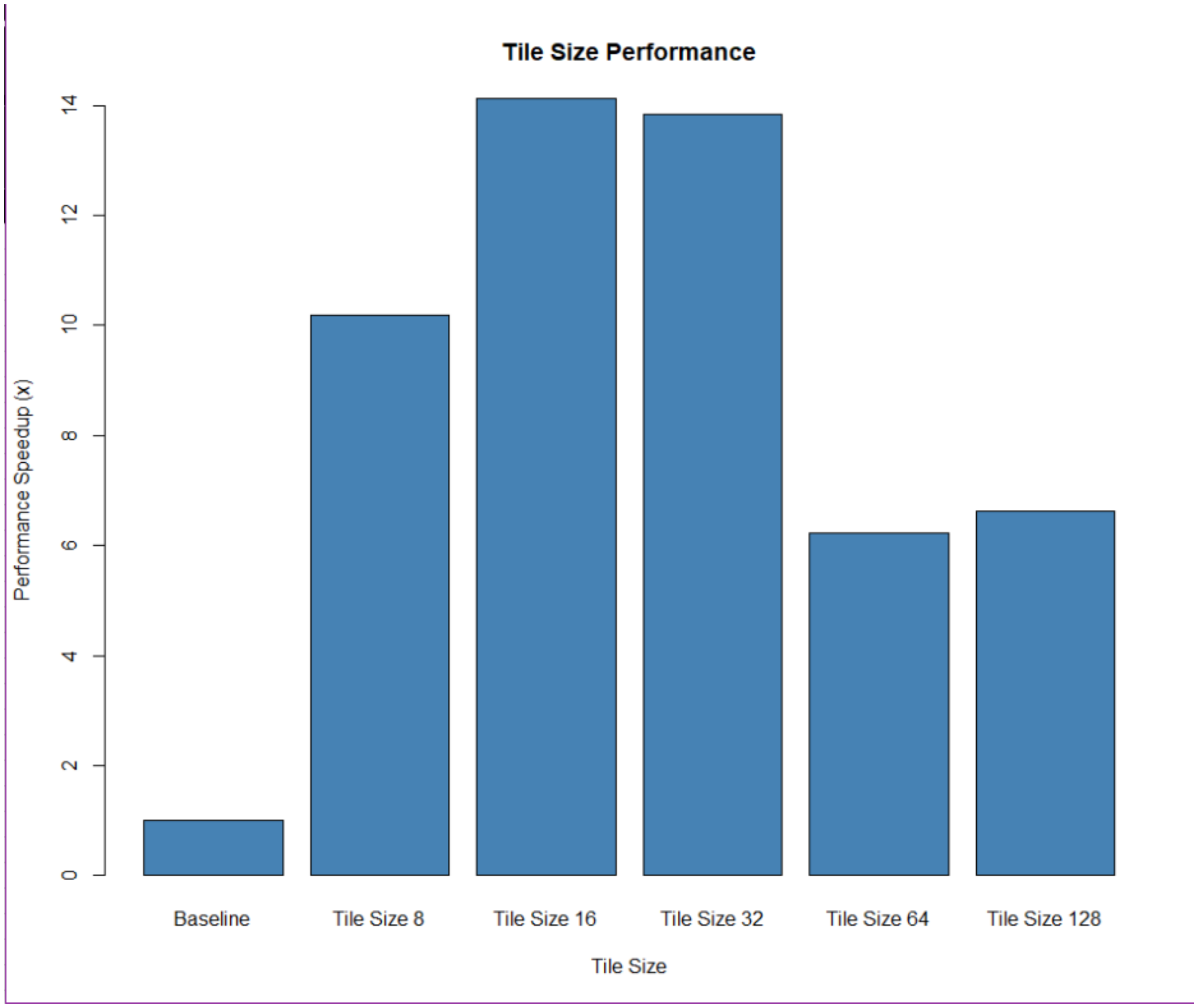
GeMM Algorithm Optimization

gemm_tile: optimized using data tiling. Below the performance of each tile size is compared with the baseline time of 483 seconds. A tile size of 16 provided the best performance.

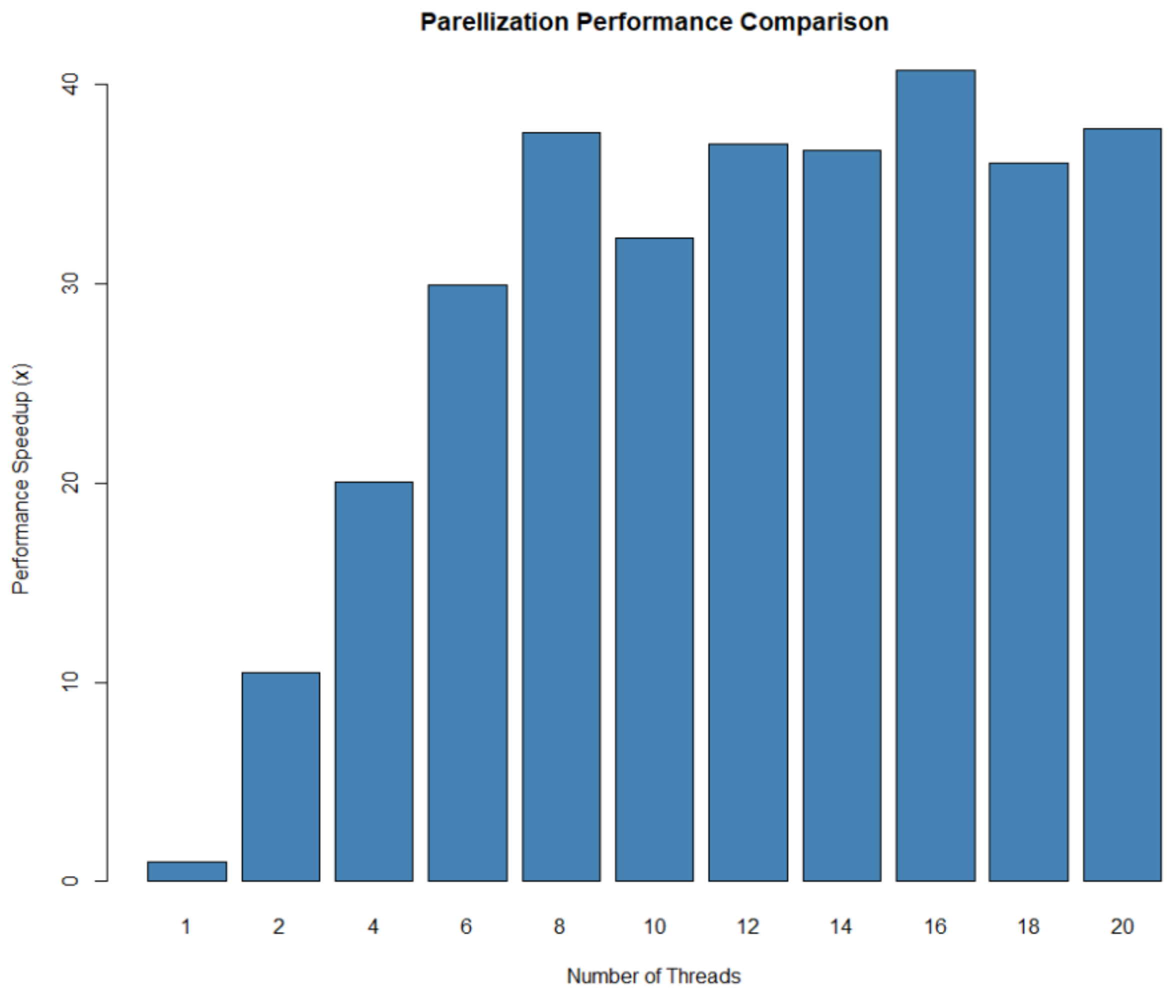


gemm_tile_simd:

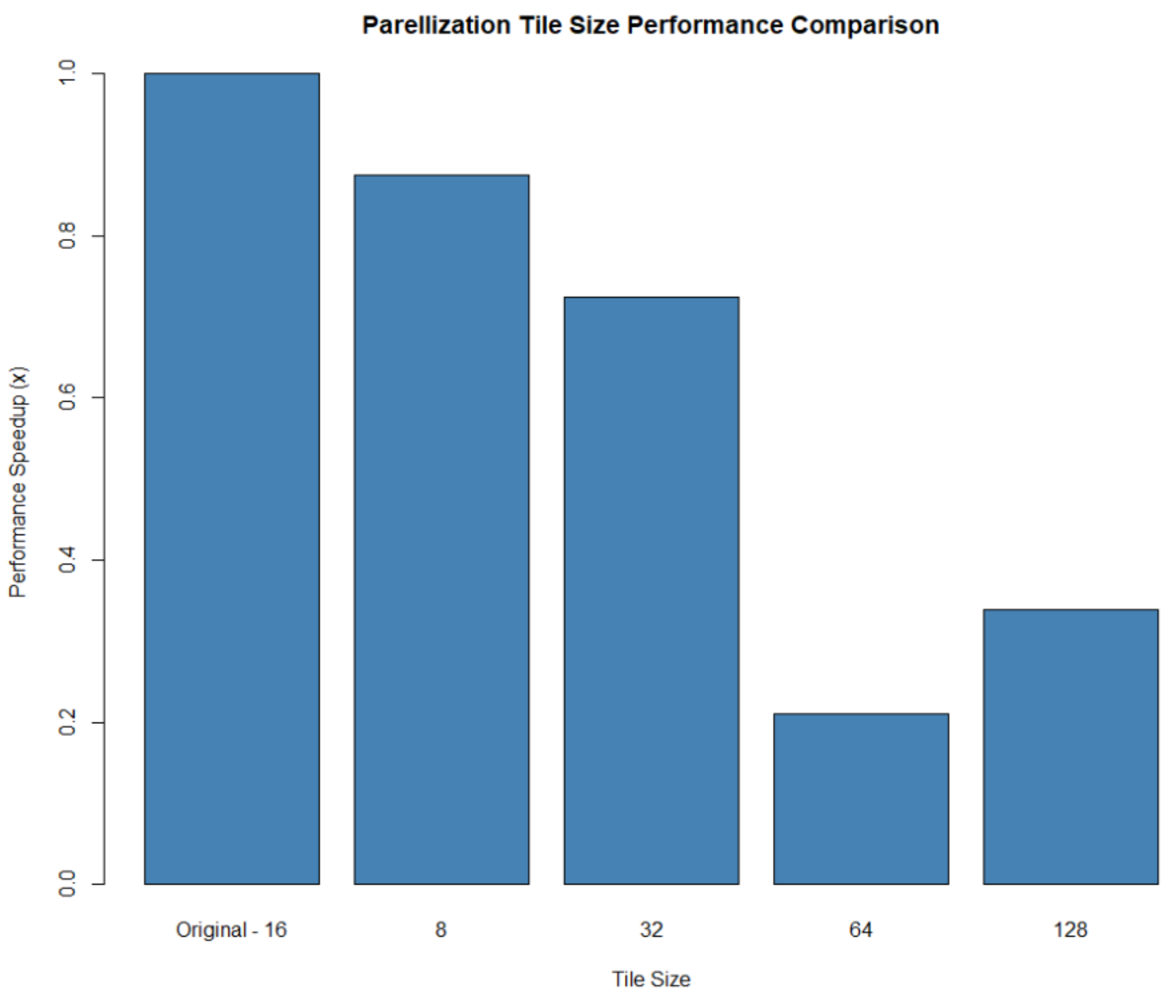
Optimized using data tiling and vectorization (SIMD). There was a 5.6 times speedup in performance over the best data tiling time.

gemm_tile_simd_par:

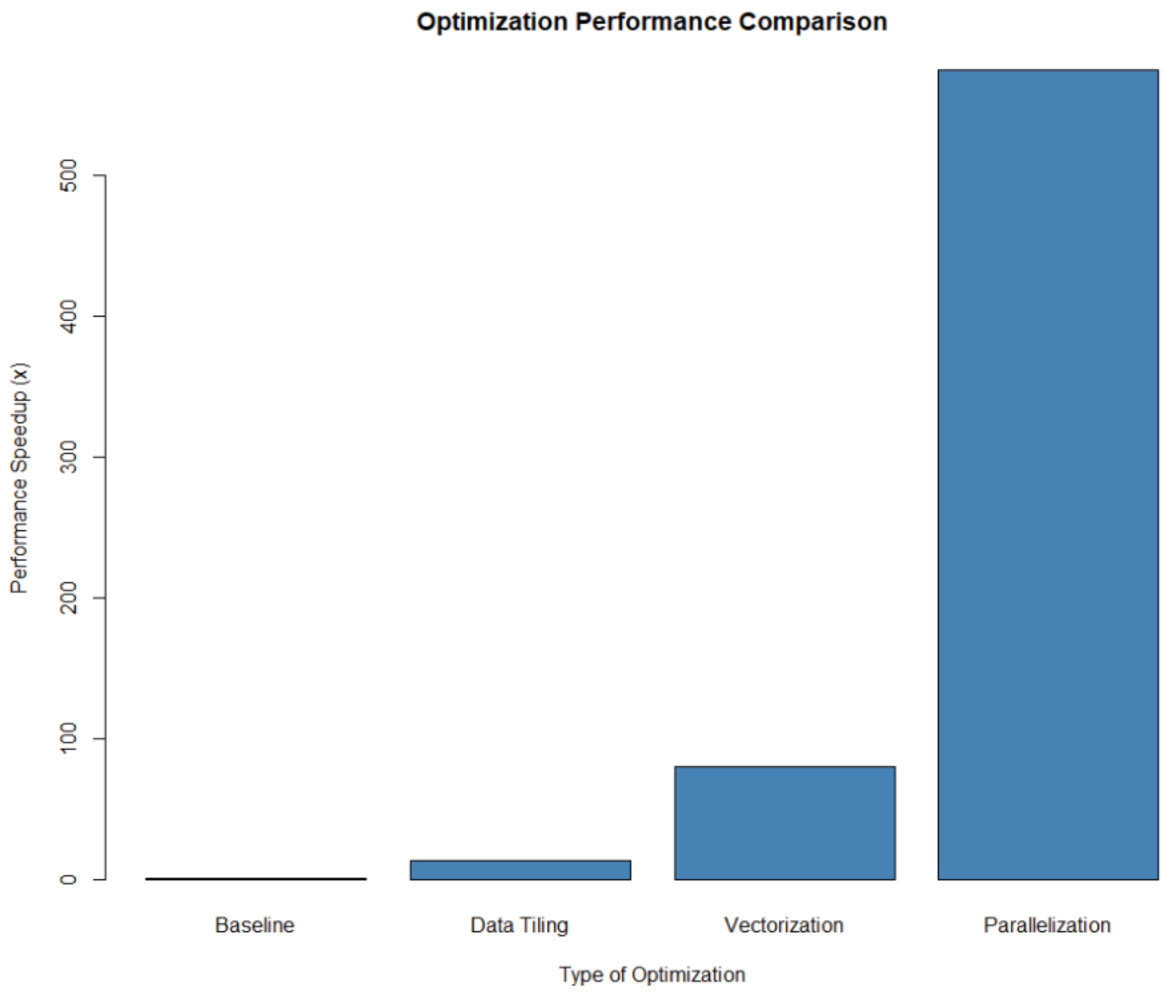
Optimized with data tiling, vectorization (SIMD), and parallelization. Below the performance of various numbers of threads are compared, including single thread which has no parallelization. 16 threads are found to have the best performance.



Below is a comparison of the performance of various tile sizes with a fixed thread count of 16. The original tile size of 16 proved to give the best performance.



Below is a comparison of optimizations performance to the baseline version. The parallelization version provided a significant performance speedup.



Performance:

We did not include other optimizations to increase performance of the program. All optimizations made used the 3 optimizations previously discussed in the report. The optimal performance is on average 0.95 seconds. This is due to the re-test as per the communication mistake of different configurations. This required an increase in the number of threads which was previously 16 in the report, up to 20. Tiling stayed the same at a size of 16.

When originally working we had optimized and stopped optimizing after achieving sub 0.9 seconds, believing we were done, and given these last-minute time constraints we were not able to optimize further.