EE/CSCI 451
Spring 2015
Programming Homework 6

Assigned: March 30, 2015
Due: April 10, 2015, before 11:59 pm, submit via blackboard
Total Points: 60

1 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 hpc-login3.usc.edu
2. source /usr/usc/openmpi/default/setup.sh
3. Go to your working directory which has 'queue.pbs' and 'scatter.c'.
4. mpicc -o go scatter.c
5. Modify the queue.pbs using your own information (working directory, email, etc.)
6. qsub queue.pbs (if you see 'qsub:script is written in DOS test format', try:
   dos2unix queue.pbs
   then
   qsub queue.pbs)
7. After you get the email saying your job is completed, check 'mpijob.output' for output and 'mpijob.error' for any possible error.

2 Pass Message in a Ring [10 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

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

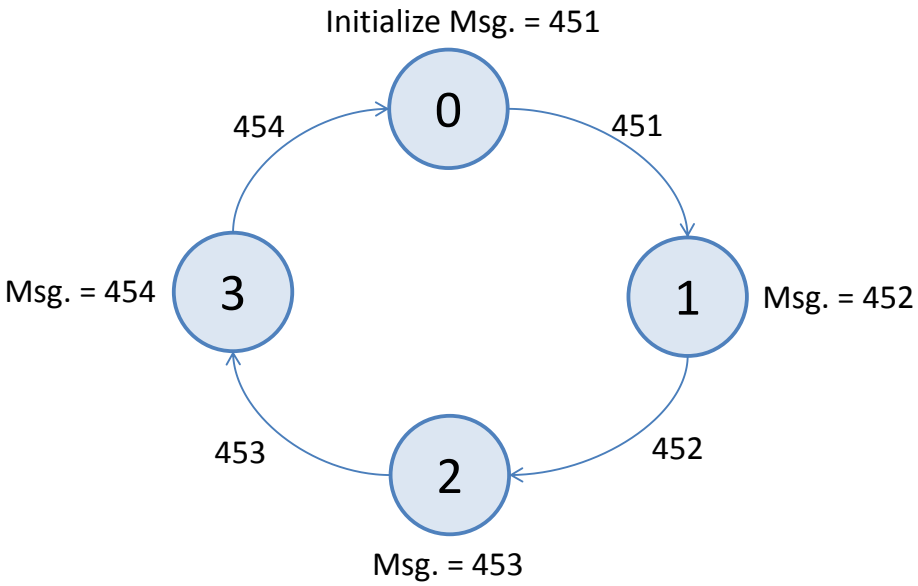

Figure 1: Example diagram

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

3 Add 64 numbers using 4 processes [30 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.

4 Bitonic Sorting using MPI [20 points]

In this problem, you will use Bitonic sort algorithm to sort an integer array. Name this program as 'p3.c'. Follow the steps to sort an 512-element array using **4** processes.

1. Process 0 creates a 512-element array. The value of each element is a **random** integer ranging from 0-2048.
2. Process 0 uses *scatter* operation to distribute the array to other processes. Each process receives 128 elements.
3. Do in parallel: each process sorts 128 element using a serial *sort* function (this can be the quicksort function which you implemented before).
4. Use the Bitonic sort approach discussed in Discussion 9 to sort the entire array.
5. Process 0 collects the sorted elements from each process using *gather* operation and prints out the entire array.

5 Submission

You may discuss. However, the programs have to be written individually. You need submit your codes, 'p1.c', 'p2_1.c', 'p2_2.c', 'p2_3.c' and 'p3.c', via blackboard. **No report** is needed for this assignment.