CS546 "Parallel and Distributed Processing"
Homework 2 - Programming

Submission:
Due by 05:00pm of 09/26/2016
This is an individual assignment!
Total points 100 - Late penalty: 10% penalty for each day late
Please upload your assignment on Blackboard with the following name:
CS546_SectionNumber_LastName_FirstName_HW2.zip.
Please do NOT email your assignment to the instructor and/or TA!

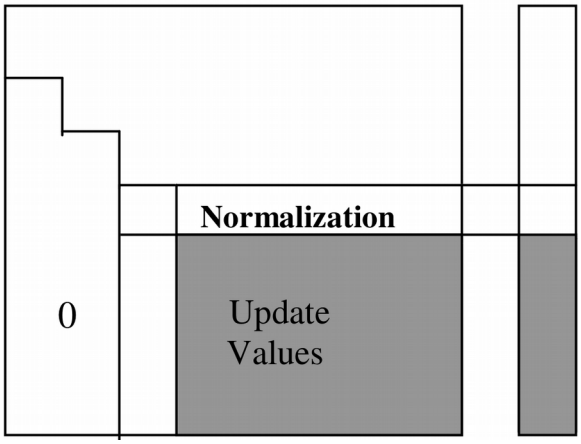
In this assignment, you will implement two parallel algorithms (one in Pthreads and the other in MPI) for solving a dense linear equations of the form A*x=b, where A is an n*n matrix and b is a vector. You will use Gaussian elimination without pivoting. You will be provided with a serial solution which you will modify to parallelize it. You should optimize your code to run as fast as possible. Note that the TA may run your program on other inputs that are not made available to you.

1. Background

The algorithm has two parts:

(a) *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.

(b) *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 sub-diagonal 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 the figure.

Hence the algorithm sweeps down the matrix from the top left corner to the bottom right corner, leaving sub-diagonal elements behind it.

2. Running Code

Your program should accept 3 command line parameters.

- `command_line_param1`
  Matrix size.
- `command_line_param2`
  Random seed.
- `command_line_param3`
  Output file name

3. Details

As usual, you may develop your code on any platform, but we will use Jarvis cluster to grade your solutions, so be sure that what you send us works on this machine.

Important: Your solution should include code that reports the elapsed time for your solution. It must also write all output to a file which you will also submit with your code for us to check the correctness. Given code does NOT do this. You should modify it.

Note: The whole point of parallel programming is performance, so you will be graded partially on the efficiency of your algorithm. Therefore, once you achieve a correct solution, you should perform some experimentation with your programs to try to optimize its performance. You should hand in two working and documented programs. You should provide a basic correctness argument for your solution (arguing that the relevant dependencies and other issues are taken care of by proper synchronization). In addition, you should describe some of the different versions of your program that you tried, what performance results you got out of them, and why you think your current version is reasonably efficient (or what more you could do to make it more efficient).

Suggestions: 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.

4. What to turn in

Your solution should be submitted as a zip file and should include:

- A written report of the assignment in either plain text(Doc) or PDF format. This is your chance to explain your approach, your optimizations, any insights gained, problems encountered, etc. Your report should include performance results for your solution on both your own machine and/or Jarvis.
- Your source code, instructions to build it on Jarvis, and instructions to run the programs (along with the arguments, etc).
- The output file with the timings of your testing. It should be consistent with the report.

5. Grading

Out of 100 possible points we split the points as follows:

Report: 40 points

- 10 points readability, use of English, organization
- 10 points for clarity of plots / figures
- 20 points for performance evaluation and analysis of results

Source code: 60 points

- 10 points for readability and cleanness of code
- 10 points for given instructions and in-code comments
- 30 points for correctness (if it is not even compiling you lose all points)
- 10 points for performance and possible optimizations you have done

Note: We encourage collaboration between you and your classmates. Discuss various approaches and techniques to come up with your solution. However, we do NOT allow copying code either from your classmates or from online sources. This is considered as cheating and falls under IIT code of honor. Penalties will be enforced. Please make sure you write your own solution. Remember this is an individual assignment! Also, make sure your code can't be copied by other students. To do that, you can change every file permission with chmod 700 [file], or you can protect your whole directory with chmod 700 /hw2/username.
GOOD LUCK!