Program 2 Result report

Joshua Landron

CSS434 Spring19

Contents

[Algorithm Documentation: 2](#_Toc8243747)

[Figure 1: Heat2d\_MPI Overview and decision flow 2](#_Toc8243748)

[Execution Output: 3](#_Toc8243749)

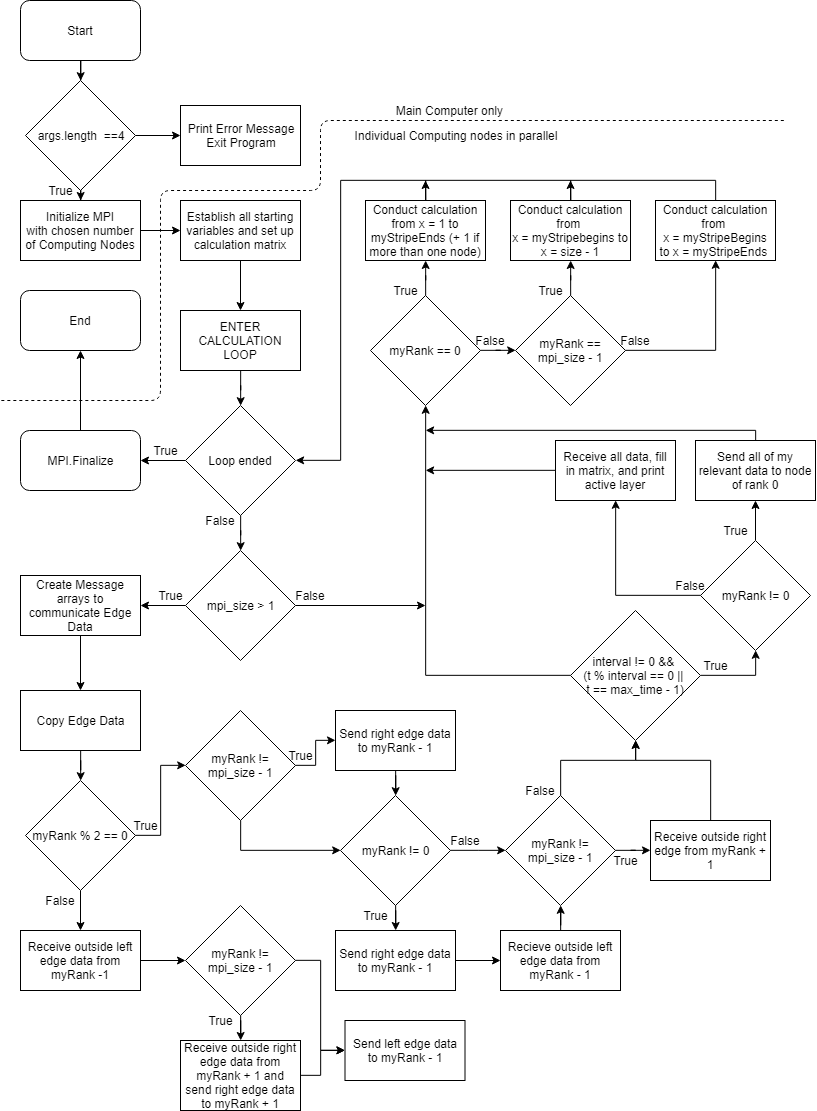
[Figure 1: Execution output image of Heat2d vs Heat2d\_mpi with 4 computing nodes 3](#_Toc8243750)

[Figure 2: Zoomed in image of Heat2d vs Heat2d\_mpi with 4 computing nodes 4](#_Toc8243751)

[Figure 3: Execution Times from 1 to 4 computing nodes. 5](#_Toc8243752)

[Discussion: 5](#_Toc8243753)

# Algorithm Documentation:

Figure 1: Heat2d\_MPI Overview and decision flow

# Execution Output:

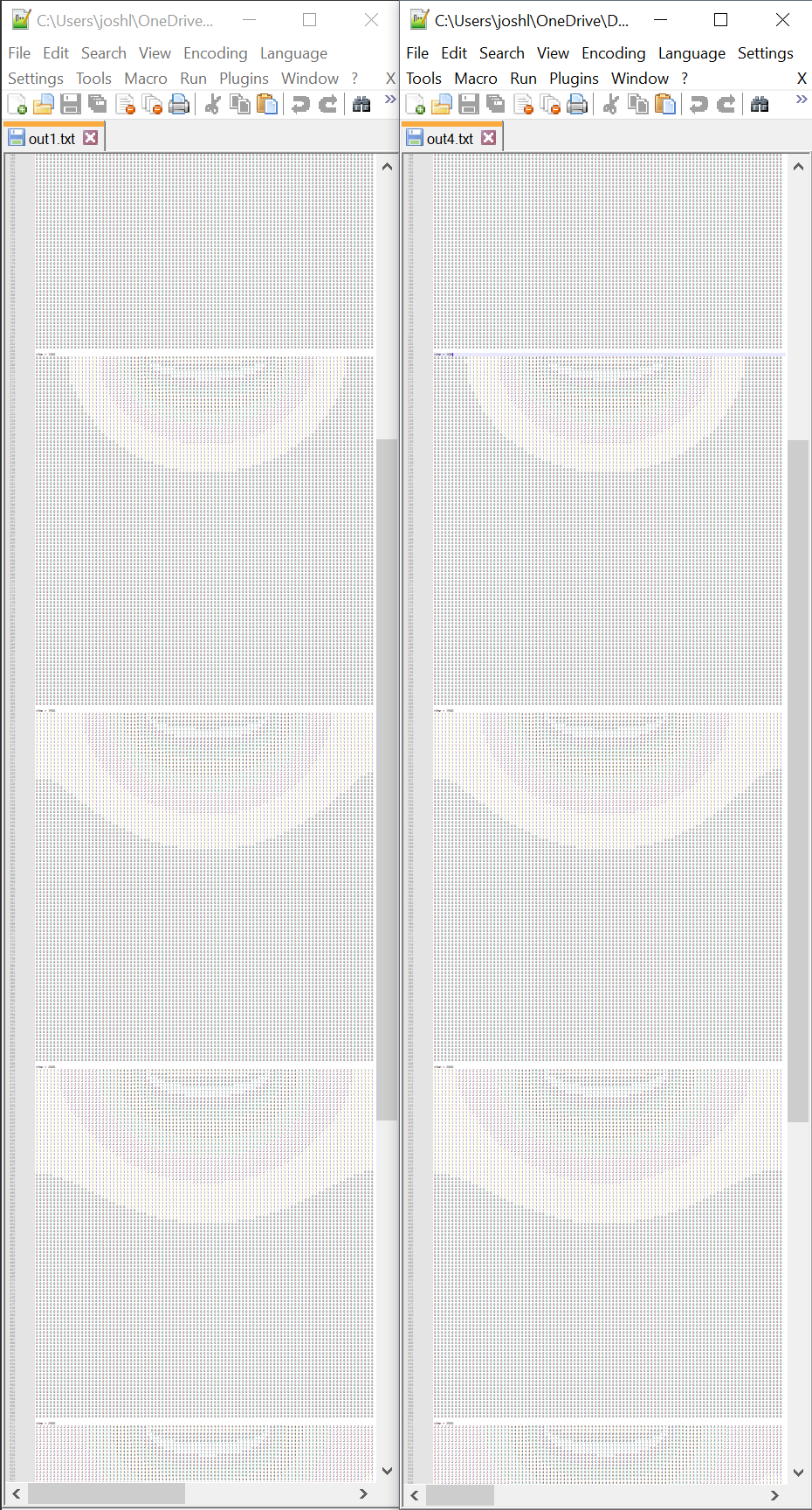
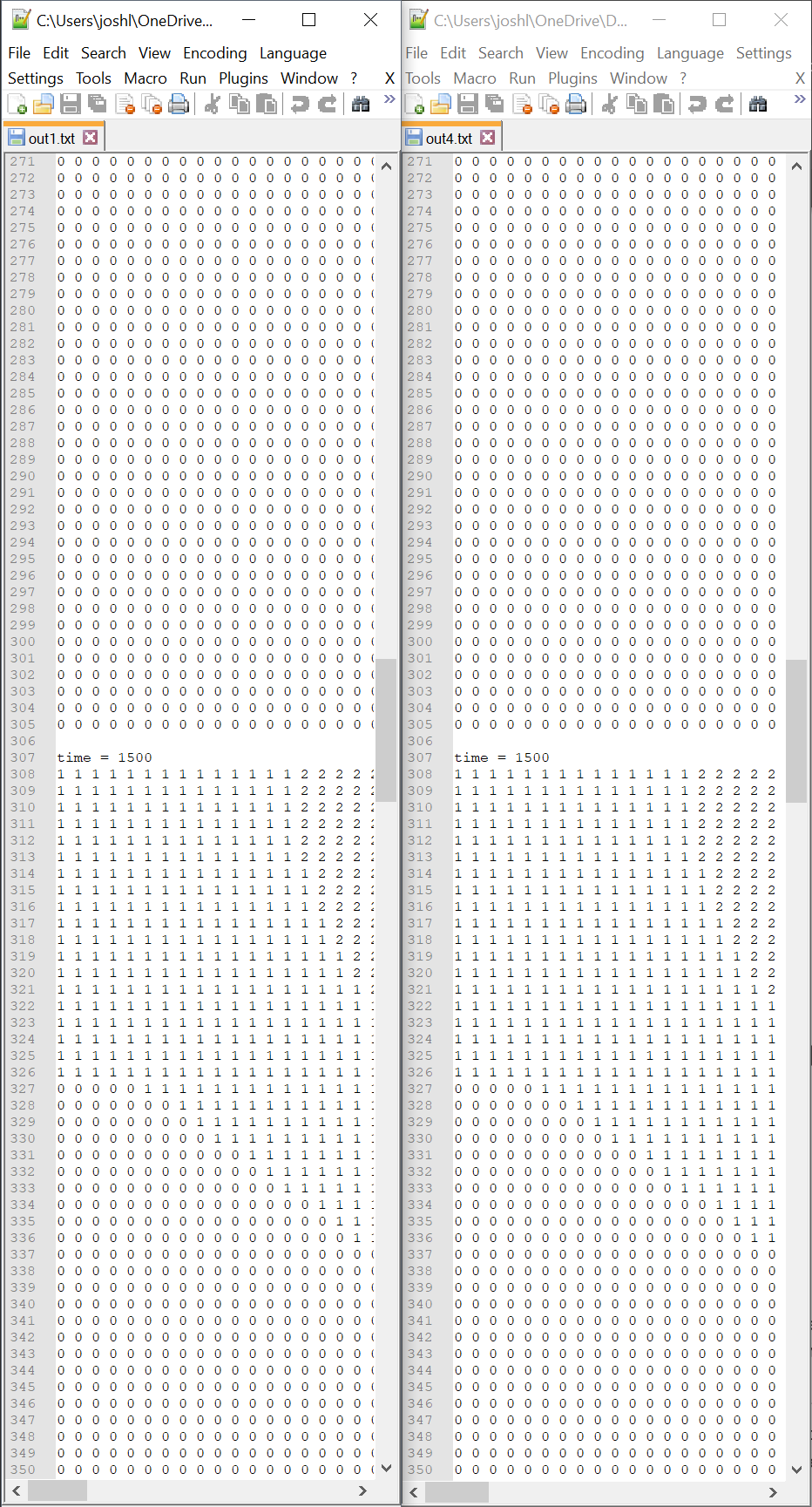
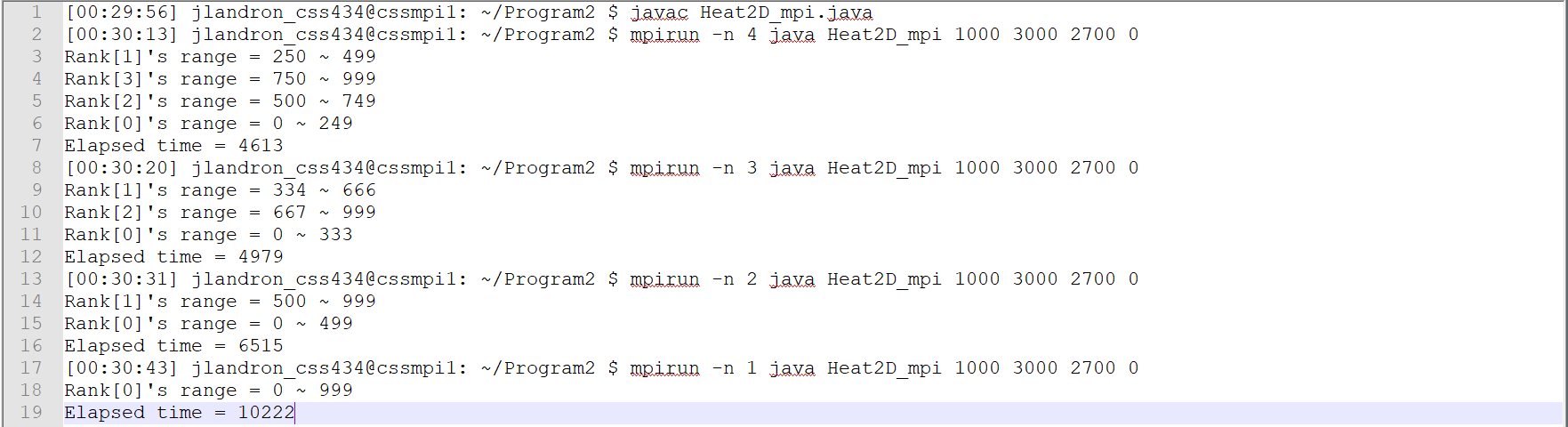
Figure 1: Execution output image of Heat2d vs Heat2d\_mpi with 4 computing nodes

Figure 2: Zoomed in image of Heat2d vs Heat2d\_mpi with 4 computing nodes

## Figure 3: Execution Times from 1 to 4 computing nodes.



Execution times have diminishing returns as more and more Computing nodes are added to the network.

# Discussion:

This parallelization improves performance over a single computing node by cutting execution time in half with 4 computing nodes. The diminishing returns of adding more nodes suggests that there is a communication time that will keep the execution time from having a limit closer to 0 than it seems to be. Given more and more nodes, I believe there will be a point where the execution performance may start to degrade due to the number of messages being passed in every loop.

I improved on the ideas presented in the assignment by narrowing down the amount of data that is passed between nodes. My method only ever passes exactly what is needed, keeping array lengths to a minimum. This improvement makes the MPI have less work to do. For example, anytime the matrix should be printed, instead of converting the entire matrix to a 1d array of length (2 \* size \* size), which for a size 1000 matrix is 2 million data points to be sent, I make each computing node send an array of size (size \* stripes[rank]) with stripes[rank] being the specific width of the stripe that computing node is working on, and to only send information from the layer that was just computed. This means with 4 nodes, each node has to send an array with ~2.5\*10^5 data points instead of 2\*10^6 data points. This order of magnitude improvement makes the messages much faster to send and receive.

The information exchange that must happen between all nodes cannot be improved, as it will always send 1 and only 1 column of data with length (size)

I believe the largest possible improvement in this algorithm would be to restrict the first 3 loops in the calculation to the stripe that is being worked on by a given node. This will only give a marginal improvement, as these loops are simple. The MPI might be improved to allow for faster communication between nodes, but that is outside of the current scope of my understanding.