Quiz-3 Oct 21

Due 22 Oct at 23:55 **Points** 25 **Questions** 10
Available 21 Oct at 10:00 - 23 Oct at 6:00 2 days **Time limit** 80 Minutes

Instructions

Quiz-3 consists of 10 MCQ/MAQ questions. The time limit for this quiz is 80 minutes. It covers the contents of first two MPI lectures (i.e., Lec7 and Lec 8 slides).

This quiz was locked 23 Oct at 6:00.

Attempt history

	Attempt	Time	Score
LATEST	Attempt 1	52 minutes	17 out of 25

Score for this quiz: **17** out of 25
Submitted 22 Oct at 22:55
This attempt took 52 minutes.

Question 1 — 0 / 2 pts

Which of the following statement is TRUE?

ou Answered

- (selected) Both `MPI_Send` and `MPI_Recv` functions must know the exact size of the message to be sent or received.

rrect answer

- As long as the message tags are set differently, messages communicated between the same pair of processes can be distinguished from each other.
- In `MPI_Send`, the argument `count` specifies the message size in bytes.

Since `MPI_Recv` can use wildcards for `source` and `tag`, a receiver process actually can't distinguish between two messages if such wildcards are used.

Question 2

3 / 3 pts

Which of the following statement(s) is(are) TRUE?

In order for a process to find out its rank, it can call `MPI_Comm_size` function.

Correct!

A communicator is a collection of processes that can send messages to one another.

Correct!

The communicator `MPI_COMM_WORLD` consists of all the processes started when the program execution begins.

Only one communicator can exist in an MPI program.

Question 3

3 / 3 pts

Which of the following statement(s) is(are) TRUE?

Correct!

MPI is useful for an implementation of MIMD/SPMD parallelism.

In MPI programs, processes communicate only by shared memory.

Correct!

A single MPI program is usually written that can run with a general number of processes.

Correct!

MPI is a message passing library specification, not a language or compiler specification.

Correct!

It is necessary to specify explicitly which part of the MPI code will run with specific processes.

Question 4

0 / 2 pts

Suppose we have three processes, A, B, and C in a communicator, and A is broadcasting two floats, x and y, within the communicator. Suppose further that on process A, $x = 5$ and $y = 10$. See the following table. When the broadcast is completed on all three processes, which of the following is TRUE regarding the values of x and y?

Time	Process A	Process B	Process C
1	`MPI_Bcast &x`	Local work	Local work
2	`MPI_Bcast &y`	Local work	Local work
3	Local work	`MPI_Bcast &y`	`MPI_Bcast &x`
4	Local work	`MPI_Bcast &x`	`MPI_Bcast &y`

Process A: $= 5, y = 10$; Process B: $x = 5, y = 10$; Process C: $x = 10, y = 5$;

You Answered

Process A: $x = 5, y = 10$; Process B: $x = 5, y = 10$; Process C: $x = 5, y = 10$;

Process A: $x = 5, y = 10$; Process B: $x = 10, y = 5$; Process C: $x = 10, y = 5$;

rrect answer

Process A: $x = 5, y = 10$; Process B: $x = 10, y = 5$; Process C: $x = 5, y = 10$;

Question 5

2 / 2 pts

Which statement is TRUE for the following MPI function?

```
MPI_Recv(buf,2,MPI_FLOAT,1,MPI_ANY_TAG,MPI_COMM_WORLD,&status);
```

The tag of the message is 2.

The rank of the sending MPI process is not determined.

Correct!

The function call receives a message with any tag from a certain sender.

The function call receives message from any source with any tag.

Question 6

2 / 3 pts

Regarding the MPI collective communication functions, which of the following statement(s) is(are) TRUE?

Correct!

Every MPI collective communication function takes one of its arguments as a communicator that defines the group of processes participating in the corresponding collective operation.

rrect answer

MPI collective communication functions do not require tags.

Correct!

A collective communication function must be called by all the processes in the communicator to participate in the collective operation.

MPI collective communication functions are non-blocking.

Question 7

0 / 2 pts

Given the following code segment, which of the following statements is TRUE?

```
    int a[10],b[10],myrank;
    MPI_Status status;
    MPI_Request requests[2];
    ......
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
    if(my_rank == 0){
        MPI_Send(a,10,MPI_INT,1,1,MPI_COMM_WORLD);
        MPI_Send(b,10,MPI_INT,1,2,MPI_COMM_WORLD);
    }
    else if(myrank==1){
        MPI_Irecv(b,10,MPI_INT,0,2,MPI_COMM_WORLD,&request[0]);
        MPI_Irecv(a,10,MPI_INT,0,1,MPI_COMM_WORLD,&request[1]);
    }
```

The two `MPI_Send` functions can finish in any order. For example, the second `MPI_Send` function can finish before the first one does.

There is an error in the code, as there are no matching receive functions to `MPI_Send` functions.

rrect answer

The two `MPI_Irecv` functions can finish in any order. For example, the second `MPI_Irecv` function can finish before the first one does.

u Answered

The code execution could cause a deadlock.

Question 8

2 / 3 pts

The following statements are about non-blocking communication operations. Which one(s) is(are) TRUE?

Correct!

`MPI_Irecv` function does not take a `status` argument, hence, it does not return `status` information.

Calling `MPI_Irecv` function followed by calling `MPI_Test` function is equivalent to calling a `MPI_Recv` function.

rrect answer

`MPI_Isend` starts a send operation but does not complete.

Correct!

The benefit of non-blocking communication is being able to overlap communication with computation.

Question 9

3 / 3 pts

Upon execution of the following code segment using 4 MPI processes, which of the following statements is TRUE after executing the `printf` statement?

```
#define SIZE 4
......
int send_buf[SIZE] = {0, 1, 2, 3};
int recv_buf[SIZE] = {0};
int my_rank;
......
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
MPI_Scan(send_buf, recv_buf, 3, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
printf("[MPI process %d] recv_buf[%d] = %d\n", my_rank, my_rank,
recv_buf[my_rank]);
```

- Process with rank 0 output: `[MPI process 0] recv_buf[0] = 2`
- Process with rank 3 output: `[MPI process 3] recv_buf[3] = 12`
- Process with rank 1 output: `[MPI process 1] recv_buf[1] = 3`

Correct!

- (●) Process with rank 2 output: `[MPI process 2] recv_buf[2] = 6`

Question 10

2 / 2 pts

Assume an MPI program has N processes in its `MPI_COMM_WORLD` communicator, and each of these N processes stores a row of an $N \times N$ matrix, $\mathbf{A}$, in a vector $\mathbf{a}$. That is process 0 stores row 0 of $\mathbf{A}$ in vector $\mathbf{a}$, process 1 stores row 1 of $\mathbf{A}$ in vector $\mathbf{a}$, and so on. Now, we want to set up a vector, $\mathbf{b}$, on each process that stores one of the columns of $\mathbf{A}$, that is, process 0 stores column 0 of $\mathbf{A}$ in vector $\mathbf{b}$, process 1 stores column 1 of $\mathbf{A}$ in vector $\mathbf{b}$, and so on. Which of the following operations would be able to achieve this in an efficient manner?

- An `MPI_Gather` followed by an `MPI_Bcast`
- An `MPI_Scatter` followed by an `MPI_Gather`
- An `MPI_Gather` followed by an `MPI_Scatter`

Correct!

- An `MPI_Alltoall`

Quiz score: **17** out of 25