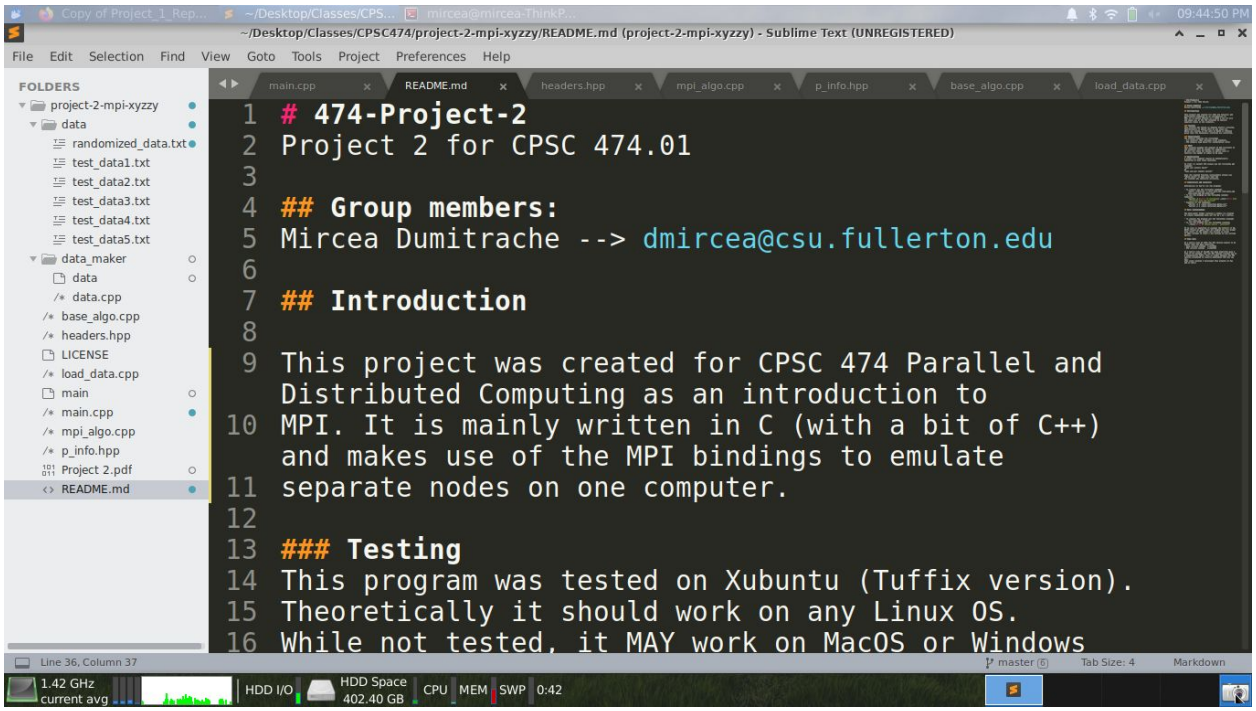
Project 2 Report

Name: **Mircea Dumitrache**
CSUF Email: dmircea@csu.fullerton.edu



Description on running the code

This code was entirely tested on Tuffix running Xubuntu 20.04. This program requires the MPI library which can be installed using the command: “sudo apt install mpich”

Those are the commands for the project:

- This is the command for compilation:
  - mpiCC -std=c++11 -o main main.cpp load_data.cpp base_algo.cpp mpi_algo.cpp
- This is the template for running the program:
  - mpirun -n <number of processes> ./main <text file including size and matrix>
  - mpirun -n 1 ./main data/test_data1.txt
  - mpirun -n 4 ./main data/test_data2.txt
- If randomized data is needed there is a program to make a random matrix:
  - cd data_maker/
  - g++ data.cpp -o data
  - ./data <size of matrix=50 by default>
  - cd ..

Code structure information

The description I make here is more or less the same as what I described in the presentation for this project, with a bit more detail. Here are some words I am making use of and their meaning:

- Scatter: refers to the use of MPI_Scatter function
- Gather: refers to the use of MPI_Gather function
- Scatter session: refers to one loop iteration that makes use of MPI_Scatter functions

The program will begin by checking the program arguments and reading the matrix data file given. Once the data is successfully read it will proceed to initialize MPI according to the mpiCC command given when the program was started.

If there is only one node running the program, the data will be computed by a base algorithm.

If there are multiple nodes running then the comparison variables will be initialized for each node, and the root node will begin to scatter the data around all nodes. If there are more rows than twice the number of nodes there will be multiple scatter sessions once all nodes finish their current workload.

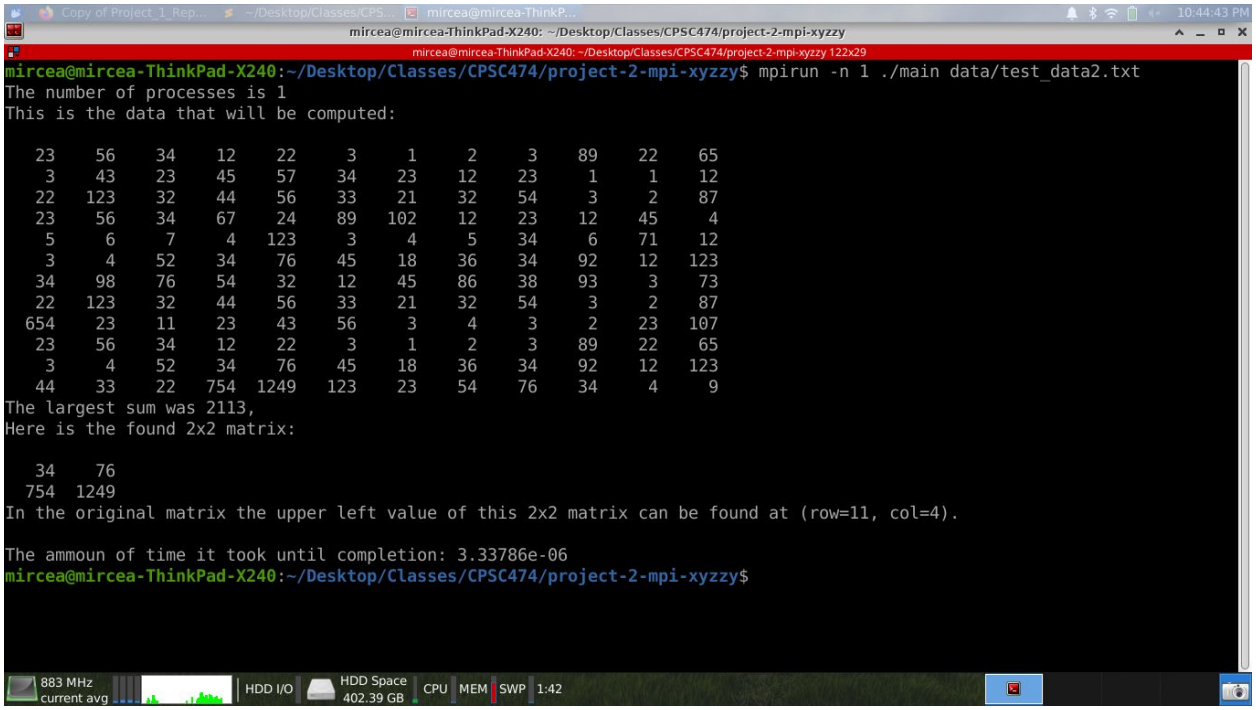
During one scatter session the rows to be sent to all nodes are divided evenly in sets of two. The nodes compute the largest sum of any squares in the data received then save the best fit. Once all nodes finish the root shifts the rows by one forward and sends the next set of data based on the new shifted value.

During the last scatter session, leftover data will be computed and the nodes will perform additional checks to make sure they do not compute information outside the range of the matrix.

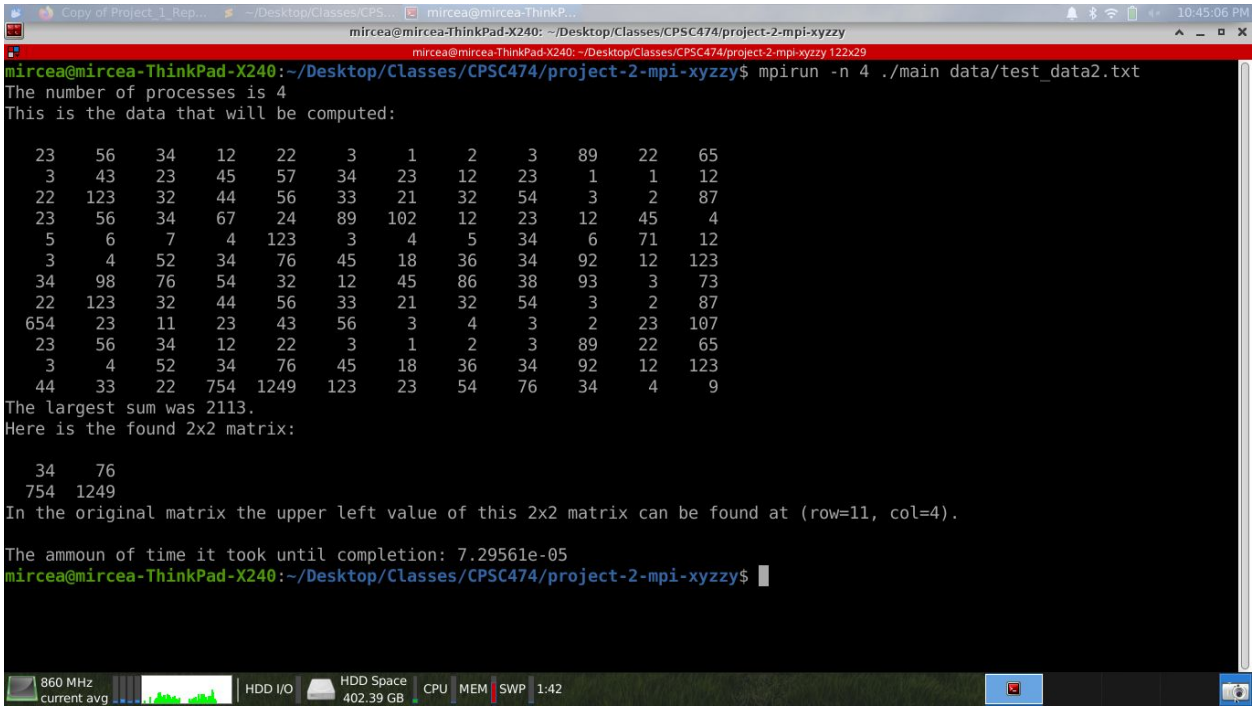
Once all rows are computed the root will begin to gather the data, and analyze the best fit by the value of the sum. The answer is then output to the screen and any dynamic allocation cleaned.

Screenshots on program running

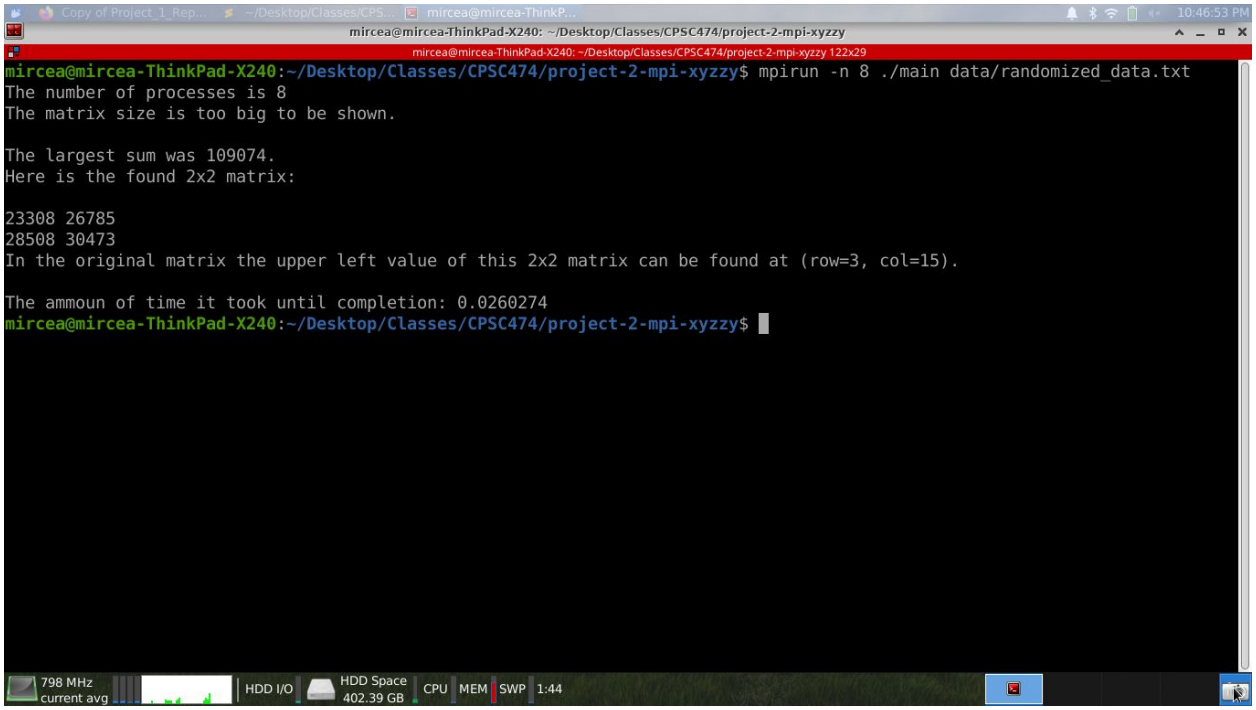
Running with one node on a 12x12 matrix (test_data2.txt)



Running with 4 nodes on a 12x12 matrix:



Running with 8 nodes and a 20x20 randomized matrix:


Copy of Project_1_Rep...
~/Desktop/Classes/CPS...
mircea@mircea-ThinkP...
10:46:53 PM
mircea@mircea-ThinkPad-X240: ~/Desktop/Classes/CPSC474/project-2-mpi-xyzzy
mircea@mircea-ThinkPad-X240: ~/Desktop/Classes/CPSC474/project-2-mpi-xyzzy 122x29
mircea@mircea-ThinkPad-X240:~/Desktop/Classes/CPSC474/project-2-mpi-xyzzy$ mpirun -n 8 ./main data/randomized_data.txt
The number of processes is 8
The matrix size is too big to be shown.
The largest sum was 109074.
Here is the found 2x2 matrix:
23308 26785
28508 30473
In the original matrix the upper left value of this 2x2 matrix can be found at (row=3, col=15).
The ammoun of time it took until completion: 0.0260274
mircea@mircea-ThinkPad-X240:~/Desktop/Classes/CPSC474/project-2-mpi-xyzzy$
798 MHz
current avg
HDD I/O
HDD Space
402.39 GB
CPU
MEM
SWP
1:44