CISC 372: Introduction to Parallel Programming
Fall 2007
Individual Programming Assignment 2
Due Date: start of class, Monday September 17, 2007

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 explore the SPMD model of parallelism and how to program in the SPMD model.

Procedure:

1. **Your First Parallel MPI Program – Storing data, locally computing and printing local results.** Write an MPI program that can be run correctly for any P>0 number of processes. Each process should assign its process id (often called its rank) to the variable N. Each process should then **dynamically allocate** space for exactly an N+1 length array of integers, fill the array with values as N*I to the Ith position, and then print its rank and the whole array that it has stored, well labeled enough to be able to tell which process is printing which array and each value of each array. You do not have to make sure that process 0 prints before process 1, etc. Just be sure it it clear which process has printed each output. You should try to minimize the number of messages that are sent to avoid undue overhead.

For example, good output might be:
Process 0: myarray[0]=0
Process 2: myarray[0]=0
Process 2: myarray[1]=2
Process 2: myarray[2]=4
Process 1: myarray[0]=0
Process 1: myarray[1]=1

Compile and run your program for 1, 8, and 16 processes. Script the runs and include the script of your compilation and all runs in your hand-in along with your program code.

2. **Your Second MPI program – Reading input and Performing Simple message passing.** Write an MPI program in which process 0 reads a single integer N>=0 from the first line of a file. This integer indicates the number of rows N and columns N of a square matrix also stored in the next few lines of the input file. So, the input file has N on the first line, and then N rows with N integers in each row. You can assume that N will be greater than P and evenly divisible by 2, 4, 8,

and 16 and you should make sure that your input file is of that form. So, N could be 16, 32, 64, 128, for instance.

Now, process 0 should read in row 0 and store it locally in a single dimension array of size N, then process 0 reads in row 1 and sends the row of N integers to process 1 which stores the row in an array of size N integers. Similarly, process 0 should send row I for I>0 to each process I until process 0 has read and sent out a row to each of the P processes. (Note there may still be more rows in the input file at this time – stay tuned.) Each process I>0 should receive their row from process 0 and print their process number followed by their received row of N numbers. For example, Process 1: 1 1 1 1
For N=4 where row 1 is 1 1 1 1.
You should try to minimize the number of messages that are sent to avoid undue overhead.

Create an input file in which N=16 and all the elements of row I = I. Compile and run your program for 1, 4, 8, and 16 processes. Script the runs and include the script of your compilation and all runs in your hand-in along with your program code.

Extra Credit: Copy program 2 and modify the program to have process 0 continue to send out the next rows to the P-1 other processes as well as assign itself to another row, and each process receive rows and print them, until all N rows have been sent, received, and printed. Compile and run your program for 1, 8, and 16 processes. Script the runs and include the script of your compilation and all runs in your hand-in along with your program code.

3. **What to Hand in.**

    a. Your 2 programs
    b. Screen shot or script of the runs as specified above for each program
    c. Discussion paragraph on any difficulties you had along the way, and/or misconceptions about SPMD parallelism that were eventually clarified.
    d. If you do the extra credit, include the code and script as specified.

4. **Criteria for Evaluation.**
    a. Program 1 code to spec: 15 points
    b. Program 2 code to spec: 15 points
    c. Screen shots of runs 5 points
    d. Discussion 5 points