Basic MPI

Tom Murphy, Dave Joiner, Paul Gray, Henry Neeman, Charlie Peck, Alex Lemann, Kristina Wanous, Kevin Hunter

Preliminaries

- What is a Supercomputer?
- MPI Version 2

Resources

- `http://www-unix.mcs.anl.gov/mpi/`
- Or, just google "MPI"
- `ftp://math.usfca.edu/pub/MPI/mpi.guide.ps`

Bindings

- C
- Fortran

MPI Commands

MPI is simple, but complex; or is it complex, but simple?

How many MPI commands are there?

- $6 + 1$

MPI Commands

MPI is simple, but complex; or is it complex, but simple?

How many MPI commands are there?

- $6 + 1$
- 128+
  - 52 Point-to-Point Communication
  - 16 Collective Communication
  - 30 Groups, Contexts, and Communicators
  - 16 Process Topologies
  - 13 Environmental Inquiry
  - 1 Profiling

Six Basic MPI commands via three fingers

Pointer Finger – Setup

- `MPI_Init()` – Allow command arguments to be modified
- `MPI_Finalize()`

Six Basic MPI commands via three fingers

Rule of Thumb – Know thy self

- `MPI_Comm_size()` – Number of MPI processes
- `MPI_Comm_rank()` – Internal process number
- `MPI_Get_processor_name()` – External processor name

Six Basic MPI commands via three fingers

Middle Finger – Message Passing

- `MPI_Send()`
- `MPI_Recv()`

MPI Concepts

MPI Types – what kind of data

- Uniformly abstract internal representation
- Heterogeneous environment through implicit representation conversion

MPI Communicator – which processes do I use

- `MPI_COMM_WORLD` represents all processes available at start-up time
- Allows processing with subsets of `MPI_COMM_WORLD`

MPI_tag – what mailbox do I look in

MPI_Processes, not processors

MPI Types – Integer

- Signed
  - `MPI_CHAR`
  - `MPI_SHORT`
  - `MPI_INT`
  - `MPI_LONG`
- Unsigned
  - `MPI_UNSIGNED_CHAR`
  - `MPI_UNSIGNED_SHORT`
  - `MPI_UNSIGNED`
  - `MPI_UNSIGNED_LONG`

MPI Types – Floating Point

- `MPI_FLOAT`
- `MPI_DOUBLE`
- `MPI_LONG_DOUBLE`

Command Syntax

Pointer Finger – Setup

- `MPI_Init(int *argc, char ***argv)`
- `MPI_Finalize()`

Rule of Thumb – Know thy self

- `MPI_Comm_rank(MPI_Comm comm, int *rank)`
- `MPI_Comm_size(MPI_Comm comm, int *size)`
- `MPI_Get_processor_name(char *name, int *resultlen)`

Middle Finger – Message Passing

- `MPI_Send(void* buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)`
- `MPI_Recv(void* buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)`

Hello World

```
#include <stdio.h>

int main(int argc, char ** argv) {

     printf("Hello World!\n");

}
```

```
gcc helloworld.c -o helloworld
./helloworld
```

Adding the Pointer Finger – Setup

- `MPI_Init(int *argc, char ***argv)`
- `MPI_Finalize()`

```
mpicc helloworld.c -o helloworld
bccd-syncdir ./hello ~/machines
mpirun -np 2 -machinefile ~/machines/tmp/<something>/helloworld
```

Hello World +2

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

int main(int argc, char ** argv) {

     MPI_Init(&argc, &argv);
     // note that argc and argv are passed by address

     printf("Hello MPI!\n");

     MPI_Finalize();
     return 0;
}
```

Adding the Thumb – Know thyself

- `MPI_Comm_rank(MPI_Comm comm, int *rank)`
- `MPI_Comm_size(MPI_Comm comm, int *size)`

```
mpicc helloworld.c -o helloworld
bccd-syncdir ./hello ~/machines
mpirun -np 2 -machinefile ~/machines/tmp/<something>/helloworld
```

Hello World +4

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char ** argv) {
int size,rank;

MPI_Init(&argc, &argv);

MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Comm_size(MPI_COMM_WORLD,&size);

printf("Hello MPI! Process %d of %d\\n",size,rank);

  MPI_Finalize();
  return 0;
}
```

Adding the Thumb – Know thyself

- `MPI_Get_processor_name(char *name, int *resultlen)`

```
mpicc helloworld.c -o helloworld
bccd-syncdir ./hello ~/machines
mpirun -np 2 -machinefile ~/machines/tmp/<something>/helloworld
```

Hello World +4(+1)

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char ** argv) {
 int size,rank;
 int length;
 char name[80];

 MPI_Init(&argc, &argv);
 MPI_Comm_rank(MPI_COMM_WORLD,&rank);
 MPI_Comm_size(MPI_COMM_WORLD,&size);
 MPI_Get_processor_name(name,&length);

 printf("Hello MPI! Process %d of %d on %s\n",size,rank,name);
 MPI_Finalize();
 return 0;
}
```

Middle Finger – Message Passing

- `MPI_Send(void* buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)`
- `MPI_Recv(void* buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)`

```
mpicc helloworld.c -o helloworld
bccd-syncdir ./hello ~/machines
mpirun -np 2 -machinefile ~/machines/tmp/<something>/helloworld
```

Hello World $6 + (+1)$ (Client Code)

```
int dest = 0;
int tag = 999;
if (rank != 0 ) {  /* I'm a client */
  MPI_Send(name,80,MPI_CHAR,dest,tag,MPI_COMM_WORLD);
}
```

Hello World $6 + (+1)$ (Client & Server Code)

```
int dest = 0;
int tag = 999;
if (rank != 0 ) {  /* I'm a client */
  MPI_Send(name,80,MPI_CHAR,dest,tag,MPI_COMM_WORLD);
}

else { /* I'm the server (rank == 0) */
  MPI_Status status;
  int source;
  for(source = 1; source < size; source++) {
    MPI_Recv(name,80,MPI_CHAR,source,tag,MPI_COMM_WORLD,&status);
    printf(" mesg from %d of %d on %s\n",source,size,name);
    }
}
```