CSE494/598 Project 2

Milestone 1 due midnight 2/27, Milestone 2 due 3/17

This project will involve writing a parallel 2D wave equation solver.

The wave equation can be succinctly represented as

$$u_{tt} = c^2(u_{xx} + u_{yy})$$

where the constant *c* has units of velocity.

This equation can be more explicitly stated as:

$$\frac{\partial^2 u}{\partial t^2} = c^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

In computing, we can decompose this equation using central differences. Time (*l)*, x-position (*i)*, and y-position (*j)* are indices of *u.* Assuming *c*=1 (which you can do),

$$\frac{u_{i,j}^{l+1} - 2u_{i,j}^{l} + u_{i,j}^{l-1}}{\Delta t^2} = \frac{u_{i+1,j}^{l} - 2u_{i,j}^{l} + u_{i-1,j}^{l}}{\Delta x^2} + \frac{u_{i,j+1}^{l} - 2u_{i,j}^{l} + u_{i,j-1}^{l}}{\Delta y^2}$$

If we assume Δx=Δy, then define $r = \frac{\Delta t}{\Delta x}$, and update in time by rearranging terms:

$$u_{i,j}^{l+1} = 2u_{i,j}^{l} - u_{i,j}^{l-1} + r^2\left(u_{i+1,j}^{l} + u_{i-1,j}^{l} + u_{i,j+1}^{l} + u_{i,j-1}^{l} - 4u_{i,j}^{l}\right)$$

This equation is the heart of your program. The boundaries will be explicitly set, so these will not be updated by the equation above. Most of the time, the boundaries will be hard set to zero.

To get the calculation started, for (*l*=0 and *l*=1) you can use *u* matrices set to zero.

You need to make sure your solver will be stable. One way to do this is to check the Courant-Friedrichs-Lewy condition:

$$C = \frac{\Delta t}{\Delta x} + \frac{\Delta t}{\Delta y} < 1$$

Basically, the more you refine your domain in space, the more you have to decrease your time step. For this project, you can hard set Δx, Δy and Δt such that this condition is always met.

Finally, to make the simulation interesting, we want to perturb the system. The easiest way is to set some part of the boundary to a fixed value. For the pulses you will implement, just set the *u* values at a few cells on the edge to a nonzero value for a several time steps.

Milestone 1:

1. Write an MPI program that implements the wave equation on a 480X480 grid, propagating a single pulse from one side for 100 time steps. Output the data for the last time step to a text file with 3 columns (*x, y,* and *u*).
2. Provide preliminary performance numbers for this program on 1, 4 and 16 processors.
3. Give one paragraph describing your strategy for implementing the additional parts of the program (described below) using MPI.

Milestone 2:

1. Now have your program find the maximum amplitude (*u*) across the entire domain. When this maximum is found to be 20% of the height of the original pulse, launch another pulse from the boundary edge 90° clockwise. Run for a 2000 time steps and provide an output file for the last time step.
2. Insert OpenMP pragmas into the program (Make this another version so you can compare the two programs.) You can compile by using `mpicc` with the `–openmp` flag. Before you run the program, you must tell the compute nodes that you will be running MPI jobs between nodes with OpenMP running on the nodes. This is done by using the `–pernode` flag with `mpiexec`:
   `mpiexec –pernode ./yourmpiprogram`
   When –pernode is run, the compute nodes will automatically send all threads to one CPU per node. To override this you must type (before running mpiexec):
   `export KMP_AFFINITY=norespect,scatter`
   Now you will be running a hybrid code.
3. Prepare a final report comparing the two implementations for two dataset sizes (1024x1024 and 2048x2048) up to 256 processors. Present speedup plots (1, 4, 16, 64, 256), profiling information (describing where time is spent in the program) and jumpshot pictures. Discuss your communication patterns (data manipulation between neighbors, use of collectives), granularity, concurrency vs. communication overhead. Justify decisions you made in your code design. You do not need to include the output routine or run for all the time steps to get your performance data.