How to Set Up a Linux Cluster and Get an MPI Program Running

Gregor Pipan, Marjan Šterk, Roman Trobec, Jaka Močnik
Jožef Stefan Institute, Ljubljana, Slovenia
E-mail: marjan.sterk@ijs.si

March 11, 2002

Abstract

This document is intended for internal use. It describes one possible way to build, set up and use Linux clusters. Every procedure in the document works for us, but is mostly not the only option and may not be optimal either.

Contents

1 Preparing Hardware

This section will tell what hardware is needed and suggest ways of interconnecting the cluster. After reading it one can proceed with buying the computers.

1.1 Individual Machines

Each machine needs the obvious: power supply, motherboard, CPU, and RAM. It is handy to have video card in each machine for testing and troubleshooting. Separate monitors and keyboards take up too much space so one of each is used, requiring the administrator to manually connect the cables when a machine that needs attention.

Warning: when a keyboard is needed, it should be connected at boot time. If a computer is booted without it, the keyboard only work after rebooting. This may not apply to all motherboards.

Although having diskless machines and booting from network is possible, this document supposes that each machine has enough disk space for the operating system, MPI libraries and some to spare. The smallest available hard drive today (Feburary 2002) is about 10 GB, which is plenty. If data-intensive applications will be run, one of the machines should have much more space, making it a *master*.

Floppy and CD-ROM drives are only needed when installing the operating system and can thus be swapped between the machines like the keyboard.

1.2 Networking Hardware

There are three basic ways to interconnect the cluster:

1. Simply adding every machine to the existing local network. This will allow them to be used as workstations (provided they have monitors, keyboards and mice), but the network bandwith available to the cluster will be smaller and will change with time, making it harder to analyze the behaviour of parallel programs. For this approach, one network interface and one slot on the switch is needed per machine.

2. Separate switch for the cluster. This requires the same amount of harware, plus a connection between the cluster network and the existing local network. Full bandwidth will be available to the cluster at any time, but still shared among all the machines in the cluster.

3. Putting more than one network interface into each machine and connecting pairs without switches or hubs. This allows any topology – our test system consists of 9 nodes connected into a 3×3 torus, making for 4 connections per node. Any pair of neighbours can communicate without slowing down communication between other pairs of neighbours. An extra connection is needed between one node and the existing local network – this node becomes a *master*. Number of network interfaces required depends on the chosen topology, but the practical limit with most available motherboards is 4 to 6.

The rest of this document applies to the third option.

2 Installing Linux

This section will bring one to the point where every machine will be up and running, but not yet connected to others.

INCOMPLETE

3 Setting Up Local Network

After this section, it should be possible to *rsh, telnet* etc. between computers. Also, all network traffic between the nodes of the cluster will be routed along the shortest path.

3.1 Organization of the Cluster

Our cluster is effectively a private network interconnected in a special way that must be described in the configuration files. IP numbers and names should be made up, for example 10.0.0.1 through 10.0.0.n with names node-1 through node-n. Each node will need to know all other nodes' addresses and first points of routes to all other nodes.

The master node is also connected to the outside world. Thus, it will have another IP number and name by which it will be visible. This is crucial as the cluster will typically be used through telnet/rsh/ssh and not by logging in locally.

3.2 Configuration Files

INCOMPLETE

3.3 Sharing Disks with NFS

The easiest way to run programs on a cluster is to have the large master node's disk shared between all the nodes and store all the data there. The local disks are then only used for booting, storing configuration files and for swap space. For example, `~user/shome` (which stands for "shared home") on each node can be the link to `node-1:~user`.

INCOMPLETE

3.4 Configuring rsh

Mpich uses rsh to run programs on other nodes, so rsh should be enabled and configured so that it doesn't ask for password. On the system level, the file `/etc/hosts` should contain IP numbers and names for all the nodes, e.g.

```
10.0.0.1 master-node
10.0.0.1 node-2
10.0.0.1 node-3
etc.
# then the stuff already there
```

The owner of this file should be root, access permission should be as set with `chmod 711 hosts`.

The other configuration file is `.rhosts` on each user's home. The users should be owners of their `.rhosts` files, permissions as set with `chmod 711 .rhosts`, and the contents:

```
master-node username
node-2 username
node-3 username
etc.
```

All of the mentioned files should be copied to all the nodes. To test rsh, login to node A and type `rsh nodeB ls`. This should list the contetns of your home directory of node B, for any combination of A and B.

3.5 Configuring ssh

Alternatively, you can use ssh instead of rsh. To set up ssh so that it won't require entering passwords, type:

```
ssh-keygen -t dsa (press enter to all questions)
cp ~/.ssh/id_dsa.pub ~/.ssh/authorized_keys2
chmod go-rwx ~/.ssh/authorized_keys2
```

Unless the home directories are shared among the machines, the file `~/.ssh/authorized_keys2` has to be copied to all machines.

4 Installing and Testing MPI

This section walks one through installation of *mpich* and running the included test programs. Subsection 4.1 describes default installation on Debian systems (other Linux distributions may have similar utilities). Subsection 4.2 describes installation for other cases.

4.1 Automatic Installation – Debian Specific

Login to the master node as root. Run the command
`apt-get install mpich`
This will install the mpich libraries, executables etc. in `/usr/lib/mpich`, while the MPI programming examples will be in `/usr/share/doc/mpich/examples/code`. It also creates symbolic links to the executables in `/usr/bin`.

As will be explained later, there are two ways of running MPI programs. One of them requires you to create a system configuration file that all users will use. Input the names of all the machines in the cluster into a text file, starting with the master node, e.g.

```
master-node
node-2
node-3
```

for a 3-node cluster and save it to `/usr/lib/mpich/util/machines/machines.LINUX`.

4.2 Manual Installation

In most cases, the following steps should work. Detailed installation instructions are located in `doc/mpichman-*.pdf` after you have untarred the archive.

1. Login as a normal user.

2. Download the "Unix - all flavours" version of mpich from `www-unix.mcs.anl.gov/mpi/mpich/`.

3. Unzip/untar the arhive to a subdirectory on your. installation instructions are located in `doc/mpichman-*.pdf`.

4. If mpich should be available to all the users, decide on the installation directory, e.g. `/usr/local/mpich`. Otherwise, installation directory is meaningless.

5. Run
   `./configure --prefix=INSTALL_DIRECTORY`

6. Compile with `make`.

7. If mpich should be available to all the users, login as root and install it with `make install`. This will create some subdirectiores in the installation directory and copy files there.

8. Add the `bin` directory of the mpich install (in the `INSTALL_DIRECTORY` if you `madee install` it, in your home otherwise to your PATH by adding something like
   `export PATH=$PATH:MPICH_DIR/bin`
   to the file `.bash_profile` in your home.

9. Edit the `machines.LINUX`[1] file as described in subsection 4.1.

4.3 Installation on Other Nodes

MPI now works on the master node, which usually suffices. If any other nodes will be used for compiling and spawning MPI programs, MPI should be installed on them as well. If you are using Debian (or other Linux distribution with install utility/mpich package available) and have the cluster organized so that every node can see internet, the process from subsection 4.1 can be repeated. Otherwise, manual installation is needed.

4.4 Configuring mpich to use ssh instead of rsh

Run the configure script, which should be somewhere in the mpich directory:
`configure -rsh=ssh`

4.5 Compiling and Running a Test Program

4.5.1 Compiling

There is a test program called `cpi` supplied with mpich. It calculates the number π, dividing the work among processors if more than one is available. To compile it, copy the source file `/usr/share/doc/mpich/examples/code/cpi.c`[2] to a directory that you have write permissions for. Compile it with the command
`mpicc -c cpi.c`
and then link with
`mpicc -o cpi cpi.o`
You should get an executable `cpi`, which you can run and check that it outputs an approximation of π.

4.5.2 Testing Parallel Execution

To test parallel execution, type `mpirun -np 2 cpi`
which stands for: run the `cpi` program on 2 processors. This will run it in parallel on the first two machines in the `machines.LINUX` file. Both processors will report themselves, outputting
```
Process 0 on master-node
Process 1 on node-2
```
After that, it will output the calculated approximation of π.

If it doesn't work and either locks up or writes "Permission denied", try `rsh node-2 ls`. This should list the contents of your home directory on node-2. If it doesn't work or asks for password, return to section 3.4.

4.5.3 Testing the Speedup

To test the speedup, it is necessary to increase the amount of calculation. Open the program source and find the commented-out part that lets the user input the number of intervals. Enable this part and comment out the next line, which hard-codes the number to 100. Recompile and run first on one

[1] For non-Linux Unices, the filename may be different, e.g. machines.solaris.
[2] This location applies to the simple Debian installation – otherwise, it can be located elsewhere.

processor, then on 2 or more with the number of intervals about 10^8. Execution on one processor should take about n times longer than on n processors.

If you study the output carefully, you will find out that the result isn't exactly the same. This happens because the order of operations is different for different number of processors.

5 Compiling and Running Your MPI Programs

This section describes the parameters and options needed to compile an MPI program using *gcc*. It also describes ways to run MPI programs with *mpich*.

5.1 Compiling Using `mpicc`

Compiling with `mpicc` and its variants is very simple. It requires correctly installed mpich and is done in two steps, as described in section 4.5.1. Compiling in one step with just
`mpicc -o cpi cpi.c`
works on some systems, but not all.

`mpicc` accepts the same parameters as `gcc` – in fact, it calls `gcc` with additional parameters to link the mpich libraries etc. For C++ programs, use `mpiCC`.

5.2 Compiling Without Using `mpicc`

On a system with miss-installed mpich, it is possible to compile programs by supplying the additional parameters manually. The `cpi` example can be compiled with
`gcc -I/mpich_include_directory -L/mpich_lib_directory -o cpi cpi.c -lmpich`
substituting the two directories as needed.

5.3 Running Using `mpirun`

Running this way requires:

- correctly installed mpich on the master node (other nodes do not need mpich as the libraries are statically linked with the program),
- correct `machines.architecture` configuration file(s),
- the program should be in the same file and directory on all the machines, which is particularly annoying.

The command to run the program is
`mpirun -np NUM_P progName PARAMS`
where `NUM_P` is the number of machines to run on and `PARAMS` are additional parameters to pass to the program. This runs the processes on the computers listed in the `machines.ARCH` configuration file, where ARCH is the name of the architecture of master node (LINUX in our case).

You can choose the architecture manually and even mix them. Suppose you would like to run a program on 5 Linux boxes, two Suns and one HP. There should be three configuration files: `machines.LINUX`, `machines.solaris` and `machines.hpux`, with the computers of each architecture listed in the right file. The command to run the program is:
`mpirun -arch LINUX -np 5 -arch solaris -np 2 -arcp hpux -np 1 progName PARAMS`
As this can get complicated, there is the `-t` (test) option to `mpirun`, which does not run any processes but only prints what would be run.

The architectures are merely an user-interface concept – they don't have to correspond to actual computer architectures. It is possible to have 3 "architecture" files: `machines.mycluster` (with computers in the cluster listed), `machines.allcomputers` (with all computers within the organisation) and `machines.localonly` (with just `localhost`). The option `-arch` will select which to use – `-arch localonly` will run all processes on just one computer.

5.4 Running Using Per-Program Configuration Files

Running this way requires a configuration file that tells which machines to run. It also tells the directory and filename of the program for each machine. Note that filename and directory need not be the same for all machines. Also, mpich need not be installed on any machine at all – it is only needed for compilation.

The `cpi` program is started with the command

```
./cpi -p4pg CONF_FILE PARAMS
```

where `CONF_FILE` is the configuration file of the following format:

```
master-node 0
node-2 1 /progs-node2/cpi
node-3 1 /progs-node3/cpi
```

The first line tells mpich to run no programs on the first node as the progam is already being run (started by user with the `./cpi` command). In the remaining lines, the first field is the name of the machine, followed by the number of processes to run on this machine and the directory and filename of the program. Currently, mpich only supports 1 process per machine. The program can have different names on different machines and need not even be the same program (of course, the latter can lead to many problems unless the programmer is an advanced MPI-er).

5.5 Taking into Account Network Topology

INCOMPLETE

6 Debugging and Profiling MPI Programs

This section gives brief descriptions of some debuggers and profilers.

INCOMPLETE

7 Using the Heart Application

This section is IJS-E6 specific and tells how to use the heat transfer simulation from preparing and modifying the model to running the simulation and extracting results.

INCOMPLETE

References