Exam Number
B142302

Course Name
Message-passing Programming

Class Test

Question 1

(a)Consider running the program on three processes: mpirun -n 3 ./classtest

i. What are the initial values of the array x on each of the three processes?

Processor0: x = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
Processor1: x = [-1]*15
Processor2: x = [-1]*15

ii. How many send or receive calls are made by each process?

Processor0: MPI_Send = 10 times, MPI_Recv = 0 times;
Processor1: MPI_Send = 0 times, MPI_Recv = 5 times;
Processor2: MPI_Send = 0 times, MPI_Recv = 5 times;

iii. Carefully explain the communications pattern, i.e. which data is transferred between which processes.

Processor send to Processor 1: x[1] = 2, x[4] = 5, x[7] = 8, x[10] = 11, x[13] = 14;
Processor send to Processor 2: x[2] = 3, x[5] = 6, x[8] = 9, x[11] = 12, x[14] = 15;
Processor 1 received from Processor 0: x[0:5] = 2,5,8,11,14;
Processor 2 received from Processor 0: x[0:5] = 3,6,9,12,15;

iv. What are the final values of the array x on each of the three processes?

Processor0: x = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
Processor1: x = [2,5,8,11,14]+[-1]*10
Processor2: x = [3,6,9,12,15]+[-1]*10

(b) Write down a SLURM script to run classtest on the compute nodes of Cirrus using 15 MPI processes distributed equally across 3 compute nodes. If you want you can modify a script previously written by yourself on Cirrus. To maintain anonymity you should assume your username is s7654321

```
#!/bin/bash

#SBATCH --job-name=classtest
#SBATCH --time=0:10:0
#SBATCH --exclusive
#SBATCH --nodes=3
#SBATCH --tasks-per-node=5
#SBATCH --cpus-per-task=36

#SBATCH —account=m22oc-s7654321
#SBATCH --partition=standard
#SBATCH --qos=standard

module load intel-compilers-19

# Change to the submission directory
cd $SLURM_SUBMIT_DIR


# Launch the parallel job
srun --cpu-bind=cores ./classtest
```

(c) Would the code still work correctly if the if block containing the receive calls came before the block containing the send calls instead of after it? Explain your answer.

Yes. Because the Receives will queue there and wait till they really receive the message from rank0 to feed them one by one asynchronously. As long as the send action is done, they will finally get the message.

(d) You are asked to rewrite the program so that the output is the same, but each process only makes a single call to either send or receive. What changes would have to be made to the program? (you should not attempt to use derived datatypes - the MPI data type should still be MPI INT / MPI INTEGER)

1) Scatter the message by preprocessing x[] array,e.g. x_2 = [2,5,8,9,11,3,6,9,12,15], count = 5, AND

2) Increase the count for MPI_Receive() from 1 to 5 for each process.

(e) In an attempt to make the program faster, the send call is replaced by the following non-blocking call:
MPI_Issend(&x[i], 1, MPI_INT, dest, tag, MPI_COMM_WORLD, &request);
There are no other changes to the code except the definition of request:
MPI_Request request; // C version
Is the new code still correct? Explain your answer. If implemented correctly, why might you expect the new code to be faster?

It should be MPI_Wait(&request, MPI_Status_Ignore). It seems that there is no improvement..?