### Introduction to MPI Programming

Schrodinger ZHU Yifan <i@zhuyi.fan>

September 29, 2021

- 1 Topology
- 2 Point-wise Communication
  - Blocking Communication
  - Probing
  - Immediate Communication
- 3 Multipoint Communication
  - Broadcast
  - Barrier
  - More
- 4 Debugging

# Topology

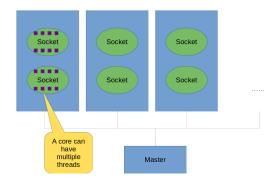


Figure: Cluster Topology

```
MPI_Send(
    void* data,
    int count,
    MPI_Datatype datatype,
    int destination,
    int tag,
    MPI_Comm communicator);
```

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

MPI datatype	C equivalent
MPI_SHORT	short int
MPI_INT	int
MPI_LONG	long int
MPI_LONG_LONG	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_UNSIGNED_LONG_LONG	unsigned long long int
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double
MPI_BYTE	char

Figure: Tags for MPI Supported Types



```
if (0 == rank) {
    data = 1999'12'08;
    for (int i = 1; i < size; ++i) {
        data = data + i;
        std::cout << "sending " << data << " to " << i <<

    std::endl;

        MPI_Send(&data, 1, MPI_INT, i, 0, MPI_COMM_WORLD);
} else {
    MPI_Recv(&data, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
    MPI_STATUS_IGNORE);
    std::cout << "received " << data << " at " << rank <<

    std::endl;
```

**Blocking Communication** 

### Blocking Point-wise Communication

```
mpirun -np 6 main
sending 19991209 to 1
sending 19991211 to 2
sending 19991214 to 3
sending 19991218 to 4
received 19991209 at 1
received 19991211 at 2
received 19991214 at 3
received 19991218 at 4
received 19991223 at 5
sending 19991223 to 5
```

Figure: Output

```
int MPI Sendrecv(
    const void *sendbuf,
    int sendcount.
    MPI_Datatype sendtype,
    int dest,
    int sendtag,
    void *recvbuf,
    int recvcount,
    MPI_Datatype recvtype,
    int source,
    int recvtag,
    MPI_Comm comm,
    MPI_Status *status);
```

**Blocking Communication** 

```
#include <mpi.h>
int main(int argc, char **argv) {
   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   std::vector<int> send(size), recv(size);
   for (auto i = 0: i < size: ++i) {
   MPI_Sendrecv(send.data(), size, MPI_INT, target, 0, recv.data(), size, MPI_INT, source, 0, MPI_COMM_WORLD,
 std::cout << "received ":</pre>
   for (auto i = 0; i < size; ++i) {
```

Point-wise Communication

0000000

```
csc4005-assignment-1/cmake-build-debug on | master [!] via \( \times \) v3.21.3
) mpirun -np 6 main
received 5 6 7 8 9 10 from 5 at 0
received 0 1 2 3 4 5 from 0 at 1
received 1 2 3 4 5 6 from 1 at 2
received 2 3 4 5 6 7 from 2 at 3
received 3 4 5 6 7 8 from 3 at 4
received 4 5 6 7 8 9 from 4 at 5
```

Figure: Output

## **Probing**

```
MPI_Get_count(
    MPI_Status* status,
    MPI_Datatype datatype,
    int* count);
MPI_Probe(
    int source,
    int tag,
    MPI_Comm comm,
    MPI_Status* status);
```

•00

000

Probing

## Probing

```
const int MAX NUMBERS = 100;
int numbers[MAX NUMBERS];
int number_amount;
if (world rank == 0) {
    // Pick a random amount of integers to send to process one
    srand(time(NULL));
    number amount = (rand() / (float)RAND MAX) * MAX NUMBERS:
   // Send the amount of integers to process one
    MPI Send(numbers, number amount, MPI INT, 1, 0, MPI COMM WORLD);
    printf("0 sent %d numbers to 1\n", number_amount);
} else if (world_rank == 1) {
    MPI Status status:
    // Receive at most MAX NUMBERS from process zero
    MPI_Recv(numbers, MAX_NUMBERS, MPI_INT, 0, 0, MPI_COMM_WORLD,
             &status):
   // After receiving the message, check the status to determine
    // how many numbers were actually received
    MPI_Get_count(&status, MPI_INT, &number_amount);
    // Print off the amount of numbers, and also print additional
    // information in the status object
    printf("1 received %d numbers from 0. Message source = %d, "
           "tag = %d\n",
           number_amount, status.MPI_SOURCE, status.MPI_TAG);
```

Probing

## Probing

```
int number amount:
if (world_rank == 0) {
    const int MAX_NUMBERS = 100;
    int numbers[MAX_NUMBERS];
    // Pick a random amount of integers to send to process one
    srand(time(NULL)):
    number amount = (rand() / (float)RAND MAX) * MAX NUMBERS;
    // Send the random amount of integers to process one
    MPI Send(numbers, number amount, MPI INT, 1, 0, MPI COMM WORLD);
    printf("0 sent %d numbers to 1\n", number_amount);
} else if (world rank == 1) {
    MPI Status status;
    // Probe for an incoming message from process zero
    MPI Probe(0, 0, MPI COMM WORLD, &status);
    // When probe returns, the status object has the size and other
    // attributes of the incoming message. Get the message size
    MPI Get count(&status, MPI INT, &number amount);
    // Allocate a buffer to hold the incoming numbers
    int* number_buf = (int*)malloc(sizeof(int) * number_amount);
    // Now receive the message with the allocated buffer
    MPI_Recv(number_buf, number_amount, MPI_INT, 0, 0,
             MPI COMM WORLD, MPI STATUS IGNORE):
    printf("1 dynamically received %d numbers from 0.\n",
```

0000

### Immediate Point-wise Communication

```
int MPI_Isend(
    const void *buf,
    int count,
    MPI_Datatype datatype,
    int dest,
    int tag,
    MPI_Comm comm,
    MPI_Request *request);
```

0000

Immediate Communication

#### Immediate Point-wise Communication

```
int MPI_Irecv(
   void *buf,
   int count,
   MPI_Datatype datatype,
   int source,
   int tag,
   MPI_Comm comm,
   MPI_Request *request);
```

### Immediate Point-wise Communication

```
int MPI_Wait(
    MPI_Request *request,
    MPI_Status *status);
int MPI_Test(
    MPI_Request *request,
    int *flag,
    MPI_Status *status);
```

0000

Immediate Communication

#### Immediate Point-wise Communication

Point-wise Communication

0000

```
MPI_Request send_req, recv_req;
MPI_Isend(send.data(), size, MPI_INT, target, 0,
→ MPI_COMM_WORLD, &send_req);
MPI_Irecv(recv.data(), size, MPI_INT, source, 0,
   MPI_COMM_WORLD, &recv_req);
int sflag = 0, rflag = 0;
do {
    MPI_Test(&send_req, &sflag, MPI_STATUS_IGNORE);
    MPI_Test(&recv_req, &rflag, MPI_STATUS_IGNORE);
} while (!sflag || !rflag);
```

### **Broadcast**

```
MPI_Bcast(
    void* data,
    int count,
    MPI_Datatype datatype,
    int root,
    MPI_Comm communicator);
```

### **Broadcast**

```
int main(int argc, char **argv) {
    int data;
    int rank;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if (0 == rank) {
        std::cin >> data;
    }
    MPI_Bcast(&data, 1, MPI_INT, 0, MPI_COMM_WORLD);
    std::cout << data << std::endl;</pre>
    MPI_Finalize();
```

Multipoint Communication

00 •0

### **Barrier**

MPI\_Barrier(MPI\_COMM\_WORLD);



#### **Barrier**

```
₹
    std::this_thread::sleep_for(std::chrono::milliseconds{100}
    * rank);
    std::cout << "hi (no bar)" << std::endl;
}
MPI_Barrier(MPI_COMM_WORLD);
{
    std::this_thread::sleep_for(std::chrono::milliseconds{100}
    * rank):
\hookrightarrow
    MPI Barrier(MPI COMM WORLD):
    std::cout << "hi (bar)" << std::endl;</pre>
}
```

### More

- Scatter and Gather: https://mpitutorial.com/tutorials/mpiscatter-gather-and-allgather/
- Reduce: https://mpitutorial.com/tutorials/mpi-reduce-and-allreduce/
- Group Division: https://mpitutorial.com/tutorials/introduction-to-groups-and-communicators/

```
#include <mpi.h>
void wrong() {
    int data = 1;
    int rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if (0 == rank) {
        MPI_Recv(&data, 1, MPI_INT, 3, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    }
}
int main(int argc, char **argv) {
    MPI_Init(&argc, &argv);
    wrong();
    MPI_Finalize();
}
```

Figure: Wrong Code

#### mpirun --timeout 5 --get-stack-traces ./main

```
TACK TRACE FOR PROC [[27102,1],0] (gru01, PIO 57162)
         Thread 5 (Thread 8x2ee391576789 (LMP 57364))
        ## Re000002ae2fe517fd3 is cooll wait () from /Libeu/Libc.co.6
        #1 0x00002ae24ebla241 in epoll_dispatch (base=0x1d521d0, tv==eptimized out=) at epoll_c:407
        #2 0x09002xc2feb3cc4a in cpal_libevent2022_event_base_loop (base=0xld521d0, flags=flags@cntry=1) at event.c:1630
        8) RobbléZac/FeaF992e is progress_engise (objrceptimized outs) at rustime/opal_progress_threads.c:186
84 0x0006Zac/Fe2F92e is start_thread () from /lib64/libpthread.so.0
         #5 0x00002xe2fe5379fd in clone () from /lib04/libc.so.6
        Thread a (Thread #220-91845h799 (LMP $7366)):
#0 009802a2/#537903 is epoil_wait () from /11664/15c.xo.6
#3 009802a2/#537903 is epoil_wait () from /11664/15c.xo.6
         #2 @c000002ac2feb3cous in cpal_libevent2022_event_base_loop (Baser0x1dab660, flags:flags@entry:1) at event.c:1630
        40 0x00002xxx0010402400 in progress_engise (obje-eptimized out+) at ratime/pmis_progress_threads.c:232
44 0x00002xxx2fe210xx0 in start.thread () from //lib04/lib0thread.io.0
        84 Dr09002ac2fe220co3 in start_thread (1 from /lib64/Lib
85 Br09002ac2fe5170fd in clone () from /lib64/Libc.so.6
Thread 1 (Thread Dr2ac38c66700 (LBP 57378)):
        #8 0x99982xe2fe52rccd im poll () from /lib64/libc.so.8
#1 0x99982xe39c2rbbf1 im ?? () from /asr/lib64/libcuda.so.1
#2 0x99982xe39c27r26a im ?? () from /asr/lib64/libcuda.so.1
         #3 0x00002xe30c2c39x0 in TT () from /usr/lib64/libcuda.so.3
        ss 0x00002xc2fe21xcc5 is start_thread () from /libs//libthread.so.0
05 0x00002xc2fe3170fd is close () from /libs//libc.so.6
Thread 2 (Thread 0x2xc2f132f0700 (LMF 071853)
         ## Re000002ae2fe537fd3 in epoll_wait () from /libes/libc.so.6
        #1 0x00002me306a9e72b in ucs_event_set_wait () from /lib64/libucs.so.0
        42 0x00002xc305a5eben in ucs_asymc_thread_func () from /lib64/libucs.se.0
81 0x00002xc305a22tea5 in start thread () from /lib64/libuthread.co.0
        #4 0x00002me2fe5379fd in clone () from /lib64/libc.so.6
        Thread 1 (Thread 8x2ex2fdb9x288 (LMP 57362)):
        ## 800000220906616527 in uct_mm_iface_progress () from /libst/libuct.so.0
        42 0c09802ac03050250 is mon_marker_propress () from /lib64(liber_nes.0
42 0c09802ac03054500 is mon_marker_propress () from /lib64(liber_nes.0
42 0c09802ac03054500 is mon_marker_propress () from /lib64(liber_nes.0)
eut>, ceam-cogtinized etc., mpi_status:new) at pml_ucc.comb

#1 000002xx27dct002c in PMPI_Recv (buf-coptinized out>, count=coptinized out>, type=coptinized out>, nource=coptinized out>, tag=coptinized out>,
com-0x203f10 <popi_mpi_com_mortd>, status=0c0) at greev.c:02
```

Figure: Stacktrace

## Parallel Debug

See descriptions on BB.