Amr Ibrahim
111342219
AMS 530 Project 1 Problem 1.1
9/28/2023

Note: Since the script snippets and output were small, I attached them to Brightspace and to this mini report. I'll experiment uploading my files into a directory named amibrahim under the class directory on seawulf here is the path: /gpfs/projects/AMS530/amibrahim/project1.

Problem Description

When running a parallel program on P processors to output "Hello from Processor X" the order is stochastic as in the order of series of Hello is random with every run. The aim of this problem is get a deterministic output that in a desired output.

Program

I used python as my programing language using mpi4py. The logic that I programmed utilizes send and recv MPI functions where any process with rank > 0 has to wait to receive an empty message from rank – 1 process before sending their "hello" printout as shown in Fig 1.

```
# project1.py
from mpi4py import MPI # import mpi using mpi4py

comm = MPI.COMM_WORLD # initialize MPI env
rank = comm.Get_rank() # get rank of processor
size = comm.Get_size() # get size of the group

if rank >  0:
        data = comm.recv(source = rank - 1) # if rank > 0 recv from rank - 1
print("Hello from Processor %d" % rank)
if rank < size - 1:
        comm.send(None,dest = rank + 1)  # send 'None' to rank + 1 if rank < max rank
```

Fig 1. Project1.py script

```
#!/bin/bash
#
#SBATCH --job-name=project1
#SBATCH --output=project1.out
#SBATCH --ntasks-per-node=20
#SBATCH --nodes=1
#SBATCH --time=02:00
#SBATCH -p short-28core

module load anaconda/2
module load mvapich2/gcc/64/2.2rc1

mpirun -np 20 python /gpfs/projects/AMS530/amibrahim/project1/project1.py
```

Fig 2. Project1.slurm script

Results

As shown in Fig 3. All "hello" printouts of the 20 processors are ordered. The stochastic element of the 1st come 1st print has been eliminated due to send recv function pair.

```
Hello from Processor 0
Hello from Processor 1
Hello from Processor 2
Hello from Processor 3
Hello from Processor 4
Hello from Processor 5
Hello from Processor 6
Hello from Processor 7
Hello from Processor 8
Hello from Processor 9
Hello from Processor 10
Hello from Processor 11
Hello from Processor 12
Hello from Processor 13
Hello from Processor 14
Hello from Processor 15
Hello from Processor 16
Hello from Processor 17
Hello from Processor 18
Hello from Processor 19
```

Fig 3. Project1.out deterministic output of Project1.py

Analysis

This program demonstrates how to overcome the stochastic nature of parallel computing by deterministically controlling the order which processors execute. By having each processor wait to receive data from another processor before printing out its rank and sending data to the next processor we ensure that order is preserved in the printout.