

Parallel Architectures and Systems MPI Information at UTD, 2007

Dr. Edwin Sha

If you have any question about MPI, first read <http://www.lam-mpi.org/tutorials/>. Especially go through the tutorial in <http://webct.ncsa.uiuc.edu:8900/public/MPI/>. Everyone should take this free tutorial course.

We have no problem in running LAM MPI in our netxx machines. If you have any problem, please ask TA. If you have tried all the possible means but fail, then go to the last section to use Open MPI on our Linux machines.

1 Setting up the Environment for LAM MPI

We will be using the LAM public domain implementation of MPI. It is most commonly used on clusters of workstations, such as the SUN machines that we have here in the department.

The machines we are using are departmental SUN workstations from **net01** to **net30**. You should “ssh -l username” to remotely login to one of these machines and run your program there. Only these machines can use *sockets* functions that are required in MPI library. Some may not be working. Use those working ones.

In your \$HOME directory, create a file called *.rhosts* that contains the following lines. You should add more netxx machines.

```
net01.utdallas.edu
net02.utdallas.edu
net03.utdallas.edu
net04.utdallas.edu
net05.utdallas.edu
net06.utdallas.edu
net07.utdallas.edu
net08.utdallas.edu
net09.utdallas.edu
net10.utdallas.edu
```

Do you know what is the purpose of this *.rhosts* file?

<http://www.lam-mpi.org/tutorials/>

contains a tutorial about how to setup, compile, and run LAM MPI programs. This handout gives you some of the information, but it is highly recommended that you go examine this tutorial.

Next, add in your *.bashrc* file the following:

```
LAMRSH="ssh -q"
export LAMRSH
```

Make sure they are above the following line of *.bashrc*:

```
if [ -z "$PS1" ] ; then
```

You also need to create an SSH identity to run LAM without typing a password or a passphrase too many times. Run the following commands:

```
$ ssh-keygen -t dsa
Enter file in which to save the key ($HOME/.ssh/id_dsa): enter to use default.
Enter passphrase (empty for no passphrase): enter for no passphrase.
Enter same passphrase again: enter again.
```

```
$ cp .ssh/id_dsa.pub .ssh/authorized_keys
$ eval `ssh-agent`
$ ssh-add
```

(Note that “eval” and “ssh-add” might not work now in 2007 for our netxx machines. But it is okay.)

Now, the first time you run `lamboot`, you’ll need to save the RSA key fingerprints of all the hosts specified in `hostfile`. At every `Are you sure you want to continue connecting (yes/no)?`, enter `yes`.

2 Writing Your First Program

You will also find a sample MPI program, `mpi.c`, in the directory (`/net/core/export/home/cs/001/e/edsha/parallel/mpi_all`). It demonstrates simple MPI message passing features. Please feel free to copy all the files there to your local directory. You can use the following UNIX command to copy all the files into your “mpi” local directory.

```
cp -rf ~edsha/parallel/mpi_all mpi
```

2.1 Calculating PI Using Numerical Integration

This program approximate the value of π by numerical integration¹ on

$$\int_0^1 \frac{1}{1+x^2} dx = \frac{\pi}{4}$$

```
#include <math.h>
#include "mpi.h"

double f(double a)
{
    return (4.0 / (1.0 + a*a));
}
```

¹Gropp, W., Lusk, E., and Skjellum, A. *Using MPI*. (Cambridge, MA: MIT Press, 1995).

```

int main(int argc, char *argv[])
{
    int done = 0, n, myrank, numprocs, i, rc;
    double PI25DT = 3.141592653589793238462643;
    double mypi, pi, h, sum, x, a;
    double startwtime, endwtime;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

    n = 0;
    while (!done) {
        if (myrank == 0) {
            if (n == 0)
                n = 100000;
            else
                n = 0;
            startwtime = MPI_Wtime();
        }
        MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
        if (n == 0)
            done = 1;
        else {
            h = 1.0 / (double) n;
            sum = 0.0;
            for (i = myrank + 1; i <= n; i += numprocs) {
                x = h * ((double)i - 0.5);
                sum += f(x);
            }
            mypi = h * sum;

            MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

            if (myrank == 0) {
                printf("pi is approximately %.16f, Error is %.16f\n",
                    pi, fabs(pi - PI25DT));
                endwtime = MPI_Wtime();
                printf("wall clock time = %f\n",
                    endwtime-startwtime);
            }
        }
    }
}

```

```
}  
MPI_Finalize();  
return 0;  
}
```

-
- The command `MPI_Init(&argc, &argv)` initializes the MPI task. This command is necessary for all MPI programs.
 - `MPI_Comm_size(MPI_COMM_WORLD, &numprocs)` is used to obtain the number of processors `mpirun` associated with the `MPI_COMM_WORLD` communicator.
 - `MPI_Comm_rank(MPI_COMM_WORLD, &myrank)` is analogous to the PVM `pvm_mytid()` – it obtains the id of the current task within the `MPI_COMM_WORLD` communicator.
 - `MPI_Wtime()` returns the wall clock time.
 - `MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD)` performs a broadcast (similar to `pvm_bcast()`) to/from all the ranks in the `MPI_COMM_WORLD` communicator.
 - `MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD)` Reduces values on all ranks to a single value – the reduction operation shown here is a global sum.
 - `MPI_Finalize()` gracefully shuts down the MPI process. This command is necessary for all MPI programs. You still need to `exit()` or return from `main()`.

3 Compiling and Running Your MPI Program

You should “telnet” (“ssh -l username”) to a net0X machine where X can be 1 to 7 to compile and run your program. Make sure you “ssh” to one of the machines listed in your hostfile that is described later.

3.1 Makefile

You really should not need to write a new `Makefile`, as the existing one ought to be sufficient for the programming tasks required for this class. Note that the slave definitions in the `Makefile` have been commented out because the parallel model is based on a combined master and slave model.

You also can simply use the following command to compile your program.

```
mpicc -o cpi cpi.c
```

3.2 The Hostfile

To run your MPI program, you first need to create a hostfile. Check the LAM tutorial to get more details about how to set up this file. Here is a sample hostfile:

```
net01.utdallas.edu  
net02.utdallas.edu
```

The machine that you will be launching the program from should be listed first in the file.

3.3 Running Your MPI Program

First read the README file. And `ssh -l username` to a net machine to run your program. The LAM environment must be started before you can run your MPI programs. The command to boot the LAM MPI environment is:

```
lamboot -v hostfile
```

where `hostfile` is the name of the text file that you created. This will spawn LAM MPI daemons on all the machines listed in your `hostfile`. If it is failed, try to figure it out yourself and ask your classmate first because I and others have tested it with no problem at all.

The command to run your MPI program under LAM is:

```
mpirun N -np 2 <program name> -- <cmd line arguments to program>
```

(note that “N” is the letter N, not a number! See the LAM tutorial for more details) Try different number after `-np` option with 1, 2, 4, etc. See what happens. **Please note that you cannot have -np number larger than the the number of computers booted by lamboot.... some minor bug in our installation.**

Please also make sure that all the machines (including your local host machine) listed in the `hostfile` are included in your `.rhosts` file.

Between each `mpirun`, especially while you are debugging your program, you can “clean out” the MPI environment with the `lamclean -v` command. This kills any running MPI programs and deletes any messages that might still be resident on the network.

When you are all finished, **be sure to shut down LAM!!**. It is considered extremely rude if you do not shut down your LAM and leave daemons running on other people’s machines. The command to shut down LAM is:

```
lamwipe -v hostfile
```

where `hostfile` is the same `hostfile` that you used with `lamboot`.

4 Run Your Programs Using Open MPI in Linux Machines

You also can try to run your MPI programs in our Linux machines after recompile the programs. We have installed Open MPI there. But I found the the overhead of setting up the communication seems heavier than Sun Machines. You can try it out.

First put the following machines in your `.rhosts` file.

```
csmpi001.utdallas.edu  
csmpi002.utdallas.edu  
csmpi003.utdallas.edu  
csmpi004.utdallas.edu  
csmpi005.utdallas.edu  
csmpi006.utdallas.edu  
csmpi007.utdallas.edu  
csmpi008.utdallas.edu
```

Then copy my directory to yours.

```
cp -rf ~edsha/parallel/openMPI .
```

I assume that you already have set up your ssh. You can ssh to one of those linux machines first. Then in that machine you can do

```
eval `ssh-agent`  
ssh-add
```

For how to compile and run your MPI programs, check <http://www.open-mpi.org/faq/>
It is very easy to compile your program:

```
mpicc cpi.c -o cpi
```

It is very easy to run your program.

```
mpirun --hostfile hosts -np 4 cpi
```

Try several times if ssh prompt (yes/no) appears.