Report Lab 2

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

I used my Lab 1 and build on the program, I had added the thread functions but I had kept the same size of two 10 x 10 matrices, so right away I was having issues seeing the change of threads in relation to execution time because the matrices where too small.

A misunderstanding also led me to have issues when it came to set up the shared array that would hold the product results of the arrays. I had kept the same result array that contained place holders for the answers, and I was trying to use the pymp.shared.array function at the same time, which obviously resulted in an issue. I eventually figured out that the result variable would hold the result dimensions of my matrix product.

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

I am aware that the Lab 2 description mentioned that the matrix sizes should be large enough to reach around 10 seconds. I used two 40x40 matrices and got 7 seconds at the longest time of execution when using a single thread.

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

I think the program was straight forward, but I did spend a lot of my time messing around with the size of the matrix to see the possible relation between threads and the time the program takes to provide results. I spent a longer time installing an IDE in my Virtual Machine and downloading the correct packages to support Python so I could test my program the correct way.

In total I would say it took me around 2-3 days 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:

0.07267531100014457

Thread 2:

0.06019822700181976

Thread 4:

0.05857520799690974

Thread 8:

0.05498577699877205

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

Although the time difference between the threads (1, 2, 4, 8) was small (due to the size of the program), it was evident that the use of more threads was increasing the speed of calculation execution. When the program was running on a single thread it took the longest around 7 seconds, but as I increased the number of threads up to 8 threads, the program took around 5 seconds to execute.

The idea behind using multiple threads is the ability to compute the same tasks at the same time instead of sequentially to increase performance. One thing to keep in mind is that the case isn’t always the more threads the better, the number of threads must be chosen considering what the physical system can support. More than 8 threads in this case wouldn’t have made a difference as they are not supported under the single core.

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

This assignment showed us the basics of parallel programming in a smaller environment, I would be interested in seeing the way programs are altered by complexity and size and how the use of multiple threads might affect the process.

**Output from the cpuInfoDump.sh program**

