CS51501 Spring 2016
Homework set # 2

Part (a): due at 11:59 pm on Monday Feb. 8, 2016
Part (b): due at 11:59 pm on Wednesday Feb. 17, 2016

Part (a)

Implement Algorithm 3.3 (BBTS: Block banded triangular solver)[1] on Page 60 in the textbook[2] to solve $\boldsymbol{L}\mathbf{x} = \mathbf{b}$, where $\boldsymbol{L}$ is a banded lower triangular matrix with m subdiagonals.

1. Your code should be implemented using OpenMP and be tested on one node of the MC cluster.
2. In step 3 of BBTS, solve $\boldsymbol{L}_i(\boldsymbol{G}_i^{(0)}, \mathbf{f}_i^{(0)}) = (\boldsymbol{R}_i, \mathbf{f}_i)$ using the Column-sweep method in Algorithm 3.1(CSweep).

Download L.mtx and b.mtx on the course web page. Use your code to solve $\boldsymbol{L}\mathbf{x} = \mathbf{b}$, and store the solution $\mathbf{x}$ to x.mtx. The order of $\boldsymbol{L}$ is $n = 8192$, with $m = 128$.

TA will test your code as follows: ./bbts A.mtx b.mtx x.mtx.

Part (b)

Implement a SPIKE-like algorithm to solve $\hat{\boldsymbol{L}}\mathbf{y} = \mathbf{f}$, where $\hat{\boldsymbol{L}}$ is a banded lower triangular matrix of order $N = 32,768 = 2^{15}$, with $t = 128$ subdiagonals on a cluster of multicore nodes. Your code should be implemented using the MPI/OpenMP programming paradigm, and be tested on four nodes of the MC cluster. The SPIKE-like algorithm should be implemented using MPI. In this algorithm, use BBTS (from Part (a)) to solve the system involvings $\hat{\boldsymbol{L}}_i$.

1. Partition $\hat{\boldsymbol{L}}$ and $\mathbf{f}$ into 4 parts, with one partition in each node, via MPI_Scatter(). Therefor, node i should contain $\hat{\boldsymbol{L}}_i$ and $\boldsymbol{R}_i$, $i = 1:4$. Note that

[1] Correction(BBTS):
step 2: doall 2: n/m
step 3: solve $\boldsymbol{L}_i(\boldsymbol{G}_{i-1}^{(0)}, \mathbf{f}_i^{(0)}) = (\boldsymbol{R}_{i-1}, \mathbf{f}_i)$

[2] Gallopoulos, Efstratios, Bernard Philippe, and Ahmed H. Sameh. Parallelism in Matrix Computations. Springer, 2015.

$\hat{\boldsymbol{L}}_i$ is the banded lower triangular matrix shown below, with $\boldsymbol{R}_i$ being an upper triangular matrix.

$$\left[\begin{array}{c|c|c|c} \hat{\boldsymbol{L}}_1 & & & \\ \hline \begin{array}{c|c} & \boldsymbol{R}_2 \\ & \mathbf{0} \end{array} & \hat{\boldsymbol{L}}_2 & & \\ \hline & \begin{array}{c|c} & \boldsymbol{R}_3 \\ & \mathbf{0} \end{array} & \hat{\boldsymbol{L}}_3 & \\ \hline & & \begin{array}{c|c} & \boldsymbol{R}_4 \\ & \mathbf{0} \end{array} & \hat{\boldsymbol{L}}_4 \end{array}\right] \left[\begin{array}{c} \mathbf{y}_1 \\ \hline \mathbf{y}_2 \\ \hline \mathbf{y}_3 \\ \hline \mathbf{y}_4 \end{array}\right] = \left[\begin{array}{c} \mathbf{f}_1 \\ \hline \mathbf{f}_2 \\ \hline \mathbf{f}_3 \\ \hline \mathbf{f}_4 \end{array}\right].$$

2. Solve

$$\begin{aligned} \boldsymbol{L}_1 \begin{pmatrix} \mathbf{g}_1 \\ \mathbf{h}_1 \end{pmatrix} &= \mathbf{f}_1, \\ \boldsymbol{L}_i \left[\begin{pmatrix} \mathbf{g}_i \\ \mathbf{h}_i \end{pmatrix}, \begin{pmatrix} \boldsymbol{M}_i \\ \boldsymbol{N}_i \end{pmatrix}\right] &= \left[\mathbf{f}_i, \begin{pmatrix} \boldsymbol{R}_i \\ \mathbf{0} \end{pmatrix}\right], \end{aligned}$$

where i=2:4, via BBTS (from Part (a)). Note that the square matrices $\boldsymbol{N}_i$ are of order t, and the vector $\mathbf{h}_i$ are of order t.

Therefor, as you have seen in the lecture, the resulting system is of the form

$$\left[\begin{array}{c|c|c|c} \boldsymbol{I} & & & \\ \hline \begin{array}{c|c} & \boldsymbol{M}_2 \\ & \boldsymbol{N}_2 \end{array} & \boldsymbol{I} & & \\ \hline & \begin{array}{c|c} & \boldsymbol{M}_3 \\ & \boldsymbol{N}_3 \end{array} & \boldsymbol{I} & \\ \hline & & \begin{array}{c|c} & \boldsymbol{M}_4 \\ & \boldsymbol{N}_4 \end{array} & \boldsymbol{I} \end{array}\right] \left[\begin{array}{c} \mathbf{y}_1 \\ \hline \mathbf{y}_2 \\ \hline \mathbf{y}_3 \\ \hline \mathbf{y}_4 \end{array}\right] = \left[\begin{array}{c} \mathbf{g}_1 \\ \mathbf{h}_1 \\ \hline \mathbf{g}_2 \\ \mathbf{h}_2 \\ \hline \mathbf{g}_3 \\ \mathbf{h}_3 \\ \hline \mathbf{g}_4 \\ \mathbf{h}_4 \end{array}\right].$$

3. In node 1, gather $\boldsymbol{N}_i$ and $\mathbf{h}_i$ from node i, $i = 2, 3$ and 4 via MPI_Gather(). Solve the reduced system

$$\begin{bmatrix} \boldsymbol{I} & & & \\ \boldsymbol{N}_2 & \boldsymbol{I} & & \\ & \boldsymbol{N}_3 & \boldsymbol{I} & \\ & & \boldsymbol{N}_4 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \\ \mathbf{h}_4 \end{bmatrix}$$

via CSweep.

4. Send $\mathbf{u}_1$ to node 2, $\mathbf{u}_2$ to node 3, $\mathbf{u}_3$ to node 4, and simultaneously compute

$$\begin{aligned} \mathbf{y}_1 &= \begin{pmatrix} \mathbf{g}_1 \\ \mathbf{h}_1 \end{pmatrix}, \\ \mathbf{y}_i &= \begin{pmatrix} \mathbf{g}_i \\ \mathbf{h}_i \end{pmatrix} - \begin{pmatrix} \boldsymbol{M}_i \\ \boldsymbol{N}_i \end{pmatrix} \mathbf{u}_{i-1}, \end{aligned}$$

where i=2:4

5. Gather $\mathbf{y}_i$ via MPI_Gather() and write it to y.mtx.

Download L_hat.mtx and f.mtx on the course web page. Use your code to solve $\hat{\boldsymbol{L}}\mathbf{y} = \mathbf{f}$, and store the solution $\mathbf{y}$ to y.mtx.

TA will test your code as follows:

mpirun -np 4 -hostfile nodes ./spike_like L_hat.mtx f.mtx y.mtx

Storage

1. vectors will be stored as dense in mtx format. Reading **b** and **f** is the same as HW1.

2. Storing a banded lower triangular matrix in dense mtx format will consume a lot of memory needlessly. So TA stores it as a sparse matrix in coordinate format. After you read the matrix from the mtx file, you should store it properly. See band storage on https://software.intel.com/en-us/node/520871. Please do not store the matrix as a 2-dimensional array of order n.

The Intel compiler

From HW2, you need to use the Intel compiler(i.e. icc, mpiicc, ifort, mpiifort) to compile your code. In later HW sets, you will learn how to use the Intel MKL library.

0.1 Set up environment for the Intel compiler

TA prepared a defs file to set up the environment for the Intel compiler.

1. Download mc.defs on the course web page and put this file in your home directory.

2. After you acces mcxx.cs.purdue.edu, type 'source mc.defs'.

Then you could use icc, mpiicc, ifort and mpiifort.

0.2 Commands to compile the OpenMP code

C/C++: icc -openmp test.c -o test

FORTRAN: ifort -openmp test.f -o test

Note that the compile-time flag -fopenmp is changed to -openmp.

Submission

Part (a): turnin –c cs51501 –p HW2a your_folder_name

Part (b): turnin –c cs51501 –p HW2b your_folder_name

Your submission should include the following files:

1. The source code.
2. A Readme file or a Makefile, which includes all the compiling commands.

Note that the submission of Part (b) should include the BBTS function from Part (a).

Please **do not** include the test cases in your submission, which would be too large to submit.