John Burke

Dr. Valentine

Comp 233 A

March 22, 2021

Jacobi Lab Report

# Abstract

The program uses Jacobi Iterations to find the heat distribution across a 2D plate. The program is run in serial and in MPI with 4, 8, and 16 cores. It was hypothesized that the addition of more cores would increase the speed. However, it was found that more cores increased the speed up to a certain point, but then the addition of more cores after that point caused a decrease in speed. The speedup and efficiency were calculated to explain the reason why this happens. It was concluded that the speed with additional cores was always faster than the serial program, but the number of cores used should be matched to the size of the problem to gain the best efficiency.

# Introduction

The purpose of this program is to find the heat distribution across a 2D plate. In the beginning the place is the same temperature across the entire surface. External sources apply different temperatures to the perimeter of the plate. The temperatures applied to the boundaries of the plate are set as:

North Boundary: 100°

East Boundary: 100°

South Boundary: 0°

West Boundary: 0°

The interior cells are set to 1°. The heat distribution across the plate is found by using Jacobi Iteration. The plate is 800 units wide by 400 units high. All of the cells are pre-set to a certain value, and the boundary cells (the cells that form the perimeter) are unchanging. The interior cells in each iteration are calculated by taking the average of the four surrounding cells. Iterations are continued until all the cells converge to a value equal to or less than an epsilon value, or when the maximum number of iterations is reached. The time it takes to meet one of the end conditions is recorded. The epsilon value is 1e-2 (.01) and the max iterations are 1,000,000.

## Hypothesis:

The speed of the MPI program will increase linearly with the addition of more cores.

## Procedure:

The matrix is formed with 400 rows and 800 columns and the temperatures of the boundary and interior cells are set. The matrix goes through Jacobi Iterations until one of the end conditions is met. The process of a Jacobi Iteration is as follows:

The new value of each cell is calculated using the following equation:

The sum of the square of the differences between the old and new values in every cell is calculated. The square root of this value is the convergence point. This can be described by the following equation:

The old values are replaced by the new values.

This process repeats until the convergence point is less than or equal to epsilon or when the number of iterations has reached the max number of iterations.

The serial version of this program follows this process exactly.

The MPI process breaks up the matrix and gives each core a section of rows. Each core goes through this process with its section of the matrix. The convergence point is calculated by summing the individual sums of the squared differences from each core and then taking the square root.

The values of the matrix after the final iteration are translated to color values using linear interpolation and then printed to form a picture.

# Analysis

The temperature of the plate converged in 196,855 iterations. After the final iteration, the temperature across the plate looks like this:

Background pattern

Description automatically generated

The middle of the plate converged to a temperature around the middle of 0° and 100°, which is around 50° and is shown by the purple. The areas closer to the boundaries converged to temperatures closer to 0°, shown by the blue, and 100°, shown by the red.

The times for the serial program and for MPI using 4, 8, and 16 cores were found. They are:

Serial: 674.717 seconds

MPI 4 Cores: 67.650 seconds

MPI 8 Cores: 119.050 seconds

MPI 16 Cores: 172.490 seconds

The hypothesis that the speed would increase with the addition of more cores was incorrect. Rather, the speed decreases with the addition of more cores. This is caused by the addition of more cores when they are not needed. A larger plate would show an increase of speed with the 8 and 16 cores, but would eventually reach the same problem of reduced speed by increasing the number of processes.

The speedup and efficiency of the program can be calculated with these equations:

The speedup and efficiency for each number of cores was calculated and organized into this table and displayed in the graphs:

|  |  |  |  |
| --- | --- | --- | --- |
| p | 4 | 8 | 16 |
| S | 9.974 | 5.668 | 3.912 |
| E | 2.493 | 0.708 | 0.244 |

The speedup and efficiency decrease with the addition of more cores.

By using Amdahl’s Law we can find the maximum speedup when using a certain number of cores. The amount of the program that is parallelized is needed to find this. In this program, the amount that is parallelized is about 85%. We can find the maximum speedup that is possible by using an infinite amount of cores in the calculation. We find this by using the following equation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| p | 4 | 8 | 16 | ∞ |
| S | 2.759 | 3.902 | 4.923 | 6.667 |

# Conclusion

It was shown that the addition of cores always caused an increase in speed when compared to the serial version of the program. The addition of cores only increases the speed up to a certain point, and then the addition of more codes causes a decrease. The number of cores used should match the amount of data being processed so that we can get the highest efficiency.

# References

Base code from

<https://www.mcs.anl.gov/research/projects/mpi/tutorial/mpiexmpl/src/jacobi/C/main.html>

Timer.h which was used in the serial version is from

Pacheco, Peter. An Introduction to Parallel Programming. Elsevier Science &amp; Technology, 2018.

Method used to print the final iteration to a binary ppm file is from

<https://rosettacode.org/wiki/Bitmap/Write_a_PPM_file#C>