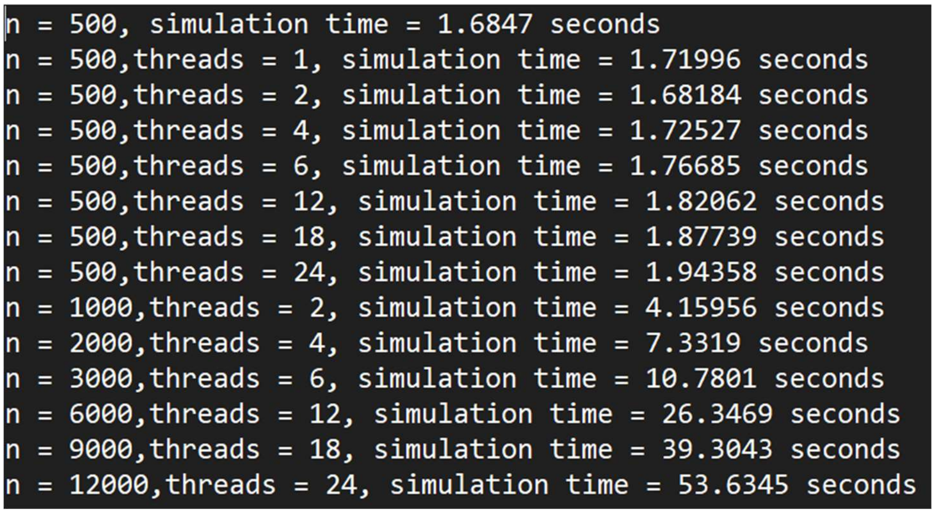
Problems

1) Code segment A is a simple parallelized program, however this does not have any included mutual exclusion, so issues could arise when running this on multiple threads accessing shared memory. B and C both include mutual exclusion. In this scenario, both B and C should perform the same, however their functionality is different in that code block B is allowing for threads to lock the array x itself while code block C will have a separate lock that will be used on the entire code block following the critical statement, which in this case is just the single line using array x.
2)
    a. On average, each thread should take 2i ms to complete. When a thread has to wait for the other, it should only wait for i ms.
    b. The execution time should not change.
    c. Adding dynamic has the possibility to decrease out execution time if the threads become out of sync, due to the dynamic assignment nature. But in worst-case, the execution time may substantially increase due to wait times and added overhead.
    d. Using firstprivate or lastprivate will help in this scenario.

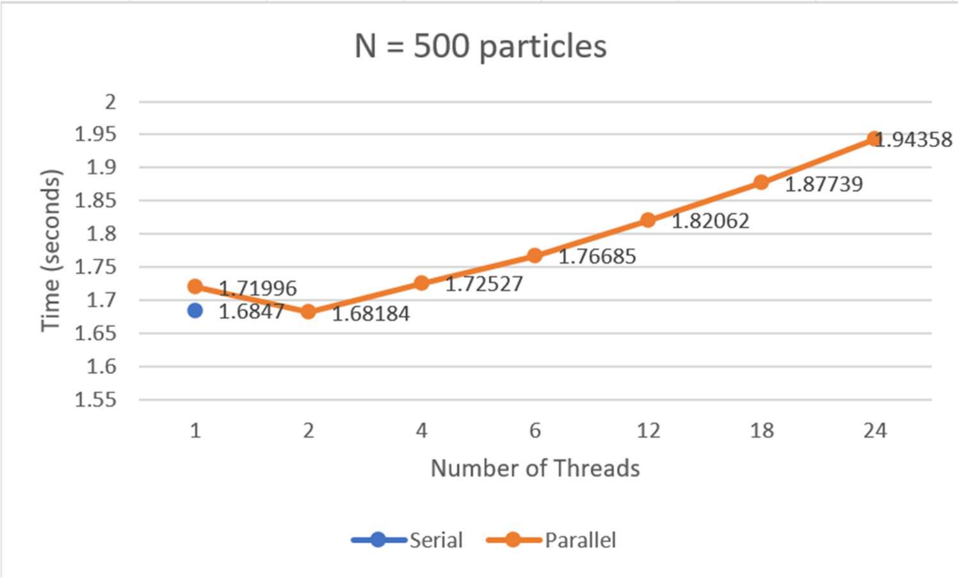
Programming Assignment

3) Programming Assignment

***My files are located under '/ihome/ece2192-2022f/cum6/ece1570/hw2'



```
n = 500, simulation time = 1.6847 seconds
n = 500,threads = 1, simulation time = 1.71996 seconds
n = 500,threads = 2, simulation time = 1.68184 seconds
n = 500,threads = 4, simulation time = 1.72527 seconds
n = 500,threads = 6, simulation time = 1.76685 seconds
n = 500,threads = 12, simulation time = 1.82062 seconds
n = 500,threads = 18, simulation time = 1.87739 seconds
n = 500,threads = 24, simulation time = 1.94358 seconds
n = 1000,threads = 2, simulation time = 4.15956 seconds
n = 2000,threads = 4, simulation time = 7.3319 seconds
n = 3000,threads = 6, simulation time = 10.7801 seconds
n = 6000,threads = 12, simulation time = 26.3469 seconds
n = 9000,threads = 18, simulation time = 39.3043 seconds
n = 12000,threads = 24, simulation time = 53.6345 seconds
```



My methodology was simply to break down the entire particle dispersal area into a series of bins. Then once we have placed all these particles into bins, we can calculate the forces from the neighboring bins. When parallelizing this, I made sure to include barriers between each phase to ensure that threads are at the same point.