Aaron Pham
Professor Anastasiu
Coen 145
October 18, 2022

Program 1 Report

1. I parallelized dense matrix multiplication two ways: with and without tiling. I parallized the multiplication of each block and tile using a "#pragma omp parallel for collapse". Inside the matrix tiling function I used one "#pragma omp parallel for" loop. I decided that it would be important to assign each thread a block or tile in order to achieve maximum speedup.

2. My timing results in **Figure 1** and **Figure 2** show my timing results for my parallel executions. I ran with a block and tile size that was proportional to the size of my matrix. A matrix with 5000 rows would have a block size and tile size of 500 while a matrix with 1000 rows would have a block size and tile size of 100.

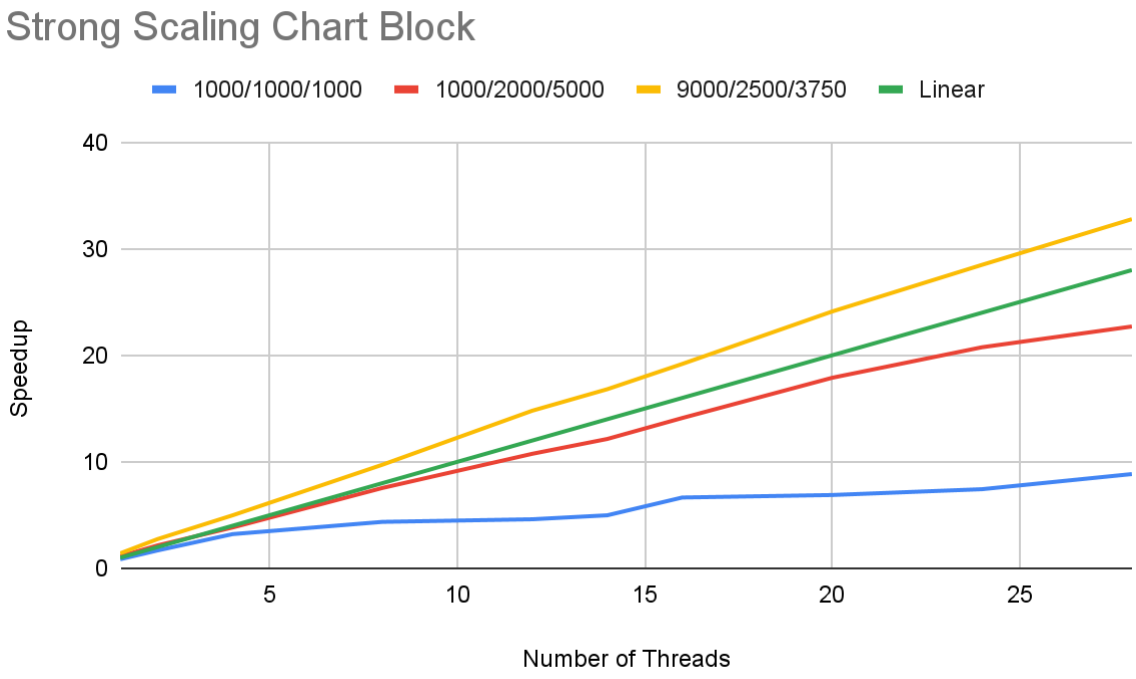

Figure 1: Strong Scaling Chart of speedup for Block parallelization

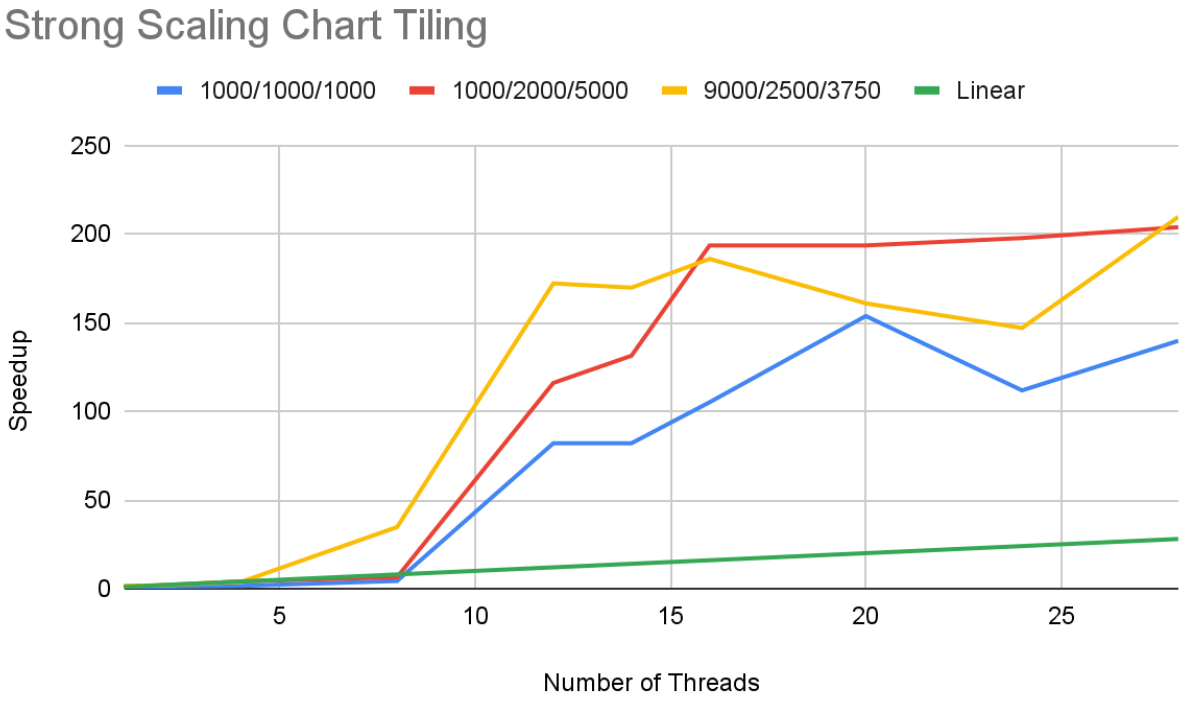

Figure 2: Strong Scaling Chart of speedup for Tiling parallelization

Analysis

3i. The results show that the number of threads has a huge impact on the runtime. This is because there is a much greater number of block sizes and tile sizes than there are a number of threads. Each thread works on a certain number of blocks or tiles and increasing the number of threads means that each thread has less blocks or tiles they have to complete in total. Less amount of work for each thread means faster completion and a faster runtime.

3ii. The size of the problem affects the runtime because it increases the amount of work each thread has to do. The block sizes and tile sizes increase as the size of the matrices increase. Matrices with more rows or columns means more multiplications that the thread has to do. More work means longer completion and a longer runtime.

3iii. My program scales fairly well. According to the strong scaling charts, it shows that a further increase in the number of threads will continue to improve. For the block parallelization, the charts show that the speedup will still improve at a more linear rate. For the tiling parallelization, however, the charts show that a speedup will still occur, but just at a lower rate.

3iv. For the block parallelization, the performance characteristics are very consistent for the different shapes of the matrices. This is shown in the strong scaling charts which show a

consistent linear pattern with all 3 different sized matrices. This consistent performance is much more apparent in the block parallelization than the tile parallelization, however the performance characteristics for the tiling are still consistent within the 3 matrix sizes I tested. Although the matrices are different sized squares and rectangles, an increase in the number of threads still results in speedup.