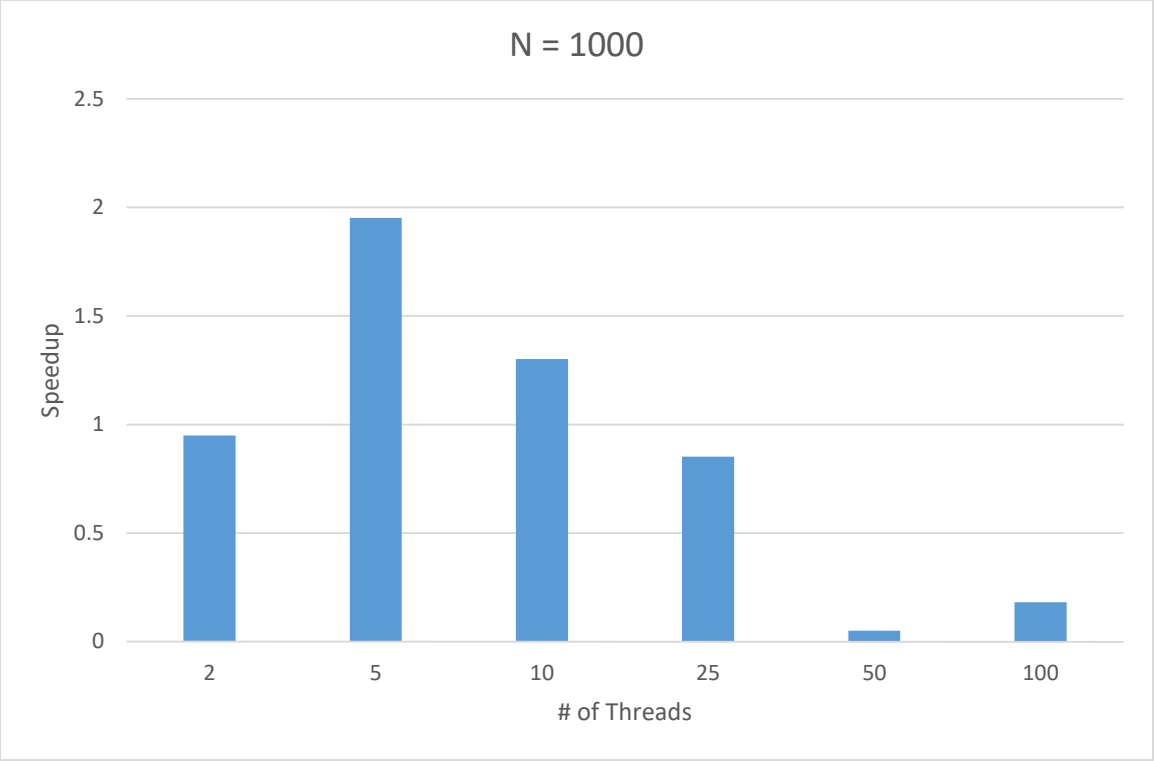


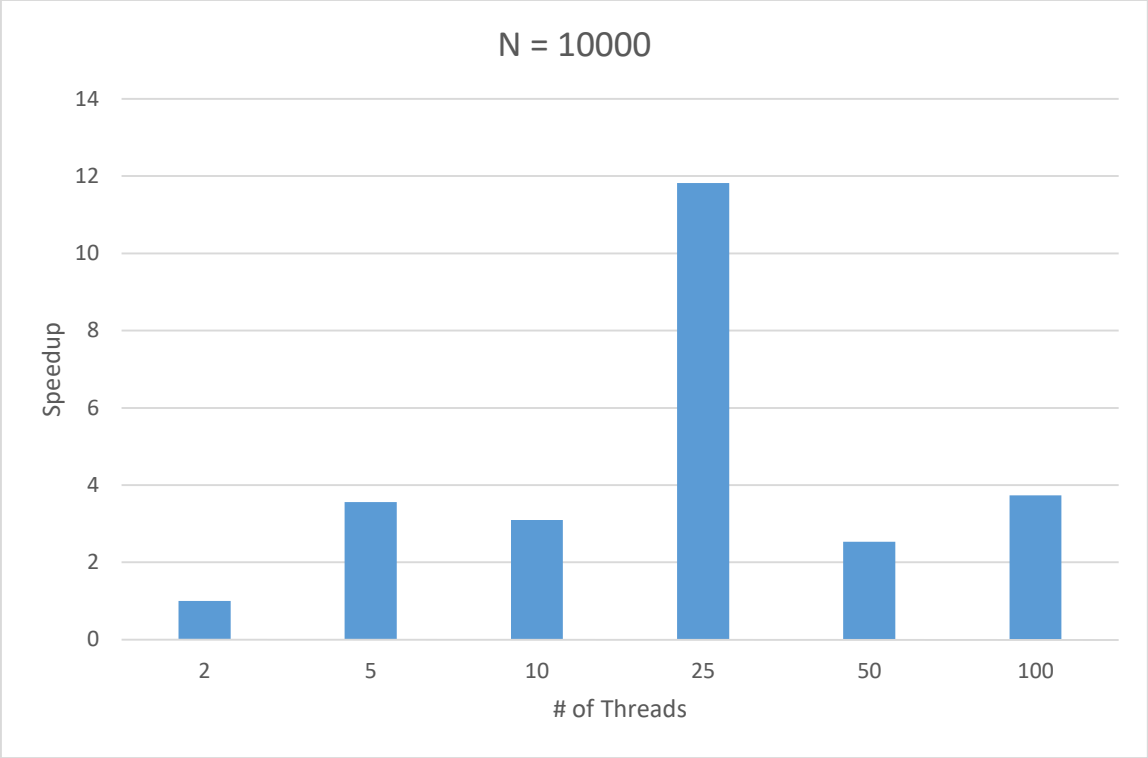
For N = 1000:

It can be seen that there is relatively no speedup when 2 threads are used. With 2 threads, the amount of calculations done per thread has significantly decreased from the sequential version. 1000 elements is also not a large data set. So the amount of time saved by having 2 threads perform calculations is outweighed when these threads must access the same global array to update the prime numbers. Since memory operations are very costly, there was no speedup.

There is some speedup when 5 or 10 threads is used because the amount of calculations decreases by a significant amount for this data set size. However, when updates to the global prime number array must occur, there is not a huge number of threads trying to access the array.

This is the problem seen as the number of threads used increases to 25, 50, and 100. Although calculation time is decreased, more threads are trying to make updates to the same array. This slows down the program since it would be more beneficial to have a smaller number of threads performing the final update to the global prime number array.

After a certain point, adding more threads does not increase speedup because the amount of data to calculate the prime numbers for is fixed. As the number of threads increases past 100, speedup will continue to get worse as synchronization and memory access overhead outweighs the benefit of lowering the calculations done per thread.



As the data size increases, so does the advantage of having parallel hardware.

With 5 threads, a speedup of approximately 4 is achieved meaning the hardware is being put to good use. In this case, there is enough data to give to each thread so that calculation time is decreased by a signification portion. The threads are still slowed down by having to update the global primes array. This contention for the global primes array becomes more of a problem as the number of threads increases for a fixed data size. At some point, it would be better to allow each thread to perform more calculations so that there is less synchronization at the end of the program.

Even though accessing the global primes array will slow down the program, having more threads calculating partitions of the data will provide a large enough boost in performance to exhibit some speedup. But as the number of threads increases past 100, the speedup will start to decrease as was seen with the N = 1000 graph for 25, 50, and 100 threads.

However, for N = 10000, there is enough data calculation to be parallelized that speedup is still seen even due to memory accesses and synchronization.