (see above).

The `dd` command takes several arguments. By default, it takes input from standard in and outputs it to standard out. (Try this by running `dd` by itself, typing a few lines, and then ending it with Ctrl-D.) Instead, we want it to take input from one hard drive and output it to another hard drive. This is the "dirty" part of "quick and dirty" - it will read it byte by byte and copy that exact sequence from one hard drive to another. We could just as easily output it to a file and make a copy of the hard drive.

The input and output are specified with `if=` (input file equals) and `of=` (output file equals). The full syntax is

```
dd if=<prepared hard drive> of=<hard drive to image> &
```

For instance, mine from above might be

```
dd if=/dev/sda of=/dev/sdb &
```

Notice this time that we're not specifying the partitions (there's no number) - we want the entire hard drive to be copied so that the new hard drive gets a swap space, too, and not just the file system. The `&` makes the process run in the background; this is fine because it won't give any output as it progresses anyway.

You can run multiple data dumps at the same time, but each one will have to be specified separately. You can use `top` to monitor how long the processes have been run and to tell when they have stopped.

Cloning Worker Nodes with Rsync

This is the second page of a two part tutorial on cloning worker nodes with `rsync`. The full tutorial includes

- **Cloning Worker Nodes**
- **Cloning Worker Nodes with Rsync**

Rsync

Rsync is an elegant tool to copy files. It does this by looking at the differences between the source and destination file system and only copying over the differences, speeding up the process over a flat copy if some of the files already exist on the destination system.

One of the differences between using a **data dump (dd)** and rsync is that the latter won't create the partition tables or boot loader on the destination drives, so that needs to be done manually. Unlike using **Updcast**, where the udp-sender broadcasts its image out to the network and all of the other nodes receive the image at the same time, a data dump essentially works point-to-point. The other main difference is that rsync's occur on one machine with the sending and receiving hard drives hooked up to it. Multiple rsyncs can be occurring at the same time, but each one is a separate process and having too many run at once will bog down the system.

Setup

The first step is to take the hard drives out of the worker nodes (unless they're already out), and put them all into another machine. This machine can be the machine holding the hard drive that's already set up, but it doesn't need to be. Next, the operating system of the machine that's going to be cloned shouldn't be running while this operation takes place. The easiest way to get around this is to use a bootable CD, like **Ubuntu's (http://www.ubuntu.com/)** live CD or **Knoppix (http://www.knoppix.org/)** . After all of the drives are hooked up, turn the machine on and boot from the CD.

Next, you'll need to become root. If you're using an Ubuntu CD, do this by issuing `sudo su -`. Otherwise, become root as you would normally. Then run

```
fdisk -l
```

If you don't run it as root, it will run without returning any drives. What you want to see is something like this:

```
root@ubuntu:~# fdisk -l

Disk /dev/hda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System

Disk /dev/hdc: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hdc1   *           1        9483    76172166   83  Linux
/dev/hdc2            9484        9729     1975995   82  Linux swap / Solaris
```

`Fdisk -l` shows all of the hard drives currently plugged into the system. In the example above, I have two

hard drives plugged in: `/dev/hda/` and `/dev/hdc`. The `hd` part refers to them being IDE drives; SATA drives will show up as `sd`-something.

Next, we need to figure out which hard drive is the one that's already set up. If you don't already have an operating system installed on the others, you'll be able to see the difference when using `fdisk` (that's the case in the above example). But if you can't tell the difference using `fdisk`, you can start mounting them one at a time and looking at the files on the hard disk until you find the master one. To mount a drive, first create a directory

```
mkdir /mnt
```

and then mount one of the hard drives

```
mount /dev/<your hard drive>1 /mnt
```

The `1` is important. Rather than mounting an entire hard drive (including the swap space), you want to only mount the part that contains a filesystem. For instance, mounting one of mine would be

```
mount /dev/hda1 /mnt
```

After you're done looking at the files, to unmount, use

```
umount /dev/<your drive>1 /mnt
```

Creating Partitions

Each worker node first needs to have its partitions set up the same way as the master image has then set up. If you need to see the master drive's partitions to remind you, run `fdisk -l` and look for the correct hard drive.

One at a time, for each worker node drive to be imaged, run

```
fdisk /dev/<drive to be imaged>
```

This will put you into an interactive mode with `fdisk`, a disk formatter. You'll be using these commands:

- `n` - create a new partition
- `d` - delete partition (if you mess up)
- `p` - print the status of partition tables
- `a` - make a partition active
- `w` - write out the changes

At a minimum, you should have two partitions: the filesystem partition, and the swap partition. Here is **a transcript** showing how to create the filesystem with one filesystem partition and one swap partition like I did for my 80 GB drive.

Sometimes, on Ubuntu (and possibly other live CDs), it will finish with an error like the following:

```
WARNING: Re-reading the partition table failed with error 16: Device or resource
busy.
The kernel still uses the old table.
The new table will be used at the next reboot.
Syncing disks.
```

This is fine; as it says, rebooting will take care of the problem. You'll have to reboot before you can create the file system and swap space.

Creating the File System and Swap

With the partitions set up, they're now ready to have the file system and swap created on them. Again, **make sure you're running these commands on the correct drives!** Creating a file system on your master image will wipe out all of your current files! Only run these commands on the new hard drives.

For each worker node, create a file system on any ext3 file system partitions. (If you've forgotten which ones those were, run `fdisk -l` again. Do this with

```
mkfs.ext3 /dev/<drive><file system partition #>
```

For instance, mine from the **transcript example** would be `mkfs.ext3 /dev/hda1`. This sets up on the file system so we'll be able to copy files over to it. (It's like formatting a drive in Windows.)

Next, for each worker node, create the swap space on the swap space partitions. Do this with

```
mkswap /dev/<drive><swap partition #>
```

Mine from the example is `mkswap /dev/hda2`.

Running Rsync

Now, we're finally ready to copy the files over. First, create a new directory for each drive, including the master image. I like to make the master image's directory slightly different so it's easy to tell them apart. Here's an example for three drives, where the `c` drive happens to be the master:

```
mkdir /tmp/mnta /tmp/mntc-master /tmp/mntd
```

Then, one at a time, mount the file system partition for each of the drives. (If you've forgotten which partition it is, run `fdisk -l`.)

```
mount /dev/<your drive><file system partition #> /tmp/<directory name>
```

Now you'll need to run `rsync` once for each drive to be imaged. You can do this in different terminal windows at the same time. To copy the files from the master image to a new drive, use

```
rsync -plarv --progress /tmp/<master location>/ /tmp/<new drive location>/
```

The trailing slashes are very important, because they indicate the copying should be for that particular directory.

Installing GRUB

All the files have now been copied over, which is excellent. There's one last step necessary in order to boot off the new drives: they need a **boot loader (http://en.wikipedia.org/wiki/Boot_loader)** installed. This is fairly easy to do. To enter the interactive mode with GRUB, a common Linux boot loader, run

```
grub
```

and then, for each new hard drive, run

```
root (hdX,0)
setup (hdX)
```

where `X` is the number for that drive. (For instance, `hda/sda` would be `0`, `hdb/sdb` would be 1, and so on.)

Done!

Once `rsync` finishes, you're done. Congratulations, you've just made a copy of your master image hard drive.

Fdisk Transcript

This is part of a tutorial on how to clone the worker nodes using rsync. The full tutorial starts with **cloning background information** and addresses rsync on the **Cloning Worker Nodes with Rsync** page.

Fdisk Transcript

```
root@ubuntu:~# fdisk /dev/hda

The number of cylinders for this disk is set to 9729.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): p

Disk /dev/hda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-9729, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-9729, default 9729): 9483

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (9484-9729, default 9484):
Using default value 9484
Last cylinder or +size or +sizeM or +sizeK (9484-9729, default 9729):
Using default value 9729

Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 82
Changed system type of partition 2 to 82 (Linux swap / Solaris)

Command (m for help): p

Disk /dev/hda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1               1        9483    76172166   83  Linux
/dev/hda2            9484        9729     1975995   82  Linux swap / Solaris
```

```
Command (m for help): a
Partition number (1-4): 1

Command (m for help): p

Disk /dev/hda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *           1        9483    76172166   83  Linux
/dev/hda2            9484        9729     1975995   82  Linux swap / Solaris

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource
busy.
The kernel still uses the old table.
The new table will be used at the next reboot.
Syncing disks.
```

Cluster Time-saving Tricks

SSH Keys

Both of the two tricks below will require authentication on each one of the worker nodes. In other words, when you SSH in to run a command, or run rsync (which runs over SSH), you'll be prompted for the password for that machine. There's an easier way than typing in the password every single time. Setting up **password-less SSH** allows the machine you're connecting from to authentication automatically with the machine you're connecting to. This can be a real time saver!

Running Commands with SSH

Whenever I need to something on all of my nodes, say an `apt-get update && apt-get upgrade`, or run a command to see what **mpi daemons** are running, it takes a lot of time to ssh into each node, wait for it to finish, then try to remember which ones I haven't done yet, and continue the process. I only have eight worker nodes and it's a pain; on a production cluster with tens or even hundreds of nodes, that would take much, much longer.

Rather than doing that, I use simple scripting to run commands. The first thing I do is keep a list of all my hosts in a file in root's home directory on the head node. It just looks like this:

```
eagle
goshawk
harrier
kestrel
kite
osprey
owl
peregrine
```

Pretty simple, but saves a lot of time not having to recreate that list all the time. Then, whenever I want to, say `apt-get update`, I write a little script at the command line from the head node. You can type it as it's show below, hitting shift+enter to create new lines -

```
gyrfalcon:~# for x in `cat ~/machines`
> do
> ssh $x apt-get update
> done
```

or all one one line, like this

```
for x in `cat ~/machines`; do ssh $x apt-get update; done
```

You can replace `apt-get update` with whatever command you want to run. If it's an interactive command (like `apt-get upgrade`), you'll have interactive session with the host until that command finishes.

Copying a File to All the Nodes

I also use my `~/machines` file on my head node for copying a file out to all the worker nodes using the command `rsync`. (You might have used rsync to image your worker nodes if you followed the **Cloning Worker Nodes with Rsync** tutorial.) If I want to copy, say, `mycoolscript` to all the nodes, I would run (from the head node)

```
gyrfalcon:~# for x in `cat ~/machines`
> do
> rsync -plarv ~/mycoolscript root@$x:~/
> done
```

or as one line,

```
for x in `cat ~/machines`; do rsync -plarv ~/mycoolscript root@$x:~/; done
```

Setting up Scientific Software

Software needs vary widely amongst clusters depending on who is using the cluster and for what purpose. My experience is limited to the scientific research done on my campus, but I'll cover how to install two commonly used molecular dynamics packages - **gromacs** and **NAMD** - and **mpiBLAST**, a parallel implementation of BLAST for searching large databases of nucleotides or proteins.

Some packages may not be available through an apt repository. (In other words, you may not have the option to `apt-get` them.) For others, the source may be better because the apt version used might be out-of-date, or a more customized installation is needed. For these reasons and more, it's often better to install scientific software packages from source, using the **Source Installation Paradigm**.

The three software packages I'll show the installation for all require an implementation MPI, a parallel programming standard, in order to run. My MPI implementation of choice is **MPICH**. MPICH or a similar MPI implementation will need to be installed before installing gromacs, NAMD, or mpiBLAST.

Where to Start

If you're unfamiliar with installing software from source, check out the **Source Installation Paradigm** page. Then, make sure you have **MPICH** installed and running. Finally, pick a software package you're interested in installing:

- **Gromacs: Molecular Dynamics**
- **NAMD: Molecular Dynamics**
- **mpiBLAST: Nucleotide/Protein Searching**

Source Installation Paradigm

Overview

Installing from source is slightly different (and more complicated) than using the built-in Debian package manager, `apt`. When new software is installed with `apt-get install`, Debian takes care of making sure all the dependencies are taken care, and sometimes even takes care of most of the initial configuration of the package. Installing from source is a "roll your own" kind of installation, in a sense.

The general outline for installing from source is

> **get the software ---> run ./configure ---> run make ---> run make install**

That's it!

When to Install From Source (and when not to!)

The Debian package maintainers go to lengths to ensure that the Debian software is stable and plays nicely together. Something that has to be given up for this stability, and that's the ability to use cutting-edge software. Installation from source should be used when important differences exist between the current version and the Debian packaged version. (This is often the case with scientific software.)

When using the `apt` package manager, it also takes care of many different little details for you during the installation. It's great to have a mostly vanilla option, but sometimes software should be customized more thoroughly for a system than the apt package would easily allow. This is another instance when installation from source should be used.

Software shouldn't be installed from source if you want it to be automatically and easily updated (like installing Apache, a web server), if you want a plain vanilla installation, and especially if the software you're installing is simple.

As a guideline, most scientific software and a lot of non-operating system software should be installed from source.

Step One: Getting the Software

The first step in installing from source is downloading a compressed version of the source code. For open source projects, this is usually easy to find in a "Downloads" or similar section. I always keep my source code in one directory so it's easy to find later; I picked up using `/usr/local/src` as a habit from my mentor. You can use a different directory if you want.

From your directory of choice, use the command `wget`. Wget is used to download a file from a website and place it in your directory. For instance, if you're **installing MPICH**, you might run

```
wget http://www-unix.mcs.anl.gov/mpi/mpich/downloads/mpich2-1.0.5p4.tar.gz
```

Source code usually comes in a format ending with `.tar.gz`. This indicates it's a tarball that has been gzipped. To untar it and ungzip it at the same time, run

```
tar xvf <your file>
```

This will spit out the contents of the tarball in a new directory with the same name, minus the `.tar.gz`. Nifty, huh?

Step Two: Configuration

The next step for installation is to run `./configure` from within your new directory and specify all of the options that you want the compiler to use when it creates your version of the source code to install.

```
./configure -help
```

will show all of the available options. (If it's longer than a page or two, run `./configure -help | less` for an easier reading experience.) A common and important option is `--prefix`. This is used to specify where all of the files installed should be located. For instance, often times I want software installed to be accessible to all of my worker nodes, so I'll specify `--prefix=/shared/` in order to install it onto my **NFS mount**.

Once you've decided on all of the options needed, run

```
./configure <option 1> <option 2> <option 3> <etcetera>
```

If you ever need to remember what options you used when you ran it, check the top of the file `config.log`. This file is automatically generated when you run `./configure`.

Step Three: make

The next step should be as easy as just running `make`. The program `make` takes all of the parameters for your system that `./configure` found out, and creates a version of the source code to be installed specific to your system. It shouldn't need any input from you, just what `./configure` created for it. However, there are a couple of show-stoppers:

```
-bash: make: command not found
```

means it's time to

```
apt-get install make
```

Run `make` again after it's done. Another possible error,

```
gyrfalcon:/shared/source/mpich2-1.0.5p4# make
Beginning make
all-local
make: all-local: Command not found
make: *** [all-redirect] Error 127
```

means you need to

```
apt-get install automake
```

Run `make` again when it's done.

Step Four: make install

Finally, `make install` is used to take the new installation created by the previous run of `make` and install it to your system, again using all of the preferences and such that you specified earlier with `./configure`. Just run

```
make install
```

And that should be it.

MPICH: Parallel Programming

The Message Passing Interface, or MPI, is a common paradigm for implementing parallel programming through message passing. This means that multiple processes can run on multiple machines or on the same machine and communicate by sending small packets of information ("messages") amongst themselves to coordinate their efforts. MPI itself is a standard for the message passing library. MPI in itself isn't an implementation of messaging passing; it's a set of functions and abilities that any implementation of the message passing library must follow. Further, there multiple versions of the standard, making it even more confusing.

However, having one (or more) implementations of MPI on a cluster is often crucial. Your users can use the MPI standard and then compile their programs by linking in one of the implementations, thereby allowing them to write software that runs in parallel. Further, many scientific software packages depend upon an implementation of MPI in order to support running them in parallel. (A few notable examples are **gromacs**, **NAMD**, and **mpiBLAST**.)

There are many different implementations of the MPI standard. The most common ones in use on clusters are MPICH, LAM/MPI, and Open MPI. I'll be using MPICH in all of my tutorials, but there's a short description on the differences below. Then we'll walk through

- **Installing MPICH**
- **MPICH: Pick Your Paradigm**
- **MPICH with Torque Functionality OR MPICH without Torque Functionality**

MPICH versus LAM/MPI versus Open MPI

MPICH (http://www.mcs.anl.gov/research/projects/mpich2/) , or Message Passing Interface Chameleon, is a project out of Argonne National Laboratory and Mississippi State University, originally headed by William Gropp and Ewing Lusk. With MPICH, users simply compile their code with the MPICH libraries, then run their code using an MPICH command. These libraries are typically available only for C/C++ and Fortran. MPICH2 takes a different route, more similar to LAM/MPI (see below).

LAM/MPI (http://www.lam-mpi.org/) , or Local Area Multicomputer/Message Passing Interface, was developed by the Ohio Supercomputing Center. Unlike MPICH1, it requires users to first boot up a LAM environment before running their code. This involved starting a daemon on each one of the nodes to be used. Then, when a user is finished, s/he shuts down the LAM environment. According to the **LAM/MPI website (http://www.lam-mpi.org/)** , most of the maintainers for the LAM/MPI project are now switching over to **Open MPI (http://www.open-mpi.org/)** , a new(er) conglomeration of several MPI implementations.

MPICH2 takes on similar characteristics to LAM/MPI, in that it can either run as MPICH1 did, or an MPI daemon (called an `mpd`) can be started on each of the nodes. Unlike in LAM/MPI, if this daemon is started by root, users' profiles can be configured to have user programs attach to and run as the root daemon, meaning they do not need to start up and tear down an environment before running a parallel program. I'll be using MPICH2 in all of the tutorials.

References

- Sloan, Joseph D.. High Performance Linux Clusters. 1st ed. Sebastopol, CA: O'Reilly Media, Inc., 2005.
- **http://www-unix.mcs.anl.gov/mpi/mpich/**
- **Wikipedia: Message Passing Interface (http://en.wikipedia.org/wiki/Message_Passing_Interface)**
- **MPICH2 User's Guide (http://www.mcs.anl.gov/research/projects/mpich2/documentation/files/mpich2-doc-user.pdf)**

References

- Sloan, Joseph D.. High Performance Linux Clusters. 1st ed. Sebastopol, CA: O'Reilly Media, Inc., 2005.
- **http://www-unix.mcs.anl.gov/mpi/mpich/**
- **Wikipedia: Message Passing Interface (http://en.wikipedia.org/wiki/Message_Passing_Interface)**
- **MPICH2 User's Guide (http://www.mcs.anl.gov/research/projects/mpich2/documentation/files/mpich2-doc-user.pdf)**

Installing MPICH

This is part one of a multi-part tutorial on installing and configuring **MPICH2**. The full tutorial includes

- **Installing MPICH**
- **MPICH: Pick Your Paradigm**
- **MPICH with Torque Functionality**
- **OR MPICH without Torque Functionality**
  - **MPICH: Starting a Global MPD Ring**
  - **MPICH: Troubleshooting the MPD**

I'll be installing MPICH2 from source. If you're unfamiliar with installing from source, the **Source Installation Paradigm** is a (gentler) introduction to it than this page.

Getting the Source

Getting the source is the first place to start when installing MPICH. Unfortunately, the `apt-get` package for MPICH isn't up-to-date, so while `apt-get install mpich-bin` will work, it will install an older version.

You'll want to keep your scientific software in one place on the **NFS mount**. My mount is kept at `/shared`, and I'm keeping all my sources in `/shared/source`. The first thing I did is to `cd /shared/source`. After doing that, grab the source code (often called the "tarball") from **the MPICH2 home page**. Choose the one under "All (Source)". After you've found the full link to the tar file, you can do

```
wget http://www-unix.mcs.anl.gov/mpi/mpich/downloads/mpich2-1.0.5p4.tar.gz
```

(making sure that the one you're wgetting is the most recent). After it's finished, untar it with

```
tar xvf mpich2*.tar.gz
```

Configuration

First, in order to install MPICH, you'll need to have Python installed. Do this with

```
apt-get install python2.5
```

Then, make sure you're in the folder that was created with the `tar xvf` (such as `/shared/source/mpich2-1.0.5p4` for me) in these next few steps.

MPICH has many, many configuration options. For a full list of them, do `./configure -help` (or, on a machine that you can't scroll on, do `./configure -help | more` to pipe the output to the program called more so that it doesn't all scroll past you at once). These are all choices that can help configure MPICH for your system, including areas like which optional parts of MPICH to install and where on your drive MPICH should install itself.

One of the more important options is `-prefix=<option>`. `prefix` is where you specify the prefix for all the directories MPICH will create. In other words, this is the directory where MPICH will install. If you're installing it onto your shared NFS partition, like I am, you'll want to put

```
--prefix=/shared/
```

Other important options include which bindings MPICH should support, meaning which programming languages. (Make sure you've installed the **compilers** before installing MPICH. It won't work unless you do! MPI supports C, C++, and Fortran. These are specified with

```
--enable-cxx
--enable-f77
--enable-f90
```

`cxx` stands for C/C++. `f77` is the Fortran 77 compiler, and `f90` is the Fortran 90 compiler.

If you'll be running MPI on machines that have more than one processor or core, you'll also want support for threading. Threading takes advantaged of the shared memory. Enable threads with the following two additions.

```
--enable-threads=multiple
--with-thread-package=posix
```

Finally, if you or your users will want to using debug programs at some point, you need to enable debugging support now, with

```
--enable-debuginfo
```

All of these options need to be given when running `./configure`. The full line, if you're using threads, looks like this:

```
./configure --prefix=/shared/ --enable-cxx --enable-f70 --enable-f90
--enable-threads=multiple --enable-debuginfo
```

When you run this, `./configure` will check your system and get `make` set up to install correctly. You'll want to scan back up through the output when it's completed to look for any errors. You also want to make sure that running the above command finishes with "Configuration completed."

Make

At this point, you're ready to run `make` from the same directory (the /mpich2-1.0.5p4 or whatever your version is one). (See the **Source Installation Paradigm** page for help with make errors.)

If it doesn't finish with errors, hooray! The phrase "Make completed" should also be one of the last lines.

Installing

Installing should be as easy as

```
make install
```

Again, if it doesn't finish with errors, you should be in business. (In this case, lack of feedback is a good thing.) `make install` also finishes with a very important message:

```
/shared/sbin/mpeuinstall may be used to remove the installation
```

This will come in handy if you want to reconfigure or update MPICH in the future and need to remove the old version.

Sanity Check

After **installing MPICH**, it's important to do a sanity check and make sure the install really worked and everything was put in its proper place. First, do an `ls` in the `/bin` subdirectory of the directory you told MPICH to install into (indicated with the -prefix= option given to `./configure`). In my case, this is

```
ls /shared/bin
```

You should have a lot of executables starting with `mpd`. If you don't, the installation didn't finish successfully, so go back and retrace your steps. You'll also want to run `which mpd` as root and as a user to make sure the directory where the MPICH commands are located are in the proper place. You should see something like `/shared/bin/mpd` returned; if you see nothing, see **Bash Profile Modifications: PATH Additions**.

If everything is go, continue on to **MPICH: Pick Your Paradigm**.

MPICH: Pick Your Paradigm

This is part two of a multi-part tutorial on installing and configuring **MPICH2**. The full tutorial includes

- **Installing MPICH**
- **MPICH: Pick Your Paradigm**
- **MPICH with Torque Functionality OR MPICH without Torque Functionality**

With or Without Torque Functionality?

MPICH has the flexibility to be run a number of different ways. Specifically, I'll be covering how to run MPICH with the resource manager/scheduler **Torque** as well as how to run MPICH without a scheduler. If you know which you'll be using, continue on. Otherwise, see below.

- **MPICH with Torque Functionality**
- **MPICH without Torque Functionality**

Overview

Most clusters will need a **scheduler/queuer**. The rare exceptions may be a single-user cluster. If you're setting up a cluster solely for yourself, and you want to monitor when your jobs are running rather than having your jobs be scheduled for you, you might not need one. (Still, you'd be saving yourself a lot of monitoring overhead if you install one, so I'd still recommend it!)

MPICH can take advantage of this with the installation of a special `mpiexec` binary.

Torque-Aware Mpiexec

When running an mpi job, this special mpiexec will take of all the MPI setup and teardown. Users will simply specify how many processes they want, and by communicating with Torque to find out how many each alloted node has, mpiexec will automatically run the nodes allocated for a job.

However, this special version of mpiexec cannot be run outside of a Torque job submission. In other words, users will not be able to test their code without submitting it to Torque, and you won't be able to test any software running over MPICH until you've installed Torque. (This can be overcome by only installing the special binary on the worker nodes, and will be covered as part of the tutorial.)

I highly recommend taking this route.

Stand Alone MPICH

Without Torque (or a different scheduler), the above option is not possible.

Each user will need a special `.mpd.conf` file, and `/etc/mpd.conf` will need to be set up on each of the worker nodes. Then an "mpd ring," a collection of MPI daemons communicating with each other, is set up across the cluster.

If Torque is installed, it will not be able to enforce which machines users access with any MPI programs; hopefully, users will "play nice" and only use their allocated machines, but this may not be the case. Alternatively, users can start their own mpd rings within Torque submission scripts, but problems will arise if a node with multiple processors is scheduled to run two MPI jobs from the same user, since only one mpd per user can be started per machine.

I recommend taking this route only if you do not have Torque installed.

MPICH with Torque Functionality

This is part three of a multi-part tutorial on installing and configuring **MPICH2**. The full tutorial includes

- **Installing MPICH**
- **MPICH: Pick Your Paradigm**
- **MPICH with Torque Functionality**

The "Other" Mpiexec

Congratulations! By going this route, you're saving yourself a lot of hassle by avoiding maintaining an mpd ring on the cluster, or helping your users set up and tear down rings. If that doesn't make sense, that's fine - that's the beauty of this option. MPI will take care of all the details for the users.

This functionality is added by using a new binary. Instead of the binary called `mpiexec` that came with the MPICH2 installation, a new `mpiexec` is installed. Unfortunately, they have the same name, but they work very differently. The "other" mpiexec - the one with Torque-functionality that we need to add to the cluster - is produced out of **Ohio Supercomputer Center**. This mpiexec only works with Torque. In other words, users will not be able to use it outside of a **qsub script**.

For this reason, you might want to only put the new mpiexec on the worker nodes, and keep the original binary for users to run outside of Torque on the head node.

Prep

Before starting, the original mpiexec needs to be moved (at least on the worker nodes) so that it is no longer in root's or users' paths (or just renamed). Find it with

```
which mpiexec
```

and then move it somewhere else as a backup.

Installation

To download the latest version of mpiexec, visit the **OSC mpiexec page** and scroll down to the downloads section. Right click on the file location. Then, from whenever your source is stored, run

```
wget http://www.osc.edu/~pw/mpiexec/mpiexec-0.83.tgz
```

or whatever the latest version is. Untar it with

```
tar xvzf mpiexec*.tar.gz
```

and then `cd` into the new directory. Mpiexec follows the standard **source installation paradigm**. Run

```
./configure --help
```

to see a list of options. Important options include

- `--prefix=` - specify where you want to have the binaries installed. They need to be accessible by all of the worker nodes. An **NFS mount** would be a good choice.
- `--with-pbs=` - necessary to get the Torque functionality! Specify the location of the Torque installation. If you followed **my Torque tutorial**, it's located at `/var/spool/pbs`
- `--with-default-comm=mpich2-pmi` - used to indicate which version of MPI

Next, run `./configure` with all the options necessary. My command looked like this:

```
./configure --prefix=/shared --with-pbs=/var/spool/pbs/
--with-default-comm=mpich2-pmi
```

Pbs_iff

To do this next part, **Torque** will need to already be installed. Mpiexec requires that a file named `pbs_iff` be on each one of the worker nodes. Normally, this file is only located on the head node and isn't installed as part of the pbs_mom installation, so it needs to be copied out from the head node to each of the other nodes.

There's an easy way to do this by scripting. The first requirement is to have a file with each of the worker nodes listed in it. Assuming Torque is running, this can be generated with

```
pbsnodes | grep -v = | grep -v '^$' >> machines
```

- `grep -v =` excludes all lines that have an equal sign in them
- `grep -v '^$'` contains a regular expression to delete all empty lines

The "machines" file of all the worker node names can then be used in a quick script to copy pbs_iff to each of the worker nodes. Find the original file with

```
updatedb && locate pbs_iff
```

(If you receive an error, `apt-get install locate` and then try again.) Then, replacing my locations below for the location you found it on your cluster, run

```
for x in `cat machines`; do rsync /usr/local/sbin/pbs_iff $x:/usr/local/sbin/; done
```

Next, pbs_iff needs to have its permissions changed to setuid root. (This means the binary runs with root privileges, even when run by a different user.) Again, to do this across all the worker nodes at once, use a script and make sure the location is correct for your setup:

```
for x in `cat machines`; do ssh $x chmod 4755 /usr/local/sbin/pbs_iff; done
```

Without these steps, users trying to run mpiexec will see errors like these:

```
pbs_iff: file not setuid root, likely misconfigured
pbs_iff: cannot connect to gyrfalcon:15001 - fatal error, errno=13 (Permission denied)
   cannot bind to reserved port in client_to_svr

mpiexec: Error: get_hosts: pbs_connect: Unauthorized Request .
```

Testing

Trying to run a program with `mpiexec` outside of a Torque job, it will give an error:

```
mpiexec: Error: PBS_JOBID not set in environment.  Code must be run from a
  PBS script, perhaps interactively using "qsub -I".
```

At least it's a helpful error! Therefore, in order to test it, mpiexec will need to be called from within a script. Continue on to the **Torque and Maui: Submitting an MPI Job** page to test.

References

- **Mpiexec - MPI parallel job launcher for PBS (http://www.osc.edu/~pw/mpiexec/index.php)**
- **Mpiexec Readme (http://svn.osc.edu/repos/mpiexec/trunk/README)**

MPICH without Torque Functionality

This is part three of a multi-part tutorial on installing and configuring **MPICH2**. The full tutorial includes

- **Installing MPICH**
- **MPICH: Pick Your Paradigm**
- **MPICH without Torque Functionality**
  - **MPICH: Starting a Global MPD Ring**
  - **MPICH: Troubleshooting the MPD**

MPICH without Torque Functionality

In this paradigm, users (and sysadmins) are responsible for much more overhead, and many more possibilities for misconfiguration occur. Unless you are planning not to have a **scheduler and queue**, I highly recommend setting up **MPICH with Torque Functionality** instead.

Creating Mpd.conf Files

Mpd stands for "multi-purpose daemon." MPICH2 separates process communication from process management by having an mpd process manage any MPI processes. The mpd's need a little extra customization that wasn't necessary in MPICH1. To begin with, each user that will (potentially) use MPICH needs to have a mpd.conf file.

User Accounts

Each user needs a `.mpd.conf` file in his/her home directory, and must only be readable and writable by that user. The file should look like this:

```
MPD_SECRET_WORD=yours3cr3tw0rd
MPD_USE_ROOT_MPD=yes
```

- `MPD_SECRET_WORD` can be unique to each user, but doesn't have to be
- `MPD_USE_ROOT_MPD` specifies that users will not start up their own mpd daemons, but will rely upon attaching to one already running under the root account

You can run a simple script to create this file for all of your users. First, create the file as shown above in root's home directory (`/root`). Make it only readable/writable by it's owner, root, by running `chmod 600 /root/.mpd.conf`. Then to make a copy for each of the user accounts, run

```
for x in `ls /shared/home/`; do rsync -plarv /root/.mpd.conf /shared/home/$x/; chown $x:users /shared/home/$x/.mpd.conf; done
```

Of course, replace all instances of `/shared/home/` with the location where your users' home directories are stored, and replace `users` with whatever group your users are in.

Future Users

In order to make sure new users automatically have this file created for them, create a `.mpd.conf` file in `/etc/skel/` on the machine you create user accounts from. Make sure it has a secret word and root mpd flag as shown above. Then, run

```
chmod 600 /etc/skel/.mpd.conf
```

This will make sure the file is created with only the user having read/write permissions.

Root's .mpd.conf

Similar to the users, root needs to have a `mpd.conf` file on each one of the worker nodes. For root, rather than being stored in a home directory, this file is located at `/etc/mpd.conf`. This file only needs one line:

```
MPD_SECRETWORD=yours3cr3tw0rd
```

You can replace `yours3cr3tw0rd` with whatever you'd like. Then, make sure it's only readable/writable by root:

```
chmod 600 /etc/mpd.conf
```

Once you've created this file on one of your nodes, you can manually create it on each of the other nodes, or see the **Cluster Time-saving Tricks** page for tips on how to script this process. Make sure that the secret word matches on all of the worker nodes.

Running on One Machine

It's wise to do a test MPI run on one machine before setting it up to run on multiple machines.

Starting a Daemon

Choose one of your worker nodes to test on. SSH into that machine and, as root, start up an mpd (multi-purpose daemon) to run in the background with

```
mpd --daemon
```

Next, type `mpdtrace -1` to verify that mpd has been starting on this host. You should see your host name with an underscore with a number (this is the MPD id) and host IP address returned to you.

Running an MPI Program

Once you've got the mpd daemon up and running, it's time to open your favorite text editor and type up an MPI program. A "hello world" type program is ideal for this, and I'll be borrowing one from the **Bootable Cluster CD project**. For this, you'll want to be on a user account, not root. Follow the instructions on **Creating and Compiling an MPI Program**.

Once you've successfully compiled it, you're ready to run it. Still as the user account (not as root), first run

```
mpiexec ./hello.out
```

You should see a message printed out from the server process, but that's it. Without specifying a number of processes, MPICH automatically only uses one process. Run it again and specify multiple processes:

```
mpiexec -np 5 ./hello.out
```

This time you should see something like this:

```
kwanous@eagle:~/mpi$ mpiexec -np 5 ./hello.out
Hello MPI from the server process!
Hello MPI!
 mesg from 1 of 5 on eagle
Hello MPI!
 mesg from 2 of 5 on eagle
Hello MPI!
 mesg from 3 of 5 on eagle
Hello MPI!
 mesg from 4 of 5 on eagle
```

Notice by the host name (in my case, `eagle`, that all of these processes are still running on the same node. To set this up across multiple nodes, first kill the mpd daemon with `mpdallexit`, then continue on to

Configuring Worker Nodes to use Root's MPD Daemon

Starting up an mpd daemon for each user each time they log in is doable, but that requires an extra step of complexity for your users to understand. Plus, they'll need to remember to start up daemons on multiple machines when they run programs that require multiple processors (not just multiple *processes*).

An easier paradigm to follow is to start a single mpd daemon on each of the worker nodes and have users' programs attach to that daemon. Continue on to **MPICH: Starting a Global MPD Ring** to implement this.

MPICH: Starting a Global MPD Ring

This is part four of a multi-part tutorial on installing and configuring **MPICH2**. The full tutorial includes

- **Installing MPICH**
- **MPICH: Pick Your Paradigm**
- **MPICH without Torque Functionality**
  - **MPICH: Starting a Global MPD Ring**
  - **MPICH: Troubleshooting the MPD**

Before you set up a ring of root mpd daemons, make sure MPICH is working correctly on a single machine. See the **MPICH without Torque Functionality** page for more information.

Mpd.conf Files

You will absolutely need mpd.conf files for any users and for root on each of the worker nodes in order for this to work. If you don't already have this set up, you can follow the instructions on the **MPICH without Torque Functionality** page to do.

Password-Less SSH

Password-less SSH will also need to be setup for all users. See the **Password-less SSH for Users** page for information on how to do this.

Starting the Mpd Ring

Starting the First Node

Once you have `/etc/mpd.conf` in place on all of your worker nodes, an mpd daemon needs to be started on each one of the worker nodes. These will be used to manage any MPI processes. The first node started up serves as a kind of focal point for all of the other mpd's. For this reason, it's important to choose (and remember) a specific node as the head MPD node.

Start by ssh'ing into this special first node, and then running

```
mpd --daemon --ncpus=<# CPUs>
```

The `--daemon` part specifies that this should be run in the background, and that the process shouldn't be killed when the SSH session ends.

Next, in order to know where exactly this daemon is running, in order to have other daemons attach to it, run `mpdtrace -l` as shown below:

```
owl:~# mpdtrace -l
owl_60519 (192.168.1.202)
```

You'll need the value after the underscore (_): this is the random port that the daemon is waiting for

communication on.

Starting the Other Nodes

Then, on the other nodes, a slightly more complicated `mpd` command is needed:

```
mpd --daemon --host=<your first host> --port=<port found with mpdtrace> --ncpus=<#
CPUs>
```

Do this one at a time on each of the other worker nodes, or see the **Cluster Time-saving Tricks** page to learn how to script it up. If you have any trouble, the **MPICH: Troubleshooting the MPD** page might help.

Checking the MPD Ring

Once you've started up an mpd daemon on each one of the worker nodes, ssh into one of the worker nodes and run

```
mpdtrace
```

This will show you all of the hosts currently hooked up as part of the ring. All of the worker nodes should be listed here. To get a quick count, run

```
mpdtrace | wc -l
```

If any are missing, those nodes should be further investigated and attempt made again to start up an mpd daemon on them.

Sanity Check: Running an MPI Program on Multiple Nodes

After the ring has been set up, it's finally time to try running an MPI job on multiple nodes. SSH into one of the worker nodes, become one of your user accounts, and follow the instructions at **Creating and Compiling an MPI Program**.

As when running multiple processes on the same machine, run the program with `mpiexec`. First, specify a number of processes smaller than or equal to the number of cpus you specified for this worker node. In my case, that's four or less.

```
mpiexec -np 4 ./hello.out
```

You should see the same hostname listed for all of the processes. This is because the mpd daemon will use all available CPUs on the host you're running on before branching out to CPUs on other hosts. To see this spread further than just one machine, ramp up the number of processes to higher than the number of cpus on this host.

```
mpiexec -np 7 ./hello.out
```

You should now be seeing different hostnames appearing in the list. The mpd on this machine automatically contacts other mpds in the ring when the host it's running on runs out of CPUs. (In MPICH1, you would have needed to specify this with a machinefile.) Pretty cool!

MPICH: Troubleshooting the MPD

This is part five of a multi-part tutorial on installing and configuring **MPICH2**. The full tutorial includes

- **Installing MPICH**
- **MPICH: Pick Your Paradigm**
- **MPICH without Torque Functionality**
  - **MPICH: Starting a Global MPD Ring**
  - **MPICH: Troubleshooting the MPD**

Stuff

If you try to start a daemon without it (using `mpd`), you'll get the following error.

```
gyrfalcon:/shared$ mpd
configuration file /shared/home/kwanous/.mpd.conf not found
A file named .mpd.conf file must be present in the user's home
directory (/etc/mpd.conf if root) with read and write access
only for the user, and must contain at least a line with:
MPD_SECRETWORD=<secretword>
One way to safely create this file is to do the following:
  cd $HOME
  touch .mpd.conf
  chmod 600 .mpd.conf
and then use an editor to insert a line like
  MPD_SECRETWORD=mr45-j9z
into the file.  (Of course use some other secret word than mr45-j9z.)
```

The error is explicit about what needs to be done to create an mpd.conf file. If you're running this as a user, create ~/.mpd.conf, or create /etc/mpd.conf for the root account. Create some secret word; the word will be used to distinguish your processes from other people's and to keep them separate. This word can be just about anything but standard password requirements (more than six characters long, containing at least one number and at least one letter) help make it more secure. Follow the instructions from the error message to insert this with the proper syntax and to change the permissions on the file. (If you don't change the permissions, you'll see something like

```
gyrfalcon:~$ mpd
configuration file /shared/home/kwanous/.mpd.conf is accessible by others
change permissions to allow read and write access only by you
```

Start mpd as a daemon in the background using

```
mpd --daemon
```

Without these arguments, on some systems you'll see an error like this:

```
kwanous@gyrfalcon:~$ mpd
gyrfalcon_53084 (mpd_sockpair 226): connect -2 Name or service not known
gyrfalcon_53084 (mpd_sockpair 233): connect error with -2 Name or service not known
```

Missing Root's /etc/mpd.conf

Sometimes you'll see an error like this:

```
osprey:~# mpdtrace -l
/shared/bin/mpdroot: open failed for root's mpd conf filempdtrace (__init__ 1171
): forked process failed; status=255
```

You'll get this message when running as root if `/etc/mpd.conf/`, root's version of `~/.mpd.conf`, doesn't exist. Use the same syntax (show in the error above) for creating the root version as for creating a user version. Once you create it, if you don't change the permissions to only be readable by root, you'll see a more helpful error:

```
osprey:~# mpdtrace -l
configuration file /etc/mpd.conf is accessible by others
change permissions to allow read and write access only by you
```

Use `chmod 600 /etc/mpd.conf` to do this and it should work.

Python Error

As of this writing, mpd (the MPI daemon) is a python program and requires the python 2.4 binary in order to run. If you don't have python installed on the machine you're trying to use MPI with, you'll see an error like this:

```
eagle:~# /usr/bin/env: python2.4: No such file or directory
```

Fortunately, it's easy enough to fix. All the hosts you're going to use MPI with need to issue

```
apt-get install python2.4
```

If this still doesn't work, try uninstalling all versions with

```
apt-get remove --purge python2.4 python
```

running

```
apt-get autoremove
```

and then finally running the apt-get install again.

If you need to do this on all of your nodes, rather than sshing into each one and doing it individually, check out the **Cluster Time-saving Tricks**.

Gromacs: Molecular Dynamics

About Gromacs

Like **NAMD**, Gromacs is a molecular dynamics program. GROMACS stands for GROningen MAchine for Chemical Simulation, originally developed at the The University of Groningen in the Netherlands. In addition, the Gromacs gmxbench suite is commonly used for benchmarking.

If you're looking for a how-to on Gromacs installation, **skip ahead** to the next section. Otherwise, read on for a basic description of the three different file types that Gromacs takes as input.

Gromacs Files

Each example file in the gmxbench suite contains three files - a .gro, .mpd, and .top. Each of these files is used by the Gromacs preprocessor (grompp), a binary that prepares the code to be run in parallel. These are what set up the simulation to be run.

.gro Files

conf.gro and .gro files in general specify the initial positions and velocities of the atoms. Output files with the results from Gromacs runs will also have this format, called Gromos87 format. Each line gives the information for one atom. A typical line might look like this:

```
 9ALA      N   87   3.492   2.996   1.618  0.1574  0.2410 -0.0154
```

- 9ALA is the name of the residue (group of atoms) that this atom belongs to
- N is the kind of atom (nitrogen in this case)
- 87 is the unique id assigned to this atom
- 3.492, 2.996, 1.618 are the x, y, and z positions of the atom
- 0.1574, 0.2410, -0.0154 are the x, y, and z velocities of the atom

For more information, see the **gro page** on the GROMACS online reference manual.

.mdp Files

.mpd files, including the file grompp.mdp included in each example gromacs setup, is a list of parameters to pass to grompp. These include directives about and for the preprocessor. It also includes some of the basic parameters about how to run the molecular dynamics simulations, including what types of processing to use and what form the output should take.

For more information, see the **mdp options page** on the GROMACS online reference manual.

.top Files

.top files, including the topol.top files included in the example, are files that grompp uses to construct the topology of a simulation, including the atoms, bonds, and angles.

For more information, see the **top example page** on the GROMACS online reference manual.

Installing Gromacs

Installing Gromacs is relatively painless and easy to do. Continue on to **Gromacs: Installation**.

References

- **GROMACS: Fast, Free and Flexible MD (http://www.gromacs.org/)**
- H.J.C. Berendsen, D. van der Spoel and R. van Drunen: GROMACS: A message-passing parallel molecular dynamics implementation Comp. Phys. Comm. 91 pp. 43-56 (1995). Available online at **http://www.gromacs.org/documentation/articles/berendsen95a.pdf**

Gromacs: Installation

This is the second part of a two-page tutorial on Gromacs. The full tutorial includes

- **Gromacs: Molecular Dynamics**
- **Gromacs: Installation**

Installing Gromacs

Gromacs should be installed from source, because compilation on the hardware it will be running on is faster than a general compilation (and guaranteed to have all the necessary binaries). Further, while there is a Debian package for gromacs, it isn't always up-to-date.

Supporting Packages

Gromacs relies upon FFTW, the Fastest Fourier Transform in the West, to do optimized Fourier transforms, and should be installed prior to compiling gromacs. To get it, run

```
apt-get install fftw3 fftw3-dev sfftw-dev sfftw2
```

`fftw3` (although not the dev packages) needs to be installed on all of the worker nodes. You can do this one node at a time, or see the **Cluster Time-saving Tricks** page to learn some scripting techniques.

If you don't install `fftw3` on all of the worker nodes, sooner or later down the line, you'll get error messages like this:

```
grompp: error while loading shared libraries: libfftw3f.so.3:
  cannot open shared object file: No such file or directory
```

Getting and Building Gromacs

The gromacs source code is available online at **the Gromacs website**. Right click the most recent .tar.gz file and then, from wherever you keep source code, use `wget` to download it. The compiled source will need to be available to all of the worker nodes, so you may want to keep the source code on an **NFS mount** as well. (For instance, my NFS mount is on `/shared`, and I'm keeping worker node source at `/shared/usr/local/src`.) From the correct directory, the wget command should look similar to this:

```
wget ftp://ftp.gromacs.org/pub/gromacs/gromacs-3.3.3.tar.gz
```

Once it has finished downloading, untar the file with

```
tar xvzf gromacs*.tar.gz
```

and then `cd` into the new directory. Gromacs follows the typical **source install paradigm**, so no surprises there. *Just make sure you're compiling it on one of your worker nodes!* Run `./configure` to see all the available options for compiling. Important ones include

- `--prefix=<location>` should be used to specify where to install, otherwise files will be placed in /usr/local/gromacs/
- `--enable-mpi` to enable MPI functionality - assuming you have **MPICH** installed
- `--with-fft=fftw3` to tell it to use the FFTW package just installed

After deciding upon any other options, run `./configure` will all of them. For instance, the line I ran was

```
./configure --prefix=/shared --enable-mpi --with-fft=fftw3
```

When compiling with MPI support, it should finish with a nice little message like this:

```
* Seems you are compiling with MPI support. You can install the MPI-
  enabled programs with suffixed names to have both MPI and non-MPI
  versions. This is useful e.g. on supercomputers where you usually
  cannot run MPI-linked programs on the login node.
  Set a suffix with e.g. --program-suffix=_mpi (or _mpi_d for double).
  You only need MPI for mdrun, so if you already have non-MPI stuff
  installed you can issue make mdrun; make install-mdrun.
```

After this, run `make`. If it finishes without errors, run `make install`. At the end of the `make install`, you should get a message like this:

```
GROMACS is installed under /shared.
Make sure to update your PATH and MANPATH to find the
programs and unix manual pages, and possibly LD_LIBRARY_PATH
or /etc/ld.so.conf if you are using dynamic libraries.

Please run "make tests" now to verify your installation.

If you want links to the executables in /usr/local/bin,
you can issue "make links" now.
```

Testing the Gromacs Install: Gromacs Test Files

At this point, the installation files should be copied out to <prefix>/bin and also <prefix>/share. installation verification sounds like a good idea. However, the test source code isn't include as part of the original tar file. Check the **gromacs site** for the most recent version of the test files, and then (still inside the gromacs source directory), wget them. For instance,

```
wget http://www.gromacs.org/content/view/169/210/
```

then untar the file,

```
tar xvzf gmxtest*
```

Then go ahead and do `make tests`.

The first time I ran it, I received these errors:

```
sh: line 1: 26201 Aborted                 mdrun >mdrun.out 2>&1
FAILED. Check files in rbl
1 out of 16 simple tests FAILED
FAILED. Check files in dec+water
1 out of 14 complex tests FAILED
All 63 kernel tests PASSED
N      Reference   This test
 10    -33.9883    -29.6936
 11    -33.9883    -29.6936
There were 2 differences in final energy with the reference file
All 45 pdb2gmx tests PASSED
```

This means there were differences in the output between the reference file and the values seen. The differences (in my case), could be viewed in `complex/dec+water/checkpot.out`, but I didn't find a file to view for rb1. It's up to you how much difference you're willing to accept.

Check the Gromacs Install: Gmxbench Suite

Another great way to test the Gromacs installation is to install the Gromacs benchmarks suite, gmxbench. These are available at the **gromacs site**. Visit this site and right-click the .tar.gz file to copy the location of the newest set of benchmarks. Then, from where you keep your source code that needs to be accessible by all the worker nodes, run the following (replacing the location if necessary):

```
wget ftp://ftp.gromacs.org/pub/benchmarks/gmxbench-3.0.tar.gz
```

Make a new directory called gmxbench (`mkdir gmxbench`) and copy the tar file into it (`mv gmxbench-* gmxbench`). Then, `cd` into the new directory and untar the file:

```
tar xvzf gmxbench*
```

Four new directories will be created; each one of these is a sample of Gromacs input to be run as a benchmark. I'm going to run the villin benchmark. This simulates the folding of a 36-amino acid protein that fits on the top of the villin molecule. `Cd` into `d.villin`.

Before running gromacs, the Gromacs pre-processor (grompp) is used to prepare the code to run in parallel. (Make sure you either have an **mpd ring** started, or that you're doing this from within a **qsub script** if you're using the Torque-enhanced MPICH - see below for a sample qsub script.) From within the `d.villin` directory, run

```
grompp -np 4 -shuffle -sort -v
```

- `-np` indicates the number of processors. I'm using four because I have four processors per machine.
- `-shuffle` distributes the loads equally amongst the processors.
- `-sort` indicates that atoms should be sorted by their coordinates, further helping parallelization.
- `-v` stands for verbose, meaning to output to the screen as it is running.

If it only finishes with a warning or two, you're good to go. Each time grompp or gromacs runs, it also gives a quote at the bottom of the output. Just FYI, not all of these are appropriate for a classroom setting!

Next, still from within the Villin directory, it's time to run gromacs using mpi. Yours should look along the lines of this:

```
mpiexec -np 4 /shared/bin/mdrun -v -c output.gro
```

- `-v` specifies to be verbose
- `-c` specifies the output file

It should finish with some nice benchmarking statics about the system it was run on.

Gromacs in Qsub

If you're using the **MPICH with Torque functionality**, not the one installed by default by the MPICH2 installation, then you won't be able to run mpirun or mpiexec outside of a qsub script. (Unless you kept the old binaries hanging around with a different name, like mpirun_old and mpiexec_old.) Not a problem, since you'll want to test what your users will be seeing and using anyway.

Below is a sample qsub script for testing the villin benchmark. Make sure you change `GPATH` to the location on your NFS mount where the villin codes are located, and `WORKDIR` to where you'd like the final output spat out. In the example below, 8 nodes are used; this can be changed as well.

```
#PBS -N villin
#PBS -l nodes=8
cd $PBS_O_WORKDIR
GPATH=/shared/usr/local/src/gmxbench/d.villin
WORKDIR=~/gromacs
grompp -np 8 -shuffle -sort -f $GPATH/grompp.mdp -p $GPATH/topol.top -c $GPATH/conf.gro
mpiexec /shared/bin/mdrun -v -c output.gro
mv output.gro $WORKDIR
rm output.gro
exit 0
```

NAMD: Molecular Dynamics

About NAMD

NAMD molecular dynamics project out of the Theoretical and Computational Biophysics Group at University of Illinois at Urbana-Champaign. The project is based on Charm++ and uses VMD for molecular graphics. At the time of this writing, the most current version is 2.6.

This tutorial will cover building NAMD and charm++ for MPI rather than "net-linux".

Building the whole project requires multiple steps:

- Installing the support packages (shown below)
- **Building and testing charm++**
- **Building namd**
- **Building vmd**

Also, I have a page with all of the errors I encountered while figuring out this process. If you follow the tutorials, hopefully you shouldn't see them, but sometimes it's easy to miss a step.

- **NAMD: Troubleshooting Errors**

Support Packages

Most of NAMD will need to be installed from source, but two supporting packages can be installed with apt: `fftw` and `tcl`.

FFTW is responsible for doing optimized Fourier transforms. To get it, run

```
apt-get install fftw3 fftw3-dev sfftw-dev sfftw2
```

Note that fftw3 is a *virtual package*. When you try to apt-get it, Debian installs `libfftw3-3` for you instead. This is fine.

Next, we need Tcl, the "Tool Command Language," which will give additional functionality to NAMD. Install it with

```
apt-get install tcl8.4 tcl8.4-dev
```

This will need to be done on each of the worker nodes. You can do it by hand or visit the **Cluster Time-saving Tricks** page to learn how to write a script to automate it.

Next...

Next, continue on to **building and testing charm++**.

References

- **NAMD - Scalable Molecular Dynamics (http://ks.uiuc.edu/Research/namd/)**

NAMD: Building charm++

This is part two of a four-part tutorial on building and testing NAMD and charm++. The full tutorial includes

- **Installing support packages**
- **Building and testing charm++**
- **Building namd**
- **Building vmd**

There is also a troubleshooting page:

- **Troubleshooting errors**

Getting the NAMD and Charm++ Source

Now that fftw and tcl have been installed, it's time to move onto NAMD and Charm++. Visit **http://www.ks.uiuc.edu/Research/namd/2.6/download/732581/** and copy the location for the source file (in case there's a more recent one than the time of this writing). You'll want to do all your work on this in a place that's accessible from every worker node, such as an NFS mount, because the binaries will still need access to some of the original untarred directories.

For instance, I normally keep my source code in `/usr/local/src`, but that's only when I only need the source code accessible to the head node. This time, I'm keeping the source code on my **NFS mount**, `/shared`, under `/shared/usr/local/src`. Make sure the source code is available to all the worker nodes. From the directory you're selecting, run

```
wget http://www.ks.uiuc.edu/Research/namd/2.6/download/732581/NAMD_2.6_Source.tar.gz
```

Untar the file with

```
tar xvzf NAMD*.tar.gz
```

and then move into the directory that was created. You should see the source code for Charm++ in the new directory; untar it as well.

```
tar xvf charm*.tar.gz
```

Symlinking Compilers

Before going into building charm++, there's one potential hiccup that needs to be moved out of the way. Apparently, charm++ doesn't like being compiled with 4.x GNU compilers. The fix for this is relatively easy. First, make sure you have an older version of gcc and g++ with

```
apt-get install g++-3.4 gcc-3.4
```

This will put the binaries in `/usr/bin`. If you do `ls -al | grep gcc`, you should see that `gcc` is just a symlink to the binary with the version information. Both the symlinks for `gcc` and `g++` need to temporarily changed to point to the 3.4 versions. To do this, cd into `/usr/bin` and remove the old symlinks with

```
rm gcc g++
```

Then, create new ones with

```
ln -s gcc-3.4 gcc
ln -s g++-3.4 g++
```

Now you're ready to proceed with building charm++. Afterward, to reverse the process, remove the existing symlinks again, and then recreate the previous ones using

```
ln -s gcc-4.3 gcc
ln -s g++-4.3 g++
```

Building Charm++...

Charm should be installed before NAMD, since NAMD will use it. Move into the new charm directory (`cd charm*`). Unlike most scientific packages, Charm doesn't follow the traditional **source installation paradigm**; there's no ./configure script. Instead, all of the action happens with `./build`.

```
./build --help
```

will show all of the options. A few important ones include

- `charm++` - you just want the base installation of Charm++
- `mpi-linux` - use Linux and **MPI** as the platform, assuming you have it installed, otherwise use net-linux
- `--base-dir=` - use this to point to where your MPICH libraries are. For instance, mine are on my **NFS mount**.
- `gcc` - compiler to use
- `-O` - use optimization

You'll need to modify this based on your system - for instance, if you're using a 64-bit architecture. The line I ran was

```
./build charm++ mpi-linux --basedir=/shared gcc -O
```

Once it finishes (and it may take a while), you'll see this message:

```
------------------------------------------------
charm++ built successfully.
Next, try out a sample program like tests/charm++/simplearrayhello
```

Testing the Charm++ Build

Now you're ready to test the installation. The test I'm using is the same as "notes.txt" provided in the NAMD tarball. Before testing, however, open up `charm-5.9/include/conv-mach.sh` and find the line that starts with `CMK_LIBS`. Add `-pthread` it to look like this:

```
CMK_LIBS='-lckqt -lmpich -pthread'
```

Let's finally test the build. You should see a new directory created in `/charm-5.9` after the installation finishes, named after the type of installation you just finished. For instance, my directory is called `mpi-linux-gcc`. Cd into the new directory, then into `tests/charm++/megatest/`, and run

```
make pgm
```

Then, make sure you have an mpd daemon running (`mpd --daemon` to run it as a daemon), and run the binary that was created with

```
./charmrun -np 2 ./pgm
```

It may take quite awhile. With two processes, mine takes about ten minutes, and an especially long amount of time on the "immediatering (gengbin)" steps. If it finishes successfully, you'll see

```
All tests completed, exiting
End of program
```

and then charm++ should be good to go. Clean up your tracks by killing the mpd with `mpdallexit` and you can continue on to **building NAMD**.

NAMD: Building namd

This is part three of a four-part tutorial on building and testing NAMD and charm++. The full tutorial includes

- **Installing support packages**
- **Building and testing charm++**
- **Building namd**
- **Building vmd**

There is also a troubleshooting page:

- **Troubleshooting errors**

Getting C Shell (csh)

You'll need to get CSH, the **C shell**, if you don't have it already. (Most base installations of Debian won't have it.) You can get this with

```
apt-get install csh
```

Tweaking Files before Building

Several files need to be tweaked before installing NAMD in order to point the build in the right places and avoid various **various errors**.

Pointing to Charm++

The first of these is in the directory `NAMD_2.6_Source/Make.charm`. There should only be one line:

```
CHARMBASE = /Projects/namd2/charm-5.9
```

Change this to

```
CHARMBASE = .rootdir/charm-5.9
```

The next file on the list to edit is `arch/Linux-<your architecture>-MPI.arch` (for instance, `Linux-i686-MPI.arch`). Open this file and change the line `CHARMACH` to

```
CHARMARCH = mpi-linux-gcc
```

or whatever other folder was created for you in the `/charm-5.9` directory when you built charm++.

Tcl Updates

The builder also needs to know where to find tcl. Open `arch/Linux-<your architecture>.tcl` (for instance, `arch/Linux-i686.tcl`). Add `-I/usr/include/tcl8.4/` to the first line and change `-ltc` in the second line so it looks like the following.

```
TCLDIR=/usr
TCLINCL=-I$(TCLDIR)/include -I$(HOME)/tcl/include -I/usr/include/tcl8.4/
TCLLIB=-L$(TCLDIR)/lib -L$(HOME)/tcl/lib -ltcl8.4 -ldl
```

Removing --static from the Compiler

Finally, open `arch/Linux-<architecture>-g++.arch`. Take all out all instances off `--static`.

Building NAMD

Now you should be ready to build NAMD. From the base source directory for NAMD, run

```
./config tcl fftw Linux-<architecture>-MPI
```

This will create a new directory named `Linux-<architecture>-MPI`. Move into this directory, then run `make`. If you receive an error, you can look it up on the **NAMD: Troubleshooting Errors** page.

When it finishes successfully, you should see new executable binaries called namd2, psfgen, and charmrun, amongst others.

Moving NAMD

Once everything is working, the binaries can be moved to a different place (still on the **NFS mount**) where they're accessible as part of users' paths. This entire directory with the newly created executables can be copied over. A symlink (like a shortcut) to the `.rootdir` directory also needs to be changed to point to the old location. (This is why the compilation and such needs to be available as an NFS mount.)

To do this, from the new location for the files, run

```
rm .rootdir
```

and then

```
ln -s <source directory> .rootdir
```

For instance, mine looked like this:

```
ln -s /shared/usr/local/src/NAMD_2.6_Source/ .rootdir
```

I'm keeping my source on my NFS mount (/shared), under /usr/local/src. My binaries for the users are now under `/shared/opt/namd`. From within that directory, I ran the above command, which creates a symlink for .rootdir back into the original source.

Next

With NAMD set up, **VMD is next**.

NAMD: Building vmd

This is part four of a four-part tutorial on building and testing NAMD and charm++. The full tutorial includes

- **Installing support packages**
- **Building and testing charm++**
- **Building namd**
- **Building vmd**

There is also a troubleshooting page:

- **Troubleshooting errors**

About VMD

VMD, short for Visual Molecular Dynamics, is a graphical program used to view and manipulation three dimensional molecules. It can take the output (and input) .gro Gromacs files and .pdb Protein Database files and display the molecules. These can then be rotated around in different ways of viewing the molecule. This is done in real time, with the user manipulating the controls.

VMD doesn't necessarily need to run on the cluster. Users can install it on their own computers and then download their files from the cluster. Sometimes, however, it's nice to have a standard install that everyone can use and the system administrator can troubleshoot in one place - like the head node of the cluster. This doesn't need to be installed on all of the nodes, just the one that the users interact with.

Users will need to have a Linux windowing system installed in order to SSH into the cluster and forward the display from VMD to their own computers. This is done with

```
ssh -X username@yourcluster
```

Installing VMD

Supporting Packages

VMD requires OpenGL supporting packages. Install them with

```
apt-get install libglu1-mesa mesa-common-dev libgl1-mesa-dri libglu1-mesa-dev
xlibmesa-gl-dev
```

It also needs the development packages for TK, and the FLTK packages for menu support. Install these with

```
apt-get install tk8.4-dev tk-dev libfltk1.1 libfltk1.1-dev
```

This will only need to be done on the head node.

Getting VMD

Like charm++ and namd, vmd can be compiled from source. However, vmd is used for displaying molecules to users, not something that worker nodes will be chugging away with. For that reason, compiling the source to optimize it for an architecture doesn't really pay off all that well. If the prebuilt binaries work for your setup, you can save yourself some time by using those. However, I had just as many difficulties with the prebuilt binaries as with compiling from source, so I'm going to detail below how I compiled vmd.

This one you'll need to go through the process of registering and such for, since they don't list the directory contents. :) If you don't need 64-bit one, the location is

```
wget wget http://www.ks.uiuc.edu/Research/vmd/vmd-1.8.6/files/vmd-1.8.6.src.tar.gz
```

Once you have the tar file, untar it with

```
tar xvzf vmd*.tar.gz
```

This will create two directories in the current directory: vmd-1.8.6 and plugins.

Building Plugins

The plugins for VMD must be compiled prior to VMD! If not, you'll see interesting errors.

`Cd` into the plugins directory. Before running `make`, the variable PLUGINDIR needs to be set in order to tell make where to install the plugins. This will also be important for VMD itself. VMD will be automatically installed to /usr/local/lib/vmd, so I'd recommend putting the plugins in /usr/local/lib/vmd/plugins. To set the PLUGINDIR variable to this, run

```
export PLUGINDIR=/usr/local/lib/vmd/plugins
```

Next, if you run `make` by itself, you'll see a list of possible architectures. Unless you know your system is 64-bit or one of the other options, you'll probably use LINUX. You can also specify directories to be including that contain TCL and TK. These are important because it won't look the right places and you'll receive errors if you don't specify where to find them. If you installed both with apt-get, your line should be the same as mine (unless you have a different architecture):

```
make LINUX TCLINC=-I/usr/include/tcl8.4/ TCLLIB=-F/usr/lib/
```

This should finish without an errors, but you won't get a nice congratulatory message. Instead, it will end without errors (just warnings about unused variables), and if you run the same command again, it will finish without any additional output.

Then run `make distrib`. This will create new directories at $PLUGINDIR/LINUX and $PLUGINDIR/noarch.

Building VMD

After building the plugins, VMD itself can be built. Cd into the VMD directory from the plugins directory with `cd ../vmd-1.8.6`. Before compiling, there needs to be a symlink (like a Windows shortcut) in this directory that points to the plugins directory.

Create a symlink with

```
ln -s ../plugins
```

Now VMD can be built. It follows the **source installation paradigm** from here. Type `./configure help` to see a list of supported architectures. Make sure you choose the same architecture as you used above for the plugins. I'm not sure why it doesn't show all the ./configure options, and `make` will only show some predetermined combinations. Looking at the Makefile, my ./configure line looked like this:

```
./configure LINUX MESA TK TCL PTHREADS
```

`PYTHON` may also be an option you're interested. Look at the Makefile for more options.

Next, cd into the src directory. Here, run

```
make veryclean
```

and then

```
make
```

This will error out the first time with the following error:

```
Compiling  PluginMgr.C  -->  PluginMgr.o  ...
PluginMgr.C:29:31: libmolfile_plugin.h: No such file or directory
PluginMgr.C: In constructor `PluginMgr::PluginMgr()':
PluginMgr.C:56: error: `MOLFILE_INIT_ALL' was not declared in this scope
PluginMgr.C:56: error: expected `;' before '}' token
PluginMgr.C:56: warning: unused variable 'MOLFILE_INIT_ALL'
PluginMgr.C: In destructor `virtual PluginMgr::~PluginMgr()':
...snipped...
```

There may be a better way to solve this (please e-mail me at kwanous <at> debianclusters <dot> org if you know of one), but the way we fixed this was but editing the Makefile in the src directory. Make for VMD doesn't automatically look in the directory the plugins were installed in but we can change this. Look for the line that starts with INCDIRS and append to it so your line looks like this:

```
INCDIRS = -I../lib/Mesa/include -I../lib/tcl/include -I../lib/tk/include
-I../plugins/include -I../plugins/LINUX/molfile -I.
-I/usr/local/lib/vmd/plugins/LINUX/molfile/
```

Once you run `make` again, you'll see that it also doesn't automatically find TCL:

```
Compiling  VMDApp.C  -->  VMDApp.o  ...
VMDApp.C:680:18: tcl.h: No such file or directory
VMDApp.C: In member function `char* VMDApp::vmd_choose_file(const char*, const char*, const char*, int)':
VMDApp.C:706: error: `Tcl_Eval' was not declared in this scope
VMDApp.C:708: error: `TCL_OK' was not declared in this scope
VMDApp.C:709: error: `Tcl_GetStringResult' was not declared in this scope
```

and this is fixed by adding

```
-I/usr/include/tcl8.4
```

to the INCDIRS line in the Makefile. Running `make` a third time will show a new error,

```
Linking  vmd_LINUX ...
/usr/bin/ld: cannot find -lmolfile_plugin
collect2: ld returned 1 exit status
make: *** [vmd_LINUX] Error 1
```

This can also be fixed by touching up the Makefile. (Again, if there's a better way of doing this, please write me at kwanous <at> debianclusters <dot> org!) Find the LIBS line:

```
LIBS = -lGL -lGLU -L/usr/X11R6/lib -lXext -lX11 -lpthread -ltk8.4 -lX11 -ltcl8.4
-lmolfile_plugin -lfltk -lX11 -lXft -lm -ldl $(VMDEXTRALIBS)
```

and replace "-lmolfile_plugin" with the actual location, "/usr/local/lib/vmd/plugins/LINUX/molfile/libmolfile_plugin.a", so that your line looks like this:

```
LIBS = -lGL -lGLU -L/usr/X11R6/lib -lXext -lX11 -lpthread -ltk8.4 -lX11 -ltcl8.4
/usr/local/lib/vmd/plugins/LINUX/molfile/libmolfile_plugin.a -lfltk -lX11 -lXft -lm
-ldl $(VMDEXTRALIBS)
```

Run `make` again for it to not find -lXft:

```
/usr/bin/ld: cannot find -lXft
collect2: ld returned 1 exit status
make: *** [vmd_LINUX] Error 1
```

This is fixed in the Makefile under LIBS again. Replace "-lXft" with "/usr/lib/libfltk.a" and then this time, make should finish with an odd little message:

```
No resource compiler required on this platform.
```

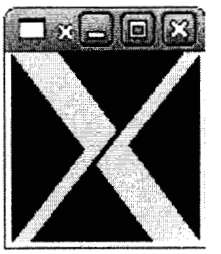
At this point, you can run `make install` and VMD should be placed successfully on your system at /usr/local/bin/vmd and in /usr/local/lib/vmd/.

Setting up the Display

Before vmd can be displayed, a graphical display server needs to be installed on the head node. This doesn't mean that a windowing environment (like Gnome or KDE) will be installed, just that the machine will be capable of rendering graphics. Users can then do `ssh yourmachine -X` to forward the graphics to their terminal (if they're running Linux). The basic linux graphics server is xserver. To install it, run

```
apt-get install xserver-xorg
```

As part of the install, you'll be prompted for which resolutions to keep. Use the space bar to select resolutions for the server to try to use. If you don't know, you're safer choosing lower resolutions and more typical resolution sizes.



There's a great little utility called `xlogo` that's used to test if the xserver is running correct and forwarding graphics. It's obtained with

```
apt-get install xutils
```

Then logout of the machine. From a machine running Linux (with a graphical display), ssh back in with the `-X` option, and then run `xlogo`. You should see a little box pop up like that shown to the right.

Now, try entering `vmd`. If it gives an error,

```
vmd: error while loading shared libraries: libstdc++.so.5: cannot open
  shared object file: No such file or directory
```

then `apt-get install libstdc++5` and you should be good to go.

Testing

To test VMD, try SSHing into the head node as a user while forwarding graphics. For example,

```
ssh -X yourclusterhere
```

Then, run `xlogo` and make sure that the little window with the logo can pop up on your local computer. If you can't display the xlogo, you won't be able to display vmd, either.

Before running vmd, let's download a protein to display. Many, many proteins can be downloaded from the **RCSB Protein Data Bank (http://www.rcsb.org/pdb)** . From within the user's home directory, download the protein ubiquitin with

VMD menus

```
wget http://dx.doi.org/10.2210/pdb4hhb/pdb
```

and then decompress it with

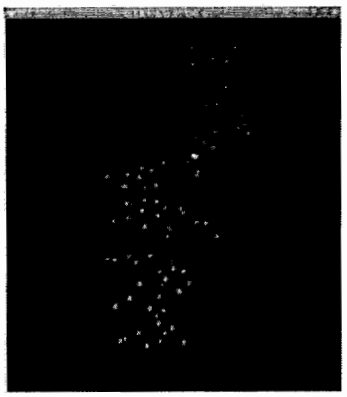
```
gunzip pdb4hhb.ent.Z
```

Then, run `vmd`. Three different windows should pop up, called VMD, VMD Main, and vmd console. Click on VMD Main, shown to the right. From here, click on the drop down menu "File" and then click on "New Molecule". A new window will pop up. In this one, click on the button "Browse". You should be placed in the home directory; if not, you can type the location in. Select pdb4hhb.ent and click on Load. Move to the "VMD" window and you should see the ubiquitin molecule!

Under the "Graphics" menu, particularly under "Representations...", you can change the way the molecule is displayed.

Displayed at right is the ubiquitin molecule displayed with the CPK drawing style. Very cool!

When you're ready to quit VMD, click on the console window and type `quit`. The program should exit gracefully.



Ubiquitin displayed by VMD

Configuring VMD

VMD has a number of different options that can be customized as part of its startup. The file in charge of these tweaks is located at `/usr/local/lib/vmd/.vmdrc`.

Unfortunately, describing these options is beyond the scope of this project. Fortunately, **UIUC covers these commands (http://www.ks.uiuc.edu/Research/vmd/current/ug/node199.html)** .

References

- **VMD Plugin Programmer's Guide: Compiling plugins from source code (http://www.ks.uiuc.edu/Research/vmd/plugins/doxygen/compiling.html)**
- Thank you to **Charlie Grosvenor (http://linux.derkeiler.com/Mailing-Lists/Debian/2004-09/2932.html)** for explaining what TCLINC and TCLLIB should point to.
- Thank you to **Axel Kohlmeyer (http://www.ks.uiuc.edu/Research/vmd/mailing_list/vmd-l/8023.html)** for explaining the reason for a missing VMD main menu.

NAMD: Troubleshooting Errors

This is the **troubleshooting errors** page of a four part tutorial on building and testing NAMD and charm++. The full tutorial includes

- **Installing support packages**
- **Building and testing charm++**
- **Building namd**
- **Building vmd**

Errors

If you follow the instructions above, you shouldn't experience any of these errors (I hope!). The solutions for them are posted right as part of the tutorial. Rather than crowding up the tutorial with what can go wrong, I decided to post all the errors (and the solutions I found for them) here at the end for anyone else experiencing these problems.

Building/Testing Charm++ Errors

C Compiler Errors

If you're trying to compile charm++ with a GNU version of gcc/g++ above 3.x, you'll see errors like this:

```
../bin/charmc  -O  -c -I. traceCore.C
In file included from converseEvents.h:5,
                 from traceCore.C:12:
traceCore.h:20: error: previous declaration of 'int Cpv__traceCoreOn_ [2]' with 'C++' linkage
traceCoreAPI.h:8: error: conflicts with new declaration with 'C' linkage
traceCore.C: In constructor 'TraceCore::TraceCore(char**)':
Fatal Error by charmc in directory /usr/local/src/NAMD_2.6_Source/charm-5.9/mpi-linux-gcc/tmp
   Command g++ -fPIC -Wno-deprecated -I../bin/../include -D__CHARMC__=1 -I. -O -I/shared/include
   -c traceCore.C -o traceCore.o returned error code 1
charmc exiting...
make[2]: *** [traceCore.o] Error 1
make[2]: Leaving directory `/usr/local/src/NAMD_2.6_Source/charm-5.9/mpi-linux-gcc/tmp'
make[1]: *** [converse] Error 2
make[1]: Leaving directory `/usr/local/src/NAMD_2.6_Source/charm-5.9/mpi-linux-gcc/tmp'
make: *** [charm++] Error 2
```

This can be solved by symlinking `/usr/bin/gcc` and `/usr/bin/g++` to older versions. See above.

mpi.h Error

You'll see errors like the following if the charm builder can't find the MPI libraries.

```
machine.c:17:17: mpi.h: No such file or directory
machine.c:144: error: syntax error before "MPI_Request"
machine.c:144: warning: no semicolon at end of struct or union
machine.c:145: warning: data definition has no type or storage class
machine.c:150: error: syntax error before '*' token
```

```
machine.c:150: warning: data definition has no type or storage class
machine.c:151: error: syntax error before '*' token
machine.c:151: warning: data definition has no type or storage class
machine.c: In function `CmiTimerIsSynchronized':
machine.c:190: error: `MPI_SUCCESS' undeclared (first use in this function)
machine.c:190: error: (Each undeclared identifier is reported only once
machine.c:190: error: for each function it appears in.)
machine.c:190: error: `MPI_COMM_WORLD' undeclared (first use in this function)
machine.c:190: error: `MPI_WTIME_IS_GLOBAL' undeclared (first use in this function)
...snipped...
Fatal Error by charmc in directory /usr/local/src/NAMD_2.6_Source/charm-5.9/mpi-linux-gcc/tmp
   Command gcc -fPIC -I../bin/../include -D__CHARMC__=1 -DFOR_CPLUS=1 -O -c machine.c -o machine.o
   returned error code 1
```

You can fix this by giving the charm++ `./build` command a `--basedir=`. Set the base directory to be the directory that encompasses where mpi.h is located. You can find this with `locate mpi.h` if you don't know. If it doesn't exist, you need to install **MPICH**, or go with a net-linux installation, which is not covered in this tutorial. (Basically, specify `net-linux` instead of `mpi-linux` in `./build`.)

Testing Charm++: No Pthreads Error

Without editing `charm-5.9/include/conv-mach.sh` to include the POSIX thread libraries, you'll see errors like this:

```
../../../bin/charmc  -o pgm  megatest.o groupring.o nodering.o varsizetest.o varraystest.o groupcast.o
   nodecast.o synctest.o fib.o arrayring.o tempotest.o packtest.o queens.o migration.o marshall.o priomsg.o
   priotest.o rotest.o statistics.o templates.o inherit.o reduction.o callback.o immediatering.o bitvector.o
   -language charm++
/shared/lib/libmpich.a(attr_get.o): In function `MPI_Attr_get':
attr_get.c:(.text+0x41): undefined reference to `pthread_getspecific'
attr_get.c:(.text+0x77): undefined reference to `pthread_setspecific'
attr_get.c:(.text+0x315): undefined reference to `pthread_getspecific'
attr_get.c:(.text+0x34b): undefined reference to `pthread_setspecific'
attr_get.c:(.text+0x3a6): undefined reference to `pthread_getspecific'
attr_get.c:(.text+0x3dc): undefined reference to `pthread_setspecific'
```

To solve it, add `-pthread` to the `CMK_LIBS` argument.

Building NAMD Errors

csh Error

Without the binary for `csh`, you won't be able to build NAMD. You'll see the following error:

```
-su: ./config: /bin/csh: bad interpreter: No such file or directory
```

This is easily fixed with

```
apt-get install csh
```

charm++.h Error

An error like the following

```
g++ -I/Projects/namd2/charm-5.9/mpi-linux/include -DCMK_OPTIMIZE=1 -Isrc -Iinc -Iplugins/include
   -I/Projects/namd2/tcl/linux/include -I/root/tcl/include -DNAMD_TCL -DUSE_NON_CONST
   -I/Projects/namd2/fftw/linux/include -I/root/fftw/include -DNAMD_FFTW  -DNAMD_VERSION=\"2.6\"
   -DNAMD_PLATFORM=\"Linux-i686-MPI\"  -O3 -march=pentiumpro -ffast-math -static -o obj/common.o
   -c src/common.C
src/common.C:22:21: charm++.h: No such file or directory
src/common.C: In function `void NAMD_quit(const char*)':
src/common.C:67: error: `CkPrintf' was not declared in this scope
src/common.C:69: error: `CmiAbort' was not declared in this scope
```

or

```
src/common.C:22:21: charm++.h: No such file or directory
make: .rootdir/charm-5.9//mpi-linux/bin/charmc: Command not found
```

indicates that make cannot find `charm++.h`. There are two locations that need to be specified for this: the first is `NAMD_2.6_Source/Make.charm`. The line should be

```
CHARMBASE = .rootdir/charm-5.9
```

or wherever else you built charm++. After you do this, you may need to delete your `Linux-<architecture>-MPI` directory and run `./configure` (with the proper options) again.

The second file to edit is `arch/Linux-<your architecture>-MPI.arch`. Open this file and change the line `CHARMACH` to

```
CHARMARCH = mpi-linux-gcc
```

or whatever other folder was created for you in the `/charm-5.9` directory when you built charm++. See **NAMD: Building namd** for more information.

tcl Error

If something's wrong with the way tcl is being pointed to, errors like these will arise:

```
g++ -I.rootdir/charm-5.9//mpi-linux-gcc/include -DCMK_OPTIMIZE=1 -Isrc -Iinc   -Iplugins/include
   -I/Projects/namd2/tcl/linux/include -I/root/tcl/include -DNAMD_TCL -DUSE_NON_CONST
   -I/Projects/namd2/fftw/linux/include -I/root/fftw/include -DNAMD_FFTW  -DNAMD_VERSION=\"2.6\"
   -DNAMD_PLATFORM=\"Linux-i686-MPI\"  -O3 -march=pentiumpro -ffast-math -static -o obj/mainfunc.o
   -c src/mainfunc.C
In file included from src/mainfunc.C:31:
src/ScriptTcl.h:19:17: tcl.h: No such file or directory
In file included from src/mainfunc.C:31:
src/ScriptTcl.h:49: error: ISO C++ forbids declaration of `Tcl_Interp' with no type
src/ScriptTcl.h:49: error: expected `;' before '*' token
src/ScriptTcl.h:50: error: `ClientData' has not been declared
src/ScriptTcl.h:50: error: `Tcl_Interp' has not been declared
src/ScriptTcl.h:50: error: ISO C++ forbids declaration of `parameter' with no type
src/ScriptTcl.h:50: error: ISO C++ forbids declaration of `parameter' with no type
```

This is fixed by editing `arch/Linux-<your architecture>.tcl` to match the following.

```
TCLDIR=/usr
TCLINCL=-I$(TCLDIR)/include -I$(HOME)/tcl/include -I/usr/include/tcl8.4/
TCLLIB=-L$(TCLDIR)/lib -L$(HOME)/tcl/lib -ltcl8.4 -ldl
```

See **NAMD: Building namd** for more information.

Multiple Definitions Error

An error that might be seen towards the very end of the build involves "multiple definitions". Pages and pages of errors may scroll by similar to

```
/usr/lib/libc.a(malloc.o):(.data+0x10): multiple definition of `__libc_malloc_initialized'
.rootdir/charm-5.9//mpi-linux-gcc/bin/../lib/libmemory-default.o:(.data+0x4): first defined here
/usr/lib/libc.a(malloc.o): In function `__malloc_check_init':
(.text+0xf90): multiple definition of `__malloc_check_init'
.rootdir/charm-5.9//mpi-linux-gcc/bin/../lib/libmemory-default.o:memory.c:(.text+0x10c1): first defined here
/usr/lib/libc.a(malloc.o): In function `_int_free':
(.text+0x1fe0): multiple definition of `_int_free'
.rootdir/charm-5.9//mpi-linux-gcc/bin/../lib/libmemory-default.o:memory.c:(.text+0x2db8): first defined here
/usr/lib/libc.a(malloc.o): In function `_int_malloc':
(.text+0x2940): multiple definition of `_int_malloc'
.rootdir/charm-5.9//mpi-linux-gcc/bin/../lib/libmemory-default.o:memory.c:(.text+0x27fc): first defined here
```

This is solved by opening `arch/Linux-<architecture>-g++.arch` and taking out all instances off `--static`.

MpiBLAST: Nucleotide/Protein Searching

About mpiBLAST

mpiBLAST is a parallel implementation of the Blast algorithm. BLAST, or the Basic Local Alignment Search Tool, is used to search large databases. "Local alignment" refers to the way BLAST compares sequences. When dealing with a potentially large sequences, rather than looking for a match in entirety, it looks for smaller patches that match. When it finds a match, it evaluates the probability that this match would occur by chance. If it's highly likely the match occurred by chance, it probably isn't very statistically significant. These matches can be searched for in either a string of nucleotides (which make up DNA) or amino acids (which make up proteins). When running BLAST, you select which type your sequence is and choose the databases of known sequences you're interested in comparing it against.

mpiBlast divides the work of searching these databases up amongst multiple processes (and processors, if available) by using MPI. First, the database to be searched is split into as many pieces are there will be processes. Then, the processes communicate amongst themselves using MPI. You'll need to have **MPICH** or another implementation of MPI installed in order for mpiblast to work.

There are two pages about mpiblast:

- **Installing mpiBLAST** - all the setup and configuration
- **Using mpiBLAST** - running mpiblast

References

- **Blast: Basic Local Alignment Search Tool (http://ncbi.nlm.nih.gov/blast/Blast.cgi)**
- The Design, Implementation, and Evaluation of mpiBLAST. A. Darling, L. Carey, and W. Feng. 4th International Conference on Linux Clusters: The HPC Revolution 2003 in conjunction with ClusterWorld Conference & Expo, June 2003. Available online at **http://www.mpiblast.org/downloads/pubs/cwce03.pdf**

Intalling mpiBLAST

Intalling mpiBLAST

This is part one of a two-part tutorial on **mpiBLAST**. The full tutorial includes

- **Installing mpiBLAST**
- **Using mpiBLAST**

Getting mpiblast

mpiblast is freely available from the site at **http://www.mpiblast.org/Downloads/Stable**. At the time of this writing, the most recent version is 1.5.0, but you may want to check yourself as versions are subject to change. From your source code directory, run

```
wget http://www.mpiblast.org/downloads/files/mpiBLAST-1.5.0-pio.tgz
```

or the location for the most recent version if there's one newer. When it's finished, untar the file with

```
tar xvzf mpiBLAST*.tgz
```

This will create a new directory, mpiblast-something. Cd into that directory. Inside this directory, mpiblast needs another file to be downloaded: a specific version of the NCBI toolkit. Do this with

```
wget ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools/old/20061015/ncbi.tar.gz
```

and then untar it with

```
tar xvzf ncbi.tar.gz
```

Building mpiblast

NCBI Toolkit

First, you'll need to patch and build the NCBI toolkit. I'm not completely sure that the patch needs to be applied (or does anything to this older version of the NCBI toolkit), but it certainly didn't hurt in my installation. To patch it, cd into the `ncbi` directory and run

```
patch -p1 < ../patches/ncbi_Oct2006_evalue_pio.patch
```

Then, cd back up one level to the mpiblast folder (`cd ..`) and run

```
./ncbi/make/makedis.csh
```

(**Note:**If you receive an error about CSH, you don't have the C shell installed, but that's easy to fix with `apt-get install csh`.)

The above command (makedis.csh) needs to be run three times. The first time you run, it will finish with an error:

```
make: *** No rule to make target `ncbimain.o', needed by `libncbi.a'.  Stop.

Fatal error building NCBI core libraries.
Please be sure that you have X11 and Motif libraries installed.
The NCBI toolkit FAQ at ftp://ftp.ncbi.nih.gov/toolbox/FAQ.html may be helpful.
```

Run the same command again,

```
./ncbi/make/makedis.csh
```

even though it finished with an error. This time, it will take quite a bit longer, and it will finish with a different error:

```
make: *** No rule to make target `ni_debug.o', needed by `libnetcli.a'.  Stop.
FAILURE primary make status = 0, demo = 0, threaded_demo = 0, net = 2
#######
#       #####   #####    ####   #####
#       #    #  #    #  #    #  #    #
#####   #    #  #    #  #    #  #    #
#       #####   #####   #    #  #####
#       #   #   #   #   #    #  #   #
#######  #    # #    #   ####   #    #
```

Run

```
./ncbi/make/makedis.csh
```

one last time, and it should finish successfully:

```
Put the date stamp to the file ../VERSION
*********************************************************
*The new binaries are located in ./ncbi/build/ directory*
*********************************************************
```

Then it's safe to continue on to building mpiblast itself.

mpiblast

From here, mpiblast follows the standard **source installation paradigm** and there shouldn't be many surprises. (See the **Source Installation Paradigm** page for a more gentle introduction on installation of software from source.) Run `./configure --help` to see all the possible options.

Important options include

- `--prefix=` - used to specify where the files should be installed. These files will need to be accessible by all of the worker nodes. A good place would be an **NFS mount**
- `--with-ncbi=` - specifies where the NCBI toolkit is found. An easy way to tell it to use "the ncbi directory within the current directory" is to use `--with-ncbi=`pwd`/ncbi`
- `--enable-MPI_Alloc_mem` - this is a new feature in MPI 2. (If you followed my **MPICH tutorial**, you installed MPICH2.) If you are using a current implementation, you'll want to enable this for performance.

My NFS mount is located at `/shared`. The ./configure line I ran was

```
./configure --prefix=/shared --with-ncbi=`pwd`/ncbi --enable-MPI_Alloc_mem
```

Once it finishes, it's time to run `make`. Make should finish successfully, unless you grabbed the wrong version of the NCBI toolkit. In my experience, the newer version didn't work - you specifically need the older version of the NCBI toolkit. With the newer version, you'll see errors like this:

```
blast_hooks.c: In function `initBLAST':
blast_hooks.c:757: error: structure has no member named `query_adjustments'
blast_hooks.c:758: error: structure has no member named `effective_db_lengths'
blast_hooks.c: In function `getQuery':
blast_hooks.c:836: error: structure has no member named `current_queryI'
blast_hooks.c: In function `writeSearchStatistics':
blast_hooks.c:1332: error: structure has no member named `stats'
blast_hooks.c: In function `runBLAST':
blast_hooks.c:1455: error: structure has no member named `calculate_statistics_and_exit'
blast_hooks.c:1457: error: structure has no member named `calculate_statistics_and_exit'
blast_hooks.c:1484: error: structure has no member named `current_queryI'
blast_hooks.c: In function `runBLASTPIO':
blast_hooks.c:1552: error: structure has no member named `calculate_statistics_and_exit'
```

Go back and wget the older NCBI toolkit and build it again, then run ./configure and make again. This time, it should work.

Once make finishes, it's time to run `make install`.

Sanity Check

It's always a good idea to make sure that the mpiblast binaries have been installed and are in the users' PATHs. As a user, run

```
which mpiblast
```

and it should return the location of the mpiblast binary, if that file is in the user's path. If it isn't, first install locate if you haven't already (`apt-get install locate`) and then you can run

```
updatedb
locate mpiblast | grep -v src
```

- The `grep -v src` part excludes any directories or files with the word "src" in them. If you keep your source files in a directory named differently, update that section. Otherwise, you'll find all of the source code as part of the search.

Once you've found the binary, you can **update user's PATHs** accordingly.

Using mpiBLAST

This is part one of a two-part tutorial on **mpiBLAST**. The full tutorial includes

- **Installing mpiBLAST**
- **Using mpiBLAST**

Organization and Setup

Before downloading a bunch of databases to use with mpiblast, it'd be a good idea to make a decision about where the databases are going to be kept. Each user will need to have a profile for it in their home directory telling mpiblast where to find the databases and also where to find work space. These need to be available to all of the worker nodes, so putting the files on an **NFS mount** would be a good idea.

Optional components are often kept under `/opt`. My NFS mount is at `/shared`, so I'll be putting all of my databases into `/shared/opt/mpiblast`.

This directory needs to be writable by users so that they can format the databases to be run over different numbers of processes. There are a few different ways to do this. The directory could be set to be world-writeable, so that anyone can write to the directory. This is done with `chmod a+w`. Another way would be to change the permissions to be group-writeable (`chmod g+w`), create a new group (`addgroup mpigroup`), add all the mpiblast users to that group, and change the owner of the directory to that group (`chown mpigroup:mpigroup`). The second method is more secure, but may take more work to setup the new group and then add the users to the group using **LDAP**.

As a third option, you could let users download their own database files and set mpiblast to point within the users' home directories using the .ncbirc file (explained below).

.ncbirc File

The file each mpiblast user needs in his/her home directory is called `.ncbirc`. It uses the same format as BLAST. The Shared variable refers to where the databases are kept. The Local variable refers to any local disk space used as a workspace; this can be part of the user's home directory. If you're using an NFS-mount, and/or your users' home directories are NFS-mounted, this will probably be in the same directory or in similar directories.

Here's an example .ncbirc file for the user kwanous:

```
[mpiBLAST]
Shared=/shared/opt/mpiblast
Local=/shared/home/kwanous/mpiblast
```

Getting Databases

Nucleotide and protein databases are available from the National Institute of Health (NIH). A list of what the different databases contained is available as part of the **Blast tutorial (http://www.ncbi.nlm.nih.gov/Education/blasttutorial.html)** and the files themselves are at

ftp://ftp.ncbi.nlm.nih.gov/blast/db/. The files under the FASTA directory are pre-formatted for FASTA, and these can be used as well.

To download a file, find the location for the one you want on the FTP site. Right-click it and copy the location of the file. Then, from wherever you're keeping your databases, use `wget`. For instance, to download the *Drosophila melanogaster'* (fruit fly) nucleotide database, run

```
wget ftp://ftp.ncbi.nlm.nih.gov/blast/db/FASTA/drosoph.nt.gz
```

Then you'll need to untar or just ungzip them. If they end in `.tg.gz`, use to tar to untar the file:

```
tar xvzf yourfile.tg.gz
```

If they just end in `.gz`, they're only gzipped, so unzip them with

```
gunzip yourfile.gz
```

Formatting the Database

In order to allow multiple processes to use the large database at the same time, the database needs first be broken down into smaller chunks. The program that does this is called `mpiformatdb`. It takes several arguments: one for the number of fragments to split the database into (this should be the number of processes that will be running mpiblast), one for the input file, and one to indicate whether it's a protein file or not.

Note: One of the great things about the mpiblast tools is that you can issue most of the commands followed by a space and a hyphen (like `mpiformatdb -`) and it will show you all the options for the command.

To segment the fruit fly database I just downloaded into four separate fragments, I ran

```
mpiformatdb -i drosoph.nt --nfrags=4 -pF
```

- `-i` specifies my input file, drosoph.nt
- `--nfrags=4` tells it to make four fragments
- `-p` specifies whether the file is a protein file or not. I put F for false, since this is a nucleotide file, not an amino acid file.

The process should look like this:

```
gyrfalcon:/shared/opt/mpiblast# mpiformatdb -i drosoph.nt --nfrags=4 -pF
Reading input file
Done, read 1534943 lines
Reordering 1170 sequence entries
Breaking drosoph.nt into 4 fragments
Executing: formatdb -p F -i /tmp/reordermc8uxl -N 4 -n /shared/opt/mpiblast/drosoph.nt -o T
Removed /tmp/reordermc8uxl
Created 4 fragments.
```

The output will be dropped into the same folder as the original file, which is why users need to have write permissions on the directory any shared databases will be kept in.

Running mpiblast

Once the database has been formatted to support multiple processes, it's time to run mpiblast.

Mpiblast takes an input file with a sequence to query the databases against. This file is usually just a testfile with a nucleotide or amino acid sequence, depending on what kind of databases are being searched against. You can create a test input sequence by opening a file with your favorite text editor (like vi or nano) and pasting one in. A greater than symbol on the first line can give the name of the input sequence. For example, I'll create `test.in` with these contents:

```
>Test
AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC
TTCTGAACTGGTTACCTGCCGTGAGTAAATTAAAATTTTATTGACTTAGGTCACTAAATACTTTAACCAA
TATAGGCATAGCGCACAGACAGATAAAAATTACAGAGTACACAACATCCATGAAACGCATTAGCACCACC
ATTACCACCACCATCACCATTACCACAGGTAACGGTGCGGGCTGACGCGTACAGGAAACACAGAAAAAAG
CCCGCACCTGACAGTGCGGGCTTTTTTTTTTCGACCAAAGGTAACGAGGTAACAACCATGCGAGTGTTGAA
GTTCGGCGGTACATCAGTGGCAAATGCAGAACGTTTTCTGCGTGTTGCCGATATTCTGGAAAGCAATGCC
AGGCAGGGGCAGGTGGCCACCGTCCTCTCTGCCCCCGCCAAAATCACCAACCACCTGGTGGCGATGATTG
AAAAAACCATTAGCGGCCAGGATGCTTTACCCAATATCAGCGATGCCGAACGTATTTTTGCCGAACTTTT
```

Mpiblast will be run just using `mpirun` or `mpiexec`, but there are a few important command line arguments. These include

- `-d` - the database to be queried against
- `-i` - the input file
- `-p` - the type of blast query to run, including
  - `blastn` for nucleotides
  - `blastp` for proteins
- `-o` - the name of the file to save the output in

The command run in its complete state should look something like this:

```
mpiexec -np 4 /shared/bin/mpiblast -d drosoph.nt -i test.in -p blastn -o results.txt
```

Of course, if you're using the **version of mpiexec with Torque functionality**, you'll need to wrap this in a **qsub script**.

When it's finished running, you should have a new `results.txt` file that looks something like this:

```
BLASTN 2.2.15 [Oct-15-2006]

Reference:
Aaron E. Darling, Lucas Carey, and Wu-chun Feng,
"The design, implementation, and evaluation of mpiBLAST".
In Proceedings of 4th International Conference on Linux Clusters: The HPC Revolution 2003,
June 24-26 2003, San Jose, CA

Heshan Lin, Xiaosong Ma, Praveen Chandramohan, Al Geist, and Nagiza Samatova,
"Efficient Data Access for Parallel BLAST".
In Proceedings of 19th International Parallel & Distributed Processing Symposium 2005,
April 3-8 2005, Denver, CO
```

```
Query= Test
         (560 letters)

Database: /shared/opt/mpiblast/drosoph.nt
           1170 sequences; 122,655,632 total letters



                                                                    Score    E
Sequences producing significant alignments:                         (bits) Value

gb|AE003681.2|AE003681 Drosophila melanogaster genomic scaffold ...    36   0.86
gb|AE002936.2|AE002936 Drosophila melanogaster genomic scaffold ...    36   0.86
gb|AE003698.2|AE003698 Drosophila melanogaster genomic scaffold ...    36   0.86
gb|AE003493.2|AE003493 Drosophila melanogaster genomic scaffold ...    36   0.86
gb|AE002615.2|AE002615 Drosophila melanogaster genomic scaffold ...    34   3.4
gb|AE003441.1|AE003441 Drosophila melanogaster genomic scaffold ...    34   3.4
gb|AE003525.2|AE003525 Drosophila melanogaster genomic scaffold ...    34   3.4


...snipped...


Query: 96     taaattaaaatttttatt 112
               ||||||||||||||||||
Sbjct: 226581 taaattaaaatttttatt 226565


>gb|AE003447.2|AE003447 Drosophila melanogaster genomic scaffold 142000013386054 section 31 of
              35, complete sequence
          Length = 304085

 Score = 34.2 bits (17), Expect = 3.4
 Identities = 17/17 (100%)
 Strand = Plus / Minus


Query: 377    cagaacgttttctgcgt 393
               ||||||||||||||||
Sbjct: 177093 cagaacgttttctgcgt 177077


  Database: /shared/opt/mpiblast/drosoph.nt
    Posted date:  Apr 18, 2008  2:48 PM
  Number of letters in database: 30,663,804
  Number of sequences in database:  292

  Database: /shared/opt/mpiblast/drosoph.nt.001
    Posted date:  Apr 18, 2008  2:48 PM
  Number of letters in database: 30,664,011
  Number of sequences in database:  293

  Database: /shared/opt/mpiblast/drosoph.nt.002
    Posted date:  Apr 18, 2008  2:48 PM
  Number of letters in database: 30,664,004
  Number of sequences in database:  293

  Database: /shared/opt/mpiblast/drosoph.nt.003
    Posted date:  Apr 18, 2008  2:48 PM
  Number of letters in database: 30,663,813
  Number of sequences in database:  292
```

Using a Scheduler and Queue

The scheduler and the queue are two essential parts for a cluster. Together, they transform a group of networked machines into a cluster, or at least something closer to one. They're what allow users, working only on the head node, to submit "jobs" to the cluster. These jobs are transparently assigned to different worker nodes, and then - without the user needing to know where the jobs were - the results are deposited back into the user's home directory.

This process requires software in two different roles: the resource manager, responsible for accepting jobs to the queue and running jobs on worker nodes, and the scheduler, responsible for deciding when and where jobs in the queue should be run in order to optimize resources. I'll be using **Torque** for the resource manager and **Maui** for the scheduler. Both of these are open source projects.

Note: Torque has a built-in scheduler that can be used instead of Maui. However, Maui integrates seamlessly and provides more options and customization than Torque's scheduler.

Installation

Before setting up Torque and Maui, **DNS** must be working. If that's not an option, this requisite can be "cheated" around by setting up `/etc/hosts` on the head node with an entry for each of the nodes and then copying this file out to each of the worker nodes. (See the **Cluster Time-saving Tricks** page for help with the copying.)

Torque needs to be installed in two parts. First, a `pbs_server` is set up on the head node and configured to know where all of the worker nodes are. Then, each of the worker nodes are set up to run `pbs_mom`, a sort of client, that will accept jobs from the `pbs_server` and run them on the worker node. A basic queue for Torque also needs to be configured.

Maui is installed only on the head node, and needs to be set up to interact with the `pbs_server`. It does not communicate with the worker nodes, but instead talks to them by way of the server.

After installing both, it's wise to try submitting a simple job before moving onto more complex configuration options. I recommend going through pages of interest in this order -

- **Resource Manager: Torque** - instructions on installing/configuring Torque
- **Scheduler: Maui** - instructions on installing/configuring Maui
- **Torque and Maui Sanity Check: Submitting a Job** - attempt submitting a simple job
    - **Troubleshooting Torque and Maui** - consult this if problems arise with the job submission process
- **Torque and Maui: Submitting an MPI Job** - assuming you have **MPICH** installed
- **Torque Queue Configuration**
- **Torque Qsub Scripts**

References

- **HOWTO Torque/Maui (http://gentoo-wiki.com/HOWTO_Torque/Maui_-_grid_scheduler_and_resource_manager)**
- **University of Cambridge Chemistry Department's Maui Administration Notes (http://www-theor.ch.cam.ac.uk/IT/servers/maui/maui-admin.html)**

Resource Manager: Torque

This is the second part of a four-part tutorial on installing and configuring a **queuing system and scheduler**. The full tutorial includes:

- **Using a Scheduler and Queue**
- **Resource Manager: Torque**
- **Scheduler: Maui**
- **Torque and Maui Sanity Check: Submitting a Job**

There is also a troubleshooting page:

- **Troubleshooting Torque and Maui**

About Torque

From the **Cluster Resources** page on Torque,

> *"TORQUE is an open source resource manager providing control over batch jobs and distributed compute nodes. It is a community effort based on the original *PBS project..."*

Because torque branched off from PBS, it still retains a lot of the old commands and names. PBS stands for **portable batch system**, and from here, I'll still call it torque, but commands may have "pbs" in them rather than "torque".

Installing Torque

Before you get and install torque, you'll want to make sure you have all the **compilers** installed that are necessary. If you don't, it will give you errors about which ones you're missing.

To get the most recent version of torque, visit **http://www.clusterresources.com/downloads/torque/** and find the most recent version of it. At the time if this writing, that happens to be torque-2.2.1.tar.gz. Copy of the link location of the file. From `/usr/local/src`, issue the following command for the most current file:

```
wget http://www.clusterresources.com/downloads/torque/torque-2.2.1.tar.gz
```

Next, untar the file with

```
tar xvf torque-2.2.1.tar.gz
```

Move into the directory that that just created with `cd torque-2.2.1`, or whatever your directory is. We're ready to run `./configure` (as part of the **Source Installation Paradigm**, which you might want to check out if this seems unfamiliar to you). We'll add a number of arguments to the compiler in order to let torque know we want a server, and how to set up the server. To see all of the possible arguments, type `./configure -help`. What we'll use is this:

```
./configure --with-default-server=<your server name>
--with-server-home=/var/spool/pbs --with-rcp=scp
```

- `--with-default-server` specifies the head node, which will run the server torque process. Be sure to replace `your server name` with your actual head node's hostname!
- `--with-server-home` sets the directory where torque will run from. `/var/spool/pbs` is by no means standard, but it's the paradigm I'll be using. Others use a directory like `/home/torque`. I don't like confusing my processes with users.
- `--with-rcp=scp` sets the default file-copying mechanism. Technically, scp (for **secure copy**) is the default, but if you don't specify it and `scp` isn't found, it'll move onto trying to find the next one, which we don't want.

If the `./configure` finishes successfully, you're ready to move onto the next step. If not, address the issues before running the command again. When it does finish successfully, it will end with a line like `config.status: executing depfiles commands`, but no message about being finished. Next, run

```
make
```

A lot of what looks like gibberish will scroll by, and it may take somewhere around five minutes. Again, it will finish without a confirmation message. The last part of the script finished on mine with

```
make[3]: Leaving directory `/usr/local/src/torque-2.2.1/doc'
make[2]: Leaving directory `/usr/local/src/torque-2.2.1/doc'
make[1]: Leaving directory `/usr/local/src/torque-2.2.1/doc'
make[1]: Entering directory `/usr/local/src/torque-2.2.1'
make[1]: Nothing to be done for `all-am'.
make[1]: Leaving directory `/usr/local/src/torque-2.2.1'
```

Finally, you're ready to run

```
make install
```

You won't get a confirmation message for this, either, and it'll finish similarly to the way the last one finished. To make sure it was installed correctly, try using `which` to locate one of the binaries, like this:

```
gyrfalcon:~# which pbs_server
/usr/local/sbin/pbs_server
```

If it can't find it, double check that the binary was installed with `ls` and `grep`:

```
gyrfalcon:~# ls /usr/local/sbin | grep pbs
pbs_demux
pbs_iff
pbs_mom
pbs_sched
pbs_server
```

If it's there in `/usr/local/sbin`, but `which` doesn't find it, you'll need to edit `/etc/login.defs`. Locate the line for `ENV_SUPATH` and add `/usr/local/bin` and `/usr/local/sbin` to it. The line for `ENV_PATH` should be right below it; add `/usr/local/bin` to it.

Configuring Torque

To start the torque server running on the head node and create a new database of jobs, issue

```
pbs_server -t create
```

Now, if you run `ps aux | grep pbs`, you'll see the server running. However, if you run a command to list the queues and their statuses,

```
qstat -a
```

you'll see nothing because no queues have been set up. To begin configuring queues for torque, we need `qmgr`, an interface to the batch system. You can run

```
qmgr
```

to start it up in an interactive mode, or enter the commands one at a time on the command line:

```
qmgr -c "set server scheduling=true"
qmgr -c "create queue batch queue_type=execution"
qmgr -c "set queue batch started=true"
qmgr -c "set queue batch enabled=true"
qmgr -c "set queue batch resources_default.nodes=1"
qmgr -c "set queue batch resources_default.walltime=3600"
qmgr -c "set server default_queue=batch"
```

Additionally, you can run commands to set the administrators' e-mail:

```
qmgr -c "set server operators = root@localhost"
qmgr -c "set server operators += kwanous@localhost"
```

Sanity Check

At this point, running `qstat -q` to view available queues should give you something like this:

```
gyrfalcon:~# qstat -q

server: gyrfalcon

Queue            Memory CPU Time Walltime Node  Run Que Lm  State
---------------- ------ -------- -------- ----  --- --- --  -----
batch              --       --       --    --    0   0 --   E R
                                               ----- -----
                                                   0     0
```

Excellent, we have a queue called "batch" and it's empty. You can also view your qmgr settings with

```
qmgr -c "print server"
```

Time to try submitting a job to the queue. First, switch over to a different user account (don't run this as root) with `su - <username>`. Then, try to submit a job that just sleeps for thirty seconds and does nothing:

```
echo "sleep 30" | qsub
```

The purpose of this is to see whether the job shows in the queue when you run `qstat` after submitting it.

Below is a script of my testing it.

```
kwanous@gyrfalcon:~$ echo "sleep 30" | qsub
0.gyrfalcon
kwanous@gyrfalcon:~$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
0.gyrfalcon               STDIN            kwanous                0 Q batch
```

Excellent, the job shows up! Unfortunately, though, it won't run... the state is "Q" (I assume for "queued"), and it needs to be scheduled. That's what we'll install **Maui** for later.

Introducing Torque to the Worker Nodes

Now we need to tell the `pbs_server` which worker nodes are available and will be running `pbs_mom`, a client that allows the the server to give them jobs to run. We do this by creating the file `/var/spool/pbs/server_priv/nodes`. With your favorite text editor, add each worker node hostname on a line by itself. If they have more than one processor, add `np=X` next to the line. Mine looks like this:

```
eagle np=4
goshawk np=4
harrier np=4
kestrel np=4
kite np=4
osprey np=4
owl np=4
peregrine np=4
```

Which that, configuration on the head node for torque is done.

Installing Torque on the Worker Nodes

Now we need to install a smaller version of torque, called `pbs_mom`, on all of the worker nodes. Move back into the directory we untarred earlier, `/usr/local/src/torque*`. There's a handy way to create the packages for the torque clients. Run

```
make packages
```

and they'll be created for you. This time you'll get a confirmation message:

```
Done.

The package files are self-extracting packages that can be copied
and executed on your production machines.  Use --help for options.
```

You'll see some new files in the directory now if you run an `ls`. The one we're interested in is `torque-package-mom-linux-*.sh` where the * is your architecture. We need to copy that file to all the the worker nodes. You can either copy it over to a **shared NFS mount**, or see my **Cluster Time-saving Tricks** on how to copy a file to all the nodes using the `rsync` command. I'm copying it over to my NFS mount with

```
cp torque-package-mom-linux-i686.sh /shared/usr/local/src/
```

Once it's on each worker node, they each need to run the script with

```
torque-package-mom-linux-i686.sh --install
```

You have a couple of options for doing this on each node. You can ssh over and run it manually, or you can check out my **Cluster Time-saving Tricks** page to learn to how to write a quick script to run the command over ssh without having to log into each node. If you're going with the second route, the command to use is

```
for x in `cat machines`; do ssh $x /<full path to
package>/torque-package-mom-linux-i686.sh --install; done
```

Before we can start up `pbs_mom` on each of the nodes, they need to know who the server is. You can do this by creating a file `/var/spool/pbs/server_name` that contains the hostname of the head node on each worker node, or you can copy the file to all of the nodes at once with a short script (assuming you've created a file at `~/machines` with the hostnames of the worker nodes as outlined in the **Cluster Time-saving Tricks** page):

```
for x in `cat ~/machines`; do rsync -plarv /var/spool/pbs/server_name
$x:/var/spool/pbs/; done
```

Next, if you're using a **NFS-mounted file system**, you need to create a file on each of the worker nodes at `/var/spool/pbs/mom_priv/config` with the contents

```
$usecp <full hostname of head node>:<home directory path on head node> <home directory path on worker node>
```

The path is the same for me on my head node or worker node, and my file looks like this:

```
$usecp gyrfalcon.raptor.loc:/shared/home /shared/home
```

Again, this file can be created on each of the worker nodes, or you can create it and copy it over to each of the nodes. If you're using the latter technique, assuming you've created a `machines` file with all the host names, and you've created a `config` file, the command to run from the head node is

```
for x in `cat ~/machines`; do rsync -plarv config $x:/var/spool/pbs/mom_priv/; done
```

After you've done that, `pbs_mom` is ready to be started on each of the worker nodes. Again, you can ssh in to each node and run `pbs_mom`, or the script equivalent is

```
for x in `cat ~/machines`; do ssh $x pbs_mom; done
```

Everyone Placing Nice on Torque

Finally, it's time to make sure the server monitors the pbs_moms that are running. Terminate the current queues with

```
qterm
```

and then start up the pbs server process again

```
pbs_server
```

Then, to see all the available worker nodes in the queue, run

```
pbsnodes -a
```

(I don't know why this command doesn't have an underscore.) Each of the nodes should check in with a little report like my node peregrine's below.

```
peregrine
     state = free
     np = 4
     ntype = cluster
     status = opsys=linux,uname=Linux peregrine 2.6.21-2-686 #1 SMP Wed Jul 11 0
     3:53:02 UTC 2007 i686,sessions=? 0,nsessions=? ,nusers=0,idletime=1910856,totme
     m=3004480kb,availmem=2953608kb,physmem=1028496kb,ncpus=8,loadave=0.00,netload=18
     0898837,state=free,jobs=,varattr=,rectime=1200191204
```

Ready to continue? Move on to **installing Maui**, the scheduler.

References

- **TORQUE Quick Start Guide (http://www.clusterresources.com/torquedocs21/a.ltorquequickstart.shtml)**
- **TORQUE Quick Start Guide - Manual Server Configuration (http://www.clusterresources.com/torquedocs21/a.ltorquequickstart-manualconfig.shtml)**

Scheduler: Maui

This is the third part of a four part tutorial on installing and configuring a **queuing system and scheduler**. The full tutorial includes:

- **Using a Scheduler and Queue**
- **Resource Manager: Torque**
- **Scheduler: Maui**
- **Torque and Maui Sanity Check: Submitting a Job**

There is also a troubleshooting page:

- **Troubleshooting Torque and Maui**

About Maui

The **Maui Cluster Scheduler**, or just Maui for short, is a cluster scheduler from **Cluster Resources**. Maui needs to be installed on just the head node, and then **Torque** is used to submit jobs to this scheduler. Maui manages the clients by way of the pbs_moms.

Installing Maui

To get Maui, first visit **http://www.clusterresources.com/downloads/maui/temp/** and find the most recent version of it. At the time if this writing, that happens to be the 27-Jun-2007 snapshot. Copy the link for the location of the file. From `/usr/local/src/`, issue the following command for the most current file:

```
wget
http://www.clusterresources.com/downloads/maui/temp/maui-3.2.6p20-snap.1182974819.tar
```

Next, untar the file with

```
tar xvf maui-3.2.6p20-snap.1182974819.tar.gz
```

Move into the directory that that just created with `cd maui-*`. We're ready to run `./configure` (as part of the **Source Installation Paradigm**, which you might want to check out if this seems unfamiliar to you). We'll add a number of arguments. To see all of the possible arguments, type `./configure -help`. What we'll use is this:

```
./configure --with-pbs --with-spooldir=/var/spool/maui/
```

- `--with-pbs` makes it compatible with **Torque**
- `--with-spooldir` sets it to use `/var/spool/maui` as its home directory

If it finishes successfully, you'll see a message and a confirmation, as shown below.

```
configure: NOTE:  link 'docs/mauidocs.html' to your local website for access to
user and admin documentation
NOTE:  latest downloads, patches, etc are available at 'http://supercluster.org/
maui'

configure successful.
```

Next, run

```
make
```

If it finishes without an error, the `make` was successful. Finally, run

```
make install
```

and again, if it finishes without an error, that's a success. In order for mine to work, I had to edit `/var/spool/maui/maui.cfg`. (If you didn't change your spool directory during `./configure`, yours will be located at `/usr/local/maui/maui.cfg`.) You should have a line like

```
#RMCFG[HEADNODE] TYPE=PBS@RMNMHOST@
```

where `HEADNODE` is your head node's hostname in capital letters. Comment out this line by adding a pound symbol, `#`, in front of it. Then create a line below it:

```
RMCFG[headnode] TYPE=PBS
```

where `headnode` is your head node's hostname in lowercase letters.

Starting Maui

Now maui can be started up on the head node. Maui installs the executable to `/usr/local/maui/bin`, so you'll want to add that as part of root's path. To do this, run

```
export PATH=$PATH:/usr/local/maui/bin:/usr/local/maui/sbin
```

(To make this a permanent addition, add the above line to your `~/.bashrc` file.) Then run

```
maui
```

You won't get any output from it, but running

```
ps aux | grep maui
```

should show maui running now. In addition, running `showq` should show give you a nice view of jobs in the queue waiting to be scheduled. Currently there are none.

```
gyrfalcon:/var/spool/maui# showq
ACTIVE JOBS--------------------
JOBNAME            USERNAME      STATE  PROC   REMAINING            STARTTIME


     0 Active Jobs       0 of    0 Processors Active (0.00%)

IDLE JOBS----------------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT            QUEUETIME


0 Idle Jobs

BLOCKED JOBS----------------
JOBNAME            USERNAME      STATE  PROC     WCLIMIT            QUEUETIME
```

Sanity Check

By this point, you should have both torque and maui installed. Great! Continue onto the **sanity check** to make sure they're talking to each other.

References

- **Maui - PBS Integration Guide** (http://www.clusterresources.com/products/maui/docs/**pbsintegration.shml)**

Troubleshooting Torque and Maui

If you're visiting this page, you should have **Torque**, a resource manager, and **Maui**, a scheduler, installed on your systems. This page will cover a few places to check for information about the problem and a few "what can go wrong" scenarios.

If you're looking for how to test if a setup is working properly, or how to submit a job to the queue, see the **Torque and Maui Sanity Check: Submitting a Job** page.

Problems with Torque/Maui

Most problems can be checked along the way while installing **Torque** and **Maui**. You'll want to make sure you can run `qstat -a` and see the queues (visit the **Torque** page if you don't), and make sure you can run `showq` to see the jobs that Maui has been working with (visit the **Maui**) page if it can't).

Troubleshooting Tips

Check for Running Programs

The first step in making sure everything flows correctly is to make sure the right components are running on the right servers.

On the head node, you'll want running

- the Torque pbs_server - run `ps aux | grep pbs | grep -v grep` to verify that it is running
- Maui - run `ps aux | grep maui | grep -v grep` to verify

On the worker nodes, you'll want running

- a Torque pbs_mom - run `ps aux | grep pbs | grep -v grep` to verify

If one of these is missing, it needs to be started with the binary of that file. If you followed my setup,

- `pbs_server` is at `/usr/local/sbin/pbs_server`
- `maui` is at `/usr/local/maui/sbin/maui`
- `pbs_mom` is at `/usr/local/sbin/pbs_mom`

Otherwise, if you can't find it, you can install locate (`apt-get install locate`), run `updatedb`, and then enter

```
locate x
```

where `x` is the binary you're trying to find. It will also potentially come up with quite a few more file names that have `x` in their path.

Check the Logs

The logs are also a great source of information. On the server, you'll want to check the Torque server logs at

<your pbs root>/server_logs/ (/var/spool/pbs/server_logs if you used **my Torque setup**). Maui logs on the server are at <your maui root>/log/maui.log (/var/spool/maui/log/maui.log if you used **my Maui setup**).

On the worker nodes, you can check the pbs_mom logs. These are at <your pbs root>/mom_logs/ (/var/spool/pbs/mom_logs if you used my setup). Additionally, you can check for undelivered files on the worker nodes - these are located at <your pbs root>/undelivered/ (/var/spool/pbs/undelivered if you used the same setup as me).

Check What Nodes the Head Node Can See

When running into troubles, checking the status of the worker nodes - according to the pbs_server (the Torque server) - can sometimes be helpful. Running

```
pbsnodes
```

will show a list of what worker nodes the head node "sees", and also their status. For instance, a typical entry for a worker node in this list might look like:

```
owl
     state = free
     np = 4
     ntype = cluster
     status = opsys=linux,uname=Linux owl 2.6.21-2-686 #1 SMP Wed Jul 11 03:53:0
     2 UTC 2007 i686,sessions=? 0,nsessions=? 0,nusers=0,idletime=1266542,totmem
     =3004480kb,availmem=2954424kb,physmem=1028496kb,ncpus=8,loadave=0.00,netloa
     d=201080783,state=free,jobs=,varattr=,rectime=1201201179
```

A node whose pbs_mom is unreachable to the head node will appear like this:

```
harrier
     state = down
     np = 4
     ntype = cluster
```

Any nodes that don't show up in the list or show up as down should be further examined. It may be helpful to check my documentation on **installing Torque on worker nodes**.

Determining Which Node a Job is Running On

Particularly if you want to check a pbs_mom log or look for undelivered files, it helps to know which worker node a job is running on. This is a little cumbersome to get to, just because to the user, it's usually transparent - they don't need to know which node their submission is running on, only that it's been submitted and is running. It's not too difficult to get to, however.

As shown above, pbsnodes will show the worker nodes and their status. To narrow it down a bit and just see the name and state lines, run

```
pbsnodes | grep -v status | grep -v ntype | grep -v np
```

No Files?

There are a number of reasons why your users might not receive their `.o#` and `.e#` files after their jobs finish.

Setting up Scp

If you're using a **mounted file system**, each of your worker nodes must have a `pbs_mom/config` file to explain how to copy files back to the head node. This file should have the following contents:

```
$usecp <full hostname of head node>:<home directory path on head node> <home directory path on worker node>
```

The path is the same for me on my head node or worker node, and my file looks like this:

```
$usecp gyrfalcon.raptor.loc:/shared/home /shared/home
```

This file needs to be stored at `<your pbs root>/mom_priv/config`. (If you used **my Torque setup**, your path will be `/var/spool/pbs/mom_priv/config`.) You can create this file on each one of the worker nodes individually, or check out the **Cluster Time-saving Tricks** page to see how to do this more quickly.

Typically, the lack of this file will result in the error and output files being lost. Your users will receive e-mails from the system saying something to this affect.

Troubles with SSH

If your users aren't getting e-mails about lost files, and you've set up scp as shown above, but your users still aren't seeing their output and standard error files after jobs finish, the problem may be with SSH configuration. Run a job as a one of your users, and pay attention to which node the job runs on by running

```
pbsnodes | grep -v status | grep -v ntype | grep -v np
```

before the job finishes. Then SSH into that node when the job completes. Check `<your pbs root>/undelivered/` (if you used [[Resource Manager: Torque | my Torque setup, that's `/var/spool/pbs/undelivered/`). If you have files ending with `.OU` and `.ER`, it's probably a delivery problem due to one of two SSH problems.

No SSH Key?

Each of your users needs to have `~/.ssh/authorized_keys2` file whose contents match the contents of their `~/.ssh/id_rsa.pub` file. If you have **NFS-mounted** home directories, you only need this once. If the home directories are different on each of the nodes, rather than mounted, you'll need the same key and same authorized_keys2 file in the home directory on each one of the worker nodes.

To create this key and file for the first time, as your user, run

```
ssh-keygen
```

Keep the default location for the file, and also hit enter twice without a password. Then, run

```
cat id_rsa.pub >> authorized_keys2
```

This file needs to only be readable by the owner. Do this with

```
chmod 600 authorized_keys2
```

Strict StrictHostKeyChecking

If you followed my **Cloning Worker Nodes** tutorial, you probably disabled `StrictHostKeyChecking` for SSH before cloning all of the worker nodes. If not, this might be the problem. If you haven't changed it, the default for this setting is `ask`, meaning that when a user (or a program acting on behalf of a user) tries to SSH from one node to another node that it hasn't encountered before, the user will be prompted as to whether they would like to accept the identification the node gives and continue SSHing in. Unfortunately, this can be a show stopper if there's no user to enter `yes` when prompted.

If you'd like to test if this is the problem, SSH into one of your worker nodes, become a normal user, and try to SSH into your head node. This is the kind of output you're likely to encounter:

```
kwanous@osprey:~$ ssh gyrfalcon The authenticity of host 'gyrfalcon (192.168.1.200)' can't
be established. RSA key fingerprint is 22:98:61:31:fd:20:e8:c6:ec:47:e9:e9:ef:99:22:0d.
Are you sure you want to continue connecting (yes/no)? </pre>
```

One solution would be to SSH from each node to your head node as each one of your users. However, even with script, this can take a lot of time, and it doesn't scale well. Plus, you'd need to do this every time you added a new user. Instead, you can disable <code>StrictHostKeyChecking for SSH.

For each of your worker nodes, you'll need to open `/etc/ssh/ssh_config` and find the line that looks like

```
#   StrictHostKeyChecking ask
```

Take out the hash (`#`) to uncomment this line, and change the value from `ask` to `no`.

If you'd rather not change it manually for each of your worker nodes, check out the **Cluster Time-saving Tricks** page to learn how to automate copying files out.

If All Else Fails...

Sometimes my server just seems to need to have Torque restarted. I haven't yet diagnosed why this happens, but it may be related to accidental power cycling (long story). When it starts to have strange errors, restarting might be a viable solution. From the head node, run

```
killall -KILL pbs_server
pbs_server
```

Torque and Maui Sanity Check: Submitting a Job

This is the last part of a four part tutorial on installing and configuring a **queuing system and scheduler**. The full tutorial includes:

- **Using a Scheduler and Queue**
- **Resource Manager: Torque**
- **Scheduler: Maui**
- **Torque and Maui Sanity Check: Submitting a Job**

There is also a troubleshooting page:

- **Troubleshooting Torque and Maui**

This part tutorial assumes you have already installed and configured **Torque** and **Maui**. If you haven't, you'll want to visit those pages first.

Torque/Maui Sanity Check: Submitting a Job

A job is one particular instance of running a particular script or program of code. You won't want to run a job as root, so first, on your head node, become one of your users. (For instance, `su - kwanous`.)

Jobs are submitted to the job queue run by torque, which maui monitors and will then schedule, and torque will tell the pbs_mom client running on the worker node that maui picks to run the job. Jobs are submitted to torque with the `qsub` command.

Test: Sleep Job

An easy job to submit and monitor is just a `sleep` command.

As one of your users, enter the command that will create a job that simply sleeps for 30 seconds, as shown below:

```
echo "sleep 30" | qsub
```

Immediately afterward, run the torque command `qstat` to see the job appear in torque's queue, and then the maui command `showq`. You can even run

```
pbsnodes | grep -v status | grep -v ntype
```

to see which node the job is running on. A script of my output is shown below.

```
kwanous@gyrfalcon:~$ echo "sleep 30" | qsub
6.gyrfalcon

kwanous@gyrfalcon:~$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
6.gyrfalcon               STDIN            kwanous                0 R batch
```

```
kwanous@gyrfalcon:~$ showq
ACTIVE JOBS--------------------
JOBNAME            USERNAME      STATE  PROC   REMAINING            STARTTIME

6                   kwanous    Running     1     1:00:00  Wed Jan 23 14:00:24

     1 Active Job         1 of   28 Processors Active (3.57%)
                          1 of    7 Nodes Active      (14.29%)

... snipped ...

Total Jobs: 1   Active Jobs: 1   Idle Jobs: 0   Blocked Jobs: 0
```

```
kwanous@gyrfalcon:~$ pbsnodes | grep -v status | grep -v ntype
eagle
     state = free
     np = 4

 ... snipped ...

peregrine
     state = free
     np = 4
     jobs = 0/7.gyrfalcon
```

Approximately thirty seconds later, the job should finish running. If you run `qstat` and `showq` again, you should no longer see the job (`6.gyrfalcon`, in my example) running.

Sleep Job Results

In the home directory of the user you've submitted the job as, you should now see two files, something like:

- `STDIN.o3`
- `STDIN.e3`

where `3` is the job ID. The file ending in `.o#` is all of the output in the form of standard out that came from the job. `.e#` is all the output from standard error. For our sleep job, both of these should be empty. `sleep` doesn't give any output to standard out or standard error.

Test: Standard Output vs Standard Error

`Qsub` can also take input in the form of files. These files can give all sorts of specifications to torque about how long the job will run and what resources it needs. (To learn more about qsub submission files, see **Using Torque and Maui**.) We'll write just a simple one. Open your favorite text editor and enter the contents of my **Standard Output/Error For Loop Script** and save this file to `submission`. This script has a simple for loop that runs from 1 to 10. If the number is less than 5, it will print a statement to standard output. If the number is greater than or equal to 5, it will print a statement to standard error.

Submit the job with

```
qsub submission
```

where `submission` is the name of the script file.

Job Results

Again, you should have `.o#` and `.e#` files in your home directory, but this time they should start with the name of the file submitted to qsub (`submission`). This time, they should have content in them. Your output file should have the first four lines, which were printed to standard output:

```
1 is less than 5
2 is less than 5
3 is less than 5
4 is less than 5
```

and your error file should have the last six, which were printed to standard error:

```
5 is greater than or equal to 5
6 is greater than or equal to 5
7 is greater than or equal to 5
8 is greater than or equal to 5
9 is greater than or equal to 5
10 is greater than or equal to 5
```

Hmm...

If you didn't get the results described on this page, visiting the **Troubleshooting Torque and Maui** page might be of help.

Standard Output/Error For Loop Script

This small script is used as a submission for the **Torque and Maui sanity check**.

Standard Error vs. Standard Output Script

```
#!/bin/bash

for x in `seq 1 10`
do
  if [ "$x" -lt 5 ]; then
    echo $x is less than 5
  else
    echo "$x is greater than or equal to 5" >&2
  fi
done
```

Torque and Maui: Submitting an MPI Job

If you've installed and configured **Torque** and **Maui**, and you've run through the **basic job submission sanity check**, you're ready to break - I mean, *test* - your system with something a little more difficult. The basic jobs in the sanity check only used bash commands - commands build into Linux and the bash shell. While they're great for testing the queuer and scheduler to make sure the two are up and running with each other, they're a little simplistic for getting real research done.

Nowadays, **MPI** is very commonly used for parallel programming. In fact, many parallel software packages like **Gromacs** or **mpiBLAST** are powered by MPI.

That's why the next step in testing out the queue is to make sure it works well with MPI. This will require that **MPICH** is installed on the head node and all of the worker nodes, and that MPICH has been set up for Torque by way of **mpiexec** or that you have a **global MPD ring** started.

Creating a Basic MPI Program

The first step is to write and compile a program that uses MPI. A "hello world" type program is ideal for this. I'll be borrowing one from the **Bootable Cluster CD (http://bccd.cs.uni.edu)** project. Follow the instructions on the **Creating and Compiling an MPI Program** page to create and compile one.

Creating a Basic Qsub Script

Next, we'll need to create a qsub script. Qsub won't just take a binary file, so you can't just run `qsub hello.out`. If you do try that, you'll get an error message:

```
kwanous@gyrfalcon:~$ qsub hello.out
qsub:  file must be an ascii script
```

Most programs will need to be submitted via a qsub script. In fact, that's what allows Maui to do the scheduling that optimizes when jobs should run. There's more about qsub scripts at the **Torque Qsub Scripts** page, but for now, let's go with this:

```
#PBS -N mpi_hello
#PBS -l nodes=8
cd $PBS_O_WORKDIR

/shared/bin/mpiexec /shared/home/kwanous/mpi/hello.out
```

This script tells Torque to call this job "mpi_hello" and that it will need eight processors. Notice that the full path to the command to run (mpiexec) must be given, as well as the full path to the executable (hello.out).

Submit the job via

```
qsub <your script name>
```

You can watch the job the same way **as in the first sanity check**. When it finishes, you should have two new files in the directory you submitted the job from: mpi_hello.eXX and mpi_hello.oXX. If mpi_hello.eXX is not

empty, you should use this information to try to diagnose the problem. If everything ran successfully, you should see something like this in mpi_hello.oXX:

```
Hello MPI from the server process!Hello MPI!
 mesg from 1 of 8 on peregrine
Hello MPI!
 mesg from 2 of 8 on peregrine
Hello MPI!
 mesg from 3 of 8 on peregrine
Hello MPI!
 mesg from 4 of 8 on owl
Hello MPI!
 mesg from 5 of 8 on owl
Hello MPI!
 mesg from 6 of 8 on owl
Hello MPI!
 mesg from 7 of 8 on owl
```

The above shows that my MPI set up is working correctly: eight processors were allocated on two nodes (the nodes each have four processors each), peregrine and owl. Peregrine only has three hellos because the server process ran on one of the four processors.

Cluster Monitoring on the Web

There are quite a few free monitoring software packages available for networks and clusters. Most of these require a web server to be installed. Often this is run on the head node, and then **IPTables** can be changed to forward web traffic to that node, just like SSH traffic is forwarded to it.

On my cluster, I run both **Nagios** and **Ganglia**. Both of these require a web server to run on. There are walkthroughs for installing and configuring both of these.

Web Server

Setting up a web server is relatively easy with Debian. There are a few optional things that can then be done to **secure the web server**.

Nagios

Nagios is often used to monitor whole networks, not just clusters. It is typically used to monitor services. For instance, it comes with options to monitor web servers, SSH servers, and do ping tests to make sure hosts are alive, amongst other options. These checks are typically run from one Nagios host, with no installation necessary on the other hosts.

- **Nagios**
- **Nagios Installation and Configuration**
- **Creating Your Own Nagios Plugin**
- **Nagios NRPE Addon Installation and Configuration** - a plugin used to execute remote commands

Ganglia

Ganglia needs to be installed on each machine to be monitored. One node then runs a meta daemon and collects the information from all of the clients. Information provided by Ganglia includes cluster load, node load, memory use, and network traffic. More information about individual hosts is also available.

- **Ganglia**
- **Installing Ganglia**
- **Configuring Ganglia: the host node (gmetad)**
- **Configuring Ganglia: the client nodes (gmond)**

Setting up a Web Server

Web Server Installation

Installing the web server software, Apache 2, is almost scarily easy. To install Apache, just run

```
apt-get install apache2
```

You won't be prompted for any options, but it's done, you should have Apache running on your system. Visit **http://www.yourserver.com/** to see be redirected to a congratulatory "It works!" page. The install drops most of its important files in `/etc/apache2`, so head there next to take a look around.

The configuration specific to the web server is in `/etc/apache2/apache2.conf` and you'll add your additions this configuration in `/etc/apache2/httpd.conf`. Settings specific to the website are in `/etc/apache2/sites-available/default`. (If this seems like a subtle distinction, don't worry about the difference.)

To get rid of the annoying redirection to the "It works!" page, open up `/etc/apache2/sites-available/default` in your favorite text editor and delete these comments and the line responsible for that redirection:

```
# This directive allows us to have apache2's default start page
# in /apache2-default/, but still have / go to the right place
RedirectMatch ^/$ /apache2-default/
```

Go ahead and delete the directory it's redirecting to, too, with `rm -rf /var/www/apache2-default/`. Then restart Apache with `apache2ctl restart`.

More Bells and Whistles

At some point, you may need to install a program (by hand) or a package may be installed with PHP or MySQL as part of its requirements. These can also be manually installed.

PHP

PHP is a scripting language that builds on top of HTML and allows dynamic content. The web server interprets the script and generates HTML from the PHP code, which is then sent along as normal to clients (people opening the web page). Sometimes PHP gets a bad wrap as taking user input with PHP can introduce a lot of security holes. However, many people use it successfully with proper precautions.

PHP is installed on Debian with

```
apt-get install libapache2-mod-php5
```

MySQL

MySQL is a database implementation. It is often used in conjunction with PHP. Websites may pull from an SQL database in order to provide different types of content. To install MySQL and have it be compatible with PHP, run

```
apt-get install mysql-server-5.0 php5-mysql
```

Securing the Webserver

What's the first thing to do once the web server has been installed? Why, put some effort into **securing it**, of course!

Securing the Web Server

Securing Apache has been the subject of many books and tutorials and will continue to be as security is an ever-changing field. By no means is this short page meant to be exhaustive. The steps below are basic recommendations for beginning to protect a web server; however, much more strenuous measures are needed to begin to truly secure one. Please use the information below as a starting place only.

If you haven't yet **set up with the web server**, you may want to look at that first.

Configuring Apache

Tweaking Defaults

One of the first things to do after installing Apache is to configure where it will serve files from, and to limit the options that people accessing the web server have. This is done by editing `/etc/apache2/sites-available/default`. Look for this section:

```
<Directory />
        Options FollowSymLinks
        AllowOverride None
</Directory>
```

This section controls how Apache treats the root directory of the file system (and by inheritance, all the files within the file system). This is somewhat secure, but a more secure configuration is better. Change the section to read like this:

```
<Directory />
        Order Deny,Allow
        Deny from all
        Options None
        AllowOverride None
</Directory>
```

This tells Apache not to serve any files at all from the file system, and also to allow no special options (such as symlinking, includes, or cgi scripts), and not to allow this to be overridden by .htaccess files in the directories. This is used to protect files that Apache shouldn't have access to. However, since we do want Apache to access files from within the `/var/www` directory, we need to edit the section below it to look like this:

```
<Directory /var/www/>
        Options FollowSymLinks MultiViews
        Order allow,deny
        Allow from all
</Directory>
```

The `allow from all` is what allows Apache to serve pages from within `/var/www`. Also, removing `Indexes` means that web users will not be able to see the contents of web directories.

You'll need to restart Apache in order for this to take effect. Restart Apache with `apache2ctl restart`.

Hiding Server Version

If you open up a web browser and visit http://www.yourserver.com/nonexistant.html, you'll see an error page like the one shown below. That gives away an awful lot of information to someone interested in attacking the system!

To obfuscate this information, open `/etc/apache2/apache2.conf`. Look for the line

```
ServerTokens Full
```

and change it to

```
ServerTokens Prod
```

You'll need to restart Apache in order for this to take effect. Restart Apache with `apache2ctl restart`. Afterward, server error pages should look something more like this:

Installing Mod_Security

Getting all the Pieces

The security module, or mod_security, is an Apache module that can be installed for closer monitoring of HTTP requests and responses as well as easy denial of packets that look suspicious. Unfortunately, due to licensing differences, mod_security is not available through the Debian repository, and so it can't be apt-gotten. Still, the module isn't terribly difficult to install.

Before obtaining the source code, there are a few other parts that can be installed through apt. These are `libxml2-dev` and `apache2-prefork-dev`. To install these, run

```
apt-get install libxml2-dev apache2-prefork-dev
```

To get the source code for mod_security, you must first create a user account with **Breach Security (https://bsn.breach.com/account/login.php)** , the developers of mod_security. After logging in, navigate to Downloads and then modsecurity-apache/. Find the latest version of modsecurity-apache...tar.gz and right-click it to save the download location. From where you keep your source code, run

```
wget <download location> --no-check-certificate
```

Untar the file with `tar xvzf` and then `cd` into the new directory. From there, cd into `apache2`. Mod_security follows the typical **source installation paradigm**, so no surprises there. Run `./configure --help` to see all available options. In most cases, none will be necessary. Go ahead and run `./configure` (with any options) and then `make`. If it finishes without an error, it's safe to `make install`.

Configuring Apache

Now that the files for mod_security have been installed, Apache needs to be told to use them. Cd into `/etc/apache2/mods-available`. A new file needs to be created to tell Apache to load the mod_security module. Call this file `modsecurity2.load` and enter the following contents:

```
LoadFile /usr/lib/libxml2.so
LoadModule security2_module /usr/lib/apache2/modules/mod_security2.so
```

Now move up one directory and then into mods-enabled (`cd ../mods-enabled`). Here, a symlink to the file needs to be created. This is done with

```
ln -s ../mods-available/modsecurity2.load
```

While we're here, mod_security also requires mod_unique_id to be running, so create a symlink to enable that one, too:

```
ln -s ../mods-available/unique_id.load
```

After this, it's time to restart Apache and make sure it loads the new file. Run `apache2ctl restart` and then look at the bottom of the Apache log with `tail /var/log/apache2/error.log`. You should see something like this:

```
[Sun Apr 06 18:54:25 2008] [notice] ModSecurity for Apache/2.5.2 (http://www.modsecurity.org/) configured.
```

If you don't, or you see any errors, double check the above and make sure you're error-free before continuing.

Configuring Mod_security

At this point, mod_security is loaded but isn't actually accomplishing anything because it hasn't yet been configured. First, it needs some rules to tell it what to look for while it's watching the web traffic. A core set of these rules can be downloaded from **Breach Security (https://bsn.breach.com/account/login.php)** , so log back into their downloads section and copy the location of the tar file containing the latest rules. From your source directory, run

```
wget <file location> --no-check-certificate
```

Untarring it won't put the contents into a directory, so don't untar it and make a mess just yet. First, create a directory to become a new home for the rules, like `mkdir /var/lib/modsecurity`. Remember this location because you'll need to reference it a few more times. Next, untar the file with

```
tar xvzf modsecurity-core-rules*.tar.gz
```

and then move all the new files into the directory you just created:

```
mv modsecurity_crs_* /var/lib/modsecurity/
```

You'll also want to tweak the location where the log files are kept. Open `modsecurity_crs_10_config.conf` and search for SecAuditLog and SecDebugLog. Change these so that they point to a file under `/var/log/apache2`.

Next, Apache needs to be told where the rules are at. Create the file `/etc/apache2/conf.d/modsecurity2.conf` and give it these contents:

```
<ifmodule security2_module>
   Include /var/lib/modsecurity/*.conf
</ifmodule>
```

Testing Mod_security

At this point, it would be nice to test mod_security and make sure it's really filtering through the traffic. This can be done with a simple `wget` test. Wget can be configured to identify itself as something other than wget, so all we need to do is change it to something that should trigger mod_security. You'll probably want to do this from somewhere other than the web server itself. If you followed my **firewall services** tutorials, for instance, it won't work unless running this command from somewhere outside of the firewall.

```
wget -O - -U "webtrends security analyzer" http://<your server>
```

If mod_security is working, the machine you issued the command from should receive a 404 error:

```
kwanous@cassowary:~$ wget -O - -U "webtrends security analyzer" http://myserver.goes.here/
--19:13:17--  http://myserver.goes.here/
          => `-'
Resolving myserver.goes.here... X.X.X.X
Connecting to myserver.goes.here|X.X.X.X|:80... connected.
HTTP request sent, awaiting response... 404 Not Found
19:13:17 ERROR 404: Not Found.
```

From your webserver, if you run

```
tail /var/log/apache2/debug.log
```

or

```
tail /var/log/apache2/audit.log
```

you'll see why!

```
[06/Apr/2008:19:13:17 --0500] [myserver.goes.here/sid#855d8f0][rid#87357f8][/][1]
  Access denied with code 404 (phase 2). Pattern match "(?:\b(?:m(?:ozilla\/4\.0 \
  (compatible\)|etis)|webtrends security analyzer|pmafind)\b|n(?:-stealth|sauditor|
  essus|ikto)|b(?:lack ?widow|rutus|ilbo)|(?:jaascoi|paro)s|webinspect|\.nasl)" at
  REQUEST_HEADERS:User-Agent. [file "/var/lib/modsecurity/modsecurity_crs_35_bad_robots.conf"]
  [line "19"] [id "990002"] [msg "Request Indicates a Security Scanner Scanned the Site"]
  [severity "CRITICAL"] [tag "AUTOMATION/SECURITY_SCANNER"]
```

Because the user agent string in the HTTP request matched the above regular expression, which includes "webtrends security analyzer", the request for the file is denied. This shows that mod_security is working!

To further crack down, additional rule sets can be downloaded from the **Got Root (http://www.gotroot.com/)** website. Be sure to place them in the same directory as the other rules (for instance, `/var/lib/modsecurity/`) and if you put them in a subdirectory, update `/etc/apache2/conf.d/modsecurity2.conf`.

Occasionally, rules in mod_security will be tripped over harmless traffic, or traffic you'd like to allow through. You can always look at audit.log or debug.log to identify the id of the rule and disable it if necessary. (Make sure to restart Apache - `apache2ctl restart` - for changes to take effect.)

Additional Mod_security Rules

The **Got Root (http://www.gotroot.com/)** website releases periodically updated mod_security rules freely available for download. These can be used in addition to the core rules package. After downloading them and placing them in `/var/lib/modsecurity/gotrootrules` (or wherever you put them), you'll need to update `/etc/apache2/conf.d/modsecurity2.conf` to point to the additional files as well as the previous ones.

References

- Ristic, Ivan. Apache Security. 1st ed. Sebastopol, CA: O'Reilly Media, Inc., 2005.

Nagios

From the official **Nagios** website, Nagios is

> *a host and service monitor designed to inform you of network problems before your clients, end-users or managers do.*

Nagios can be used to monitor many different aspects of a cluster through a web interface. It runs on one node and can check various services on the host node and other nodes, including doing ping tests to see if machines are up, attempting to access web services, and checking that SSH is up and running. It can also be configured to notify different users in the events of errors. Depending on user level, it can be configured very simply or with a lot complexity.

Below is a section for anyone new to Nagios on understanding the basic definitions necessary for setup. There are also tutorials on

- **Nagios Installation and Configuration**
- **Creating Your Own Nagios Plugin**
- **Nagios NRPE Addon Installation and Configuration** - executing remote commands

Understanding Nagios Basics

Although Nagios can be very simple (or very complicated, depending on the setup), it does require some very specification declarations (which it calls definitions). In order to get the most of Nagios, you'll want to know about host definitions, hostgroup definitions, service definitions, and servicegroup definitions.

Host Definitions

Each host that Nagios will monitor needs a host definition. These look something like the following:

```
define host {
        host_name   gyrfalcon.raptor.loc
        address     192.168.1.200
        use         generic-host
}
```

- `host_name` is the fully qualified name
- `address` is the IP address of the host
- `use generic-host` means this host will use the generic-host template, but this can be further customized.

If you have fifty hosts to monitor, you'll need 50 host declarations. The **Nagios Installation and Configuration** tutorial will walk you through scripting this so you don't have to write it by hand.

Hostgroup Definitions

For ease of reference in the service definitions, hosts are grouped together with hostgroup definitions. These look something like this:

```
define hostgroup {
        hostgroup_name nodes
        alias          Worker Nodes
        members        kestrel.raptor.loc, owl.raptor.loc, goshawk.raptor.loc, ...snipped..., peregrine.rapto
}
```

- `hostgroup_name` is how you'll refer to these hosts
- `alias` is what will show up in the Nagios web interface
- `members` is a comma-separated list of the fully qualified domain names of each of the hosts in this group

Service Definitions

Each monitoring command to be run is defined as a service. Services are defined as shown below.

```
define service {
        hostgroup_name                  ping-servers
        service_description             PING
        check_command                   check_ping!100.0,20%!500.0,60%
        use                             generic-service
        notification_interval           0 ; set > 0 if you want to be renotified
}
```

- `hostgroup_name` is a comma separated list of all the hostgroups that have this service monitored
- `service_description` will show up on the Nagios web interface
- `check_command` is the name of the command to be executed as well as a an exclamation point (!) separated list of arguments
- `use generic-service` specifies that this service should use the generic template
- `notification-interval` is how often the admin should be renotified if the service stays down

Servicegroup Definition

The last definition I'll cover is a servicegroup definition. These aren't strictly necessary, but they can help you group hosts and services together to better view them in the Nagios web interface. (These are accessed under the "Servicegroup" links on the left bar.) They basically group hosts together based on services. You define which hosts and which services are grouped together, and this is strictly for display and notification purposes. A servicegroup definition could look like this:

```
define servicegroup {
        servicegroup_name       pbsmoms
        alias                   All PBS Nodes
        members                 gyrfalcon.raptor.loc, MPI, goshawk.raptor.loc, MPI, ...snipped... peregrine.ra
}
```

- `mpi_group` is how this servicegroup will be referred to
- `alias` will show up in the Nagios web interface
- `members` is a comma separated list of this form: host 1 full name, host 1 service, host 2 full name, host 2 service, etcetera. All of mine in the above example are running the `MPI` service, but that doesn't have to be the case.

References

- **http://www.nagios.org/**

Nagios Installation and Configuration

Installing Nagios

First, you're going to need a list of all the hosts you want to monitor, but listed as their fully qualified names. For example, mine are `eyrie.raptor.loc`, `gyrfalcon.raptor.loc`, etcetera - one for each host I want to monitor.

Nagios must be installed on a machine running a **web server**. Install it with

```
apt-get install nagios2
```

*Note: As part of the package install, this will also install **exim**, a mail transfer agent. More documentation to follow soon (hopefully) about setting up mail.*

Configuring Nagios

Web

At this point, you should be able to visit **http://yourserver/nagios2/** and be prompted for a username and password. Great! What are they? Well, they haven't been created yet. Create them with

```
htpasswd -c /etc/nagios2/htpasswd.users nagiosadmin
```

and specify the password you want. Reload the page, enter the username and password, and you should see something **like this**. If you click on anything under the "Monitoring" headline, you'll get a nice little error message: "Whoops! Error: Could not read host and service status information!" That's because there isn't yet any information for it to be reading. We need to go ahead and specify this.

Configuration: Hosts

Nagios drops a configuration files into `/etc/nagios2/conf.d/` as part of the base install. We'll use these as bases for configuration Nagios. First, we'll need a template file. Copy `host-gateway_nagios2.cfg` to `/tmp/cluster-nagios.skel` with

```
cp host-gateway_nagios2.cfg /tmp/cluster-nagios.skel
```

Next, open `/tmp/cluster-nagios.skel` in a text editor. Cut out the top line and change the values to variables we'll be replacing, so the file looks like this:

```
define host {
        host_name       HOSTNAME
        alias           ALIAS
        address         IP
        use             generic-host
}
```

We'll need to create a host entry for each machine to monitor. If you have that file with the hostnames, IP addresses, and fully qualified names, copy that to `/tmp/nagioshosts`. If you don't have one, make one,

because you'll need it. Make sure your file of hosts is at `/tmp/hosts`. Paste this script into a new file, make it executable (`chmod o+x yourfilename`) and then run it (`./yourfilename`).

```
#!/bin/bash
# Short script to generate nagios host entries
# Assumes you have a file called /tmp/hosts with format
# Hostname  IP  Fully qualified name
#
# Also assumes you have a /tmp/cluster-nagios.skel file that looks like this
# define host {
#        host_name    HOSTNAME
#        alias        ALIAS
#        address      IP
#        use          generic-host
# }

for x in `cat /tmp/hosts`
  do
    y=`echo $x | cut -d . -f1`
    z=`host $x | awk '{print $3}'`
    echo "Creating entry host_name = $x , alias = $y, IP = $z"
    sed "s/HOSTNAME/$x/g" /tmp/cluster-nagios.skel | sed "s/IP/$z/g" | sed "s/ALIAS/$y/g" >> /tmp/cluster-nagi
  done
```

After you run it, `/tmp/cluster-nagios.cfg` should have one entry for each host. Copy this file to the nagios2 configuration directory:

```
cp /tmp/cluster-nagios.cfg /etc/nagios2/conf.d/
```

Next, try to start up (or restart) nagios with

```
/etc/init.d/nagios2 restart
```

Check the log to make sure it started up correctly. If it did, you should have one entry about "no services associated" for each host.

```
gyrfalcon:/etc/nagios2/conf.d# tail -n 20 /var/log/nagios2/nagios.log
[1197252453] Nagios 2.10 starting... (PID=7448)
[1197252453] LOG VERSION: 2.0
[1197252453] Finished daemonizing... (New PID=7449)
[1197253356] Caught SIGTERM, shutting down...
[1197253356] Successfully shutdown... (PID=7449)
[1197256272] Nagios 2.10 starting... (PID=8371)
[1197256272] LOG VERSION: 2.0
[1197256272] Warning: Host 'eagle.raptor.loc' has no services associated with it!
[1197256272] Warning: Host 'eyrie.raptor.loc' has no services associated with it!
[1197256272] Warning: Host 'goshawk.raptor.loc' has no services associated with it!
[1197256272] Warning: Host 'gyrfalcon.raptor.loc' has no services associated with it!
[1197256272] Warning: Host 'harrier.raptor.loc' has no services associated with it!
[1197256272] Warning: Host 'kestrel.raptor.loc' has no services associated with it!
[1197256272] Warning: Host 'kite.raptor.loc' has no services associated with it!
[1197256272] Warning: Host 'osprey.raptor.loc' has no services associated with it!
[1197256272] Warning: Host 'owl.raptor.loc' has no services associated with it!
[1197256272] Warning: Host 'peregrine.raptor.loc' has no services associated with it!
[1197256273] Finished daemonizing... (New PID=8372)
```

Configuration: Services

Hoorah. Now we've given Nagios the names of the hosts to monitor, but we haven't told them what we want to monitor. The easiest thing to monitor is whether a given host is currently up, and we can do that with ping. Again, we'll need our list of all the hosts. To create a comma-separated list of all the hosts to add, issue

```
for x in `cat /tmp/hosts`; do echo "$x," >> /tmp/hoststoadd; done
```

Then open `/tmp/hoststoadd` in a text editor and delete all the blank lines so all of these are on one line. Then copy the contents of the file. We'll need to add all the hosts to `/etc/nagios2/conf.d/hostgroups_nagios2.cfg`. Open `hostgroups_nagios2.cfg` and look for this section:

```
# nagios doesn't like monitoring hosts without services, so this is
# a group for devices that have no other "services" monitorable
# (like routers w/out snmp for example)
define hostgroup {
        hostgroup_name  ping-servers
                alias           Pingable servers
                members         gateway
        }
```

Delete `gateway` (it should be in your list of hostnames anyway), then paste your list in. Again, restart nagios:

```
/init.d/nagios2/restart
```

Cleaning up Extra Cruft

Installing the nagios2 package causes some groups and services to automatically be created for you, and you'll see those entries on the Nagios web-site of things. I prefer to only have what I've configured showing.

To clean up, first create a backup directory. I did

```
mkdir /etc/nagios2/old-conf.d-stuff
```

Then, move `localhost_nagios2.cfg`, `host-gateway_nagios2.cfg`, and `extinfo_nagios2.cfg` into that directory. Move a copy of `hostgroups_nagios2.cfg` into that directory.

```
mv /etc/nagios2/conf.d/localhost_nagios2.cfg /etc/nagios2/old-conf.d-stuff/
mv /etc/nagios2/conf.d/extinfo_nagios2.cfg /etc/nagios2/old-conf.d-stuff/
mv /etc/nagios2/conf.d/host-gateway_nagios2.cfg /etc/nagios2/old-conf.d-stuff/
cp /etc/nagios2/conf.d/hostgroups_nagios2.cfg /etc/nagios2/old-conf.d-stuff/
```

Next, edit `/etc/nagios2/conf.d/hostgroups_nagios2.cfg` and take out (or comment out) the sections for the following host_groups:

- debian-servers
- ssh-servers
- http-servers (or change the value in this entry from `localhost` to the fully qualified name of your web server, ie `gyrfalcon.raptor.loc`

Leave no references to `gateway` or `localhost`.

After that, you'll need to comment out any services that you're no longer using in the file `/etc/nagios2/conf.d/services_nagios2.cfg`

Finally, restart nagios. If you haven't gotten out all the entries for localhost or gateway, or you missed moving a file, you'll see any error like this -

```
gyrfalcon:/etc/nagios2/conf.d# /etc/init.d/nagios2 restart
Restarting nagios2 monitoring daemon: nagios2
Nagios 2.10
Copyright (c) 1999-2007 Ethan Galstad (http://www.nagios.org)
Last Modified: 10-21-2007
License: GPL

Reading configuration data...

Error: Could not find any host matching 'localhost'
Error: Could not expand member hosts specified in hostgroup
(config file '/etc/nagios2/conf.d/hostgroups_nagios2.cfg', starting on line 25)

***> One or more problems was encountered while processing the config files...

     Check your configuration file(s) to ensure that they contain valid
     directives and data defintions.  If you are upgrading from a previous
     version of Nagios, you should be aware that some variables/definitions
     may have been removed or modified in this version.  Make sure to read
     the HTML documentation regarding the config files, as well as the
     'Whats New' section to find out what has changed.

errors in config!
 failed!
```

At least the error message is pretty helpful - it will give you the name of the file with the error. Edit/move that file, then try restarting again.

Creating Your Own Nagios Plugin

The Good News

Creating your own custom Nagios plugin turns out to be surprisingly easy. This is a selling point for Nagios: it's easily customizable and extensible. As far as I can tell, you can implement your script in just about any language (Bash has worked well for simple ones for me), keeping in mind only a few minor points:

- This script will be run by the `nagios` user, so it can't run anything requiring root privileges.
- The script must execute locally, unless you're implementing this with the **NRPE Nagios client** plugin.
- The script exit with specific values:
  - 0 - all ok
  - 1 - warning
  - 2 - critical
  - 3 - unknown

Implementing your plugin is basically a three-step process: writing the plugin, adding the commands for the plugin, and then adding the service(s) for the plugin.

Writing the Plugin

As I stated above, you can use any language you want to create your plugin, as far as I know, as long as it can print to standard output (like Bash's `echo` or C's `printf`) and as long as it can return with an integer value (0, 1, 2, or 3). For one of my simpler plugins, I decided to write a script to check whether the pbs_mom on any given node was up and available. I wrote a little script in Bash to do this: **Nagios Pbs_Mom Plugin**.

Notice that my script expects the hostname of the target node to be given to it. Nagios will take care of that part for us when we set up the command. Notice also that my script ends with different values depending on the status of the service (the pbs_mom service, in this case).

Nagios keeps all the basic plugins in `/usr/lib/nagios/plugins/`, and I like to put mine in `/usr/lib/nagios/plugins/local/`, but you can put yours whenever you want. You'll be specifying the full path to the plugin later, anyway.

When you're writing the plugin, remember that you need to indicate to Nagios what kind of result you found by returning 0 (ok), 1 (warning), 2 (critical), or 3 (unknown). Standard output from the program will also be captured and displayed in the web interface, so it's helpful to write a little message about what the success/error was.

Adding the Plugin Commands to Nagios

Each one of the built-in plugins comes with a file describing its "commands", the name you put under `check_command` in a `services` directive. (If this doesn't make sense to you, you might want to check out the **Nagios installation and configuration tutorial**). For instance, the ping service runs a command called `check_ping`, which is defined in `/etc/nagios-plugins/config/ping.cfg`. You'll need a create a new file in this same directory. You can name it whatever you want, as long as it ends in `.cfg`, but make sure you choose a name that correlates well with whatever the plugin will monitor.

A plugin can have more than one possible command. For each command, you'll need an entry in the `.cfg` file with this format:

```
define command {
        command_name <command name>
        command_line <full path to plugin> $HOSTADDRESS$
}
```

My command for the pbs_mom plugin looks like this:

```
define command {
        command_name check_pbsmom
        command_line /usr/lib/nagios/plugins/local/check_pbsmom $HOSTADDRESS$
}
```

You can add extra arguments as needed with whatever flags they need, as well. For instance, one of the default Nagios plugins has a command defined in `ping.cfg` as shown below:

```
# 'check_ping' command definition
define command{
        command_name    check_ping
        command_line    /usr/lib/nagios/plugins/check_ping -H $HOSTADDRESS$ -w $ARG1$ -c $ARG2$
}
```

Adding the Plugin Service(s) to Nagios

Finally, the command can be implemented as a service in Nagios. You can put this under `/etc/nagios2/conf.d/services_nagios2.cfg` or your own custom-made `.cfg` file in the same directory. At any rate, you'll need to define a service with the syntax shown below:

```
define service {
        hostgroup_name <hostgroup to check>
        service_description <short description>
        check_command <command>
        use generic-service
}
```

- `hostgroup_name` is where you list all of the hostgroups that this check should be performed on. You can have more than one hostgroup listed as long as they're comma-separated.
- `service_description` is a short (a few letters) name of the service. These are typically uppercase by convention... things like FTP, WEB, MPD, or such. These will be displayed in the Nagios web interface.
- `check_command` is the command you defined in the previous section of this tutorial.
- `use generic-service` means that this should use the template called `generic-service`. To further customize it, you can write your own template.

My service definition for the pbs_mom plugin looks like this:

```
define service {
        hostgroup_name raptor_nodes
        service_description PBSMOM
        check_command check_pbsmom
        use generic-service
}
```

Following up with the more arguments example, the built-in ping service example looks like this:

```
# check that ping-only hosts are up
define service {
        hostgroup_name                  ping-servers
        service_description             PING
        check_command                   check_ping!100.0,20%!500.0,60%
        use                             generic-service
        notification_interval           0 ; set > 0 if you want to be renotified
}
```

Notice the arguments are separated with exclamation marks.

Restarting Nagios

At this point, you're ready to restart Nagios with

```
/etc/init.d/nagios2 restart
```

If the restart fails, check the file on the line-number it gives you at the beginning of the error. Once it starts correctly, open up the web interface and make sure you're getting the correct output. If you see a response "out of bounds", make sure your command definition is accessing the plugin at the correct place!

Restarting Nagios

At this point, you're ready to restart Nagios with

```
/etc/init.d/nagios2 restart
```

If the restart fails, check the file on the line-number it gives you at the beginning of the error. Once it starts correctly, open up the web interface and make sure you're getting the correct output. If you see a response "out of bounds", make sure your command definition is accessing the plugin at the correct place!

References

- **Writing an own Service with Nagios (http://lena.franken.de/nagios/own_service.html)** - the English here is a bit shaky, but it's still a good resource.
- **IBM: Leverage Nagios with plug-ins you write (http://www.ibm.com/developerworks/aix/library/au-nagios/index.html)**

Nagios Pbs Mom Plugin

This is part of the **Creating Your Own Nagios Plugin** tutorial and is only meant as a simple example.

check_pbsmom

```
#!/bin/bash
# File: /usr/lib/nagios/plugins/check_pbsmom

if [ -z "$1" ]
then
  echo "No hostname given."
  exit 3
fi

hostname=$1

pbsoutput=`pbsnodes $hostname 2>&1`

if [ "$?" -gt 0 ]
then
  error=`echo $pbsoutput | grep MSG | awk -F "MSG=" '{print $2}'`
  echo Error: $error
  exit 1
fi

state=`pbsnodes $hostname | awk 'NR==2 {print $3}'`

if [ "$state" = "down" ]
then
  echo Pbsmom is unreachable!
  exit 2
elif [ "$state" = "free" ]
then
  echo Pbsmom is running.
  exit 0
else
  echo Other pbsmom state: $state
  exit 1
fi

exit 3
```

Nagios NRPE Addon Installation and Configuration

Nagios is a service that, by default, runs on only one node. That host node can be used to check various services on other nodes - including SSH, ping, web services, and many others - but it can't execute commands on remote machines. In order to do that, you need to install the Nagios NRPE plugin, sometimes called the Nagios client. This plugin has two components: a simple plugin for Nagios on the host machine, and an NRPE daemon. The daemon needs to be installed on every machine than the host will be running remote commands on.

Installation on the Host Node

First, you'll need to install the NRPE plugin on the Nagios host node. This is the host that runs the web server and that you've already set up Nagios on. (If you haven't already set up Nagios, **this tutorial** will help you, and you should do that first.)

To install the host side of NRPE, issue

```
apt-get install nagios-nrpe-plugin
```

Installation on the Clients

Each one of the clients will need to have this installed and configured. On each machine that the Nagios host will contact to execute plugins remotely, issue

```
apt-get install nagios-nrpe-server
```

- When prompted for `Workgroup/Domain Name:` enter your internal domain name
- When asked, `Modify smb.conf to use WINS settings from DHCP?`, keep the default of no

Depending on the version of Debian you're running, this may or may not install all of the default Nagios plugins. If you want them (if you're not just running your own **custom plugins**), install them with

```
apt-get install nagios-plugins
```

Configuration on the Clients

Next, `/etc/nagios/nrpe_local.cfg` needs to be edited on each one of the clients. Add the line

- `allowed_hosts=<Nagios host IP>`

and put in the IP address of the machine that runs Nagios. Then, for each plugin you want to be able to run remotely, add a line like this:

- `command[<command name>]=<full path to plugin and any arguments>`

Then you'll need to restart the NRPE daemon with

```
/etc/init.d/nagios-nrpe-server restart
```

If you create this file once, you can use my **Cluster Time-saving Tricks** to copy it over to the rest of the nodes and also restart the NRPE on all of them.

Configuration on the Head Node and Sanity Check

Once you've finished with all of the clients, you should be ready to implement a remote check on one of the user nodes. I'm going to use one of the built-in plugins, so I'm assuming you've installed those. If you haven't, a custom plugin will work just as well.

First, you'll need to edit `/etc/nagios-plugins/config/check_nrpe.cfg`. I changed this file to get it to work for me, because I wanted to use the base `check_nrpe` without having an extra incorrect argument. I commented out the first command (`check_nrpe`) and changed the second command definition from `check_nrpe_1arg` to just `check_nrpe`.

Then, restart Nagios:

```
/etc/init.d/nagios2 restart
```

Now you're ready to check it. Whatever command you're going to run, you need to have this command set up on the client side as shown above. Some of these are already defined in `/etc/nagios/nrpe.cfg`, but if the plugins weren't automatically installed for you (and you didn't apt-get install them), they might not work. I'm going to run the command `check_users`. To do this, issue

```
/usr/lib/nagios/plugins/check_nrpe -H <fully qualified host name> -c check_users
```

If it finishes correctly, you should see the number of users currently logged into the system.

Possible Problems

There are a number of potential problems at this point. You'll see an error like this,

```
Connection refused by host
```

if you didn't put the correct IP address in the nrpe_local.cfg file on the client (see above), or if you did it correctly but forgot to restart NRPE on the client.

Another error,

```
NRPE: Command 'check_users' not defined
```

indicates that you didn't define the commands you wanted, again on the client, or you didn't restart NRPE on the client after you defined them.

Finally,

```
NRPE: Unable to read output
```

usually means that the path to the plugin to run is incorrect on the client. If you change it, remember to restart NRPE again.

Bringing it All Together

Once you get it working, you're finally ready to implement the command into Nagios on the Nagios host. You'll do this by defining a service. Either in `/etc/nagios2/conf.d/services_nagios2.cfg` or in a file of your own creation in the same directory, add a section like this:

```
define service {
        hostgroup_name <hostgroup to take monitor>
        service_description <short description>
        check_command check_nrpe!<command to run remotely>
        use generic-service
}
```

Then open your web interface and enjoy the new functionality!

References

- **Nagios NRPE Documentation (http://nagios.sourceforge.net/docs/nrpe/NRPE.pdf)**
- **Debian Help: Complete Nagios Configuration and NRPE Addon Configuration (http://www.debianhelp.co.uk/nagiosconfig.htm)** - this is a bit out of date for the package
- **FreeBSD: Tutorial for the configuration of Nagios and the NRPE plug-in to execute check scripts with several parameters (http://www.acodedb.com/70/tutorial-for-the-configuration-of-nagios-and-the-nrpe-plug-in-to-exe**

Ganglia

About Ganglia

Ganglia is a monitoring system out of Berkeley designed specifically for clusters and grids. It can be used to monitor workload at a glance. Unlike **Nagios**, which runs by default on a single host machine, roles are split amongst machines with Ganglia. The head node or **web server** can run the Ganglia Meta Daemon (called "gmetad"), and each worker node runs a Ganglia Monitoring Daemon ("gmond").

If your web server runs on your head node, this is easy. Otherwise, you may want to install it on the web server instead of the head node.

The tutorial for Ganglia is broken into three parts:

- **Installing Ganglia**
- **Configuring Ganglia: the host node (gmetad)**
- **Configuring Ganglia: the client nodes (gmond)**

References

- **Ganglia Monitoring System (http://www.ganglia.info/)**
- **Ganglia Wiki: Readme (http://ganglia.wiki.sourceforge.net/ganglia_readme)**

Ganglia: Host Configuration

This is part two of a three-part tutorial on installing and configuring **Ganglia** on Debian. The full tutorial includes

- **Installing Ganglia**
- **Configuring Ganglia: the host node (gmetad)**
- **Configuring Ganglia: the client nodes (gmon)**

Reminder

By this point, you should have already installed the Ganglia meta daemon on the head node with

```
apt-get install gmetad
```

/etc/gmetad.conf

The base of controls for gmetad in Debian is `/etc/gmetad.conf`. Going through the file from the beginning to the end, here are a few values to possibly change. You can search for these inside your favorite text editor.

Required Changes

Some of these are commented out by default (they have a # in front of the line). They need to be uncommented to work.

- `authority` - This should be set to yourhost.yourdomain.com/ganglia. If you're behind a firewall and the URL appears as the firewall's, you should use that. For instance, my webserver is gyrfalcon, but through **NAT with IPTables**, my url appears as eyrie.mydomain.edu, and so I use that URL for `authority`.
- `trusted_hosts` - If your webserver has multiple domain names, they should all be listed here. Otherwise, this can remain empty.

Optional Changes

- `gridname` - If you don't like having the overall wrapper named "Grid", you can change it to something else.
- `rrd_rootdir` - Ganglia needs to store a lot of data for RRD. If you want this stored some place other than `/var/lib/ganglia/rrds`, change this value.

Restarting Ganglia

After any changes, gmetad will need to be restarted. Do this with

```
/etc/init.d/gmetad restart
```

Configuring Gmon

The host running gmetad is probably also running gmon, if you want to monitor this host. It will need to be configured as **a client node** also.

Ganglia: Installation

This is part one of a three-part tutorial on installing and configuring **Ganglia** on Debian. The full tutorial includes

- **Installing Ganglia**
- **Configuring Ganglia: the host node (gmetad)**
- **Configuring Ganglia: the client nodes (gmon)**

Supporting Packages

Ganglia uses RRDTool to generate and display its graphs and won't work without it. Fortunately, rrdtool doesn't need to be installed from source. Just run

```
apt-get install rrdtool librrds-perl librrd2-dev
```

In order to see the pie charts in PHP, you'll need an additional package:

```
apt-get install php5-gd
```

Getting Ganglia

Except for the frontend, Ganglia is in the Debian repository and easy to install. On the webserver, you'll need both packages, so

```
apt-get install ganglia-monitor gmetad
```

On all of the worker nodes (and head node, if it isn't running your webserver), you'll only need `ganglia-monitor`. You can `apt-get` this one at a time or see the **Cluster Time-saving Tricks** for tips on how to write a script.

Getting the Ganglia Frontend

Unfortunately, the frontend requires an entire build of Ganglia. That doesn't detract from the convenience of having the other two packages in the Debian repository, though, since you'll only need to build this one.

Visit **Ganglia's SourceForge download page** and copy the file location of the most recent version (the file should end in `.tar.gz`). Then, from whenever you keep your source files, run

```
wget
http://downloads.sourceforge.net/ganglia/ganglia-3.0.7.tar.gz?modtime=1204128965&big_
```

or similar. Then untar the file with

```
tar xvzf ganglia*.tar.gz
```

and cd into the new directory. Ganglia follows the typical **source installation paradigm**. Run `./configure --help` to see all of the options. Important ones include

- `--prefix=` - specify where the binaries should be installed, optional ones are often put in `/opt`
- `--enable-gexec` - use gexec support
- `--with-gmetad` - compile and install the metad daemon

The full line to enter should look similar to this:

```
./configure --prefix=/opt/ganglia --enable-gexec --with-gmetad
```

If it errors out and is unable to find rrd.h, make sure you installed everything listed above under "RRDTool". When it finishes successfully, you should see a screen like this:

```
Welcome to..
     ______                    ___
    / ____/___ _____  ____ _/ (_)___ _
   / / __/ __ `/ __ \/ __ `/ / / __ `/
  / /_/ / /_/ / / / / /_/ / / / /_/ /
  \____/\__,_/_/ /_/\__, /_/_/\__,_/
                   /____/

Copyright (c) 2005 University of California, Berkeley

Version: 3.0.7 (Fossett)
Library: Release 3.0.7 0:0:0

Type "make" to compile.
```

Go ahead and enter `make`, and after that finishes, `make install`.

Finally, when it's done, create a ganglia directory and copy all of `/web` to it:

```
mkdir /var/www/ganglia
cp web/* /var/www/ganglia
```

Apache

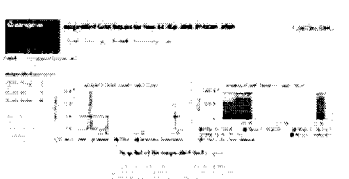
For convenience, you can update Apache to redirect any HTML requests for the root directory (ie, yourserver.yourdomain.com) to go straight to `/ganglia`. Open `/etc/apache2/sites-enabled/000-default` and find the block of code for the `/var/www` directory and add a redirect. The block should now look like this:

```
<Directory /var/www/>
        Options FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
        allow from all
        RedirectMatch ^/$ /ganglia/
</Directory>
```

After this, restart Apache:

```
apache2ctl restart
```

When you visit yourserver.yourdomain.com for the first time, you should be redirected to yourserver.yourdomain.com/ganglia, and you should have an "unspecified Grid report" as shown to the right. Cool!



First view of Ganglia.

Next...

Next, it's time to get rid of some of those "unspecifieds". Continue on to **configuring the Ganglia host's gmetad**.

Ganglia: Node Configuration

This is part three of a three-part tutorial on installing and configuring **Ganglia** on Debian. The full tutorial includes

- **Installing Ganglia**
- **Configuring Ganglia: the host node (gmetad)**
- **Configuring Ganglia: the client nodes (gmond)**

Reminder

By this point, you should have run

```
apt-get install ganglia-monitor
```

on each of the nodes you're configuring to be monitored by Ganglia.

/etc/gmond.conf

The file responsible for connecting each node appropriate to the server hosting Ganglia is `/etc/gmond.conf`. This file needs to be edited appropriately for each node. This can be done individually, or one file can be created on one node and it can be **scripted and copied out** to each one of the nodes.

The following values need to be edited:

- `name` - This is the name of the cluster this node is associated with. This will show up on the web page.
- `owner` - Different owners will be used to separate different clusters into administrative domains. If you only have one cluster, it's not such a big deal.
- `mcast_if` - If the node has multiple interfaces, the one to be used to connect to the host should be specified.
- `num_nodes` - The number of nodes in the cluster.

Restarting Ganglia Monitoring

After making the changes, gmond needs to be restarted on the node. Do this with

```
/etc/init.d/ganglia-monitor restart
```

Restarting Ganglia Host

After making the changes on all the nodes, gmetad on the webserver needs to be restarted. Do this with

```
/etc/init.d/gmetad restart
```

You may need to wait around ten minutes to see your changes take affect.