# CSCE 659 Fall 2017

## HW 2: Jacobi Iterative Solver for 2D Laplace Equation

Due: 11:59pm Tuesday, October 10, 2017

1. (70 points) You are required to develop an MPI-based parallel implementation of the Jacobi iterative solver for solving the Laplace equation on a unit square. Compute the solution to the Laplace problem for a uniform mesh of size (N+2)×(M+2) with the following boundary conditions on the solution:

$$u_{0,1:M} = u_{N+1,1:M} = 0 \qquad \text{and} \qquad u_{1:N,0} = u_{1:N,M+1} = 1$$

At the k[th] iteration, the algorithm updates the solution at each grid point using the solution at the preceding iteration:

$$u_{i,j}^{k} = \frac{1}{4}\left[ u_{i+1,j}^{k-1} + u_{i-1,j}^{k-1} + u_{i,j+1}^{k-1} + u_{i,j-1}^{k-1} \right], \quad i = 1,\ldots,N, \quad j = 1,\ldots,M$$

The algorithm terminates when $\left| u_{i,j}^{k} \right| \le \varepsilon$ at all grid points, where $\varepsilon$ is a user-specified error tolerance.

You will be provided with the file `parallel_jacobi_2d_laplace_stripped.c` that has some code missing in places marked `// INSERT CODE HERE`. See illustration for an overview of the parallelization strategy. Once the required code has been inserted, the file can be compiled and executed to solve the problem using multiple processors. Note that even without any modification, the code compiles and produces correct results when using a single processor. This can be used to verify the correctness of your multiprocessor implementation. Your implementation should be able to partition the mesh into an $p_x \times p_y$ array of subgrids for arbitrary values of $p_x$ and $p_y$. Compute the solution to the Laplace problem for a mesh of size $1026 \times 1026$. Use $\varepsilon = 10^{-4}$ as the error tolerance.

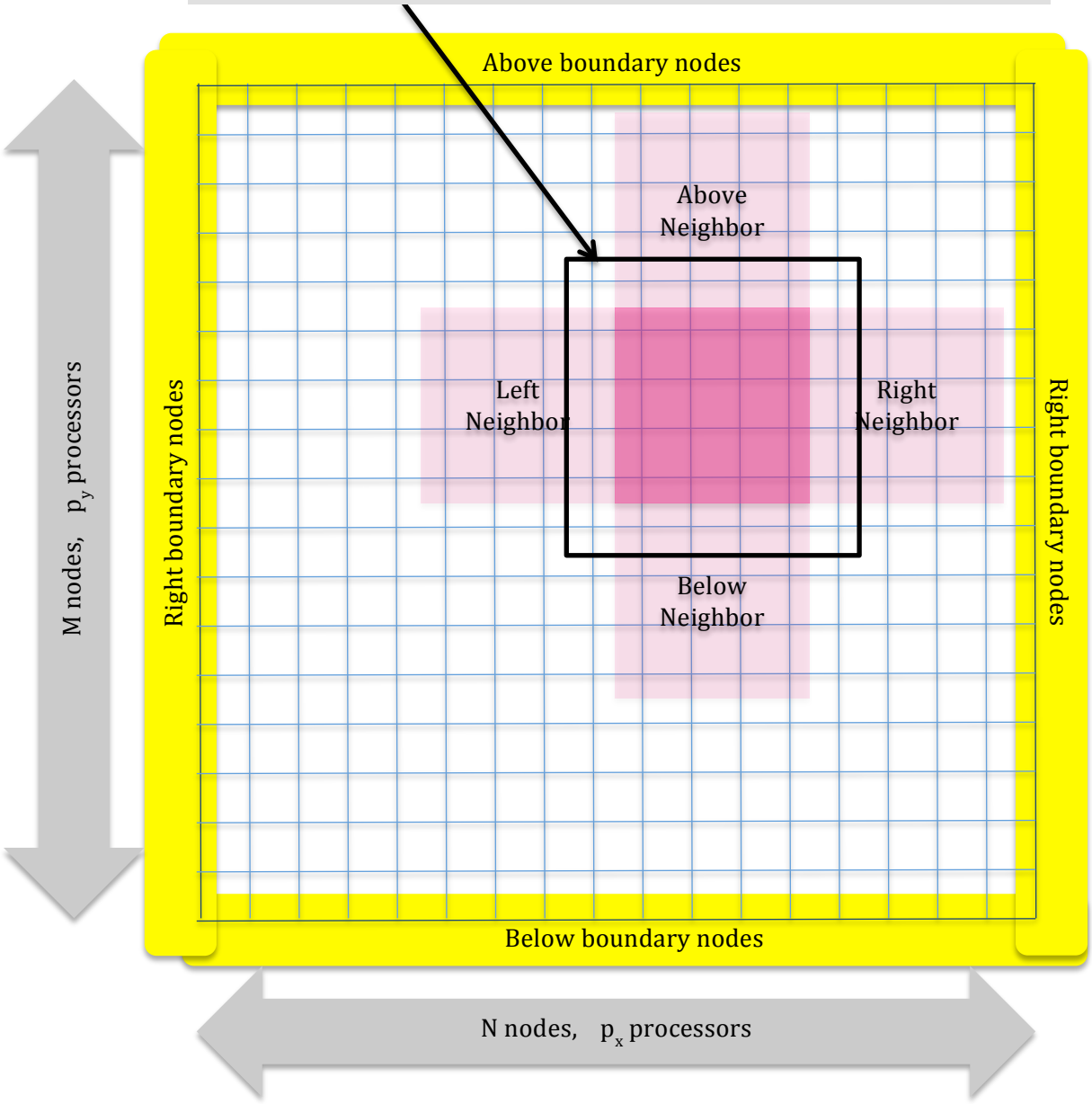
2. (15 points) Plot the parallel time taken by your code to solve the problem using p processes for p = 1, 2, 4, 8, and 16. Conduct two sets of experiments: the first one should use $p_y$=1 and the second one should use ($p_x$, $p_y$)= (1,1), (1,2), (2,2), (2,4), and (4,4), respectively. Estimate the speedup and efficiency of your code for each case. Comment on the experimental results. For these experiments, you may terminate the Jacobi iterations when the number of iterations exceeds 512.

3. (15 points) Plot the parallel time taken by your code to solve the problem on an (N+2)×(N+2) grid using 16 processes, where N = $2^t$, t = 8, 9, 10, 11. Conduct experiments with $p_y$=1 and the second one should use $p_y$=⌈√p⌉. Estimate the speedup and efficiency in each case as the problem size is changed. Comment on the experimental observations.

Process stores local mesh in U[0..n+1][0..m+1] where U[1..n][1..m] stores the unknowns assigned to this process.

- U[0][*] stores the left boundary,
- U[n+1][*] stores the right boundary
- U[*][0] stores the below boundary
- U[*][m+1] store the above boundary

These boundary values are initialized by values received from adjacent neighbor process or by domain boundary values if neighbor is absent




Parallelization Strategy

**Submission:** You need to upload the following to ecampus:

1. Submit the file `parallel_jacobi_2d_laplace_stripped.c.`
2. Submit a single PDF or MSWord document that includes the following.
   - A description of the changes made to the code, the parallel computer used, and how to compile and execute the code on the parallel computer
   - Responses to Problems 2 & 3.

**Helpful Information:**

1. Source file(s) are available on ecampus.
2. Load the Intel software stack prior to compiling and executing the code. Use:

   ```
   module load intel/2017A
   ```
3. Compile C programs using `mpiicc`. For example, to compile `code.c` to create the executable `code.exe`, use

   ```
   mpiicc –o code.exe code.c
   ```
4. The run time of a code should be measured when it is executed in dedicated mode. Create a batch file as described on hprc.tamu.edu and use the following command to submit it to the batch system:

   ```
   bsub < batch_file
   ```