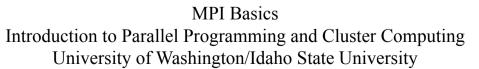
Message Passing Interface MPI Basics

Charlie Peck
Earlham College











Preliminaries

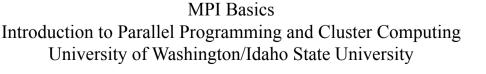
answering: "What is a cluster"

- To set-up a cluster we must:
 - Configure the individual computers
 - Establish some form of communication between machines

- Run the program(s) that exploit the above
- MPI is all about exploitation









So what does MPI do?

"What does the coder do?"

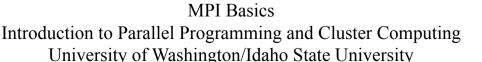
- Simply stated:
 - MPI allows moving data and commands between processes
 - Data that is needed
 - for a computation

or

- from a computation
- Now just wait a second, shouldn't that be processors?











Simple or Complex?

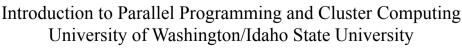
for now it's simple

- MPI has 100+ very complex library calls
 - 52 Point-to-Point Communication
 - 16 Collective Communication
 - 30 Groups, Contexts, and Communicators
 - 16 Process Topologies
 - 13 Environmental Inquiry
 - 1 Profiling
- MPI only needs 6 very simple (complex) library calls













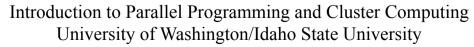
Six Basic MPI commands

- Start and stop
 - MPI_Init(...)
 - MPI Finalize(...)
- Know yourself and others
 - MPI_Comm_rank(...)
 - MPI Comm size(...)
- Message Passing
 - MPI_Send(...)
 - MPI Recv(...)













Essential MPI Organization

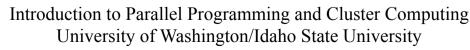
that sometimes get in the way

- Data Representation is Standardized
 - MPI data types
- Harnessing Processes for a Task
 - MPI Communicators
- Specifying a kind of message
 - MPI Tags
- How many: Processes and Processors
 - **-**np
 - -machinefile













Data Representation

Exact -> Integer Types

- Signed
 - MPI CHAR
 - MPI SHORT
 - MPI INT
 - MPI_LONG
- Unsigned
 - MPI UNSIGNED CHAR
 - MPI UNSIGNED SHORT
 - MPI UNSIGNED
 - MPI UNSIGNED LONG









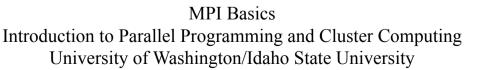
Data Representation

Appoximate -> Floating Point

- MPI_FLOAT
- MPI DOUBLE
- MPI_LONG_DOUBLE











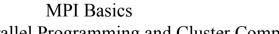
Data Representation

Special Cases

- MPI BYTE
 - Device independent
 - Exactly 8 bits
- MPI_PACKED
 - Allows non-contiguous data
 - MPI_Pack(...)
 - MPI Unpack(...)











Under the hood of the Six

How do I start and stop

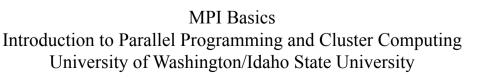
- MPI_Init(int *argc, char ***argv)
 - We gotta change (int argc, char **argv)

since

- MPI uses it to pass data to all machines
- MPI_Finalize()











Under the hood of the Six

Know thyself (and others)

- MPI Comm rank(MPI Comm comm, int *rank)
- MPI Comm size(MPI Comm comm, int *size)









Under the hood of the Six

The actual message passing

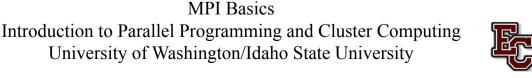
- MPI Send(void* buf, int count, MPI Datatype datatype, int dest, int tag, MPI Comm comm)
- MPI Recv(void* buf, int count, MPI Datatype datatype, int source, int tag, MPI Comm comm, MPI Status *status)

MPI Basics

University of Washington/Idaho State University







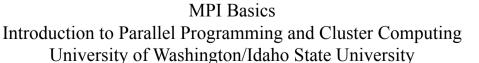


A fugue in six parts

- 1. Including the Right Stuff
- 2. General Declarations
- 3. MPI Setup
- 4. Client-side Code
- 5. Server-side Code
- 6. The Grand Finale









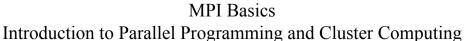


Part 1: Including the right stuff

```
#include <mpi.h>
#include <stdio.h>
#include <string.h>
#define SERVER 0
#define TAG 2
```







University of Washington/Idaho State University



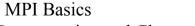


Part 2: General Declarations

```
int main(int argc, char **argv) {
  int my_rank, world_size;
  int destination=SERVER, source;
  int tag=TAG, length;
  char message[256], name[80];
  MPI_Status status;
```











Part 3: MPI Setup





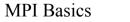


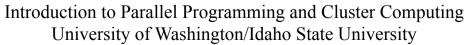


Part 4: Client-side Code













Part 5: Server-side Code









Part 6: The Grand Finale

```
fprintf(stderr, "Calling Finalize %d\n", my_rank);
MPI_Finalize();
```





