EC526 PARALLELIZATION FRAMEWORK ANALYSIS

May 2, 2021

Boston University
Justin Hafling
Hans Walker

MEMBER CONTRIBUTIONS

Justin Hafling

- Proposal Presentation
- Final Presentation Slides 1-7
- Report
- MPI Code
- Serial Code
- Makefile

Hans Walker

- OpenACC Code

INTRODUCTION

The goal of this project was to specifically analyze the performance differences between OpenACC and MPI when running parallel threads on the CPU. It should be noted that running the OpenACC threads on the GPU would've run even faster, but this was done on the CPU for this analysis to provided a more consistent comparison with less confounding variables. This report will be analyzing the timings of running a Heat Flow analysis of different sizes comparing the 2 methods that have been mentioned. This report will reveal the differences in how the threads communicate and handle the division of labor when solving the identical problem. Specifically analyzing the rates of converges for grids of square grids of size 64, 128, and 256. The amounts of threads were also varied for each size: 4,8,16.

HEAT FLOW EQUATION

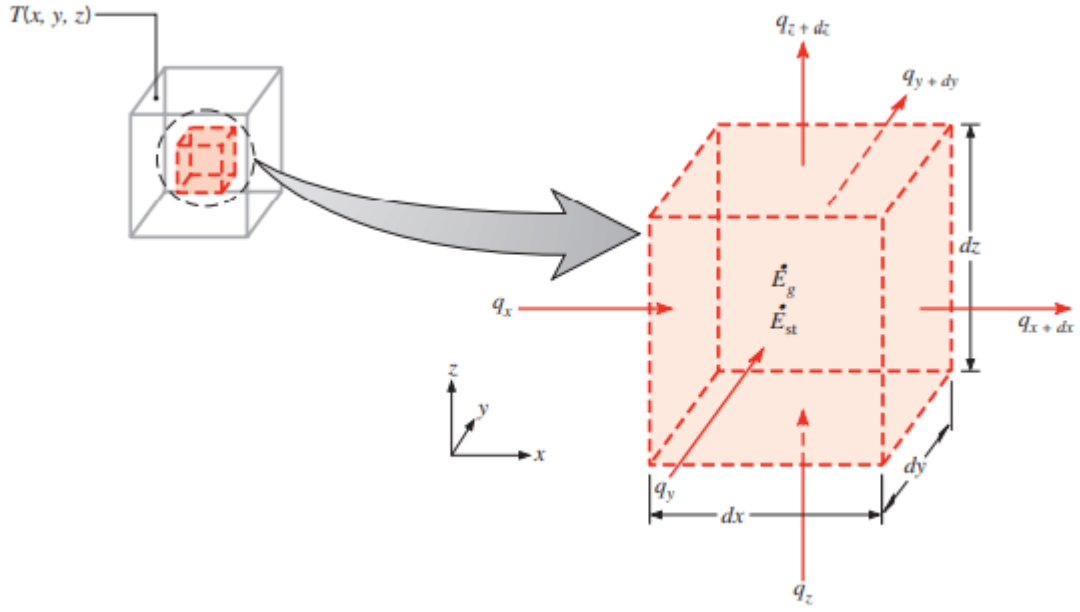
The scenario chosen to compare the two frameworks was the time dependent 2D-Heat Flow Equation. This equation is listed below **Equation 1**:

$$\frac{\partial}{\partial x}\left(k\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial T}{\partial y}\right)+q=\rho c_P\frac{\partial T}{\partial t} \tag{1}$$

In the above 2D-Heat Flow equation, T is the Temperature, t is time, k is the thermal conductivity of the element, ρ is the density, and c_P is the specific heat at constant pressure. This finite element. The thermal conductivity, density, and specific heat are material dependent properties that vary depending on what material is being analyzed. This project looked at materials such as Gold, Aluminum, Silver, Copper, and Diamond to note the times that they were able to converge within to confirm the phenomena and the accuracy of the equation being implemented. These constants were taken from Heat and Mass Transfer Bergman [1].

In order to implement this equation, more knowledge of heat transfer was needed in order to accurately convert the model to an iterative algorithm.

Figure 1: Finite Element Heat Transfer [1]



From the diagram below, the convection in the x,y,z (Note 2D was used for project but 3D provided better image) were easily calculated using Fourier's Law of Cooling defined by **Equation 2**.

$$q_x^{"} = -kA_x \frac{\partial T}{\partial x} = -kA_x \frac{\Delta T}{\Delta x} \tag{2}$$

The specific implementation of this function was shown below. Note that the x and y directions were swapped but fundamentally it is fine that as long as its consistent with going into or out of the element that was analyzed. Using a 2D Temperature Array, the following calculation was done to calculate the 2D Heat Flow Equation. Coordinates x,y represent the current element being analyzed.

The Heat Flow into the element in the X-Direction.

$$q_x^{"} = -k \frac{T[x][y-1] - T[x][y]}{dx} \tag{3}$$

The Heat Flow our of the element in the X-Direction.

$$q_{x+dx}^{"} = -k \frac{T[x][y] - T[x][y+1]}{dx} \tag{4}$$

The Heat Flow into the element in the Y-Direction.

$$q_y^{"} = -k\frac{T[x+1][y] - T[x][y]}{dy} \tag{5}$$

The Heat Flow out of the element in the Y-Direction.

$$q_{y+dy}^{"} = -k\frac{T[x][y] - T[x-1][y]}{dy} \tag{6}$$

The calculated change in temperature of the element.

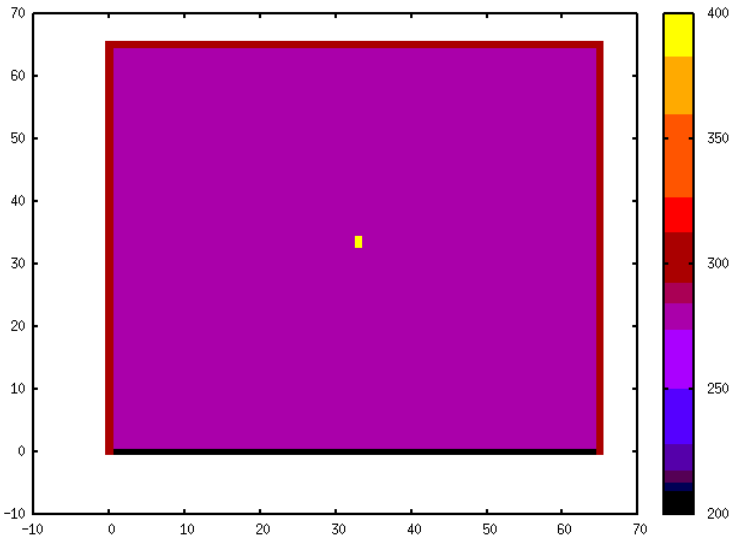
$$\Delta[x][y] = \left(q_x^{"} + q_y^{"} - q_{x+dx}^{"} - q_{y+dy}^{"} + q\right)/\rho c_P \tag{7}$$

Combining Equations 3-7 allowed the algorithm to be implemented as a parallel jacobi iteration. This allowed each thread to take a small chunk of the grid without worrying about needed to know the current value. Each iteration updated the overall temperature grid.

INITIAL CONDITIONS

Below were the initial conditions of the grids when testing for convergence. No heat was generated, but a pseudo source was used at the center which was kept at 400K. This was considered one of the boundary conditions of the grid that was used.

Figure 2: Initial Conditions of Temperature Heat Map.



Initial Boundary Conditions

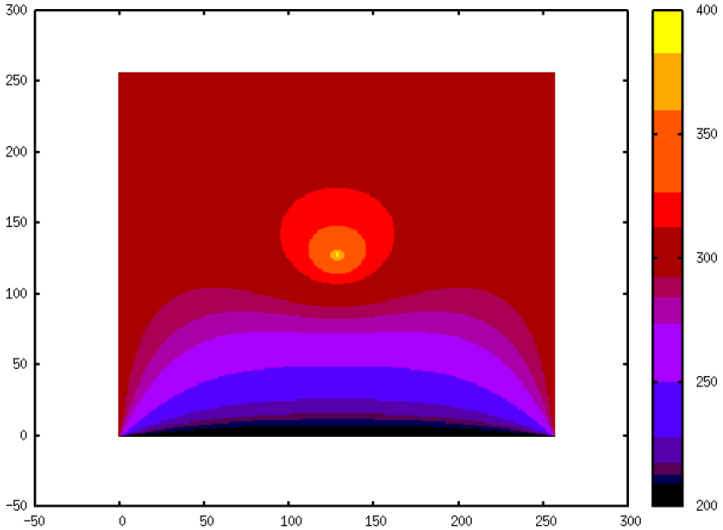
- Top Wall: 300K
- Side Walls: 300K
- Bottom Wall: 200K
- 2x2 Center Source: 400k
- Else: 273.15K

The above boundary conditions were run on all of the different grids and allowed for an observable and expected convergence.

CONVERGENCE/RESIDUAL

The algorithm was using a time-independent source so that convergence was comparable across the various threads and grid sizes. The algorithm was considered to converge when the total sum of the temperature differences calculated (How much the jacobi iteration wanted to change) varied by less than $1e^{-}10$ from one run to the next. This high precision residual was used in order to provide smooth output plots like shown below.

Figure 3: Converged 256x256 Grid of Aluminum.



CONCLUSION

The group found that the OpenACC implementation was able to converge to a solution considerably faster than the MPI solution. This was beleived to be due to the fact of how MPI had to coordinate message passing between the threads and had waitall calls that were sort of synchronizing the threads. The OpenACC threads were run with an asynchronous flag which allowed them to not have to wait for the other threads to be running their individual operations to completion before moving on. This was likely the source of the large discrepancy in their performance. MPI showed to improve logarithmically when the amount of threads were doubled whereas OpenACC converged and adding further threads stopped improving performance. This is believed to be do to memory congestion. The timing plots are shown below.

Figure 4: MPI Timing

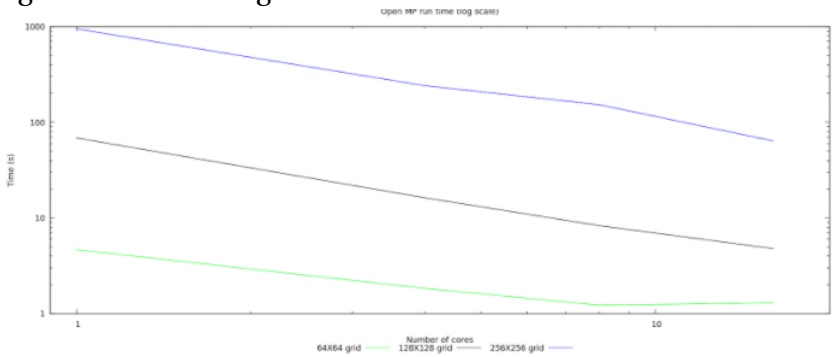
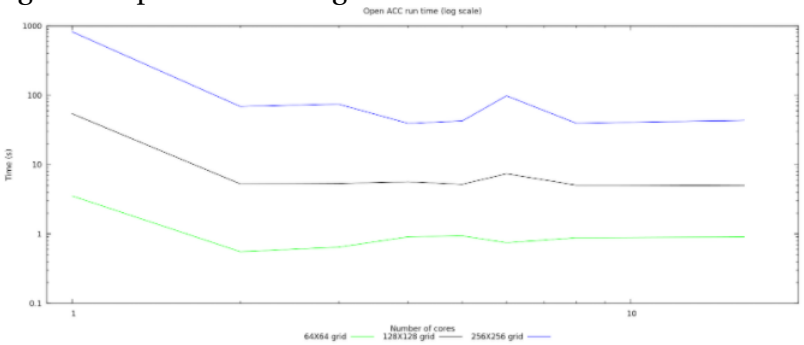


Figure 5: OpenACC Timing



REFERENCES

[1] BERGMAN, THEODORE FUNDAMENTALS OF HEAT AND MASS TRANSFER. WILEY, 2020.