Programming Assignment
MPI and CUDA implementations of the Jacobi method

OBJECTIVES

- Practice MPI and CUDA programming

DESCRIPTION

This project can be done by a group of two students. The Jacobi method is a commonly used iterative method to solve partial differential equations. It begins with an initial estimate for the solution and successively improves it until the solution is as accurate as desired. For example, applying the Jacobi method to solve the *Laplace equation* on the unit square

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

with boundary conditions shown in the following figure, we get

$$u_{i,j}^{(k+1)} = \frac{u_{i-1,j}^{(k)} + u_{i,j-1}^{(k)} + u_{i+1,j}^{(k)} + u_{i,j+1}^{(k)}}{4}.$$

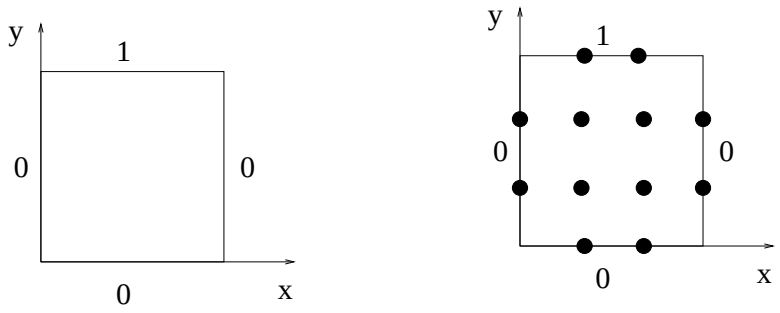

Figure 1: Boundary conditions (left) and mesh(right) for Laplace equation

In this project, you will be given a sequential program for a generalized Jacobi method, where

$$u_{i,j}^{(k+1)} = a_1 * u_{i-1,j}^{(k)} + a_2 * u_{i,j-1}^{(k)} + a_3 * u_{i+1,j}^{(k)} + a_4 * u_{i,j+1}^{(k)}.$$

The parameters a_1, a_2, a_3 and a_4, as well as three other parameters, a_5, a_6 and a_7, will be given in a file proj3.input, whose format is '`N a1 a2 a3 a4 a5 a6 a7`'. The parameters a_5 and a_6 specify the boundary condition. Assuming that the problem domain is given by Mesh[0..N+1][0..N+1], where Mesh[1..N][1..N] are the interior grid points. The boundary condition will be specified as $Mesh[i][0] = a_5 * i$, $Mesh[i][N+1] = a_6 * i$, $Mesh[0][i] = 0$, $Mesh[N+1][i] = 0$. The interior grid points should be initialized to be 0 at the beginning. The parameter a_7 specifies the error that is allowed in the solution. The program should perform

the Jacobi computation iteratively until $|u_{i,j}^{(k+1)} - u_{i,j}^{(k)}| < a_7$, for any i and j, in which case, the program should stop and write the following two vectors Mesh[N/2, 0..N+1] and Mesh[0..N+1, N/2] (as binary data) to file proj3.output. Proj3.output will contain Mesh[N/2, 0], Mesh[N/2, 1], ..., Mesh[N/2, N+1], Mesh[0, N/2], Mesh[1, N/2], ..., Mesh[N+1, N/2]. The data type for all the variables we discussed should be *float*.

Your task is to convert the sequential program to MPI and CUDA. Your program should work for any sized mesh and any number of processes in MPI.

DEADLINES AND MATERIALS TO BE HANDED IN

- **Feb. 13, 2013**. You should demonstrate this project and submit the whole project directory as a tar file to me. You also need to write a checker program compares the output files of the programs (or you can make sure that binary output files for different implementations (different number of threads, processes, etc) are exactly the same). In the directory, you should have a README file describing how to compile and run you program and the known bugs in your program.

GRADING POLICY

You must make sure that your programs meet the following conditions.

1. Your program must produce correct result for all cases.
2. Your MPI and CUDA program must achieve **a speedup of more than 2** on linprog and GPU.
3. In the MPI implementation, each MPI process should only store a fraction of the mesh (the array size should be roughly equal to $N \times N/P$, where P is the number of processes (10 points for this feature).
4. You are also responsible for showing that your programs produce correct results (via the checker program or the cmp utility).

The grading for each part of the project is as follows.

- Proper README file (10)
- Proper makefile (10)
- MPI implementation (35)
- CUDA implementation (35)
- demo, checker program, etc (10)