



Parallel Computing Workshop (Day 1)

Dr. Yifeng Zhu

Dr. Bruce Segee

Electrical and Computer Engineering

University of Maine

1

Introduction to MPI

- What is MPI?
 - MPI stands for “**M**essage **P**assing **I**nterface”
 - In ancient times (late 1980’s early 1990’s) each vendor had its own message passing library
 - » Non-portable code
 - » Not enough people doing parallel computing due to lack of standards

2

What is MPI?

- April 1992 was the beginning of the MPI forum
 - Organized at SC92
 - Consisted of hardware vendors, software vendors, academicians, and end users
 - Held 2 day meetings every 6 weeks
 - Created drafts of the MPI standard
 - This standard was to include all the functionality believed to be needed to make the message passing model a success
 - Final version released may, 1994

3

What is MPI?

- A standard library specification!
 - Defines syntax and semantics of an extended message passing model
 - It is not a language or compiler specification
 - It is not a specific implementation
 - It does not give implementation specifics
 - » Hints are offered, but implementers are free to do things however they want
 - » Different implementations may do the same thing in a very different manner
 - <http://www.mpi-forum.org>

4

Getting Started with MPI

- MPI contains 125 routines (more with the extensions)!
- Many programs can be written with just 6 MPI routines!
- Upon startup, all processes can be identified by their *rank*, which goes from 0 to N-1 where there are N processes

5

6 Basic Functions

- **MPI_INIT**: Initialize MPI
- **MPI_Finalize**: Finalize MPI
- **MPI_COMM_SIZE**: How many processes are running?
- **MPI_COMM_RANK**: What is my process number?
- **MPI_SEND**: Send a message
- **MPI_RECV**: Receive a message

6

Accessing UMaine Supercomputer

- Website: <http://www.clusters.umaine.edu>
- Gateway: panopticon.clusters.umaine.edu
- Need to login onto **fawlty**, **kearney2** or **bender** in order to compile your code
- Bender: OS X, Dual G5 2GHz, 2GB Ram
- Fawlty: Gentoo Linux, Dual G5 2GHz, 2GB Ram
- Kearney: Fedora, Dual Pentium III, 1GHz

7

Running Jobs

Must be submitted to the batch system (PBS)

1. Create a "PBS Script" which tells PBS how to run your job
2. Submit job(s) to PBS: `qsub myscript.ps`
3. Wait (`qstat` to check status; `qstate jobNumber` to delete a job)

Sample PBS Script Runs Over Ethernet

```
#!/bin/bash
#PBS -l nodes=13:ppn=2
#PBS -A ece574
#PBS -l walltime=00:00:05
#PBS -q linux-spool
#PBS -o out
#PBS -e err

cd ~/examples/EX00_Hello_Ethernet
echo `date`
/usr/bin/mpixec -np 25 --mca btl tcp,self ~/examples/EX00_Hello_Ethernet/hello
echo `date`
```

8

Running Jobs

Must be submitted to the batch system (PBS)

1. Create a "PBS Script" which tells PBS how to run your job
2. Submit job(s) to PBS: `qsub myscript.ps`
3. Wait (`qstat` to check status; `qstate jobNumber` to delete a job)

Sample PBS Script Runs Over Myrinet

```
#!/bin/bash
#PBS -l nodes=13:ppn=2:myrinet
#PBS -A ece574
#PBS -l walltime=00:00:05
#PBS -q linux-spool
#PBS -o out
#PBS -e err

cd ~/examples/EX00_Hello_Myrinet
echo `date`
/usr/bin/mpixec -np 25 --mca btl gm,self ~/examples/EX00_Hello_Myrinet/hello
echo `date`
```

9

Example Of MPI codes

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

int main( int argc, char *argv[] )
{
    int rank, size;
    MPI_Init( &argc, &argv );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    MPI_Comm_size( MPI_COMM_WORLD, &size );
    printf( "I am %d of %d\n", rank, size );
    MPI_Finalize();
    return 0;
}
```

Compile: `mpicc -o hello hello.c`

10

Using SPMD Computational Model

```
main( int argc, char *argv[] )
{
    MPI_Init(&argc, &argv);
    .
    .
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
    /*find process rank */

    if (myrank == 0)
        master();
    else
        slave();
    .
    .
    MPI_Finalize();
}
```

where `master()` and `slave()` are to be executed by master process and slave process, respectively.

11