# Programming Project #4

CS4379: Parallel and Concurrent Programming
CS5379: Parallel Processing
Spring 2020

~~Due 5/12, 11:59 p.m. (hard deadline)~~ **Due 5/12, noon 12 p.m. (hard deadline) Please submit a soft copy on Blackboard. No submissions accepted after that.**

**Please submit a single tarball/zipped file containing all your source codes and documents. Please name your submission file starting as "LastName_FirstName_PP4".**

***Please note that we may request a 5-10 mins quick demo for grading.***

## MPI programming

**Source code samples:** Please checkout source code samples with executing the following command on the HPCC cluster:

```
git clone https://discl.cs.ttu.edu/gitlab/yongchen/cs4379cs5379.git
```

If you have already checked out a copy of the repo earlier, you can run the following command to update to the latest source code repo:

```
git pull
```

**Q1.** Please compile and run the provided *mpi_matrixmul.c* code with matrix size 2000*2000, and report run times for 1, 2, 4, 8, 16, 32, and 64 processes on HPCC cluster using 2 nodes. Please plot a chart to report the results.

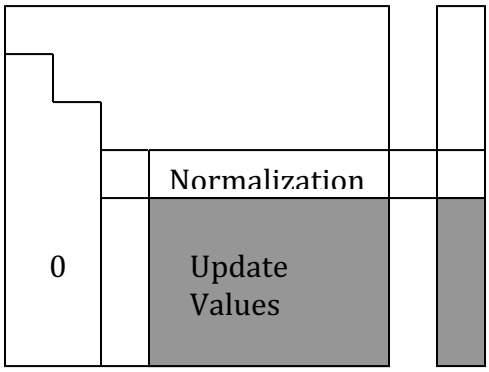
**Q2.** In this problem you are asked to write a parallel program using MPI for solving a set of dense linear equations of the form A*x=b, where A is an n*n matrix and b is a vector. You will use Gaussian elimination without pivoting. You had written a shared address space parallel program using Pthread and OpenMP. Now you need to write the program with a message passing distributed address space 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**) Please write a parallel program using MPI point-to-point communication routines (you can still use the MPI_Barrier() collective routine).

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

(**Part c**) Please develop a Makefile to automate the compilation and please report run times for 1, 2, 4, 8, 16, 32, and 64 processes on HPCC cluster using 2 nodes for Part a and Part b, respectively, with matrix size 2000*2000. Please use MPI timers for portability, i.e. the MPI_Wtime() calls.

Suggestions:

- You can consider using 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 in your MPI program.
- 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.

- You may observe performance slowdown because of the synchronization overhead among processes.

**Grading Criteria:**

***Please note that we may request a 5-10 mins quick demo for grading.***

| Q1 | Percentage % | Criteria |
|---|---|---|
| 20% | 100 | Successfully carry out all specified test cases and produce a plot to report the results. |
| **Q2** | **Percentage %** | **Criteria** |
| 80% | 10 | Inline comments to briefly describe your code |
| | 50 | Correct use of MPI APIs and library functions to complete the program. Correct Makefile (at least automate compilation and cleanup). |
| | 20 | Correct parallelization and correct results. |
| | 10 | Successfully carry out specified test cases |
| | 10 | A brief document to report plotted results, any other solutions you have tried, and your findings in general. |

**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.**

**Reference Materials:**

- MPI**:**
  - MPI tutorials, https://computing.llnl.gov/tutorials/mpi/
  - MPI 3.1 Specification, https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf

- Makefile**:**
  - http://www.gnu.org/software/make/manual/make.pdf or
  - http://www.gnu.org/software/make/manual/html_node/index.html