Programming Project 3

CS4379: Parallel and Concurrent Programming
CS 5379: Parallel Processing
Spring 2014

Instructor: Yong Chen **Office:** EC 211-I **e-mail:** yong.chen@ttu.edu
Instructor Office Hours: 10 a.m. – 11 a.m., TR, or by appointment
TA Name: Yin Lu **Office:** 201B **e-mail:** yin.lu@ttu.edu
TA Office Hours: Wed./Fri. 1-2:30

Due 5/2 (Fri.), 5 p.m. Please submit a soft copy on Blackboard. Late submissions are accepted till 5/6 (Tue.), 5 p.m. with 10% penalty each day. No submissions accepted after 5/6 (Tue.), 5 p.m.

MPI programming

Note: please see lecture 23 slides for the instruction to compile and run MPI programs on DISCFarm cluster. Please find sample MPI codes posted on the Blackboard as well.

Q1. Please complete the MPI Program 6.3 on page 258 of the textbook. Please report running times for 1, 2, 4, 8, 16, and 32 processes on DISCFarm cluster (use up to 4 nodes). Evaluate the performance of the program for a given matrix size (2000*2000) on the cluster. USE THE MPI TIMERS FOR PORTABILITY, i.e. the MPI_Wtime() calls even though this measures elapsed times and not CPU times.

Q2. In this problem you are asked to write a parallel program using MPI for solving a set of dense linear equations of the form A*x=b, where A is an n*n matrix and b is a vector. You will use Gaussian elimination without pivoting. You had written a shared memory parallel program using Pthread and OpenMP. Now you need to write the program with a message passing distributed memory version using MPI.

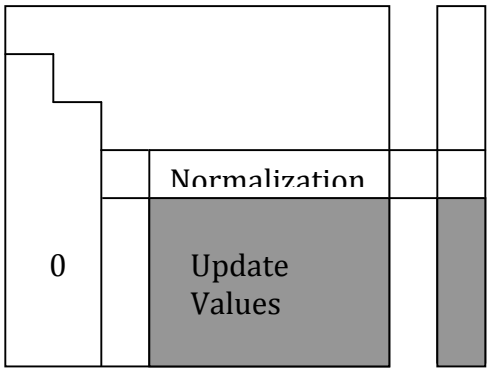
Assume that the data for the matrix A and the right hand-side vector x is available in process 0. You can generate the data randomly, but note that you will need to distribute the data to all other processes. Finally the results will have to be collected into process 0 which will print the final results.

The algorithm has two parts:

- *Gaussian Elimination*: the original system of equation is reduced to an upper triangular form Ux=y, where U is a matrix of size N*N in which all elements below the diagonal are zeros which are the modified values of the A matrix, and the diagonal elements have the value 1. The vector y is the modified version of the b vector when you do the updates on the matrix and the vector in the Gaussian elimination stage.

- *Back Substitution*: the new system of equations is solved to obtain the values of x.

The Gaussian elimination stage of the algorithm comprises (N-1) steps. In the algorithm, the ith step eliminates nonzero subdiagonal elements in column i by subtracting the ith row from row j in the range [i+1,n], in each case scaling the ith row by the factor A_{ji}/ A_{ii} so as to make the element A_{ji} zero. See figure below:



Hence the algorithm sweeps down the matrix from the top corner to the bottom right corner, leaving subdiagonal elements behind it.

(**Part a**) Write a parallel program using MPI point-to-point communication routines (you can still use the MPI_Barrier() collective routine).

(**Part b**) Write a parallel program using MPI collective communication routines (whenever possible).

(**Part c**) Report running times for 1, 2, 4, 8, 16, and 32 processes on DISCFarm cluster for Part a and Part b respectively. Evaluate the performance of the program for a given matrix size (2000*2000) on the cluster. USE THE MPI TIMERS FOR PORTABILITY, i.e. the MPI_Wtime() calls even though this measures elapsed times and not CPU times.

Suggestions:

- The whole point of parallel programming is performance, so you need to consider carefully for the efficiency of your algorithm and program.
- You can consider adopting advanced MPI features, such as creating a derived datatype to be used in the communication routines and using non-blocking communication routines for better efficiency.
- Consider carefully the data dependencies in Gaussian elimination, and the order in which tasks may be distributed.
- Gaussian elimination involves $O(n^3)$ operations. The back substitution stage requires only $O(n^2)$ operations, so you are not expected to parallelize back substitution.
- The algorithm should scale, assuming N is much larger than the number of processors.
- There should be clear comments in the code explaining your program.

Note on cheating: We have zero-tolerance policy for cheating. Working in groups is fine for discussing approaches and techniques. Copying problem solutions or code is cheating. Both the person copying and the person giving the answers will be equally penalized. Make sure you do your own work.