EE/CSCI 451
Fall 2024
Programming Homework 4

Assigned: October 8, 2024
Due: October 18, 2024, submit via BrightSpace
Total Points: 100

General Instructions

- You may discuss the algorithms. However, the programs have to be written individually.
- Submit **the source code, makefile and a simple report** via **BrightSpace**. The report should include the screenshots on the output from HPC. The source codes should be named 'p1.c', 'p2_1.c', 'p2_2.c' and 'p2_3.c'. Put all the files in a zip file with file name `<firstname>_<uscid>_phw<programming homework number>.zip` (do not create an additional folder inside the zip file). For example, `alice_123456_phw1.zip` should contain only the source codes and the report.
- Your program should be written in C or C++. You can use any operating systems and compilers to develope your program. However, we will test your program on a x86 linux with the latest version of g++ and gcc. Make sure you makefile could compile the executables for **all** the problems (hint: set multiple targets for the makefile) and the name of the executables are correct. If your program has error when we compile or run your program, you will lose at least 50% of credits.

Examples

The "mpi examples" folder includes the source codes used in discussions and a pbs file 'queue.pbs '. To run an mpi program, for example, the 'scatter.c', follow the steps:

1. login to USC CARC
2. Go to your working directory which has 'job.sl' and 'scatter.c'.
3. mpicc -o go scatter.c

4. sbatch job.sl
5. Check 'mpijob.out' for output and 'mpijob.err' for any possible error.

1 Pass Message in a Ring [40 points]

Write an MPI program that passes a value around a ring of 4 processes using the following steps.

1. Process 0 initializes $Msg = 451$ and prints value of Msg
2. Process 0 sends the value of Msg to Process 1
3. Process 1 receives the value of Msg, increases it by 1, prints the value and sends the current value of Msg to Process 2
4. Process 2 receives the value of Msg, increases it by 1, prints the value and sends the current value of Msg to Process 3
5. Process 3 receives the value of Msg, increases it by 1, prints the value and sends the current value of Msg to Process 0
6. Process 0 receives the value of Msg from Process 3 and prints the value

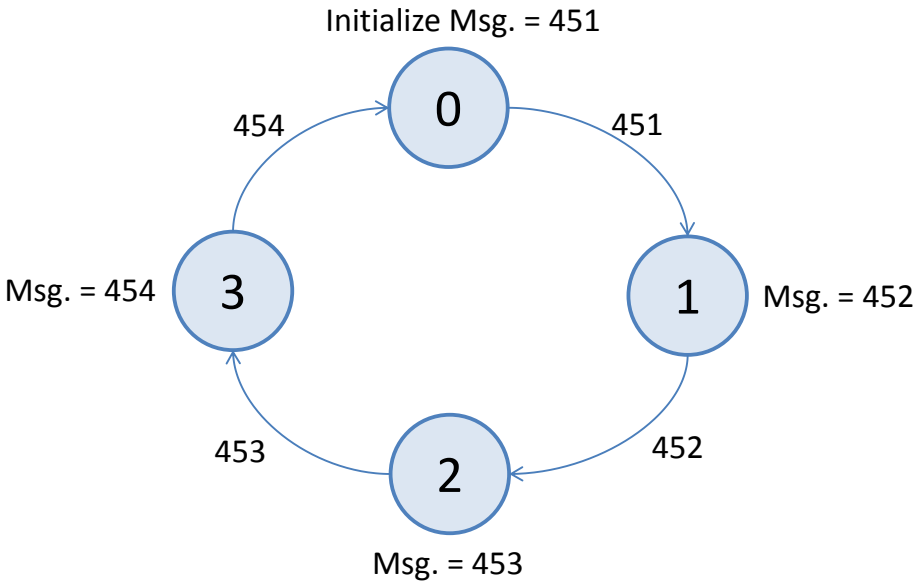

Figure 1: Example diagram

Name this program as 'p1.c'. Figure 1 illustrates the steps. The output messages look like:

- Process 0: Initially $Msg = 451$
- Process 1: $Msg = 452$
- Process 2: $Msg = 453$
- Process 3: $Msg = 454$
- Process 0: Received $Msg = 454$. Done!

2 Add 64 numbers using 4 processes [60 points]

In the "number.txt" file, you can find 64 numbers. Your task is to write an MPI program with 4 processes to compute the sum of these 64 numbers. There are 3 approaches:

1. Approach 1, name this program as p2_1.c:
   - Each process reads the entire *array*.
   - Do in parallel: Process 0 computes $\sum_{i=0}^{i=15} array[i]$; Process 1 computes $\sum_{i=16}^{i=32} array[i]$; Process 2 computes $\sum_{i=32}^{i=47} array[i]$; Process 3 computes $\sum_{i=48}^{i=63} array[i]$.
   - Process 1,2,3 send their partial *sum* to Process 0.
   - Process 0 computes the sum of all the partial *sum*s and prints it out.
2. Approach 2, name this program as p2_2.c:
   - Process 0 reads the *array*
   - Process 0 broadcasts the entire *array* to every process
   - Do in parallel: Process 0 computes $\sum_{i=0}^{i=15} array[i]$; Process 1 computes $\sum_{i=16}^{i=32} array[i]$; Process 2 computes $\sum_{i=32}^{i=47} array[i]$; Process 3 computes $\sum_{i=48}^{i=63} array[i]$.
   - Process 0 uses MPI_SUM reduction to sum these partial *sum*s.
   - Process 0 prints out the result.
3. Approach 3, name this program as p2_3.c:
   - Process 0 reads the array and scatters the entire *array* to every process using the **scatter** operation.
   - Each process sums up the portion of the *array* it receives.
   - Process 0 uses the **gather** operation to gather these partial *sum*s, computes the sum of all the partial *sum*s and prints it out.