Andrew Plant
9/11/22

Results of Running LuDecomposition

1. Tables

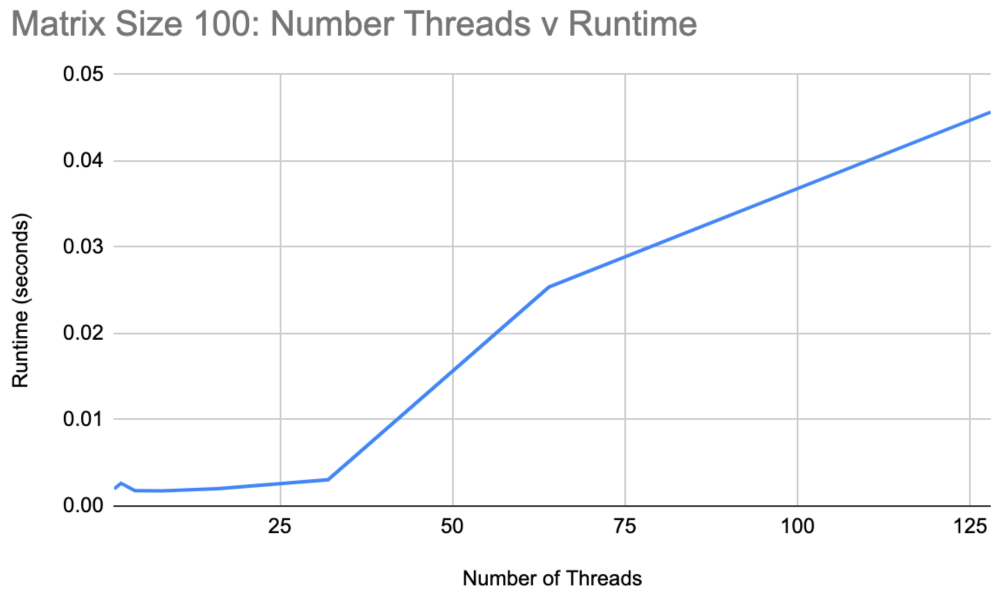

Figure 1. Matrix Size 100

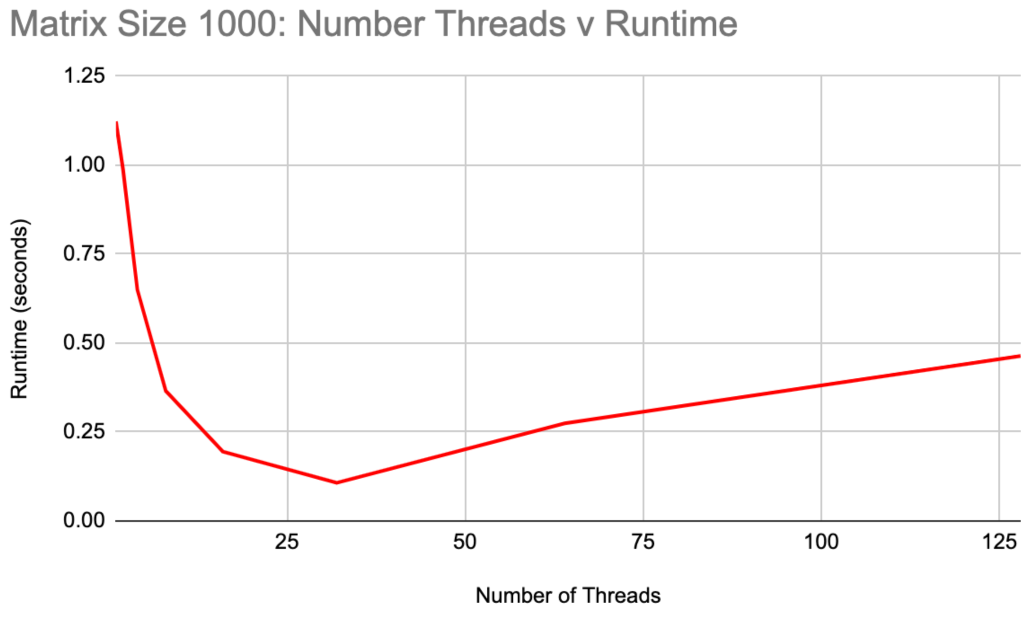

Figure 2. Matrix Size 1000

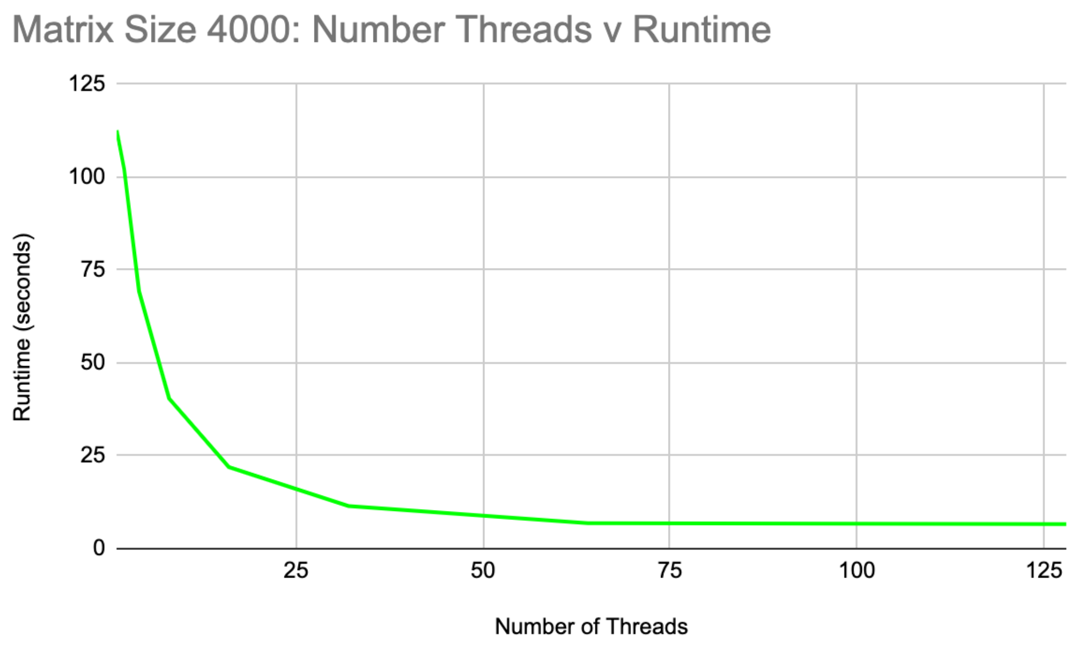

Figure 3. Matrix Size 4000

2. Observations

For matrices of size 100, it is apparent that the overhead of an increased number of threads creates a longer running time. The optimal number of threads in this case was 8 with the fastest running time of 0.00175878 seconds and any larger number of threads negatively affected the runtime.

For matrices of size 1000, an increased number of threads decreased runtime until the number of threads reached 32 and afterwards the overhead of establishing more threads created a longer runtime. It is important to note that the x-axis increases in a logarithmic scale.

For matrices of size 4000, increased number of threads continually decreased the runtime but in lower and lower increments. There would be a definable number of threads in which the runtime would begin to increase at approximately 256 or 512 threads. This size matrix has a considerably large runtime relative to sizes 100 and 1000, I theorize that this could be due to memory access for such a large matrix.