Stephanie Galvan

Parallel Computing

MW 4:30PM – 5:50PM

Lab 2: Parallel Matrix Multiplication

Description:

With the help of pyMP library available in python, the following lab aims to run multiplication of two square matrices in parallel. The performance of the program will be tested with 1, 2, 4, and 8 threads. Similar to lab 1, all matrices are assumed to be square and some utility functions were provided to help in the finalization of the program.

Approach:

The given python file contained valid arguments to pass on the terminal (such as size and value for the matrix), so an additional argument was created to represent the num of threads used for parallel. Below is a sample entry for the terminal to run the program:



Similar to the previous lab, three nested for loops were used to multiply two given matrices.

Problems/Bugs:

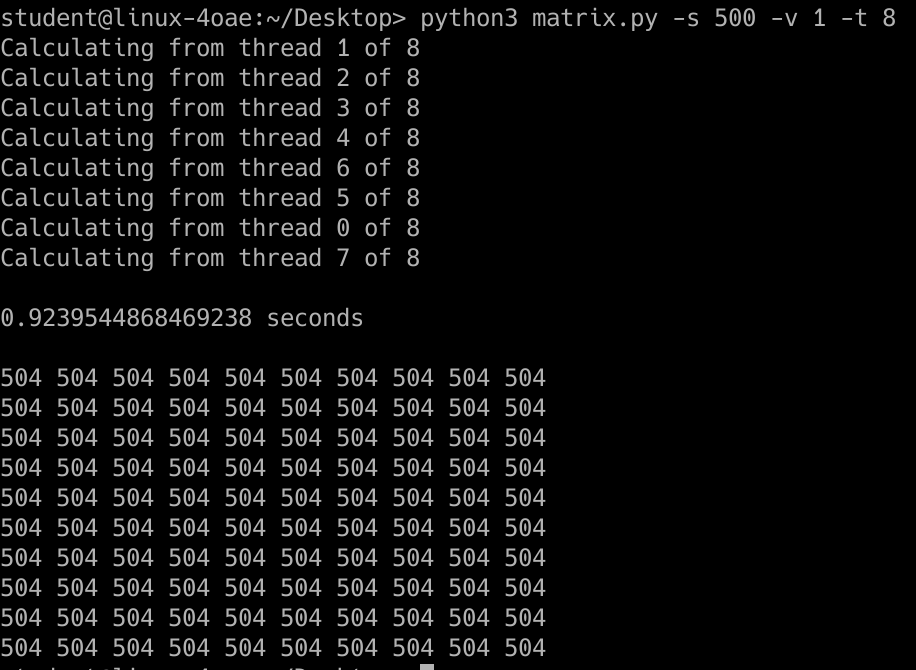
During the first run with pymp, the program took around 9 minutes, but this was later fixed by changing *range* to *p.range*. The first attempt at parallel programming failed due to implementing the wrong loop.

Also, the program does not implement a shared array and instead uses the generate array function provided. It is speculated that a shared array might provide different results; however, multiplying with a shared array resulted in erroneous answers and thus the lab did not follow this approach.

Finally, in order to obtain the correct resultant matrix, the values of the two matrices that are used to multiply had to be multiplied by the number of threads currently used in the program. Here is a sample code:



The code above works correctly when the matrices are of only one value and when the number of threads is less than 6. When the number of threads is 8, it results in the wrong result matrix as follows:



The resultant matrix should be composed of 500 and not 504; however, this bug has not been solved. Moreover, using a matrix of size 1024 does result in the correct resultant matrix with 8 threads; thus, the testing aspect of this lab will focus on matrices of size 1024.

Time Completion:

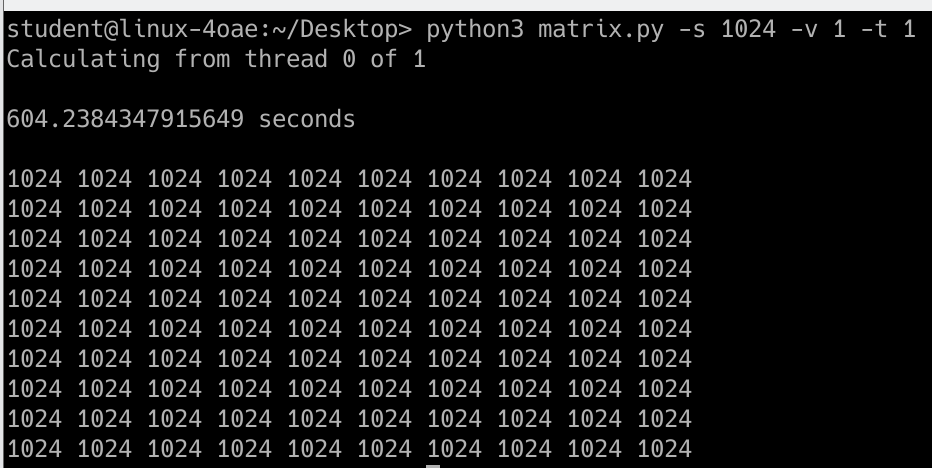
Disregarding the time to read documentation and understand the pymp library, it took around 2 hours to complete this lab. This time includes implementation and testing.

Performance Measurements:

The following tests will use matrices of size 1024 and value 1. To compare performance, the tests will use 1, 2, 4, and 8 threads. The results are shown below.

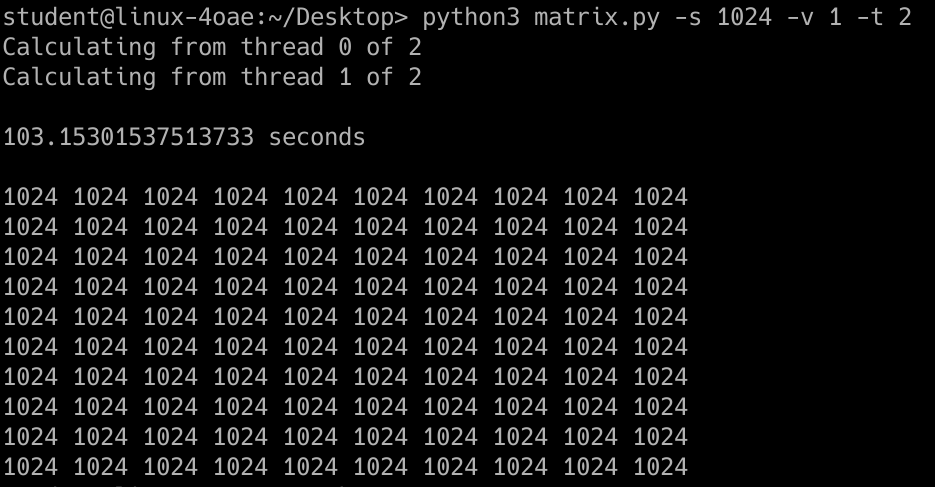
One (1) Thread:

A single thread resulted in a runtime of 604.24 seconds or around 10 minutes.



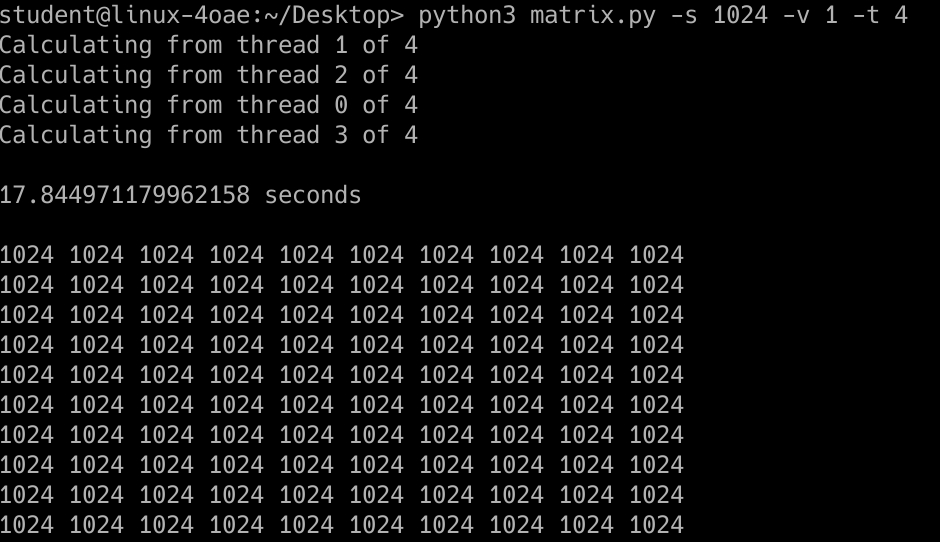
Two (2) Threads:

Two threads resulted in a runtime of 103.15 seconds or almost 2 minutes.



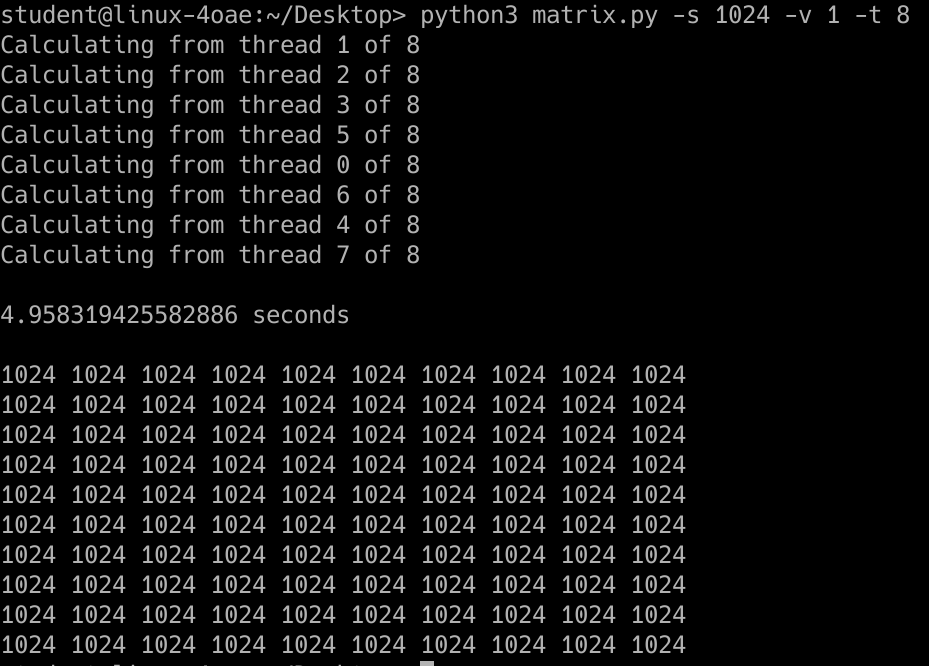
Four (4) Threads:

Four threads resulted in a runtime of 17.85 seconds.



Eight (8) Threads:

Eight threads resulted in a runtime of almost 5 seconds.



Analysis:

As demonstrated on the test results, an increasing number of threads resulted in a decreasing runtime. This happened because multithreading allows the program to run faster because processes are divided between threads.