Name: Brijesh Mavani
CWID: A20406960
University: Illinois Institute of Technology
Course: Parallel and Distributed Processing
Final Project - README

The codes for each implementation are already compiled on the comet system.
1. ProgrammingProject_2DConvolution_MPISendReceive_BrijeshMavani.c – Executable mpip2p is created after compilation for part A.
2. ProgrammingProject_2DConvolution_MPICollective_BrijeshMavani.c – Executable mpicoll is created after compilation for part B.
3. ProgrammingProject_2DConvolution_MPITask_BrijeshMavani.c – Executable mpitask is created after compilation for part C.

Following steps can be executed for code execution:

1) C codes are already compiled as mentioned above. In case re-compilation is required you can execute the following command:
mpicc -o <executable file name> <C code file name>
eg: mpicc -o mpip2p ProgrammingProject_2DConvolution_MPISendReceive_BrijeshMavani.c – For part A implementation.

mpicc -o mpicoll ProgrammingProject_2DConvolution_MPICollective_BrijeshMavani.c – For part B implementation.

mpicc -o mpitask ProgrammingProject_2DConvolution_MPITask_BrijeshMavani.c – For part C implementation.

2) For code execution there are two ways. Both ways are provided below:

1) Creating a job and submitting.
    a. Create the job file for execution. There are jobs created for each processor size (1, 2, 4, and 8).
    File name format: <MPIP2P/MPICollective/MPITask><no of processors>_job.sh
    eg:
    MPIP2P1_job.sh -- For MPI point to point communication (Part A) for 1 processor
    MPIP2P2_job.sh -- For MPI point to point communication (Part A) for 2 processors

    MPICollective1_job.sh -- For MPI collective communication (Part B) for 1 processor
    MPICollective2_job.sh -- For MPI collective communication (Part B) for 2 processors

    MPITask8_job.sh – For implementation of task and data parallelism (Part C) for 8 processors

Sample .sh file content

```
------------------------------------------------------------------------------------------------------------------------

#!/bin/bash
#SBATCH --job-name="<executable file>"
#SBATCH --output="MPI_P2P_1_.%j.%N.out"
#SBATCH --partition=compute
#SBATCH --nodes=X
#SBATCH --ntasks-per-node=1
#SBATCH --export=ALL
#SBATCH -t 00:10:00

ibrun -np X ./<executable file>
------------------------------------------------------------------------------------------------------------------------
```

eg:ibrun -np 1 ./mpip2p
where,

- X is number of processors i.e. 1,2,4,8. For Part C implementation, this should be multiple of 4.
- executable file -- name of the executable file name given during compilation. eg. mpip2p, mpicoll, mpitask.

Same script can be created for other implementation (part B and C) by changing details for X and executable file.

b. Execute the job with sbatch <job name>.
eg: $ sbatch MPICollective1_job.sh
$ sbatch MPIP2P1_job.sh
$ sbatch MPITask8_job.sh

c. The result of the implementation will be stored in the file as below:
   1. mpi_sendrecv_output – Output of MPI point to point communication implementation (part A). The format is same as the provided input files.
   2. mpi_collective_output – Output of MPI Collective communication implementation (part B). The format is same as the provided input files.
   3. mpi_task_output – Output of task and data parallelism implementation (part C). The format is same as the provided input files.

d. Each job execution will create the output file with unique name for with the execution details about elapsed, communication and computation timings. For simplicity, execute the command: ls –ltr to find out which file created at last.

e. cat <output file name> -- To view the contents of the job output file or the program output file. As program output file can be large, you can use the following command to view certain lines of the output file.

   tail -n 10 <output file name> -- for viewing only last 10 lines of the output file.

2) Alternatively, below command can be executed to execute the code without submitting the job:

mpirun –np X ./<executable file>
where,

- X is number of processors i.e. 1,2,4,8. For Part C implementation, this should be multiple of 4.
- executable file -- name of the executable file name given during compilation. eg. mpip2p, mpicoll, mpitask.

Eg: mpirun –np 1 ./mpip2p