SE498 Parallel Computing
Lab 3: Using openMP to simulate heat transfer
Due: October 9, 2013

1. Objectives

- Explain how the Jacobi algorithm works to calculate future diffuse values
- Construct an openMP program which uses the Jacobi Algorithm to simulate heat transfer over a grid.
- Simulate the heat flow in a simple system
- Construct the fastest application to perform a given simulation.

2. Introduction

In physics, you learned about the basics of heat transfer. In short, heat always flows from hot to cold. We can restrict heat flow using insulation. Insulation has an R-value, which essentially indicates how much the material resists the flow of heat from one location to another.

In this lab, we are going to construct a program which simulates the propagation of heat throughout a building. The building is represented as a 2D grid of points. Each point has multiple attributes. The first attribute, t(x, y) represents the current temperature at the given point in degrees F. A larger value represents a higher temperature. The second attribute, r(x, y) represents the insulation factor for the given location. A larger value indicates that the given location does not conduct heat as well as a location with a smaller value. The third attribute e(x,y) indicates if there is an energy source at a given location which causes that location to have a fixed temperature. For example, a cell with a fire burning in it might have a fixed temperature of 1000, indicating that the given cell is at 1000 degrees due to the fire.

Figure 1 indicates a set of initial conditions as well as subsequent temperature changes over time. In all cells except for the center cell on the bottom, the initial temperature is 0 degrees. The bottom cell in the middle (red) has a temperature of 1000 degrees, indicting there is a fire at that location. Each cell has an RFactor of 1, indicating that the cell has approximately the same conductivity as air. The other colors (yellow and orange) indicate increases in the temperature of those cells as the heat spreads over time.

To calculate the value of t+1 for each location, the following formula is used:

$$t_{n+1}[x,y] = \frac{t_{n+1}[x,y] \times r[x,y] + t_{n[x-1,y]} + t_n[x,y-1] + t_n[x+1,y] \mp t_n[x,y+1]}{r[x,y]+4}$$

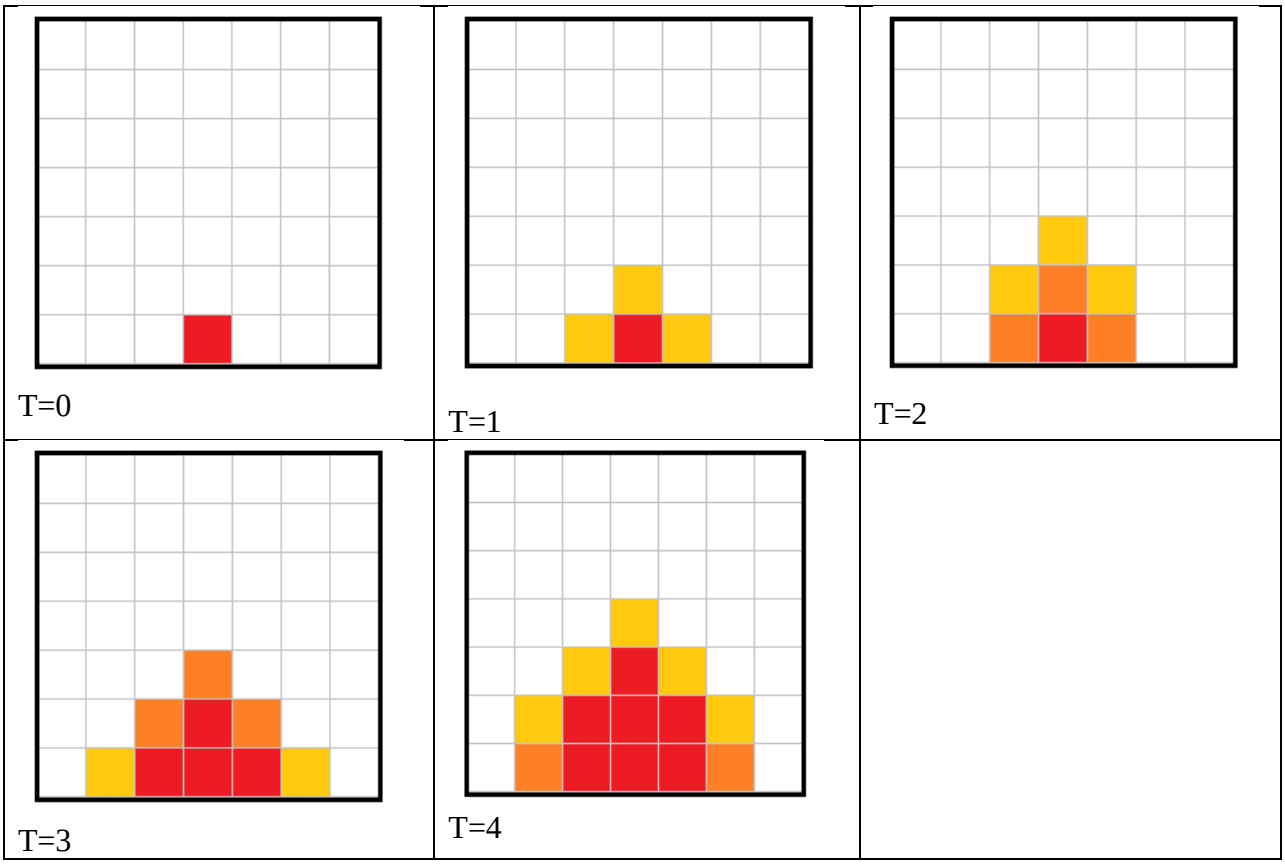

Figure 1 Graphical images showing the change of temperatures over time.

The program steps through the simulation and dumps the images to a file. After all the images have been rendered, a video is created showing all of the frames together.

The code, as it is currently structured, runs sequentially. There is no parallelism at all in the process. Your first task is to parallelize the code so that it uses openMP to run in parallel. Your goal is to attempt and achieve as much parallelism as is possible in the system, but to also make the system run as efficiently as possible. Once this is done, you are to use the example scenario provided and calculate the speedup and efficiency as the task is spread to additional processors.

3. Specific Steps

1. Download the source code from the website and understand its structure. It's written in C.
2. Add appropriate timing calls to the code so that you are able to time the execution of the program.
3. Parallelize the computations to run which compute the next cell
4. Parallelize the renderings of the image so that an image is rendered when the next computation is occurring.
5. Determine the speedup and performance gains for part 1, part 2, and part 1 and 2 combines together. Which overall is the most effective usage of parallelization?

4. Deliverables

Each lab team is responsible for submitting a report with the following

1. What did you learn by doing this lab?
2. What things went right and wrong when doing this lab?
3. What was the minimum processing time you were able to achieve when the code was parallelized, and what is the maximum speedup you were able to achieve when executing the model? Include a plot of your execution times, speedup, and efficiencies for the Glenn supercomputer.
4. If you re-run the program on the Oakley supercomputer, how does the execution compare with the Glenn supercomputer? Do the speedups exhibit the same general trend?
5. What conclusions can you draw from this experience?

5. Challenges

This lab has three challenges. The first challenge is to obtain the maximum speedup through parallelization. The second challenge is to maximize the performance so that the code executes in the minimum amount of time. The third challenge is to obtain the maximum efficiency as new processors are added.