CISC 372: Introduction to Parallel Programming
Fall 2006
Individual Programming Assignment 2
Due Date: start of class, Thursday, September 21, 2006

Individual Work: You should be doing the work for this lab on your own. You may ask the TA or instructor for help, but there should be no collaboration among students in this class or outside this class (with the exception of sharing and comparing output generated by programs).

Objectives: The objective of this assignment is to write some simple MPI programs to learn how to accomplish communication between processes, to debug and time MPI programs, and make some observation about different programming approaches.

Procedure:

1. **Your First Parallel MPI Program.** *Centralized Summation:* Write an MPI program that computes the global sum of results from computations on the data local to each process in the following way. First, for ease of creating local data, use the rank of each process as its local data. To have some computation at each node, have each node locally compute the summation 1 to 10000 of the local rank (myrank+myrank+...myrank), performing the additions explicitly, not by multiplying 100000 times myrank! We're just creating some fair amount of local computation! The overall centralized summation is the sum of these local sums. Assuming that there are N processes, the global sum should be computed by having processes 1 through N-1 send their local result to process 0, and having process 0 receive their local results and compute the global sum of the local results. The processes should not have to send their local results in any particular order, and process 0 should not have to receive them in any particular order. That is, process 0 should not explicitly try to receive the results in order 1 through N-1. This causes undue synchronization. Your centralized summation program should be written in a general manner in order to work for any number of processes N > 1, without recompilation of the program code.

2. Take 3 timings each for 4 processes, 8 processes, and 16 processes. The goal of the timings are to compare the three algorithms. So, in order to compare them, we need to time all of each algorithms starting after the initial MPI init, comm rank and comm size commands, and before the printing of the results. The printing takes up too much time. So, you want to place your timing statements such that process 0 does the calls, and does the _rst call before it does any work (including its own computation of its local sum), and after it gets all the results from everyone and does the global summation. Typcally, process 0 starts on the machine you are logged into and submitting the mpirun command, so it will get started before anyone else, so this will include the complete computation time of all local sums and the global sum and all communication.

3. **Input and Data Distribution.** Copy your centralized summation program to a new file. Modify the new file such that process 0 reads in a data file of 10,000 integers, and evenly distributes the numbers (using a single dimension array) as load balanced as possible to the number of processes for the current run. Have the processes sum their local data and send back for the global summation as above. (You can create your own data files and share data files with others.) Insert timing statements to time the data distribution code segment for each of the same number of processes, and separately time the computation part as before. Record your data distribution and your computation times for 4, 8, and 16 processes in a table.

4. **An Improved Parallel Algorithm.** Design and implement an improved program for data distribution and summation, which you believe will speed up the program. Insert the same timing commands into your program to fairly compare the data distribution time and the computation times with the first summation program. Record your times in a table.

5. **What to Hand in.**

   a. Your three programs: centralized summation with fake data summation, centralized summation with data input and distribution, and the improved parallel algorithm.
   b. Screen shot or script of a run with 4 or 8 processes for each program.
   c. Discussion paragraph summarizing your observations and conclusions based on the three programs. The more detailed and analytical your discussion, the better.

6. **Criteria for Evaluation.**
   a. Centralized summation with fake data: 15 points
   b. Data input and distribution: 15 points
   c. Improved parallel algorithm: 15 point
   d. Screen shots of runs 5 points
   e. Tables of timings for each program for 4, 8, and 16 processes 15 points (5 each)
   f. Discussion 10 points
7. **Extra Credit.**
   Copy and modify one of your last two programs to support any number of data inputs in terms of maintaining a good load balance of data distribution and computation. Show that your program works for an unknown number of inputs by showing 2 separate runs with different input sizes.