CS546 “Parallel and Distributed Processing”

Program Assignment 2

Submission:
Due by 11:59pm of 10/8/2019
Total points 100 - Late penalty: 10% penalty for each day late
Please upload your assignment with the following name:
CS546_SectionNumber_LastName_FirstName_PA2.
Please do NOT email your assignment to the instructor and/or TA!

Problem Statement:
Use OpenMP application program interface to parallelize a Jacobi iteration. Compare your solution with serial version of the algorithm and show how your code scales with increasing number of threads. Additionally, compare the answer of the parallel code to that from the serial code and, if they differ, explain why that might legitimately be the case.
Details:
Jacobi method is an algorithm for determining the solutions of a diagonally dominant system of linear equations. Each diagonal element is solved for, and an approximate value is plugged in. The process is then iterated until it converges. This algorithm is a stripped-down version of the Jacobi transformation method of matrix diagonalization.
For example

Let

$$A\mathbf{x} = \mathbf{b}$$

be a square system of *n* linear equations, where:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

Then *A* can be decomposed into a diagonal component *D*, and the remainder *R*:

$$A = D + R \qquad \text{where} \qquad D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} \text{ and } R = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}.$$

The solution is then obtained iteratively via

$$\mathbf{x}^{(k+1)} = D^{-1}(\mathbf{b} - R\mathbf{x}^{(k)}),$$

where $\mathbf{x}^{(k)}$ is the *k*th approximation or iteration of $\mathbf{x}$ and $\mathbf{x}^{(k+1)}$ is the next or *k* + 1 iteration of $\mathbf{x}$. The element-based formula is thus:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n.$$

Running Code:

- The whole package should compile and run as is.

What to turn in:

- 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, which includes running with different numbers of threads for the same problem size, as well as running with the same number of threads for different problem sizes (i.e., both strong and weak scaling)

- Your source code and instructions to build it along with instructions to run the program.
- Output of the file with timings of your testing. It should be consistent with your report.

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 better understand the questions. However, we do NOT allow copying solutions or code. This is considered as cheating and falls under IIT code of honor. Penalties will be enforced. Please make sure you write your own solutions. GOOD LUCK!