HOW TO BUILD A LINUX CLUSTER

Disclaimer: These notes DO NOT substitute the textbook for this class. The notes should be used IN CONJUNCTION with the textbook and the material presented in class. If a statement in these notes seems to be incorrect, report it to the instructor so that it be fixed immediately. These notes are only distributed to the students taking this class with A. Gerbessiotis in Fall 2004; distribution outside this group of students is NOT allowed.
Cautionary introduction
Building a Linux cluster

A note of caution. There is an insecure element in the building of the cluster described in these notes. This is the use of the r commands. These are rcp, rexec, rsh, rlogin. They are instrumental in distributing a copy of an executable program to the nodes of a cluster. When you do an mpirun, bsprun, ccp to execute one of the homework problems you implicitly use one of these commands.

In BSPlib these commands can not be avoided. Depending on the LAM-MPI installation, one can substitute ssh commands for the r commands. So one could use scp, ssh instead of rcp, rsh. However, there will be an annoyance if one does so: a password prompt that needs to be answered as many times as the machines one uses for a specific parallel task. One can only imagine how convenient this might be if one runs a parallel program on 16 machines. We will show at the very end how one can make ssh equivalent to rsh, i.e. no password is required for login.

Some terminology first. From now on, a node is a PC box; a node may have one or more processors. In our cluster of pcc16, pcc15, pcc14, pcc13, each node has two CPUs. The following assumptions are made in the remainder of this discussion.

- You know how to install a certain linux distribution. The instructions below work for Red Hat linux version 7.1 and up (includes, 7.1, 7.2, 8.0 and 9.0). They would probably well equally well under the Fedora distribution as well. Not many changes are required for other distributions.

- You have already installed linux, say Red Hat 9.0 on a machine.

- You know networking, at least the basics, even if you are not very familiar with protocols etc. For example, IP address, mask, DNS, gateway are familiar words. Certain networking choices in some configuration files below will not be explained. Take a networking course to learn more or what they actually mean! Or even better, read the manual pages available on-line!

- You have root access to a linux node.

The actions in the following sections must be repeated for all nodes. Of course, one can write a set of scripts and these actions take place automatically. On my web-page you can find such a set of scripts that do what i describe below and more!

(c) Copyright Alex. Gerbessiotis. All rights reserved. (CIS 786 NOTES Fall 2004)
Step 1: Networking essentials (hosts,host.conf,resolv.conf)
Building a Linux cluster

Go to directory /etc as root and start working.
The /etc/hosts file should look like

127.0.0.1 localhost.localdomain localhost
128.235.32.113 pcc13.njit.edu pcc13
128.235.32.114 pcc14.njit.edu pcc14
128.235.32.115 pcc15.njit.edu pcc15
128.235.32.116 pcc16.njit.edu pcc16

We put all information in the hosts file to eliminate the need of defining a gateway if we choose to do so for extra security. Make sure that host.conf is set up properly. In brief, the first line says, that for routing purposes, use the hosts file before you use anything else to resolve network addresses/names. Note that in host.conf there is no ending s in host as opposed to /etc/hosts.

order hosts,bind
multi on
nospoof on

The resolv.conf file determines that if one uses pcc17 he/she really means pcc17.njit.edu. For pcc16 the hosts files resolves this as the pcc16 and pcc16.njit.edu entries there are considered equivalent for IP address 128.235.32.116, so one could use any one of them interchangeably. If an address cannot be resolved by the hosts file then the DNS nameserver is used.

search njit.edu
nameserver 128.235.251.10
Step 2: Securing things a little
Building a Linux cluster

The files hosts.allow and hosts.deny indicate what connections are allowed or not and from which machines. A hosts.allow file looks like the one below. The word ALL is a catchall that allows all connections from the machine listed after the :

```
#
# hosts.allow This file describes the names of the hosts which are
# allowed to use the local INET services, as decided
# by the '/usr/sbin/tcpd' server.
#
ALL:128.238.32.113
ALL:128.238.32.114
ALL:128.238.32.115
ALL:128.238.32.116
sshd:above.njit.edu,probe.njit.edu,pcc69.njit.edu,*.njit.edu
sshd_fwd-X11:*.njit.edu
```

A hosts.deny file looks like.

```
# hosts.deny This file describes the names of the hosts which are
# allowed to use the local INET services, as decided
# by the '/usr/sbin/tcpd' server.
#
# The portmap line is redundant, but it is left to remind you that
# the new secure portmap uses hosts.deny and hosts.allow. In particular
# you should know that NFS uses portmap!
#ALL:ALL=ALL,PARANOID
ALL:ALL
```

The way things work are as follows. Connections are allowed from the hosts.allow as long as they are not forbidden in hosts.deny. Our hosts.deny file denies any connection from any machine by default. The only ones allowed are those of the hosts.allow file. All connections from pcc13, pcc14, pcc15, pcc16 are allowed. The only other connection allowed are secure shell connections (the name of the secure shell daemon named sshd is defined on the left of a :) from all NJIT specific addresses. In the example above, the catch all *.njit.edu makes all three previous entries redundant. The next line allows X Windows forwarding to remote NJIT-based machines.
SECURITY ALERT! THE ACTIONS BELOW ARE A SECURITY RISK!

The file `hosts.equiv` defines what systems are considered equivalent in the cluster. For equivalent systems authentication (a password prompt at any login attempt) is not required.

```plaintext
pcc13.njit.edu
pcc14.njit.edu
pcc15.njit.edu
pcc16.njit.edu
```

The lines above indicate that for our cluster of the four machines no password will be required if a user logsins from another node of the cluster.

The `r` commands by default are disabled. Even if you define `hosts.equiv` above, you will still be prompted for a password, unless you complete the steps below.

The daemon that provides Internet services in a linux system is `xinetd`. Its configuration file is `xinetd.conf` which looks like

```plaintext
# # Simple configuration file for xinetd
#
# Some defaults, and include /etc/xinetd.d/

defaults {
    instances = 60
    log_type = SYSLOG authpriv
    log_on_success = HOST PID
    log_on_failure = HOST
    cps = 25 30
}

includedir /etc/xinetd.d
```

(c) Copyright Alex. Gerbessiotis. All rights reserved. (CIS 786 NOTES Fall 2004)
Step 4: Allowing for the r commands
Building a Linux cluster

The important files that `xinetd` uses for configuration are in directory `/etc/xinetd.d`. Make sure that there are at least three file entries for `rexec`, `rsh`, `rlogin` named after the commands and the contents of these entries are as follows.

```
# default: off
# description: Rexecd is the server for the rexec(3) routine. The server \
# provides remote execution facilities with authentication based \
# on user names and passwords.
service exec
{
    socket_type = stream
    wait = no
    user = root
    log_on_success += USERID
    log_on_failure += USERID
    server = /usr/sbin/in.rexecd
    disable = yes
}

description: rlogind is the server for the rlogin(1) program. The server \
# provides a remote login facility with authentication based on \ 
# privileged port numbers from trusted hosts.
service login
{
    disable = no
    socket_type = stream
    wait = no
    user = root
    log_on_success += USERID
    log_on_failure += USERID
    server = /usr/sbin/in.rlogind
}
```

(c) Copyright Alex. Gerbessiotis. All rights reserved. (CIS 786 NOTES Fall 2004)
Step 4: continued
Building a Linux cluster

In LINUX after you create these files you don’t need to reboot (although it is OK if you do so). You can just restart the xinetd services. Go to /etc/rc.d/init.d and do a

```
% ./xinetd restart
```

which is equivalent to

```
% ./xinetd stop
% ./xinetd start
```

The network script can be restarted if you also did the changes of the early steps above without rebooting or restarting already. Note however, that the correct sequence of stopping/starting network and internet services is as follows: stop first the Internet services, then the network ones, then restart the network services, and finally the Internet services.
Step 5: Checking networking
Building a Linux cluster

Go to `/etc/sysconfig` and check the `network` file. It could look like

```
NETWORKING=yes
HOSTNAME=pcc16.njit.edu
GATEWAY=aaa.bbb.ccc.ddd
```

The GATEWAY entry might not be defined. If it is replace, `aaa.bbb.ccc.ddd` appropriately after you consult with the person responsible for security/networking at your Organization.

Assuming that you only have one network card, the `/etc/sysconfig/network-scripts/ifcfg-eth0` file could look like

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=static
IPADDR=128.235.32.116
NETMASK=255.255.252.0
GATEWAY=aaa.bbb.ccc.ddd
```

If you have set-up a firewall, make sure that it allows services such as ssh, and certain ports used by the r-commands. These ports are described in `/etc/services`. An `iptables` might look like as follows (next page).
Building a Linux cluster

Step 5: continued

Go to `/etc/sysconfig` and check the network file.

```
# Firewall configuration written by lokkit
# Manual customization of this file is not recommended.
# Note: ifup-post will punch the current nameservers 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 --sport 22 --dport 22 -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p tcp --sport 2049 --dport 2049 -j ACCEPT
-A RH-Lokkit-0-50-INPUT -p udp --dport 6000:6009 -j ACCEPT
COMMIT
```

Or

```
# Firewall configuration written by lokkit
# Manual customization of this file is not recommended.
# Note: ifup-post will punch the current nameservers 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 --sport 22 --dport 22 -j ACCEPT
-A RH-Lokkit-0-50-INPUT -i lo -j ACCEPT
COMMIT
```

The latter table is much more forgiving than the former.
Step 6: Ready!
Building a Linux cluster

The cluster is (or should be ready). You know how to fully install BSPlib. If you need to install LAM MPI (every Red Hat distribution preinstalls it), grab the latest copy from http://www.lam-mpi.org and install it with rpm.

```
% rpm --install lam-7.1.1-2.i586.rpm
```

However, this install process does not work with the ssh option defined on the next page. If you are security cautious and don’t like the r-commands you can avoid them altogether. For lam to work however, you need to recompile the source on every node with the ssh option on. Check the installation manual for details. BSPlib DOES NOT SUPPORT THE SSH OPTION.
Step 7: Ssh logins without passwords
Building a Linux cluster

The instructions below, assume that both machine pccL, the local node, and pccR, the remote node, have SSH2 installed. The instructions for SSH1 are different. We also assume that both machines use OpenSSH (the default Red Hat linux version).

Generate a public/private DSA key pair on pccL.

```
ppcL % ssh-keygen -t dsa -f ~/.ssh/id_dsa
```

When you are asked for a passphrase, leave it empty. Now send the public key to pccR. The example below assumes that your user name is user0.

```
ppcL % cd .ssh
ppcL % scp id_dsa.pub user0@pccR:~/.ssh
```

Please note the difference in file name above. They are not typos. The filename id_dsa stores the private key that it is being generated. The file id_dsa.pub holds the public key. You are allowed to copy the PUBLIC KEY BUT NOT THE PRIVATE KEY!

Then, log in on pccR with supplying a password (for the last time) and add the public key to the list of authorized keys.

```
ppcL% ssh user0@pccR

pccR% cd .ssh
pccR% cat id_dsa.pub >> authorized_keys2
pccR% chmod 640 authorized_keys2
pccR% rm -f id_dsa.pub
```

Note that the filename is authorized_keys2, not authorized_keys. That’s it; you’re ready to ssh from pccL to pccR without having to enter a password.