**Parallel Floyd Warshall Lab Report**

Leslie Daniela Sosa

**What problems you encountered completing the assignment and how you overcame them.**

I spent a couple of hours trying to figure out why I was getting an out of bounds error, only to realize that I hadn’t included the variable [i] from the loop within the matrix operations, I had made a typo and had only included [x][y] which where the variables iterating through the matrix (the variable matrix contains the contents of file ‘fwTest.txt’).

**Any problems you couldn’t overcome, or any bugs still left in the program.**

There’s an issue when I try to execute 8 threads, their times not only get retrieved at different times, but the time gets long, and the last thread never gets retrieved.

I noticed that as I increased the number of threads the printed results were not being retrieved in order, I’m not sure if that is an issue within the placement of my code or if it’s a common recurrence.

**About how long it took you to complete the assignment.**

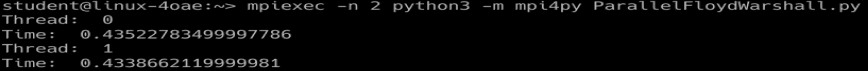
In total I would say it took me around 6 hours fixing up the code, making sure it ran, and creating a detailed report about the specific process I followed testing the multiple threads.

**Performance measurements (given in seconds) for 1, 2, 4, and 8 threads.**

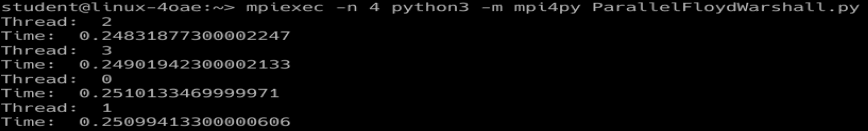
**Thread 1:**



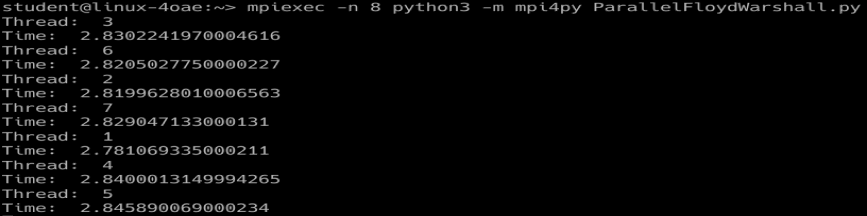
**Thread 2:**



**Thread 4:**



**Thread 8:**



**A short analysis of why the program behaves as it does with an increasing number of threads**

I had completed the assignment using a hardcoded matrix at first, and then I decided to attempt to read off the file ‘fwTest.txt’. My variable matrix remained the same throughout my program simply taking over the contents of file ‘fwTest.txt’ (I commented out my previous hardcoded matrix code).

This program reads file ‘fwTest.txt’ and splits it line by line into matrix, the program then opens the file ‘results.txt’ to write the program results.

The time differences aren’t really noted until 4 threads are executed. For threads 1 and 2 the execution times are around 0.4, with a decrease of 0.04 in time from 1 to 2. The use of 4 threads cut down the time taken to execute in half which is expected because the threads are incremented by twice the amount.

**Any observations or comments you had while doing the assignment**

This assignment had the most obvious time change, and it reflected the correct execution times it terms of size, meaning the more threads the faster the program executed, imitating a parallel execution.

**Output from the cpuInfoDump.sh program**

