Beowulf Cluster Design and Setup

Amit Jain Department of Computer Science Boise State University

April 26, 2006

This material is based upon work supported by the National Science Foundation under grant No. 0321233, Development of Tools to Enable the Port of Software to a Beowulf Cluster. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Download from: http://cs.boisestate.edu/~amit/research/beowulf/beowulf-setup.pdf

Contents

1	\mathbf{Des}	Designing a Beowulf Cluster								
	1.1	Nodes	4							
		1.1.1	Hardware	4						
		1.1.2	Benchmarking	4						
		1.1.3	Node Software	5						
	1.2	Networ	king Hardware	5						
		1.2.1	What Type of Network to Use?	5						
		1.2.2	Network Topology Design	6						
		1.2.3	Ethernet Protocol	6						
		1.2.4	Ethernet Packets	6						
		1.2.5	TCP/IP Addresses	7						
		1.2.6	Network Topology Options	7						
		1.2.7	Network Interface Cards	8						
	1.3	Networ	k Hardware Design: Case Studies	9						
		1.3.1	Flat Neighborhood Networks	16						
		1.3.2	Switch-Less Designs	18						
		1.3.3	A 5 Node Ring	18						
		1.3.4	A n-Node Ring	19						
		1.3.5	A 8 Node Hypercube	19						
		1.3.6	A 64 Node Hypercube	20						
		1.3.7	Closing Comments on Design	21						
	1.4	Exercis	es	22						
2	Beo	wulf Sy	vstem Software Setup	23						
	2.1	Overvie	ew of Cluster Software	23						
	2.2	2.2 Network Setup Options								
	2.3	Setting	up a Cluster	25						
		2.3.1	Networking Setup	26						
		2.3.2	Setting up Remote Shell	28						
		2.3.3	Secure Shell: Alternative to Remote shell	31						

	2.3.4	Parallel Shell	32
	2.3.5	Time Synchronization	34
	2.3.6	Firewall Setup	36
	2.3.7	IP Masquerading	37
	2.3.8	Network File System	38
	2.3.9	Domain Name Service	39
	2.3.10	Enabling Mail from the Compute Nodes	39
2.4	Setting	g up Parallel Software	40
2.5	Install	ing the Cluster Automatically	42
	2.5.1	Introduction	42
	2.5.2	Installing the Cluster using YACI	42
2.6	Manag	ging a Cluster	43
	2.6.1	User Accounts etc	43
	2.6.2	Cluster Monitoring	43
	2.6.3	Workload Management	43
2.7	Integra	ated Solutions	44

1 Designing a Beowulf Cluster

The first step is the hardware selection. There are two issues involved in the hardware selection:

- Specifications of cluster nodes.
- Interconnection design.

1.1 Cluster Nodes

1.1.1 Hardware

- Buy generic PCs and shelves. May want keyboard switches for smaller configurations. For larger configurations, a better solution would be use the serial ports from each machine and connect to a terminal server.
- Build custom PCs from commodity off-the-shelf components. Requires a lot of work to assemble the cluster but we can fine-tune each node to what we want.
- Custom rack-mount nodes. More expensive but saves space. May complicate cooling issues due to closely packed components.

Some desirable characteristics:

- Dual-processor motherboards may be a good idea. Currently the price/performance curve falls off for quad or more processors per board. Having dual-processor boards may cause more load on the network cards in each node. Certain network cards have dual-network processors in them, making them better candidates for dual-processor motherboards.
- Spend as much as possible on memory.
- Don't need a fancy graphics card: either a minimal graphics card or none.
- No keyboards, CD-ROM, mouse, monitors needed for the compute nodes.

1.1.2 Benchmarking

An excellent collection of benchmarks is the Beowulf Performance Suite (available from paralogic.com). It consists of the following well-known public domain benchmarks.

• bonnie - hard drive performance

- stream memory performance
- netperf general network performance
- netpipe detailed network performance
- nas nas NASA parallel tests
- unixbench general Unix benchmarks
- lmbench micro Linux benchmarks

It is also worthwhile to download the latest version of netpipe benchmark. The latest version allow you to compare network performance with raw TCP, PVM, and MPICH, LAM/MPI among others.

For parallel benchmarks, NAS parallel are a reasonable test (especially if you will be running numerical computations on your cluster).

Use the above and other benchmarks to evaluate different architectures, motherboards, network cards before making a decision.

1.1.3 Node Software

Just Install Linux. For parallel libraries, install PVM and MPI both. For example, PVM and LAM/MPI come preinstalled with Fedora Core Linux (if you install everything or use custom install and select Scientific Computation packages).

Obviously, you can stick the CD in each machine to install Linux. However, there are some choices available to allow you to install the cluster over the network. This would be needed if the nodes do not have CD drives. We will discuss these options later.

1.2 Networking Hardware

1.2.1 What Type of Network to Use?

The following table compares various networking options for a Beowulf cluster. The price per node includes the price of the network interface cost and the amortized cost of network switches. Be aware that the network switch price can have a lot of variation. The prices shown for reasonable choices.

Network type	maximum bandwidth	latency	price/node
	(Mbits/second)	(microsecs $)$	dollars
Fast Ethernet	100	200	40
Gigabit Ethernet	1000	29 - 62	225
Myrinet	2000	6.8	1100
Myrinet	4000	6.8	1600
Dolphin Wulfkit	4500 - 10000	2	1200 - 2000
Infiniband	10000	8	1500 - 2000

Ethernet is the most cost-effective choice. Ethernet is a packet-based serial multi-drop network requiring no centralized control. All network access and arbitration control is performed by distributed mechanisms. The price of Gigabit Ethernet is dropping rapidly.

1.2.2 Network Topology Design

Some ranges of possibilities.

- Shared multi-drop passive cable, or
- Tree structure of hubs and switches, or
- Custom complicated switching technology, or
- One big switch.

Before we discuss network topology design, let us look at Ethernet in some more detail.

1.2.3 Ethernet Protocol

In an Ethernet network, machines use the Carrier Sense Multiple Access with Collision Detect (CSMA/CD) protocol for arbitration when multiple machines try to access the network at the same time. If packets collide, then the machines choose a random number from the interval (0, k) and try again. On subsequent collisions, the value k is doubled each time, making it a lot less likely that a collision would occur again. This is an example of an exponential backoff protocol.

1.2.4 Ethernet Packets

The Ethernet packet has the format shown below.



Note that the Maximum Transmission Unit (MTU) is 1500 bytes. Messages larger than that must be broken into smaller packets by higher layer network software. Higher end switches accept Jumbo packets (up to 9000 bytes), which can improve the performance significantly.

For many network drivers under Linux, the MTU can be set on the fly without reloading the driver!

Use the program ethereal to watch Ethernet packets on your network!

1.2.5 TCP/IP Addresses

The most prevalent protocol in networks is the Internet Protocol (IP). There are two higherlevel protocols that run on top of the IP protocol. These are TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).

IPv4 protocol has 32-bit addresses while the IPv6 protocol has 128-bit addresses. IP address range is divided into networks along an address bit boundary. The portion of the address that remains fixed within a network is called the **network address** and the remainder is the *host address*. Three IP ranges are reserved for private networks.

- 10.0.0.0 10.255.255.255
- 172.16.0.0 172.31.255.255
- 192.168.0.0 192.168.255.255

These addresses are permanently unassigned, not forwarded by Internet backbone routers and thus do not conflict with publicly addressable IP addresses. These make a good choice for a Beowulf cluster.

We will often use 192.168.0.0 - 192.168.0.255 in our examples. The address with all 0's in the host address, that is, 192.168.0.0, is the network address and cannot be assigned to any machine. The address with all 1's in the host address, that is, 192.168.0.255 is the network broadcast address.

1.2.6 Network Topology Options

Hubs and Switches.

- **Direct wire**. Two machines can be connected directly by a Ethernet cable (usually a Cat 5e cable) without needing a hub or a switch. With multiple NICs per machine, we can create networks but then we need to specify routing tables to allow packets to get through. The machines will end up doing double-duty as routers.
- Hubs and Repeaters/ All nodes are visible from all nodes and the CSMA/CD protocol is still used. A hub/repeater receives signals, cleans and amplifies, redistributes to all nodes.
- Switches. Accepts packets, interprets destination address fields and send packets down only the segment that has the destination node. Allows half the machines to communicate directly with the other half (subject to bandwidth constraints of the switch hardware). Multiple switches can be connected in a tree or sometimes other schemes. The root switch can become a bottleneck. The root switch can be a higher bandwidth switch.

Switches can be **managed** or **unmanaged**. Managed switches are more expensive but they also allow many useful configurations. Here are some examples.

- **Port trunking** (a.k.a Cisco EtherChannel). Allows up to 4 ports to be treated as one logical port. For example, this would allow a 4 Gbits/sec connection between two Gigabit switches.
- Linux Channel Bonding. Channel bonding means to bond together multiple NICs into one logical network connection. This requires the network switch to support some form of port trunking. Supported on Linux kernel version 2.4.12 or newer.
- Switch Meshing. Allows up to 24 ports between switches to be treated as a single logical port, creating a very high bandwidth connection. useful for creating custom complicated topologies.
- Stackable, High bandwidth Switches. Stackable switches with special high bandwidth interconnect in-between the switches. For examples, Cisco has 24-port Gigabit stackable switches with a 32 Gbits/sec interconnect. Up to 8 such switches can be stacked together. All the stacked switches can be controlled by one switch and managed as a single switch. If the controlling switch fails, the remaining switches hold an election and a new controlling switch is elected. Baystack also has stackable switches with a 40 Gbits/sec interconnect.

1.2.7 Network Interface Cards

The Ethernet card, also known as the Network Interface Controller (NIC), contains the Data Link Layer and the Physical Layer (the two lowest layers of networking). Each Ethernet card has a unique hardware address that is know as its MAC address (MAC stands for Media Access Controller). The MAC address is usually printed on the Ethernet card. The command **ifconfig** can be used to determine the MAC address from software.

NICs with PXE boot support are essential for larger Beowulf clusters. These greatly simplify installation of the cluster.

Another issue to consider s that having dual-processor boards may cause more load on the network cards in each node. Certain network cards have dual-network processors in them, making them better candidates for dual-processor motherboards. For example, the HP Broadcomm 5782 NIC has dual-RISC processors so it can handle a send/recv simultaneously. Check the website http://cs.boisestate.edu/ amit/research/benchmarks.html for a comparison.

1.3 Network Hardware Design: Case Studies

Design I

Here are some sample designs for a 64-node Beowulf cluster with Gigabit Ethernet network.



Price: ???? HP 9308gl (supports 232 GbE ports) \$100,000







HP5308gl 8–slot chassis \$1,919 4–port GbE module x 8 \$5,624 total/chassis \$7,563 Total: \$7563 x 4 = \$30,252



 3 x Cisco 3750G-24-T-S switch\$3,716 x 3
 3 x 1.6ft Cisco Stackwise cable Total: \$11,150







32000 MBits/sec link

Each Compute Node: dual 2.4 GHz Xeons, 1GB RAM, 80GB ATA100 drive Gigabit Switch: Cisco 24–port Gigabit stacking cluster switch

1.3.1 Flat Neighborhood Networks

A Flat Neighborhood Network uses multiple NICs per machine in such a way that we can get from any machine to any machine in one hop. It requires special routing to be setup at each node. However, it does not require the use of managed switches, which may save cost and get higher performance at the same time. The following FNNs were designed using a Genetic Algorithm (available on the website: http://aggregate.org/FNN).

Here is the layout for a 24-node FNN using 2 16-port Gigabit switches.

The FNN design found that requires the fewest switches uses 3 16-port switches to connect 24 PCs each using no more than 2 NICs/PC. The FNN is given below in the format switch-number: list-of-PCs-connected.

The following diagram shows the layout of the 24-node FNN.



A 24–node FNN

The FNN design found that requires the fewest switches uses 3 48-port switches to connect 64 PCs each using no more than 2 NICs/PC. The FNN is given below in the format switch-number: list-of-PCs-connected.

0:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22		
	23	24	25	26	27	28	29	30	31	32	33	34	35	42	43	44	45	46	47	48	49	50	51	52	53
1:	12	13	14	15	16	17	18	19	20	21	22	23	30	31	32	33	34	35	36	37	38	39	40		
	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63		
<u>.</u>	0	4	0	2	Λ	F	C	7	0	0	10	4 4	04	05	06	07	00	00	26	27	20	20	40		
2:	0	T	2	3	4	Э	ю	(ð	9	10	ΤT	24	25	20	21	28	29	30	31	30	39	40		
	41	54	55	56	57	58	59	60	61	62	63														

The FNN design found that requires the fewest switches uses 11 24-port switches to connect

64 PCs each using no more than 4 NICs/PC. The FNN is given below in the format switch-number: list-of-PCs-connected.

0: 9 10 11 36 37 38 39 40 41 54 55 56 57 58 59 1: 5 12 13 14 15 16 17 30 31 32 33 34 35 48 49 50 51 52 53 2: 5 18 19 20 21 22 23 42 43 44 45 46 47 60 61 62 63 3: 5 24 25 26 27 28 29 42 43 44 45 46 47 60 61 62 63 4: 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 5: 10 11 30 31 32 33 34 35 54 55 56 57 58 59 60 61 62 63 6: 9 10 11 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 7: 12 13 14 15 16 17 30 31 32 33 34 35 42 43 44 45 46 47 54 55 56 57 58 59 8: 12 13 14 15 16 17 36 37 38 39 40 41 48 49 50 51 52 53 60 61 62 63 9: 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 10: 18 19 20 21 22 23 24 25 26 27 28 29 48 49 50 51 52 53 54 55 56 57 58 59

The following diagram illustrates the setup, along with a cost analysis.



Flat Neighborhood Network Designs VII and VIII

Note that for our design case of 64 nodes, we were not able to achieve a very cost-effective FNN. That would, however, change with time as more unmanaged and commodity Gigabit switches with 24 port or more become available at lower costs than managed Gigabit switches. Note that the FNN designs will give lower latency (since all communications go through only one switch) and higher bandwidth as well.

The catch is that a fairly complicated routing needs to be set at each node. The designers of FNN have programs that can generate this automatically. Parallel libraries like PVM and MPI would also has to be modified to follow the specialized routing.

1.3.2 Switch-Less Designs

We can use the routing capabilities of Linux kernel and do away altogether with switches. All we need then are the network interface cards and some Ethernet cables! Let us see some examples.

1.3.3 A 5 Node Ring

To setup a ring, each machine has two NICs: eth0 and eth1. The private IP addresses assigned to the nodes are shown in the figure below.

<i>1.1</i> Node 0 0.1	1.2 Node 1 0.2	1.3 Node 2 0.3	1.4 Node 3 0.4	1.5 Node 4 0.5
hostname n0	<i>n1</i>	n2	<i>n3</i>	n4
eth0 192.168.0.1 –	192.168.0.2 -/	192.168.0.3 -	192.168.0.4 –	192.168.0.5 –
eth1 192.168.1.1	192.168.1.2	192.168.1.3	192.168.1.4	192.168.1.5

A 5 node ring

For the ring to work, we must set the routing properly. Let us consider Node n0. To route to node n1, we can simply make 192.168.0.1 be the gateway. Similarly to route to node n4, we can make 192.168.1.1 be the gateway. Things get more interesting for the remaining two nodes. For node n2, it is shorter to route via node n1, while for node n3, it is shorter to route via node n4. The following script, which can be run at system boot up time, shows the routing for node n0.

route add -host 192.168.0.2 gw 192.168.0.1 route add -host 192.168.0.5 gw 192.168.1.1 route add -host 192.168.0.3 gw 192.168.0.1 route add -host 192.168.0.4 gw 192.168.1.1

Similar scripts can be generated for each of the other four nodes.

1.3.4 A n-Node Ring

In order to create larger rings, we can write scripts or programs to automate the task of coming up with the route commands. We can also use the concept of net masks to simplify the routing commands. Suppose we have 15 machines in our ring, with IP addresses in the range 192.168.0.1–192.168.0.15. Then consider the network address 192.168.0.0 (which implicitly selects all 254 machines in the range 192.168.0.1–192.168.0.254, 192.168.0.255 is the broadcast address). If we use a netmask of 255.255.255.255.0, then we are selecting all addresses in that network. However, a netmask of 255.255.255.248 selects machines with addresses in the range 192.168.0.1–192.168.0.7. Similarly, a netmask of 255.255.255.248 selects machines with addresses machines in the range 192.168.0.8–192.168.0.254, which for us would be the furthest 8 machines. So now we can use the following script to set up the routing on node n0.

route add -net 192.168.0.0 netmask 255.255.255.248 gw 192.168.0.1 route add -net 192.168.0.0 netmask 255.255.255.7 gw 192.168.1.1

1.3.5 A 8 Node Hypercube

The following figure shows a 8-node hypercube with direct connections. Each node has three NICs: eth0, eth1. and eth2. The convention is that the *i*th NIC corresponds to the *i*th dimension. The addresses are shown below.

Node	eth0	eth1	eth2
n0	192.168.0.1	192.168.1.1	192.168.2.1
n1	192.168.0.2	192.168.1.2	192.168.2.2
n2	192.168.0.3	192.168.1.3	192.168.2.3
n3	192.168.0.4	192.168.1.4	192.168.2.4
n4	192.168.0.5	192.168.1.5	192.168.2.5
n5	192.168.0.6	192.168.1.6	192.168.2.6
n6	192.168.0.7	192.168.1.7	192.168.2.7
n7	192.168.0.8	192.168.1.8	192.168.2.8

The following diagram shows the hypercube.



Now for each node, we have three direct connections to its neighbors. For the remaining nodes, we will use a simple routing scheme. If the node is in the *i*th dimensions, route packets to it using the *i*th NIC. For example, the following shows the routing script for node n0.

```
route add -host 192.168.0.2 gw 192.168.0.1
route add -host 192.168.0.3 gw 192.168.1.1
route add -host 192.168.0.5 gw 192.168.2.1
route add -host 192.168.0.4 gw 192.168.1.1
route add -host 192.168.0.6 gw 192.168.2.1
route add -host 192.168.0.7 gw 192.168.2.1
route add -host 192.168.0.8 gw 192.168.2.1
```

Again we can write scripts/programs to generate the routing files automatically. We leave the reader to ponder if there is a better way to route than the one we have presented.

1.3.6 A 64 Node Hypercube

A switch-less 64-node hypercube would requires 6 network cards nodes per machine. That gives us a cost of $64 \times 50 \times 6 = 19,200$ for the network cards. A 64-node hypercube would require $(64 \times \log 64)/2 = 192$ network cables. Assuming \$5 per cable adds another \$960 for a total amount of \$20,160.

A layout for the 64-node hypercube is shown next.



1.3.7 Closing Comments on Design

Some observations about the switch less designs.

- Note that switch less design can improve bisection width significantly but at the expense of latency.
- They are not necessarily less expensive. For example, to build a 64-node hypercube, we would need 6 Gigabit cards per machine, which would cost \$300 per node, which is higher than the cost of our Design number VI presented earlier with Gigabit switches.
- Direct connections can be used to supplement switched designs. An excellent example is at the website http://www.cita.utoronto.ca/webpages/mckenzie/tech/networking/.
- An interesting website is an automated Cluster Design Website (http://aggregate.org/CDR/). Give it a try!

1.4 Exercises

- 1. Write the routing rules for the 24-node FNN using 3 16-port switches discussed earlier in this section.
- 2. Write a program or script to generate the routing commands for a *d*-dimensional hypercube built from PCs without switches.
- 3. Design network and routing for a 8×8 mesh using PCs and direct cables without any switches.
- 4. Generalize your solution to an $\sqrt{n} \times \sqrt{n}$ mesh.
- 5. Can you suggest an improvement to our Beowulf Cluster design V using the idea of direct wiring? The improvement should cost less than \$5,000 (based on prices used in this document)?
- 6. Can you suggest a simple way of automatically routing around nodes that are down?

2 Beowulf System Software Setup

2.1 Overview of Cluster Software

For effective use of a Beowulf cluster as a shared, production tool, the first five of the following list could be considered essential and the others very desirable.

- 1. automated cluster installation
- 2. remote hardware management (remote power on/off, monitoring CPU temperature etc.)
- 3. cluster management (monitoring, system administration etc.)
- 4. job scheduling
- 5. libraries/languages for parallel programming
- 6. tuning and analysis utilities
- 7. integrated distributions

Let us consider these topics in order. Remote hardware management has made substantial strides in recent years. Improved mother boards with sensors and corresponding monitoring software makes it possible to monitor CPU temperature, fan speed etc and be able to remotely power on and off. The area of cluster management has seen a lot of activity as well. Some examples of cluster management software include Ganglia[8], utilities from the scyld distribution[9], ClusterWorX[10], ClusterEdge[11] and MSC.Linux[12]. Graphical programs are also available to check on the state of the cluster either as part of these software or separately. Software for other system administration tasks is in various stages of development. Many of the above cluster management utilities also support automated cluster installation. Good examples are YACI (Yet Another Cluster Installer) [24], OSCAR (Open Source Cluster Application Resources) [26], LUI [20] and ROCKS [27].

With multiple users on a cluster, a sophisticated job scheduling software is required. Comprehensive solutions like Condor[13], LSF(Load Sharing Facility)/ClusterWare[14], PBS (Portable Batch System)[15], SLURM[21], among others, are available. However, they do have a learning curve associated with them.

In the area of parallel libraries/languages, there is a plethora of choices, which can be hard to choose from. There are data-parallel languages like HPF (High-Performance Fortran)[16], ZPL[17] and many others. However the problems of conversion/rewriting of existing Fortran, C, C++ codes into these languages as well as how to extract high performance deters many scientists. Parallelizing compilers are available for HPF. But the performance of the resulting code is, in general, not very high. As a result, libraries that augment existing sequential languages like Fortran, C, C++ are popular. Another choice is between distributedshared memory and distributed memory. For performance reasons, most scientists choose distributed memory. The two most widely used libraries in scientific programs are MPI[7] and PVM[6], which are based on the distributed memory model.

Less research has been done in the area of tuning and analysis for parallel programs. Some visualizations programs are available for PVM[6] and MPI[7]. Another project in this area is TAU[18]. There have also been recent efforts at providing integrated distributions with all software bundled together. Examples are scyld[9], OSCAR[19] and LUI[20].

In the rest of this section, we will focus on more detailed setups for clusters.

2.2 Network Setup Options

There are three possibilities.

- Stand alone cluster. A cluster with all private IP addresses. Requires no Internet connection but has very limited access.
- Universally accessible cluster. All machines have public IP addresses and can thus be accessed from anywhere. This can be a high maintenance security nightmare! It can also be hard and expensive to get so many public IP addresses.
- Guarded cluster. There are two scenarios here.
 - One machine has two NIC's: one with a public IP address and the other with a private IP address. All other machines have only private IP addresses. Thus only machine is vulnerable. IP Masquerading can be used to give the private nodes access to the Internet, if needed. This is the most common Beowulf configuration.
 - The cluster machines still have public addresses but firewall software limits access to all but one machine from a limited set of outside machines. One machine is still publicly accessible. This is more flexible but requires a careful firewall setup. This would be the case when a system administrator wants to configure already existing machines into a cluster.

When assigning addresses and hostnames, it helps to keep them simple and uniform. For example: ws01, ws02, ws03, ..., ws08. Or ws001, ws002, ..., ws255. Labeling each machine with its host name, IP address and MAC address can be handy. It is also possible to use dynamically assigned addresses and names for each node, using one machine as the DHCP (Dynamic Host Configuration Protocol) server. See the reference [3] for more details.

2.3 Setting up a Cluster

We will assume that Red Hat Linux 9.0, Fedora Core 1/2/3 and later releases are being used. The changes for other versions of Linux should be none or minor. We will use two running examples.

Example 1. Guarded Beowulf Cluster with public addresses. The machine pp00.boisestate.edu is the master node, while pp01.boisestate.edu,..., pp07.boisestate.edu are the compute nodes. The host names are publicly addressable Internet names and they map to public Internet addresses. This could also be the scenario where the machines already exist for other reasons and we want to band them together into a cluster.

Here are the IP addresses for these machines. This addresses are assigned by the system administrator for the DNS service of your organization.

132.178.208.230 pp00.boisestate.edu 132.178.208.231 pp01.boisestate.edu 132.178.208.232 pp02.boisestate.edu 132.178.208.233 pp03.boisestate.edu 132.178.208.234 pp04.boisestate.edu 132.178.208.235 pp05.boisestate.edu 132.178.208.236 pp06.boisestate.edu 132.178.208.237 pp07.boisestate.edu

Example 2. Guarded Beowulf with private addresses. The master machine is onyx.boisestate.edu. It has two network interfaces. The public interface has the name onyx, while the private interface has the name ws00 associated with it. There are 16 machines on the private network. Their names are ws01,...,ws16. Here is the /etc/hosts file on each of the machines. Note the use of private IP addresses in the range 192.168.0.xxx.

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1
                localhost.localdomain
                                         localhost
132.178.208.159 onyx.boisestate.edu
                                         onyx
                ws00
192.168.0.1
192.168.0.101
                ws01
192.168.0.102
                ws02
192.168.0.103
                ws03
192.168.0.104
                ws04
192.168.0.105
                ws05
192.168.0.106
                ws06
192.168.0.107
                ws07
192.168.0.108
                ws08
192.168.0.109
                ws09
```

192.168.0.110	ws10
192.168.0.111	ws11
192.168.0.112	ws12
192.168.0.113	ws13
192.168.0.114	ws14
192.168.0.115	ws15
192.168.0.116	ws16

2.3.1 Networking Setup

Example 1. Here the machines already exist with the specified network parameters, so we can skip this section.

Example 2. The main network setup file on the master node is the following.

/etc/sysconfig/network

Here is that file on the private Beowulf cluster example master node.

NETWORKING=yes HOSTNAME=onyx.boisestate.edu GATEWAY=132.178.208.1

The directory /etc/sysconfig/network-scripts/ contains specific setup files for the various network interfaces. On our example private Beowulf cluster with a public and private network interface, the relevant files in the directory /etc/sysconfig/network-scripts/ are ifcfg-eth0 and ifcfg-eth1.

Here is the *ifcfg-eth1* file for our example cluster.

DEVICE=eth1 BOOTPROTO=none BROADCAST=132.178.223.255 IPADDR=132.178.208.159 NETMASK=255.255.240.0 NETWORK=132.178.208.0 ONBOOT=yes USERCTL=no PEERDNS=no GATEWAY=132.178.208.1 TYPE=Ethernet

And here is the ifcfg-eth0 file for our example cluster.

DEVICE=eth0 BOOTPROTO=none BROADCAST=192.168.0.255 IPADDR=192.168.0.1 NETMASK=255.255.255.0 NETWORK=192.168.0.0 ONBOOT=yes USERCTL=no PEERDNS=no TYPE=Ethernet

After setting up these files, restart the network using the following command.

/sbin/service network restart

You can use the /sbin/ifconfig command to check the network setup. On our example cluster, here is the output of the ifconfig command.

```
[amit@onyx network-scripts]$ ifconfig
```

```
eth0
         Link encap:Ethernet HWaddr 00:E0:81:27:4D:27
          inet addr:192.168.0.1 Bcast:192.168.0.255 Mask:255.255.255.0
         UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
         RX packets:78936337 errors:0 dropped:0 overruns:0 frame:0
         TX packets:88868442 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
         RX bytes:3448909838 (3289.1 Mb) TX bytes:45533193 (43.4 Mb)
          Interrupt:48 Base address:0xd800 Memory:fe9e0000-fea00000
eth1
         Link encap:Ethernet HWaddr 00:E0:81:27:4D:26
          inet addr:132.178.208.159 Bcast:132.178.223.255 Mask:255.255.240.0
         UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
         RX packets:46147977 errors:0 dropped:0 overruns:26 frame:0
         TX packets:14629806 errors:0 dropped:0 overruns:7 carrier:0
         collisions:0 txqueuelen:100
         RX bytes:1941881538 (1851.9 Mb) TX bytes:850990172 (811.5 Mb)
          Interrupt:17 Base address:0xb000 Memory:fe7fd000-fe7fd038
10
         Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
         UP LOOPBACK RUNNING MTU:16436 Metric:1
         RX packets:10029390 errors:0 dropped:0 overruns:0 frame:0
         TX packets:10029390 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:2790547401 (2661.2 Mb) TX bytes:2790547401 (2661.2 Mb)
```

You can also use the graphical network configuration tool that comes with Red Hat Linux

9.0 and later releases.

The network setup files for the compute nodes are generated automatically by the cluster installation software, for example, using YACI (see Section 2.5.2).

2.3.2 Setting up Remote Shell

The **rsh** service is insecure but it is universally available and is useful when combined with appropriate restrictions for a universally accessible cluster. In a private Beowulf cluster, the **rsh** service makes more sense since all the traffic is on the private network and not visible from the public network.

Example 1. Guarded Beowulf Cluster with public addresses. The machine pp00.boisestate.edu is the master node, while pp01.boisestate.edu,..., pp07.boisestate.edu are the compute nodes. The host names are publicly addressable Internet names. We want to allow rsh access in the cluster but only allow secure shell access from outside the cluster.

Here are sample setup files for the xinetd daemon that controls access to services.

Access to rsh service is only from pp00.boisestate.edu. Here is the file /etc/xinetd.d/rsh for this setup. This file would be the same on all nodes.

```
#location: /etc/xinetd.d/rsh
service shell
{
        socket_type
                                 = stream
        wait
                                 = no
        user
                                 = root
        log_on_success
                                 += USERID
        log_on_failure
                                 += USERID
                                 = /usr/sbin/in.rshd
        server
                                 = UNLIMITED
        instances
        only_from
                                 = pp00.boisestate.edu
}
```

After editing this file, reload the **xinetd** configuration using the following command on each machine.

/sbin/service xinetd reload

The file /etc/hosts.equiv lists equivalent machines that can use rsh. Here is the /etc/hosts.equiv on each node.

pp00.boisestate.edu pp01.boisestate.edu pp02.boisestate.edu pp03.boisestate.edu pp04.boisestate.edu pp05.boisestate.edu pp06.boisestate.edu pp07.boisestate.edu

Next add a line containing **rsh** to **/etc/securetty** to allow root to login to the whole cluster without a password. Here is a sample file from all nodes.

vc/1 vc/2 vc/3 vc/4 vc/5 vc/6 vc/7 vc/8 vc/9 vc/10 vc/11 tty1 tty2 tty3 tty4 tty5 tty6 tty7 tty8 tty9 tty10 tty11 rsh

The last step is to create a .rhosts file for the root user on each node. This file \sim root/.rhosts contains the following line.

pp00.boisestate.edu

Now we are set. Test it out.

[amit@pp00 amit]:rsh pp00 date Mon Dec 1 15:46:06 MST 2003 [amit@pp00 amit]: for i in 1 2 3 4 5
> do
> rsh pp0\$i date
> done
Mon Dec 1 15:44:56 MST 2003
Mon Dec 1 15:45:00 MST 2003
Mon Dec 1 15:44:57 MST 2003
Mon Dec 1 15:44:25 MST 2003
Mon Dec 1 15:46:29 MST 2003

Example 2. Guarded Beowulf Cluster with private addresses. The machine ws00 is the master node, while ws01,..., ws16 are the compute nodes. The setup for enabling rsh is the same as above. The difference is that we use the private name for the master node instead of the public name. Here is the /etc/hosts.equiv file.

ws00 ws01 ws02 ws03 ws04 ws05 ws06 ws07 ws08 ws09 ws10 ws11 ws12 ws13 ws14 ws15 ws16

Here is the /etc/xinetd.d/rsh file.

```
service shell
{
    socket_type = stream
    wait = no
    user = root
    log_on_success += USERID
    log_on_failure += USERID
```

```
server = /usr/sbin/in.rshd
disable = no
only_from = ws00
```

The file $\sim root/.rhosts$ contains the following line on each node.

ws00

}

The file /etc/securetty is the same as in the previous example.

2.3.3 Secure Shell: Alternative to Remote shell

Secure shell (ssh) can be used as a replacement for remote shell (rsh). Below we will compare the two approaches for Beowulf clusters.

- The security of secure shell is based on the encryption of any passwords that are sent over the network. The remote shell does not use any encryption. Hence you should never type a password when using the remote shell. However that is not an issue with Beowulf clusters since access without passwords is desired for the cluster.
- The secure shell also encrypts the data being transmitted. So we take a performance hit as compared to the remote shell.
- The remote shell uses privileged ports (ports with numbers ; 1024). There are only so many of these ports, so running large number of remote shell commands simultaneously can run out of ports. However, in practice, this is not a significant issue. The secure shell is not limited by this restriction.

Example 2. For Example 2 (private Beowulf), using **remote shell** gives the best performance without losing on security. For Example 1 (Guarded Beowulf with public addresses), using secure shell may be a better solution. It is however more complex to setup, if we want it to be more secure than remote shell.

Example 1. Here we can setup automatic login with secure shell. The only way to do this and be more secure than the remote shell is on a per user level. Each user would need to run a secure shell agent. See the man page for **ssh-agent** on secure shell agent. Normally, the agent is automatically started when you are running X windows.

1. Look in your ~/.ssh directory on the master node. There should be two files, id_rsa and id_rsa.pub. If not, create them using ssh-keygen -t rsa.

- Check the file mode of ~/.ssh/authorized_keys on the remote machine. It must be 600 (not readable by anyone other than yourself). You can assure this with chmod 600 *l*.ssh/authorized_keys.
- 3. Append your local id_rsa.pub to the remote host's /.ssh/authorized_keys.

Now you should be able to **ssh** to the remote host without a password. To use secure shell as your remote shell, we will need to define a few environment variables in the system wide /etc/bashrc file.

export DSH_PATH /usr/bin/ssh export PVM_RSH /usr/bin/ssh export LAM_RSH /usr/bin/ssh

2.3.4 Parallel Shell

Here we will discuss pdsh, a simple, open-source parallel shell that is quite useful for system administration. It relies on either the remote shell or the secure shell to be properly configured on the cluster. The website for pdsh is http://www.llnl.gov/linux/pdsh/pdsh.html.

Example 1 and **Example 2**. The simplest way to setup pdsh is get the RPM file and install it on the master node.

rpm -ivh pdsh-1.7-6.i386.rpm

Set up a file called /usr/local/etc/machines that lists the hostnames of all the machines in the cluster. Now we are all set to use pdsh.

```
pdsh -a
pdsh> date
ws16: Mon Dec
              1 15:40:07 MST 2003
ws04: Mon Dec 1 15:40:07 MST 2003
ws14: Mon Dec 1 15:40:07 MST 2003
ws07: Mon Dec 1 15:40:07 MST 2003
ws01: Mon Dec 1 15:40:07 MST 2003
ws12: Mon Dec 1 15:40:07 MST 2003
ws08: Mon Dec 1 15:40:07 MST 2003
ws06: Mon Dec 1 15:40:07 MST 2003
ws09: Mon Dec 1 15:40:07 MST 2003
ws10: Mon Dec
              1 15:40:07 MST 2003
ws02: Mon Dec 1 15:40:07 MST 2003
ws03: Mon Dec 1 15:40:07 MST 2003
ws00: Mon Dec 1 15:40:07 MST 2003
ws15: Mon Dec 1 15:40:07 MST 2003
```

```
ws05: Mon Dec 1 15:40:07 MST 2003
ws13: Mon Dec 1 15:40:07 MST 2003
ws11: Mon Dec 1 15:40:07 MST 2003
pdsh> exit
[root@onyx sysconfig]#
```

The parallel shell allows the user to execute the same command across the cluster. You can also use it to run one command at a time and pipe the output to a filter.

```
pdsh -a cat /etc/resolv.conf | dshbak
  -----
ws06
_____
search boisestate.edu
nameserver 132.178.16.3
nameserver 132.178.208.127
_____
ws00
_____
search boisestate.edu
nameserver 132.178.16.3
nameserver 132.178.208.127
_____
ws09
_____
search boisestate.edu
nameserver 132.178.16.3
nameserver 132.178.208.127
_____
. . .
_____
ws08
_____
search boisestate.edu
nameserver 132.178.16.3
nameserver 132.178.208.127
_____
ws02
_____
search boisestate.edu
nameserver 132.178.16.3
nameserver 132.178.208.127
_____
ws16
```

33

search boisestate.edu
nameserver 132.178.16.3
nameserver 132.178.208.127
------[root@onyx sysconfig]#

To copy files to the cluster, use the pdcp command, which is part of the pdsh package. For example:

```
pdcp -x ws00 -a /etc/passwd /etc/passwd
```

executed on ws00 copies the password file from the master to all the nodes. Check the man page for pdcp and pdsh more on the options.

2.3.5 Time Synchronization

The Network Time Protocol (http://www.ntp.org) is an accepted standard for synchronizing time on a system with a remote time server. You will need to find either a network time server or get a card which receives synchronized time from satellite broadcasts. Either way, you have a machine whose time is accurate to within a few milliseconds.

On the cluster, we would synchronize the master with the remote time server. Then we would setup the compute nodes to synchronize their time with the master.

Example 1. The setup for a cluster with publicly addressable compute nodes is the same as for Example 2 below, except for using the appropriate addresses.

Example 2. In the following example, assume that we are setting up NTP on the private cluster with ws00 (192.168.0.1) as the master node.

On each compute node, the file /etc/ntp.conf contains the following lines in the appropriate sections.

restrict 127.0.0.1 server 192.168.0.1

The files /etc/ntp/ntpservers and /etc/ntp/step-tickers contain the following line.

192.168.0.1

On the master node ws00, we want to set the server to actual time server that we are contacting (132.178.208.127, in this case) and allow the compute nodes to get time from us. The /etc/ntp.conf file contains the following entries in appropriate sections.

Prohibit general access to this service.

restrict default ignore restrict 132.178.208.127 mask 255.255.255.255 nomodify notrap noquery # Permit all access over the loopback interface. This could # be tightened as well, but to do so would effect some of # the administrative functions. restrict 127.0.0.1 # -- CLIENT NETWORK ------# Permit systems on this network to synchronize with this

time service. Do not permit those systems to modify the # configuration of this service. Also, do not use those # systems as peers for synchronization. restrict 192.168.0.0 mask 255.255.255.0 notrust nomodify notrap

. . .

server 132.178.208.127

The file /etc/ntp/ntpservers and /etc/ntp/step-tickers contain the following line.

192.168.0.1

You may also add additional servers for the master (as a fallback measure).

You will also have punch a hole in the firewall for the NTP packets to get through. Add the following line to /etc/sysconfig/iptables and restart the firewall on the master node.

-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 132.178.208.127 --sport 123 -d 0/0 -j ACCEPT

Then restart the firewall as follows:

/sbin/service iptables restart

Now we can start the NTP service on the master.

/sbin/service ntpd start

Watch for messages in /var/log/messages to ensure that the time service is operational. At this point, you can also start up the time service on the compute nodes.

pdsh -a -x ws00 /sbin/service ntpd start

2.3.6 Firewall Setup

Example 1. For the Example 1 (cluster with public addresses), the simplest option would be to trust all the machines in the network. Below we show the lines we would add to /etc/sysconfig/iptables to get the desired effect.

Allow all TCP traffic between the nodes

```
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp -s 132.178.208.230/32 -d 0/0 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp -s 132.178.208.231/32 -d 0/0 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp -s 132.178.208.232/32 -d 0/0 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp -s 132.178.208.233/32 -d 0/0 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp -s 132.178.208.234/32 -d 0/0 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp -s 132.178.208.235/32 -d 0/0 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp -s 132.178.208.236/32 -d 0/0 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp -s 132.178.208.237/32 -d 0/0 --syn -j ACCEPT
# Now for the UDP traffic
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 132.178.208.230/32 -d 0/0 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 132.178.208.231/32 -d 0/0 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 132.178.208.232/32 -d 0/0 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 132.178.208.233/32 -d 0/0 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 132.178.208.234/32 -d 0/0 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 132.178.208.235/32 -d 0/0 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 132.178.208.236/32 -d 0/0 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 132.178.208.237/32 -d 0/0 --syn -j ACCEPT
```

Then restart the firewall on all nodes.

pdsh -a /sbin/service iptables restart

Example 2. Here we will discuss using iptables firewall for a private Beowulf cluster. Since the master is the only node accessible from a public network, the firewall is used only on the master node. The goal is that we allow only limited access to the master via the public network but allow full access from the private network. The limited access might include secure shell (via port 22), sendmail (via port 25), domain name service (via port 53), NTP time service (via port 123) and a few selected others.

In the example configuration file shown below, **eth1** is the network interface connected to the public network, while **eth0** is the network connected to the private network.

```
# Note: ifup-post will punch the current name servers through the
# firewall; such entries will *not* be listed here.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
```

```
:OUTPUT ACCEPT [0:0]
:RH-Lokkit-0-50-INPUT - [0:0]
-A INPUT -j RH-Lokkit-0-50-INPUT
-A FORWARD -j RH-Lokkit-0-50-INPUT
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp --dport 22 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp --dport 25 --syn -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth0
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 0/0 --sport 67:68 -d 0/0 --dport 67:68 -i eth1
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 132.178.208.127 -- sport 123 -j ACCEPT
-A RH-Lokkit-0-50-INPUT -i lo -j ACCEPT
-A RH-Lokkit-0-50-INPUT -i eth0 -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 132.178.16.3 --sport 53 -d 0/0 -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -s 132.178.208.127 --sport 53 -d 0/0 -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p tcp -m tcp --syn -j REJECT
-A RH-Lokkit-0-50-INPUT -p udp -m udp -j REJECT
COMMIT
```

Here the ports 67,68 are for remote booting that we will use for DHCP and cluster install. The machines 132.178.16.3 and 132.178.208.127 are providing Domain Name Service. The machine 132.178.208.127 is our NTP time server.

You can restart the firewall with the new configuration with the command:

/sbin/service iptables restart

2.3.7 IP Masquerading

There are two options for a Beowulf cluster with private network.

- The compute nodes only have network access to the master node. This is the easy setup. The cluster installer (YACI) discussed later does this by default.
- The compute nodes have access to the public network but entities in the public network have no access to the compute nodes. They only see one machine: the master node.

To have the second setup, we need to use IP Network Address Translation and IP Masquerading.

First of all, the kernel should have NAT support in it. The stock kernel in Red Hat Linux 9.0 and later systems have NAT capability by default. On the master, the file /etc/rc.d/rc.local should have the following entries, assuming ethO is the private interface and eth1 is the public interface.

#set up masquerading for the clients

```
modprobe iptable_nat
iptables -t nat -F
echo " FWD: Allow all connections OUT and only existing and related ones IN"
iptables -A FORWARD -i eth1 -o eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
#iptables -A FORWARD -j LOG
echo " Enabling SNAT (MASQUERADE) functionality on eth1"
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

```
#turn on IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Create a script, say /usr/local/adm/bin/restart_nat that contains the above commands. Make it writeable/executable only by the superuser.

chmod 744 /usr/local/adm/bin/restart_nat

Then run the script to enable NAT and masquerading without having to reboot the master node.

```
/usr/local/adm/bin/restart_nat
```

On the compute nodes, the file /etc/rc.d/rc.local should have the following line in it.

route add default gw 192.168.0.1

To enable the routing the command can be given manually without having to reboot the compute nodes.

pdsh -a -x ws00 route add default gw 192.168.0.1

Now the compute nodes have full access to the public network but they cannot be accessed from the public network.

2.3.8 Network File System

Example 1. Please read the discussion for Example 2 below. A Network File System considerably simplifies the cluster environment and should be setup even for a publicly addressable cluster, if possible.

Example 2. The simplest setup to use is to let users have the same home directory on all nodes. In addition, allowing /usr/local/ to be shared is useful.

The most common choice for a common file system is NFS (Network File System). There are faster network file systems in the offing but currently NFS is the established system that has kernel level support.

To start NFS, use the following command.

/sbin/service nfs start

On the master node, which is the NFS server as well, create the file /etc/exports, which lists the directories to be exported to the compute nodes. Here is the exports file from onyx (or ws00).

/tftpboot	192.168.0.0/24(rw,no_root_squash)
/home	192.168.0.0/24(rw,no_root_squash)
/usr/local	192.168.0.0/24(rw,no_root_squash)
/var/spool/mail	192.168.0.0/24(rw,no_root_squash)
[amit@onyx etc]\$	

Now to export the directories, use the command:

/usr/sbin/exportfs -va

On the client side, the following lines should be in the file /etc/fstab

```
ws00:/home /home nfs soft,rw,suid 0 0
ws00:/usr/local /usr/local nfs soft,rw,suid 0 0
ws00:/var/spool/mail /var/spool/mail nfs soft,rw,suid 0 0
```

Notice we use ws00 since the compute nodes are connected to the private network interface on onyx. This way, the shared directories are mounted automatically at bootup time.

2.3.9 Domain Name Service

Example 1. This step is does not apply to Example 1.

Example 2. Set aliases (canonical names) for the private name of the master node to be equal to the public name of master node in the Domain Name Service. This turns out to be handy for running some system management software. You will have to contact your system administrator to accomplish this step.

2.3.10 Enabling Mail from the Compute Nodes

Example 1. This step is does not apply to Example 1.

Example 2. This step is not necessary for mist clusters unless you want the capability of using mail from any of the compute nodes.

Sendmail and postfix are the two most popular mail transport agents for Linux. Rather than getting into their specifics, we will describe what needs to be done. To allow compute nodes to use mail, three steps are needed.

- In each compute node, set relay host to be the master node. This would be dependent upon whether you chose to use postfix or sendmail.
- NFS mount /var/spool/mail on each compute node.
- Set aliases (canonical names) for the compute nodes to be equal to the master node's public name in DNS. This would allow systems that check the validity of an Internet address before accepting email to be placated. For example, here is the result of doing a lookup for ws03 in our example cluster.

[amit@onyx amit]\$ host ws03
ws03.boisestate.edu is an alias for onyx.boisestate.edu.
onyx.boisestate.edu has address 132.178.208.159
[amit@onyx amit]\$

2.4 Setting up Parallel Software

PVM and LAM/MPI already come pre-installed with Red Hat Linux 9, Fedora Core 1/2 (if you choose to install everything or install Scientific Libraries package group).

The only tricky issue is if you have multiple network cards on the master node. You have to modify PVM and MPI to use the right network interface. In our example, we want the PVM and MPI applications to use ws00 instead of onyx, assuming that is our master node. For MPI, the solution is simple: set a global environment variable MPI_HOST=ws00;export MPI_HOST. This can be done in /etc/bashrc file so that all users see the value when they login.

For PVM, modify the invocation string lines in the script /usr/local/lib/pvmd. The original invocation string is:

exec \$PVM_ROOT/lib/\$PVM_ARCH/pvmd3\$sfx \$@

Modify it to (for each occurrence):

exec \$PVM_ROOT/lib/\$PVM_ARCH/pvmd3\$sfx -nws00 \$@

This needs to be done only on the master node.

It is also handy to setup other environment variables needed for PVM and MPI in the global /etc/bashrc file.

#setup for PVM3

```
PVM_ARCH=LINUXI386
export PVM_ARCH
PVM_ROOT=/usr/share/pvm3
export PVM_ROOT
MANPATH=$MANPATH:$PVM_ROOT/man
export MANPATH
PVM_DPATH=$PVM_ROOT/lib/pvmd
export PVM_DPATH
PATH=$PATH:$PVM_ROOT/lib:$HOME/pvm3/bin/$PVM_ARCH:
export PATH
XPVM_ROOT=/usr/share/pvm3/xpvm
export XPVM_ROOT
PVM_TMP=/tmp
export PVM_TMP
PVM_DEBUGGER=$PVM_ROOT/lib/debugger.ddd
export PVM_DEBUGGER
PVM_EXPORT=DISPLAY:PATH:PVM_DEBUGGER
export PVM_EXPORT
#for ddd debugger
export XEDITOR=gvim
# create PVM directories auto-magically for users
if ! [ -d ~/pvm3 ]
then
     mkdir ~/pvm3
     mkdir ~/pvm3/bin
     mkdir ~/pvm3/bin/$PVM_ARCH
fi
#-----
             _____
# setup for LAM/MPI
LAMRSH=rsh; export LAMRSH
#______
```

2.5 Installing the Cluster Automatically

2.5.1 Introduction

All of the above discussion on setting up clusters assumed that you had Linux installed on each cluster already. In Example 1, we had a set of machines that already existed with Linux on them, so installation was not an issue.

In Example 2, we did not specify how to get the systems up. Obviously we can install them manually one by one but that would be unacceptably slow and error prone process. A cluster installation software automates the process. Good examples are YACI (Yet Another Cluster Installer) [24], OSCAR (Open Source Cluster Application Resources) [26], LUI [20] and ROCKS [27].

2.5.2 Installing the Cluster using YACI



YACI (Yet Another Cluster Installer) is a flexible cluster installer that is open-source. See its webpage [24] for details about downloading and using it.

YACI builds a tarball image of a compute node on the master node. Each compute node boots up and sends a DHCP request. The DHCP server running on the master node has the MAC addresses of the compute node hard-coded so it can give them back a fixed address via DHCP. Once the compute node has a address, then it uses tftp (Trivial File Transfer Protocol) to transfer a PXE image (Pre-Execution Environment). The PXE image boots and then downloads via TFTP a Linux kernel. This kernel starts booting and then acts as a installation kernel. It creates partitions on the local disk, creates the file systems and then copies the image to the local hard disk via NFS (or using Multicasting, if available). Once the installation is finished, the compute node boots into the newly installed system.

Using YACI, we can do all of the steps discussed earlier in this documents automatically. For example, we were able to install a 61 node cluster from bar disk to fully working in 12 minutes using YACI.

YACI comes with excellent documentation. There are a few caveats in its use. Please contact amit@cs.boisestate.edu, if you have questions.

2.6 Managing a Cluster

2.6.1 User Accounts etc

Simple scripts are sufficient for pushing password and related files across the cluster. Network Information System (NIS) can also be used to keep the user information consistent although there is anecdotal evidence against the scalability of NIS.

The files to be kept consistent include /etc/passwd, /etc/shadow, /etc/group, /etc/bashrc and possibly others depending upon the installation.

2.6.2 Cluster Monitoring

Use LM_SENSORS to monitor cluster node hardware. There are several commercial packages that provide an integrated view of the state of the cluster. See Section 2.1 for more details.

We are developing a web-based cluster monitoring system, named **clusmon**. Check its web page at: http://tux.boisestate.edu/clusmon/chopshop.php.

2.6.3 Workload Management

A workload management software is essential for a larger Beowulf installation. The most popular software is the Portable Batch System (PBS). A open-source version is available for clusters of size up to 32 nodes and a commercial version is available for larger clusters. An improved open source version of PBS is available from the Ohio State Supercomputer Center (website for PBS: http://www.osc.edu/ pw/pbs/). Please consult the guide at http://cs.boisestate.edu/ amit/research/beowulf/pbs.pdf for nuances on installing and using PBS.

A highly scalable and completely open source solution is the SLURM software from Lawrence Livermore National Labs [21]. Other solution includes Condor [13], and LSF [14].

2.7 Integrated Solutions

Some interesting solutions for a integrated single system image for Beowulf clusters are bproc (http://bproc.sourceforge.net), scyld [9] and MOSIX (http;//www.mosix.org).

References

- [1] http://www.beowulf.org.
- [2] http://www.beowulf-underground.org.
- [3] Becker D., Savarese, D. F., Sterling, T., Dorband, J. E., U. A. Ranawake and C. V. Packer, "Beowulf: A Parallel Workstation for Scientific Computation, Proceedings of the 24th International Conference on Parallel Processing, I:11–14, 1995.
- [4] Sterling, T., Salmon, J., Becker, D. and D. Savarese, "How to Build a Beowulf," The MIT press, 1999.
- [5] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and V. Sunderam. "PVM: Parallel Virtual Machine: A User's Guide and Tutorial for Networked Parallel Computing," The MIT Press, 1994.
- [6] "PVM: Parallel Virtual Machine," main website: http://www.csm.ornal.gov/pvm.
- [7] "The Message Passing Interface (MPI) Standard," Grop W., and E. Lusk, website: http://www-unix.mcs.anl.gov/mpi.
- [8] Ganglia Project, http://ganglia.sourceforge.net.
- [9] Scyld Computing Corporation, Document #3720-2-1, http://www.scyld.com.
- [10] ClusterWorX by Linux NetworX, http://www.LinuxNetworX.com.
- [11] ClusterEdge by Scali High Performance Clustering, http://www.scali.com.
- [12] MSCLinux, http://www.mscLinux.com.
- [13] Condor, http://www.cs.wisc.edu/condor.
- [14] LSF/ClusterWare by Platform Computing, http://www.platform.com.
- [15] PBS (Portable Batch System), http://www.openpbs.org.
- [16] HPF, http://www.crpc.rice.edu/HPFF.
- [17] The ZPL Project, http://www.cs.washington.edu/research/projects/zpl/.
- [18] TAU: Tuning and Analysis Utilities, http://www.cs.uoregon.edu/research/paracomp/tau.
- [19] OSCAR: Open Source Cluster Application Resources, http://www.openclustergroup.org.
- [20] LUI: Linux Utility for Cluster Installation, http://oss.software.ibm.com/developerworks/projects/lui.

- [21] SLURM: Simple Linux Utility for Resource Management, http://www.llnl.gov/linux/slurm/slurm.html.
- [22] Distributed Production Control System, http://www.llnl.gov/icc/lc/dpcs/.
- [23] Globus, http://www.globus.org.
- [24] YACI: Yet Another Cluster Installer, http://www.llnl.gov/linux/yaci/yaci.html.
- [25] PDSH: Parallel Distributed Shell, http://www.llnl.gov/linux/pdsh/pdsh.html.
- [26] OSCAR: Open Source Cluster Application Resources, http://oscar.sourceforge.net.
- [27] Rocks Cluster Distribution, http://www.rocksclusters.org/Rocks/.