Aaron Pham
Professor Anastasiu
Coen 145
November 8, 2022

Program 2 Report

1. I parallelized sparse matrix multiplication by first breaking the matrix into blocks equaling in size. The last block would get extra if it did not split evenly. Each thread would get a block and do the multiplications and put the answers into another csr block. I used a reduction in order to sum of the number of non zeros in all the blocks. I used this non zeros variable to reserve C. After reserving C, I went through each block and put the values, indices, and pointer values into C.

2. My timing results in **Figure 1**, **Figure 2, Figure 3, and Figure 4** show my timing results for my parallel executions with Fill Factors: 0.05, 0.10, 0.15, and 0.20. As the fill factors got larger the speedup stopped increasing as much. They leveled out around the 12 - 14 mark even when the number of threads increased. My block sizes depended on the number of threads and a larger number of threads for a similar size matrix would mean small blocks for each thread.

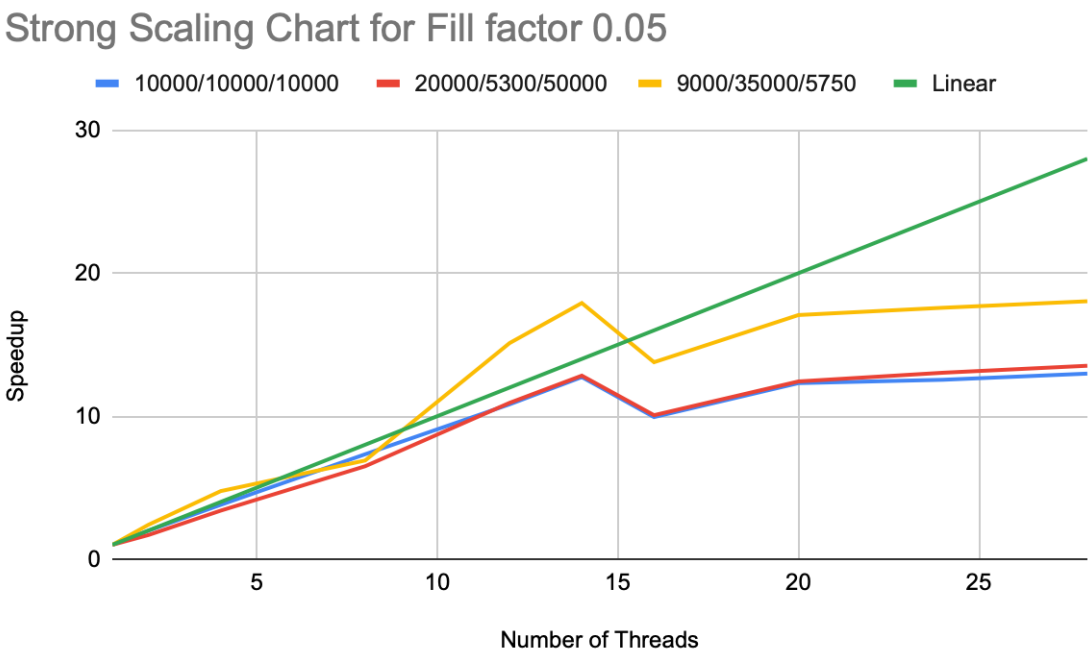

Figure 1: Strong Scaling Chart of speedup for Fill Factor of 0.05

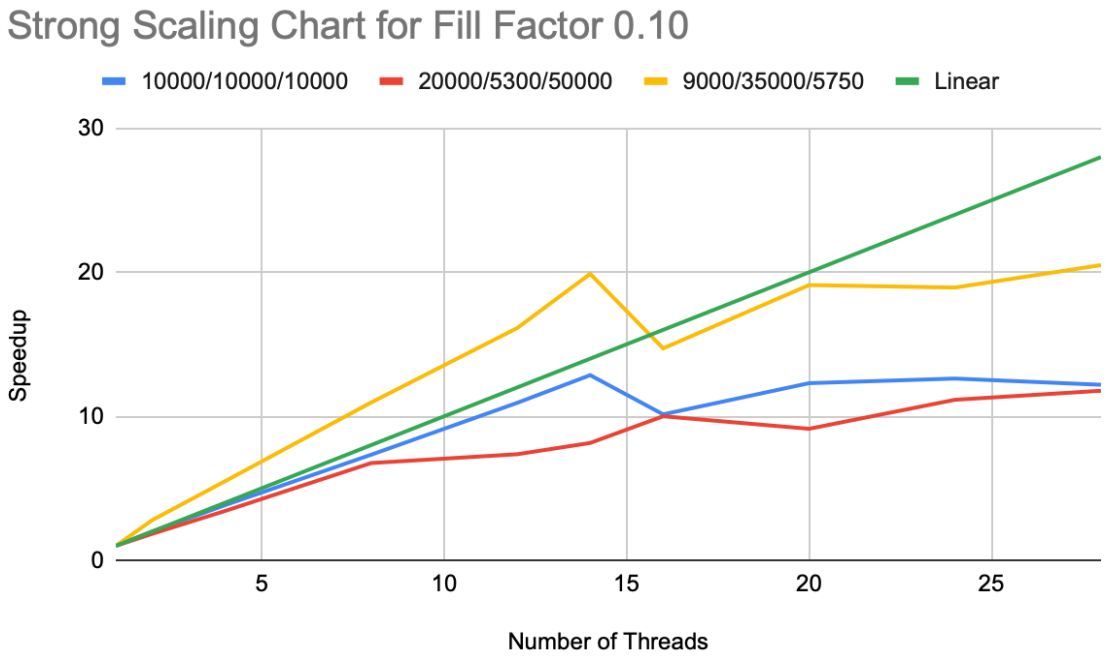

Figure 2: Strong Scaling Chart of speedup for fill factor of 0.10

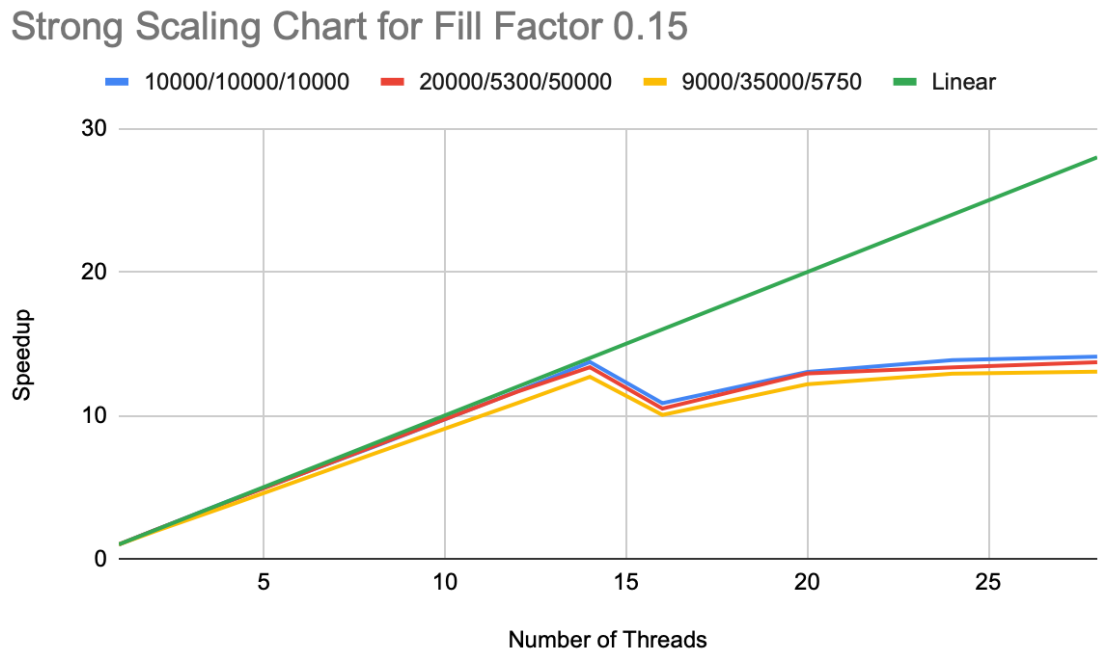

Figure 3: Strong Scaling Chart of speedup for fill factor of 0.15

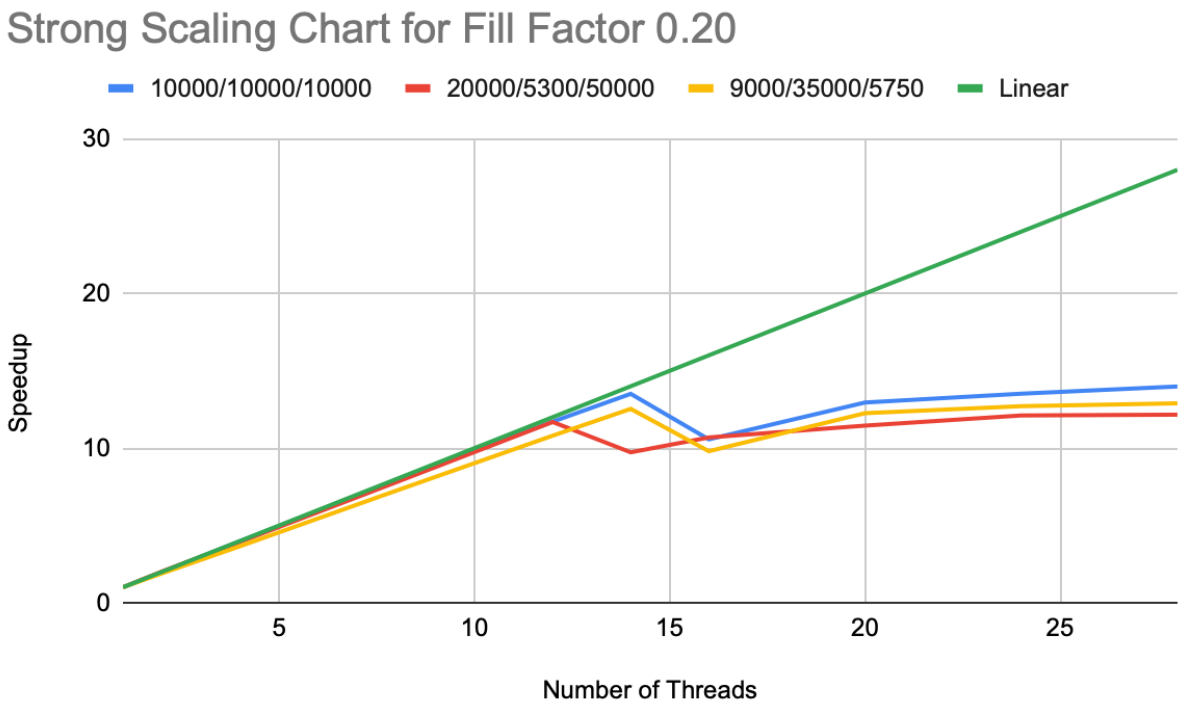

Figure 4: Strong Scaling Chart of speedup for fill factor of 0.20

Analysis

3i. The results show that the number of threads has a huge impact on the runtime. This is because each thread gets an even split of work and multiplications. More threads means more blocks which means less work for each thread. Less work for each thread means that it finishes faster increasing the speedup and runtime.

3ii. The size of the problem affects the runtime because it increases the amount of work each thread has to do. Since the number of blocks is consistent if the input of nthreads is the same, a larger matrix size means larger blocks for each thread. This means more multiplications 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 the speedup. More threads means more blocks means less work for each block.

3iv. For the block parallelization, the performance characteristics are fairly consistent for the different shapes of the matrices. For both fill factors, the 9000, 35000, 5750 matrix was slightly faster than the other two matrix shapes, but generally the speedup matches the other sizes. I think the 9000, 35000, 5750 one was slightly faster because the B matrix was the matrix that was slightly larger than the A matrix, however the B matrix was more sparse, which meant less numbers to multiply.