Lab 7

CSEN 145L: MPI Programming

Message Passing Interface

In parallel computing, efficient data exchange between processes is crucial for maximizing performance and scalability. MPI (Message Passing Interface) provides a suite of collective communication operations that facilitate coordinated data movement across processes, enabling effective parallel computation. This lab explores key MPI collective functions such as broadcast, reduction, scatter and gather, all-to-all communication, and all-to-all personalized communication. You will learn how your program code and data can be transferred over different nodes to utilize maximum compute resources.

You are provided with merge_sort_mpi.cpp and and squared_sum_mpi.cpp files and you are required to complete MPI related code for each problem.

To get Started, Clone or download the lab06 folder from github.

1. Start implementing the remaining code for both the .cpp files.

2. To compile/execute and test your code, you will need to request resources from WAVE HPC by either requesting interactive (srun method) or non-interactive (sbatch) resources.

A. Srun method:

srun --partition=cmp --cpus-per-task=4 --mem=4G --nodes=2 --ntasks-per-node=4 --pty /bin/bash

This configuration will let you run your code on a total of 8 cores across 2 nodes.

After you are allotted with resources your current node will change from login1/2/3 to cpuXX or memXX.

3. Loading the environment
   Load Open MPI module using module load
   module load OpenMPI

or
module load OpenMPI/4.1.5

4. Now to compile and execute your code, you will need to compile it using **mpicc**

To, Compile .cpp files using
Ex: mpicc -o merge_sort_mpi merge_sort_mpi.cpp

To run your code,
mpirun -np 8 ./**merge_sort_mpi (to distribute workload across 8 cores)**
or
mpirun -np 16 ./**merge_sort_mpi (to distribute workload across 16 cores)**

B. Sbatch Method:

This method follows all the steps explained above since they are already present in the given **lab07_job.sh** bash file.

```
#!/bin/bash
#SBATCH --partition=cmp       # Specify the partition, gpu or hub
#SBATCH --nodes=2             # Number of nodes
#SBATCH --ntasks-per-node=4         # Number of tasks (1 task)
#SBATCH --cpus-per-task=4      # Number of CPU cores per task
#SBATCH --mem=2G              # Memory allocation
#SBATCH --time=00:10:00       # Max runtime (1 hour)
#SBATCH --output=mpi_output.log # Log standard output to file
#SBATCH --error=mpi_error.log   # Log error output to file

# Email notifications
#SBATCH --mail-type=BEGIN,END,FAIL  # Send email on job start, end, or failure
#SBATCH --mail-user=your_email@example.com  # Replace with your email address

# Load the OpenMPI module
module load OpenMPI/4.1.5

# Compile the MPI  code
mpicc -o merge_sort_mpi  merge_sort_mpi.cpp
mpirun -np 8  ./merge_sort_mpi
```

Note: This batch file is for merge_sort_mpi and you will need to edit or create a new file for squared sum.