MPI Introduction

Lecture Objectives:

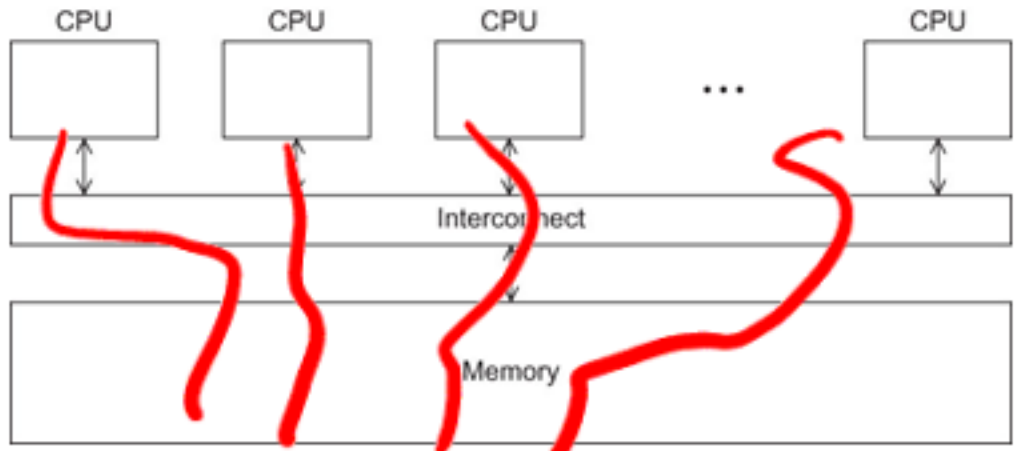
1) Explain the limitations of OpenMP
2) Define the acronym MPI.
3) Explain the advantage of MPI over OpenMP in terms of hardware architecture.
4) Define in terms of MPI the term communicator, rank, and size.
5) Explain how to execute an MPI program on a UNIX system.
6) Draw a picture showing how an MPI program is compiled.
7) Explain the purpose for the MPI Init, Finalize, Comm Size, and Comm rank methods.
8) Construct a simple MPI program using MPI Send and MPI receive.

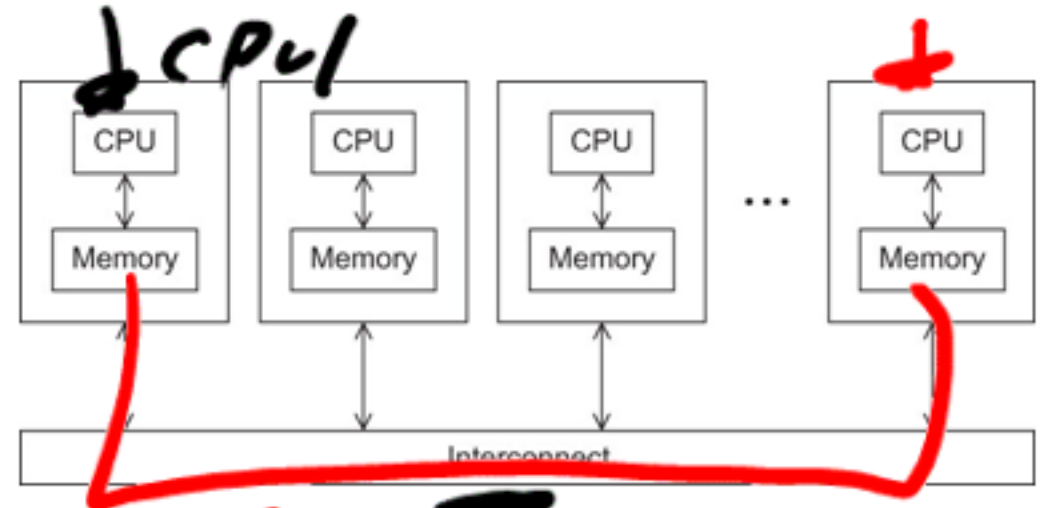
- What is wrong with it?

Open MP

Limited
No control over
how it is parallelized.
- Requires compiler support.
- But interprocess comm is
a problem Big issue.
- Architectural issues

Norma versus UMA and NUMA





UMA

NUMA

NORMA

System

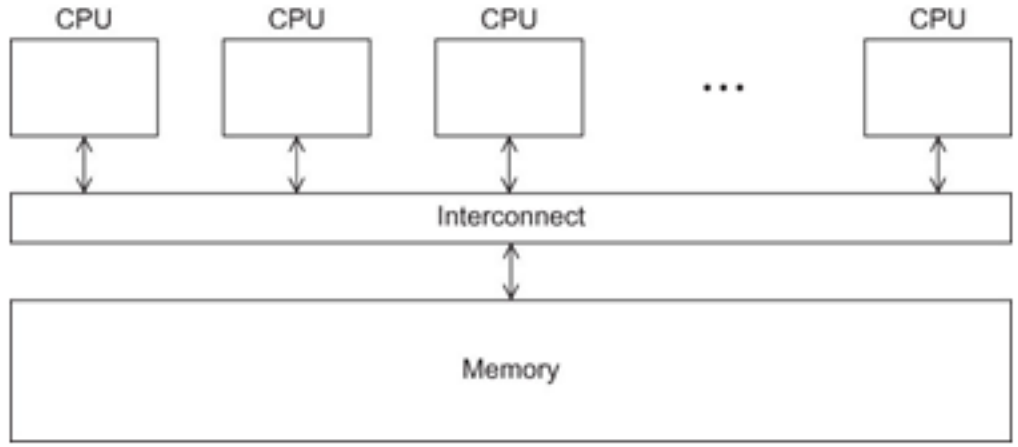
No remote

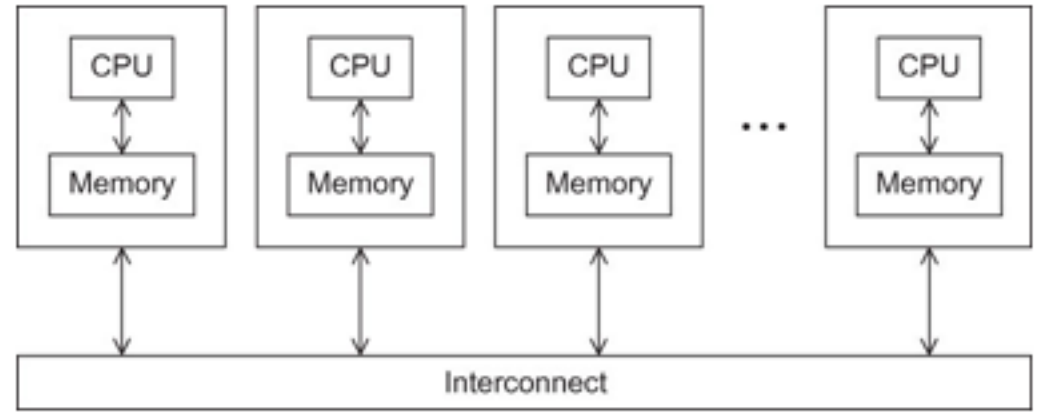
memory

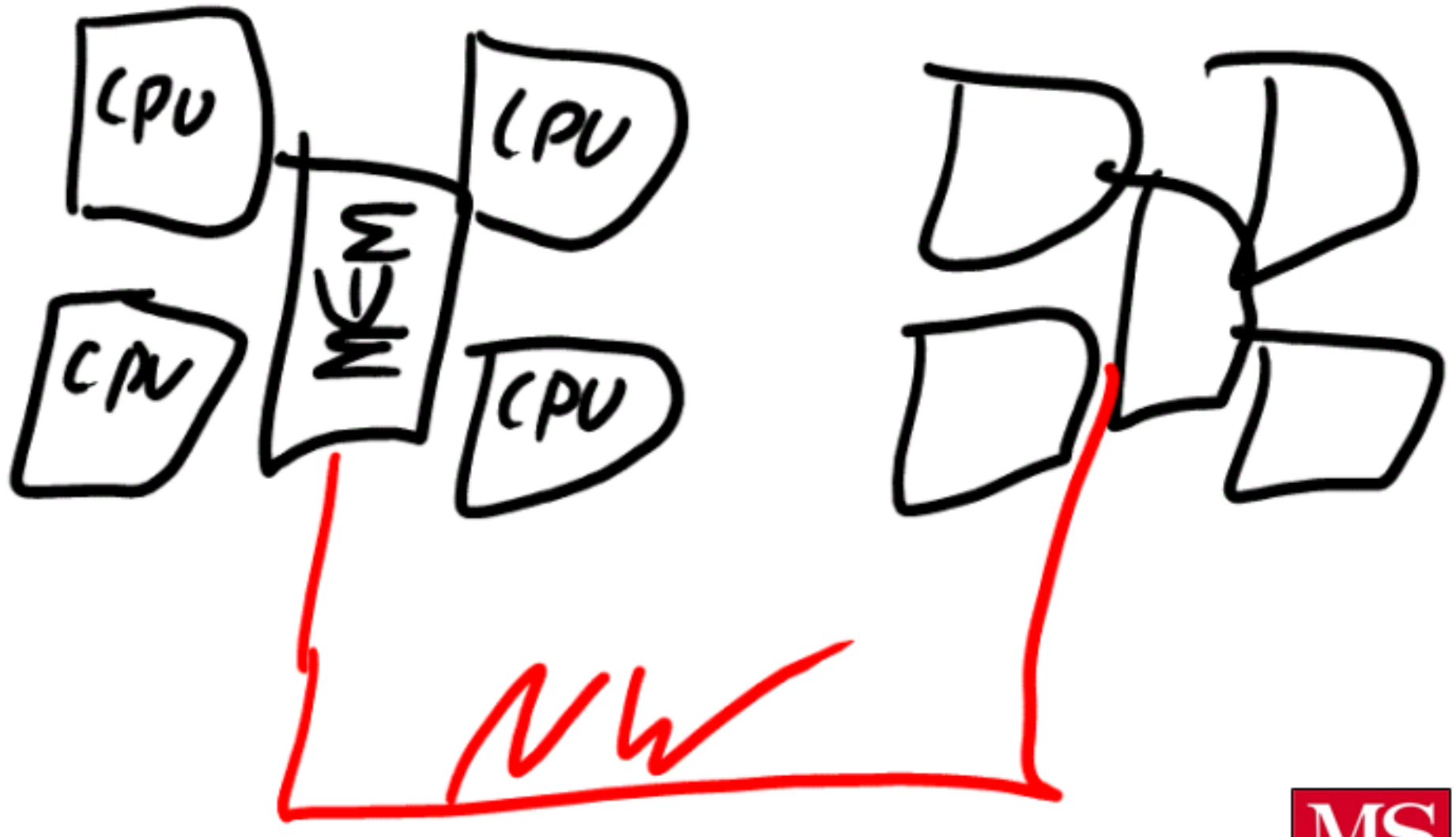
Access

MSOE

Norma versus NUMA and UMA







What is MPI?

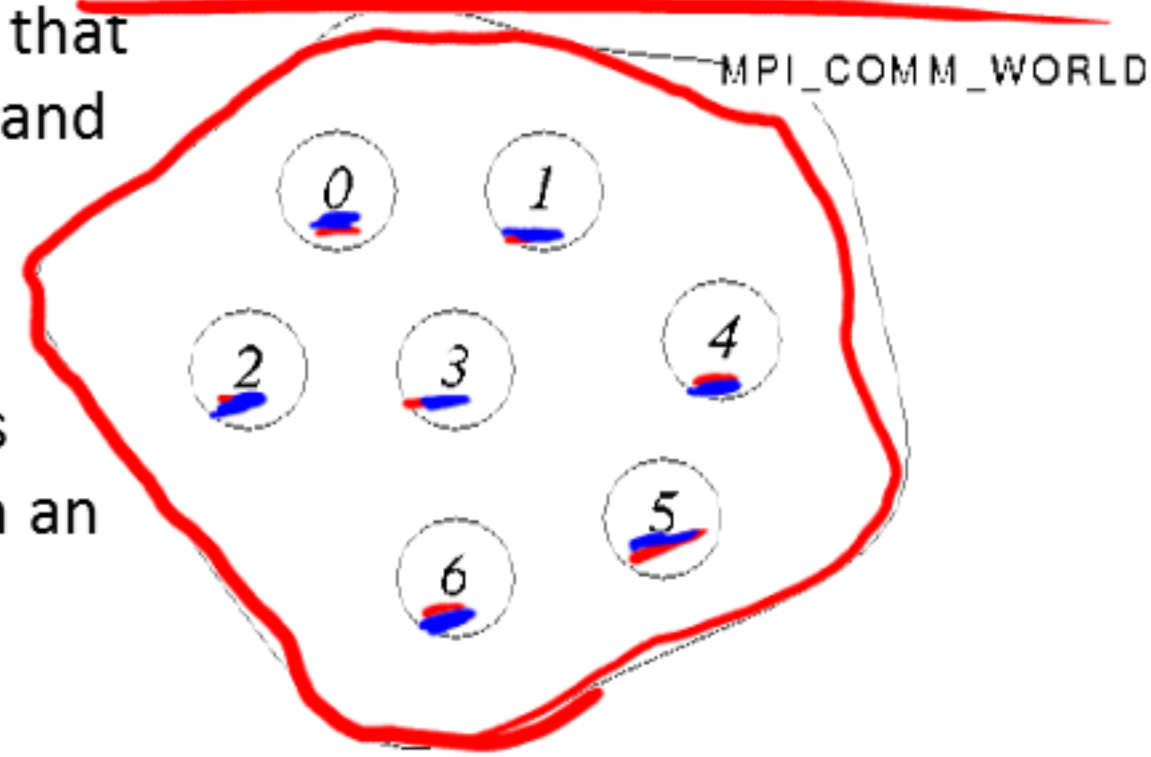
- A *message-passing library specification*
  - extended message-passing model
  - not a language or compiler specification
  - not a specific implementation or product
- For parallel computers, clusters, and heterogeneous networks
- Full-featured
- Designed to provide access to advanced parallel hardware for
  - end users
  - library writers
  - tool developers

MPI Sources

- The Standard itself:
  - at http://www.mpi-forum.org
  - All MPI official releases, in both postscript and HTML
- Books:
  - *Using MPI: Portable Parallel Programming with the Message Passing Interface*, by Gropp, Lusk, and Skjellum, MIT Press, 1994.
  - *MPI: The Complete Reference,* by Snir, Otto, Huss-Lederman, Walker, and Dongarra, MIT Press, 1996.
  - *Designing and Building Parallel Programs*, by Ian Foster, Addison-Wesley, 1995.
  - *Parallel Programming with MPI*, by Peter Pacheco, Morgan-Kaufmann, 1997.
  - *MPI: The Complete Reference Vol 1 and 2,*MIT Press, 1998(Fall).
- Other information on Web:
  - at http://www.mcs.anl.gov/mpi
  - pointers to lots of stuff, including other talks and tutorials, a FAQ, other MPI pages

MPI Basic Concepts

- MPI Communicator
  - A collection of processes that can send messages back and forth to each other
- MPI Size
  - The number of processes working in parallel within an MPI Communicator
- MPI Rank
  - The order for a specific process within the MPI Communicator
  - If we have s processes, the rank varies between 0 and s-1

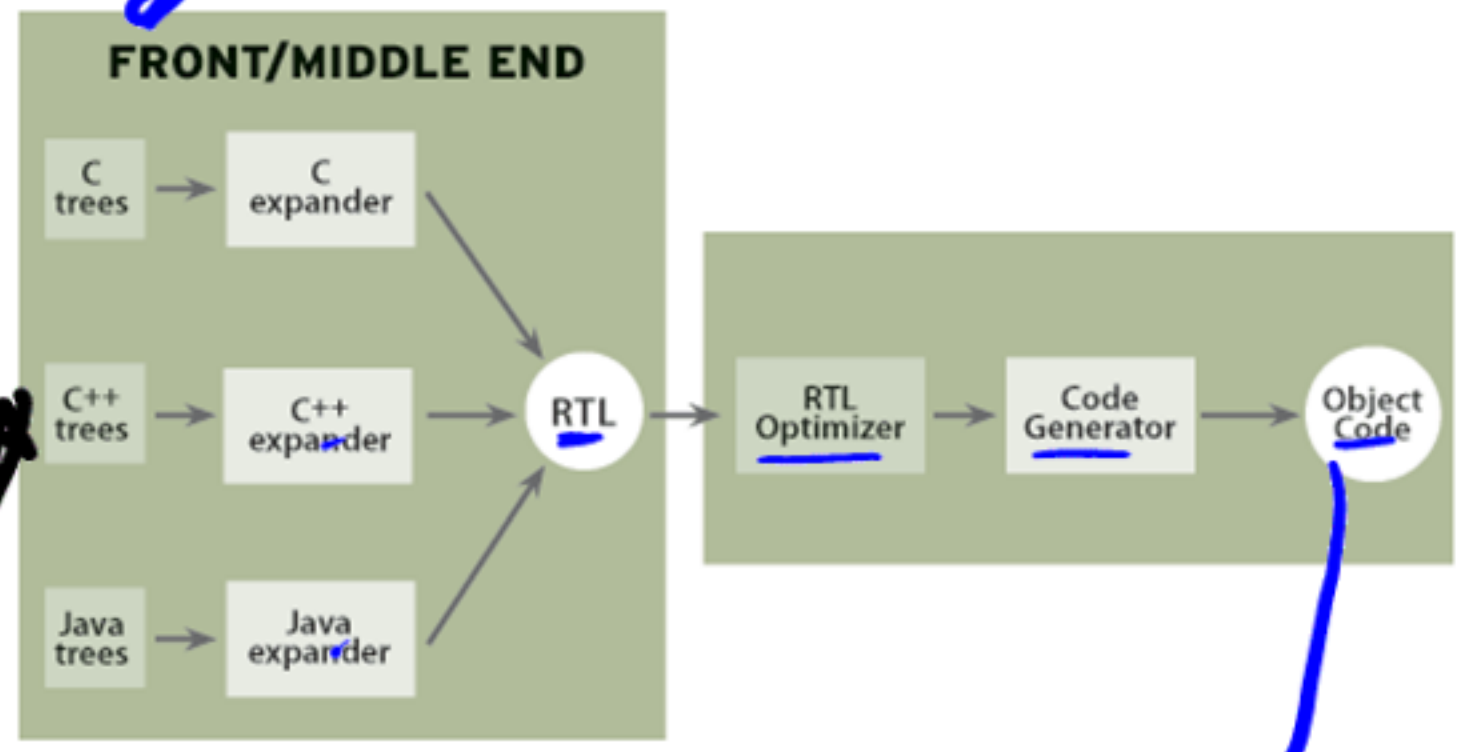


C/C++ code

MPI Compilation

MPI
Preprocessor
Translates
So MPI

C/C++ code



Link/Execute

***From Source to Binary: The Inner Workings of GCC by Diego Novillo**

MSOE

Compilation

Calls gcc or whatever other compiler.

wrapper script to compile

All warnings

source file

```
mpicc -g -Wall -o mpi_hello mpi_hello.c
```

output

Source files

produce debugging information

create this executable file name (as opposed to default a.out)

turns on all warnings

• A first MPI Program

```
28  int main(void) {
29      char        greeting[MAX_STRING];  /* String storing message */
30      int         comm_sz;               /* Number of processes    */
31      int         my_rank;               /* My process rank        */
32      /* Start up MPI */
33      MPI_Init(NULL, NULL);
34      /* Get the number of processes */
35      MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
36      /* Get my rank among all the processes */
37      MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
38      if (my_rank != 0) {
39          /* Create message */
40          sprintf(greeting, "Greetings from process %d of %d!", my_rank, comm_sz);
41          /* Send message to process 0 */
42          MPI_Send(greeting, strlen(greeting)+1, MPI_CHAR, 0, 0, MPI_COMM_WORLD);
43      } else {
44          /* Print my message */
45          printf("Greetings from process %d of %d!\n", my_rank, comm_sz);
46          for (int q = 1; q < comm_sz; q++) {
47              /* Receive message from process q */
48              MPI_Recv(greeting, MAX_STRING, MPI_CHAR, q, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
49              /* Print message from process q */
50              printf("%s\n", greeting);
51          }
52      }
53      /* Shut down MPI */
54      MPI_Finalize();
55      return 0;
56  }  /* main */
57
```

MPI

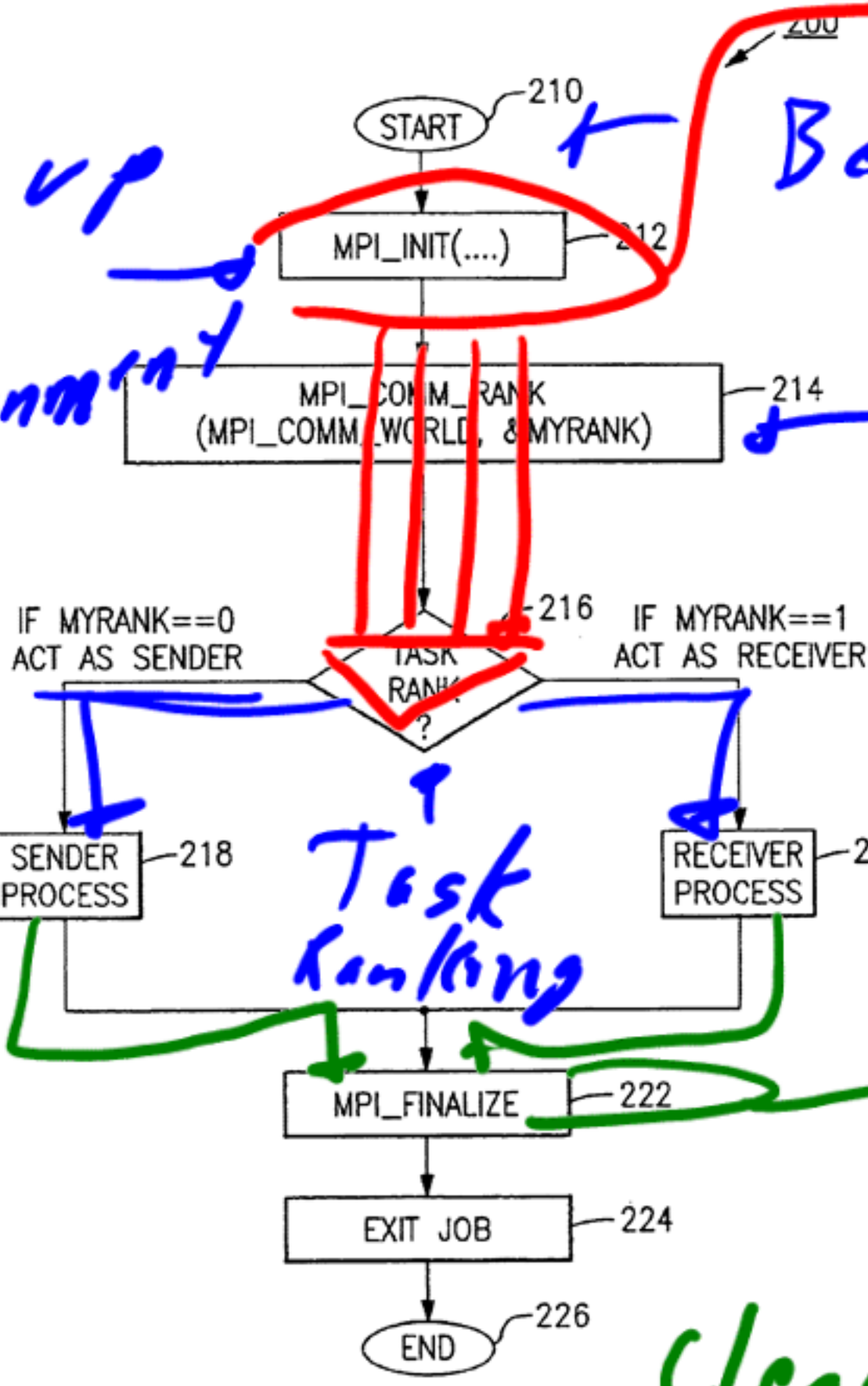
options

Slave Processes

receive all the other messages.

MSOE

MPI Code Compilation and Execution



Fork

Beginning

Sets up environment

Where do I fit in

How many

MPI Basic flow

Task Ranking

Stop parallel process

Cleans up

MPI Methods

- MPI Init
- MPI Finalize
- MPI_Comm_Size
- MPI_Comm_Rank

⇒ Sets up parallel cores

⇒ Shutdown the parallel cores

⇒ How many nodes in communicator

⇒ Who Am I?

Parallelize code and figure out our Roles.

MSOE

MPI Send

http://www.open-mpi.org/doc/v1.4/man3/MPI_Send.3.php

int MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)

- Buf
  - Initial address of send buffer (choice).
- Count
  - Number of elements send (nonnegative integer).
- Datatype
  - Datatype of each send buffer element (handle)
- Dest
  - Rank of destination (integer).
- Tag
  - Message tag (integer).
- Comm
  - Communicator (handle).

MPI Receive

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

- count
  - Maximum number of elements to receive (integer).
- datatype
  - Datatype of each receive buffer entry (handle).
- source
  - Rank of source (integer).
- tag
  - Message tag (integer).
- comm
  - Communicator (handle).

Message matching

```
MPI_Send(send_buf_p, send_buf_sz, send_type, dest, send_tag,
         send_comm);
```

MPI_Send

src = q

MPI_Recv

dest = r

```
MPI_Recv(recv_buf_p, recv_buf_sz, recv_type, src, recv_tag,
         recv_comm, &status);
```

r

q

MPI Data Types

MPI datatype	C datatype
MPI_CHAR	signed char
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long int
MPI_LONG_LONG	signed long long int
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED_SHORT	unsigned short int
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long int
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double
MPI_BYTE	Don't have C reps.
MPI_PACKED	

- mpiHelloWorld

Lets look at another example