CS546 Parallel and Distributed Processing

Program Assignment 3

Submission:
Due by 11:59pm of 10/22/2019
Total points 130 - Late penalty: 10% penalty for each day late
Please upload your assignment on Blackboard with the following name:
CS546_SectionNumber_LastName_FirstName_PA3.
Please do NOT email your assignment to the instructor and/or TA!

Problem Statement:

1. Use MPI application program interface to implement the PPT algorithm.
2. (optional, additional 30 points) Use MPI application program interface to implement the PDD algorithm.

Details:

A Partition Method for Parallel Processing:

A tridiagonal system is a linear system of equations

$$Ax = d, \tag{1}$$

where $x = (x_1, \cdots, x_n)^T$ and $d = (d_1 \cdots, d_n)^T$ are n-dimensional vectors, and A is a tridiagonal matrix with order n:

$$A = \begin{bmatrix} b_0 & c_0 & & & & \\ a_1 & b_1 & c_1 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & a_{n-2} & b_{n-2} & c_{n-2} \\ & & & & a_{n-1} & b_{n-1} \end{bmatrix} = [a_i, b_i, c_i] \tag{2}$$

To solve Eq. (1) efficiently on parallel computers, we partition A into submatrices. For convenience we assume that $n = p \cdot m$, where p is the number of processors available. The matrix A in Eq. (1) can be written as

$$A = \tilde{A} + \Delta A,$$

where $\tilde{A}$ is a block diagonal matrix with diagonal submatrices $A_i (i = 0, \cdots, p-1)$. The submatrices $A_i (i = 0, \cdots, p-1)$ are $m \times m$ tridiagonal matrices. Let e_i be a column vector with its ith $(0 \leq i \leq n-1)$ element being one and all the other entries being zero. We have

$$\Delta A = [a_m e_m, c_{m-1} e_{m-1}, a_{2m} e_{2m}, c_{2m-1} e_{2m-1}, \cdots, c_{(p-1)m-1} e_{(p-1)m-1}] \begin{bmatrix} e_{m-1}^T \\ e_m^T \\ . \\ . \\ e_{(p-1)m-1}^T \\ e_{(p-1)m}^T \end{bmatrix} = V E^T,$$

where both V and E are $n \times 2(p-1)$ matrices. Thus, we have

$$A = \tilde{A} + V E^T.$$

Based on the matrix modification formula originally defined by Sherman and Morrison [8] for rank-one changes, and assuming that all A_i's are invertible, Eq. (1) can be solved by

$$x = A^{-1} d = (\tilde{A} + V E^T)^{-1} d \tag{3}$$

$$x = \tilde{A}^{-1} d - \tilde{A}^{-1} V (I + E^T \tilde{A}^{-1} V)^{-1} E^T \tilde{A}^{-1} d. \tag{4}$$

Let

$$\tilde{A}\tilde{x} = d \tag{5}$$

$$\tilde{A} Y = V \tag{6}$$

$$h = E^T \tilde{x} \tag{7}$$

$$Z = I + E^T Y \tag{8}$$

$$Z y = h \tag{9}$$

$$\Delta x = Y y. \tag{10}$$

Equation (4) becomes

$$x = \tilde{x} - \Delta x. \tag{11}$$

In Eqs. (5) and (6), $\tilde{x}$ and Y are solved by the LU decomposition method. By the structure of $\tilde{A}$ and V, this is equivalent to solving

$$A_i[\tilde{x}^{(i)}, v^{(i)}, w^{(i)}] = [d^{(i)}, a_{im} e_0, c_{(i+1)m-1} e_{m-1}], \tag{12}$$

The Parallel Partition LU(PPT) Algorithm

Based on the matrix partitioning technique described previously, using p processors, the PPT algorithm to solve (1) consists of the following steps:

Step 1. Allocate $A_i, d^{(i)}$ and elements $a_{im}, c_{(i+1)m-1}$ to the *ith* node, where $0 \le i \le p-1$.

Step 2. Use the *LU* decomposition method to solve

$$A_i[\tilde{x}^{(i)}, v^{(i)}, w^{(i)}] = [d^{(i)}, a_{im}e_0, c_{(i+1)m-1}e_{m-1}]$$

Step 3. Send $\tilde{x}_0^{(i)}, \tilde{x}_{m-1}^{(i)}, v_0^{(i)}, v_{m-1}^{(i)}, w_0^{(i)}, w_{m-1}^{(i)}$ from the *ith* node to the other nodes $0 \le i \le p-1$.

Step 4. Use the LU method to solve Zy = h on all nodes

Step 5. Compute in parallel on p processors

$$\Delta x^{(i)} = [v^{(i)}, w^{(i)}] \begin{bmatrix} y_{2i-1} \\ y_{2i} \end{bmatrix}$$

$$x^{(i)} = \tilde{x}^{(i)} - \Delta x^{(i)}$$

The Parallel Diagonal Dominant (PDD) Algorithm

Using the PDD algorithm consists of the following steps:

Step 1. Allocate $A_i, d^{(i)}$, and elements $a_{im}, c_{(i+1)m-1}$ to the ith node, where $0 \le i \le p-1$.

Step 2. Solve (12). All computations can be executed in parallel on p processors.

Step 3. Send $\tilde{x}_0^{(i)}, v_0^{(i)}$ from the ith node to the $(i-1)$th node, for $i = 1, \cdots, p-1$.

Step 4. Solve

$$\begin{bmatrix} 1 & w_{m-1}^{(i)} \\ v_0^{(i+1)} & 1 \end{bmatrix} \begin{pmatrix} y_{2i} \\ y_{2i+1} \end{pmatrix} = \begin{pmatrix} \tilde{x}_{m-1}^{(i)} \\ \tilde{x}_0^{(i+1)} \end{pmatrix}$$

in parallel on the ith node for $0 \le i \le p-2$. Then send y_{2i} from the ith node to the $(i+1)$th node, for $i = 0, \cdots, p-2$.

Step 5. Compute (10) and (11). We have

$$\Delta x^{(i)} = [v^{(i)}, w^{(i)}] \begin{pmatrix} y_{(2i-1)} \\ y_{2i} \end{pmatrix}$$

$$x^{(i)} = \tilde{x}^{(i)} - \Delta x^{(i)}$$

Testing case:
The testing data we provided are the a.csv, b.csv, c.csv and d.csv, please use these matrixes as your input and save your output (the value is rounded to three decimal places) as .csv file.
Based on Eq. (2), the matrix A is $[a_i, b_i, c_i]$. For the input matrix A= $[a_i, b_i, c_i]$, where $0 \leq i < 4096$, where $a_i = 1$, for $0 < i < 4096$, $c_i = 1$, $0 \leq i < 4095$, and $b_i = -2$, for $0 \leq i < 4096$. You can load these three no-zero vectors from a.csv, b.csv and c.csv. In addition, please create at least 2 test cases to prove the correctness of your code. You must submit all the inputs and outputs you use.

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 any insights gained, problems encountered, results, etc.
- At least three sets of inputs and outputs (.csv files), to prove your implementation is correct.
- Your source code and instructions to build it along with instructions to run the program.

Grading:

- Out of 100 possible points we split the points as follows:
  - Report: 40 points
    - 10 points readability, use of English, organization
    - 30 points for approach, results 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 possible optimizations you have done

Hint: Please check the provided implementations of the sequential tridiagonal matrix algorithm: https://en.wikibooks.org/wiki/Algorithm_Implementation/Linear_Algebra/Tridiagonal_matrix_algorithm

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!