Erdem Ege Maraşlı
21602156

CS426 - Project 3 - Report

1.Parallelization Strategy and Design Choices

I figured out that in the given input files there are some zero values given as non zero values. The main purpose of compressed sparse row representation is to reduce stored data and its computation by only including non zero values. Because of this, I recomputed the non zero value count in the given text files.

Parallel implementation was pretty straight forward. The only critical part was scheduling. Static scheduling is prefered for load-balanced loops while dynamic scheduling is prefered if iterations have widely varying loads, but ruins data locality. Load balance of the given data is not known. Because of this, I choose to assume a general case which is uniform distribution so that I use dynamic scheduling. I assumed that each row contains nearly equal non zero values. This choice can cause low performance if rows in the one chunk contains a lot of non zero values while rows in the other chunks contains small amount of non zero values. If the input data specify its non zero distribution type we could specify different schedulings for the given inputs.

Note: Resulting vector can give nan values on the high number of repetitions because of the stack overflow of the double values since the decimal part gets really long after some point.

2. Benchmark Results

CPU: Intel Core i5-8259U @ 2.30GHz
RAM: 16 GB 2133 MHz

All of the data points are taken as the minimum execution time of the 10 execution

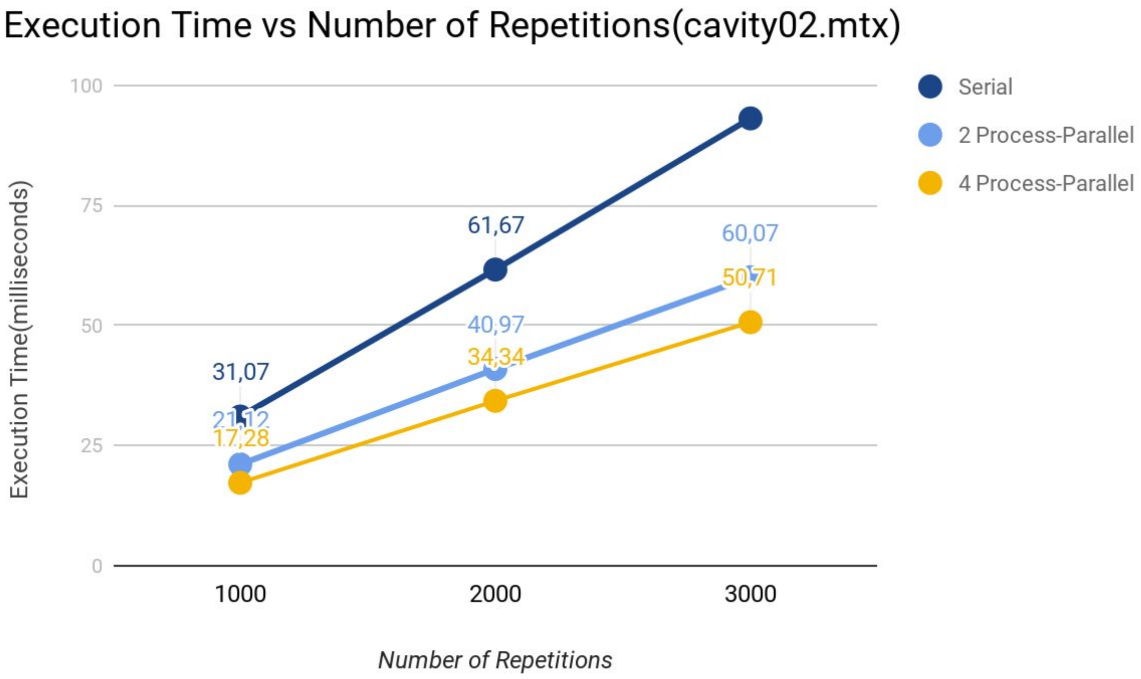

Figure 1: Execution Time vs Number of Repetitions(cavity02.mtx)

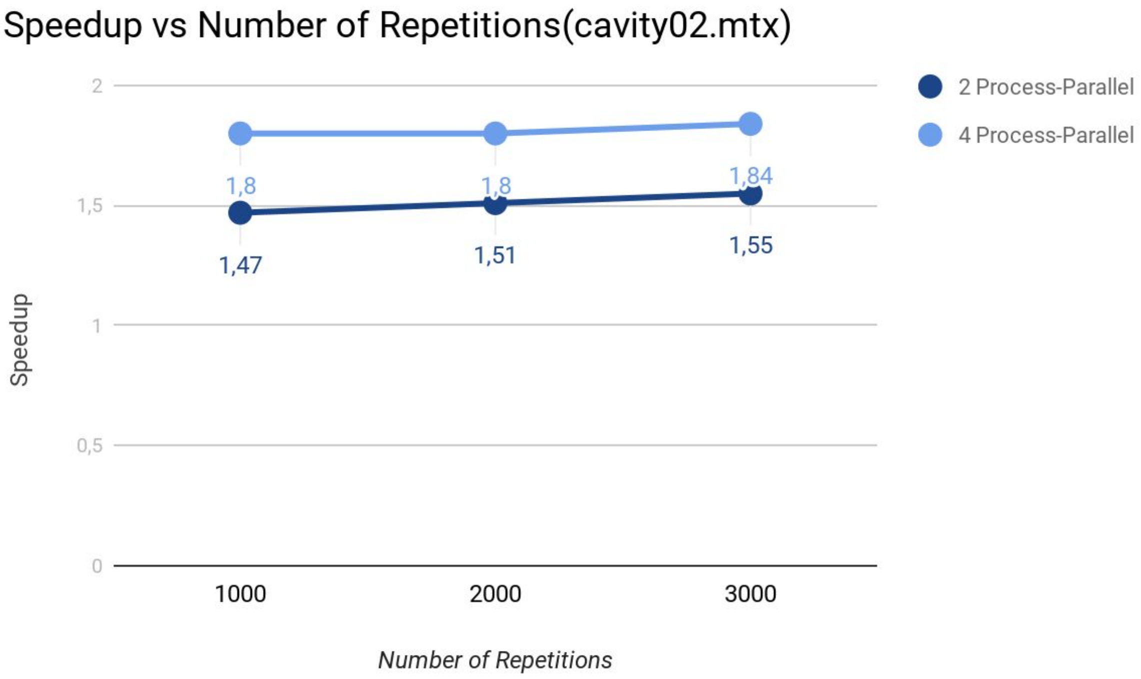

Figure 2: Speedup vs Number of Repetitions(cavity02.mtx)

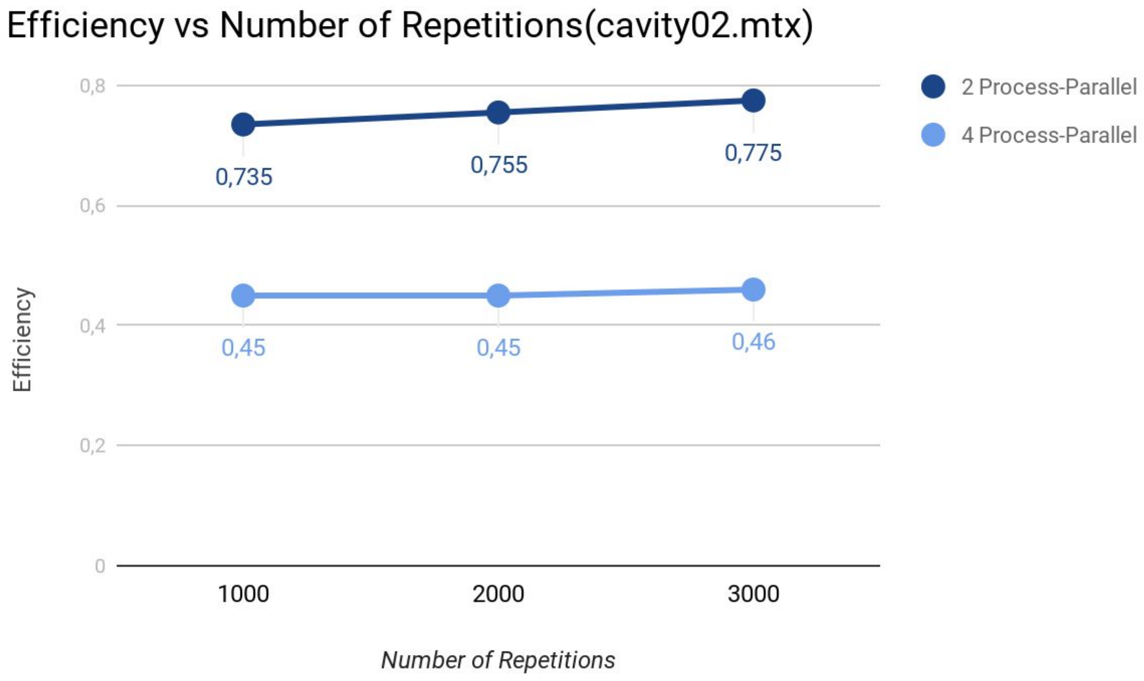

Figure 3: Efficiency vs Number of Repetitions(cavity02.mtx)

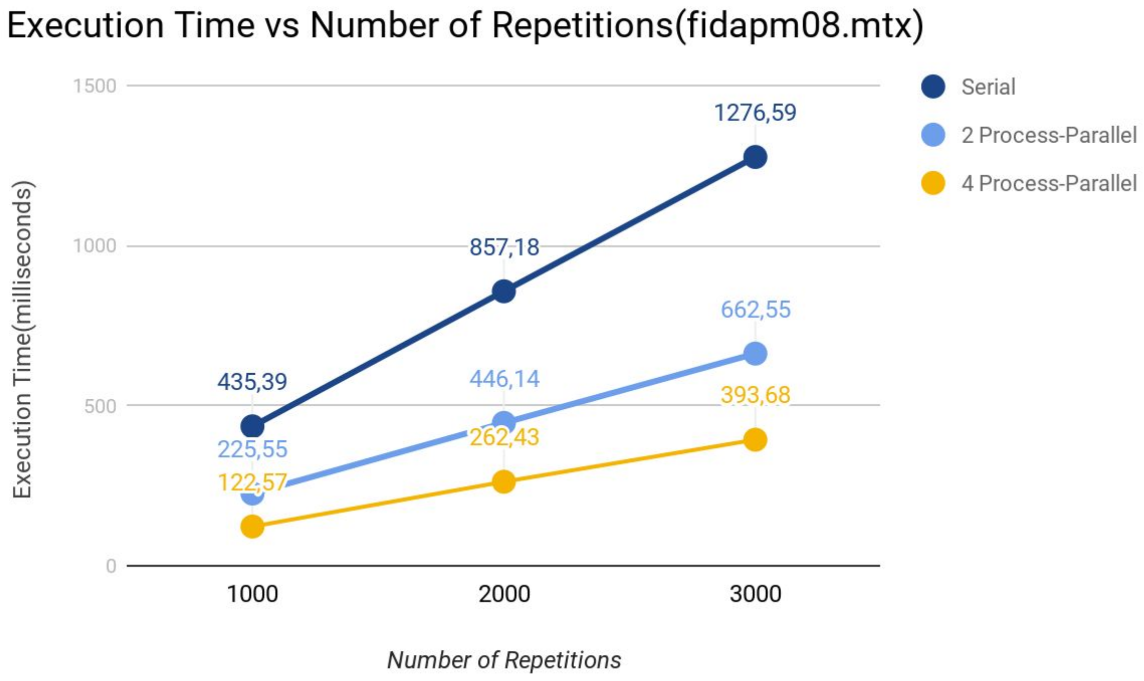

Figure 4: Execution Time vs Number of Repetitions(fidapm08.mtx)

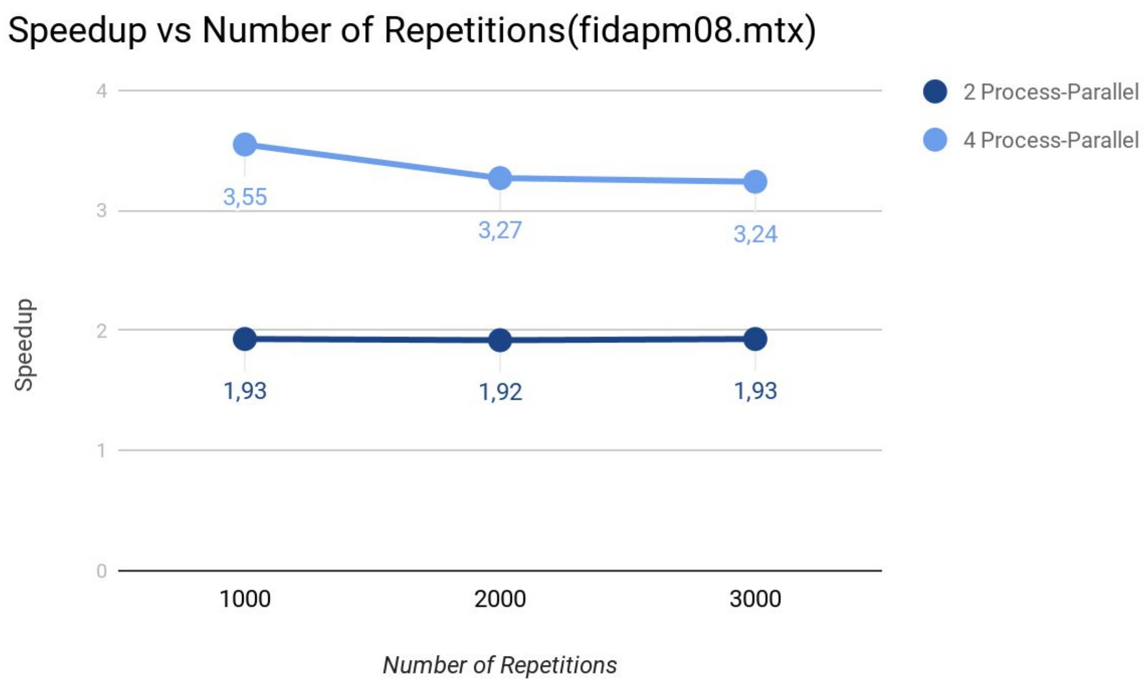

Figure 5: Speedup vs Number of Repetitions(fidapm08.mtx)

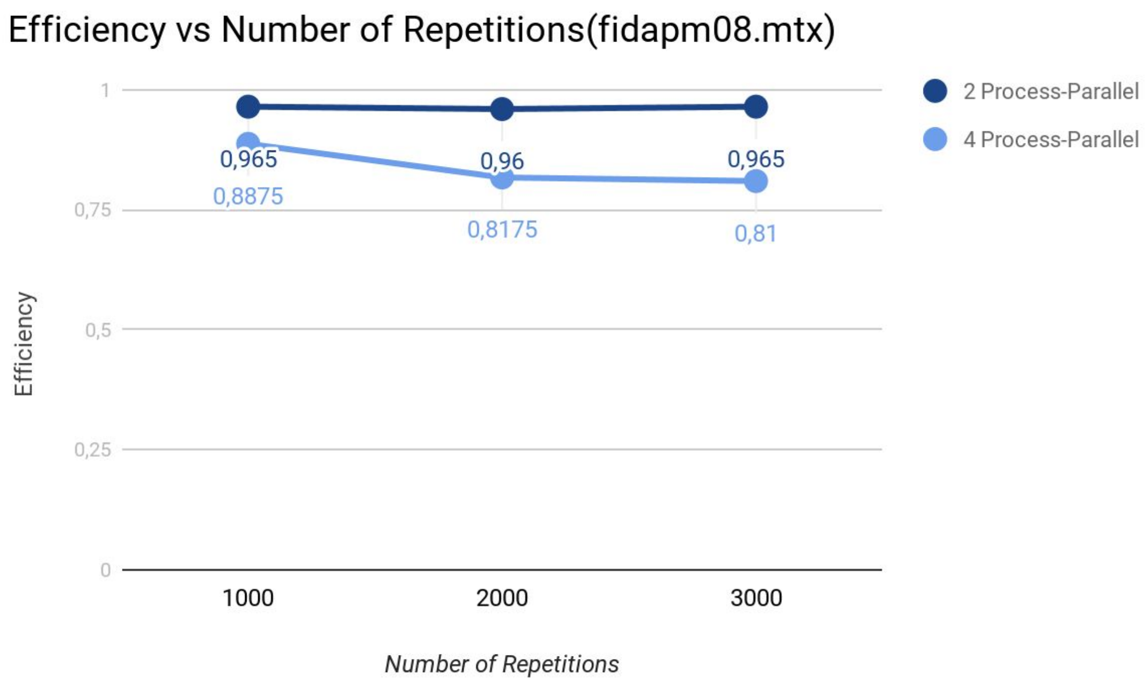

Figure 6: Efficiency vs Number of Repetitions(fidapm08.mtx)

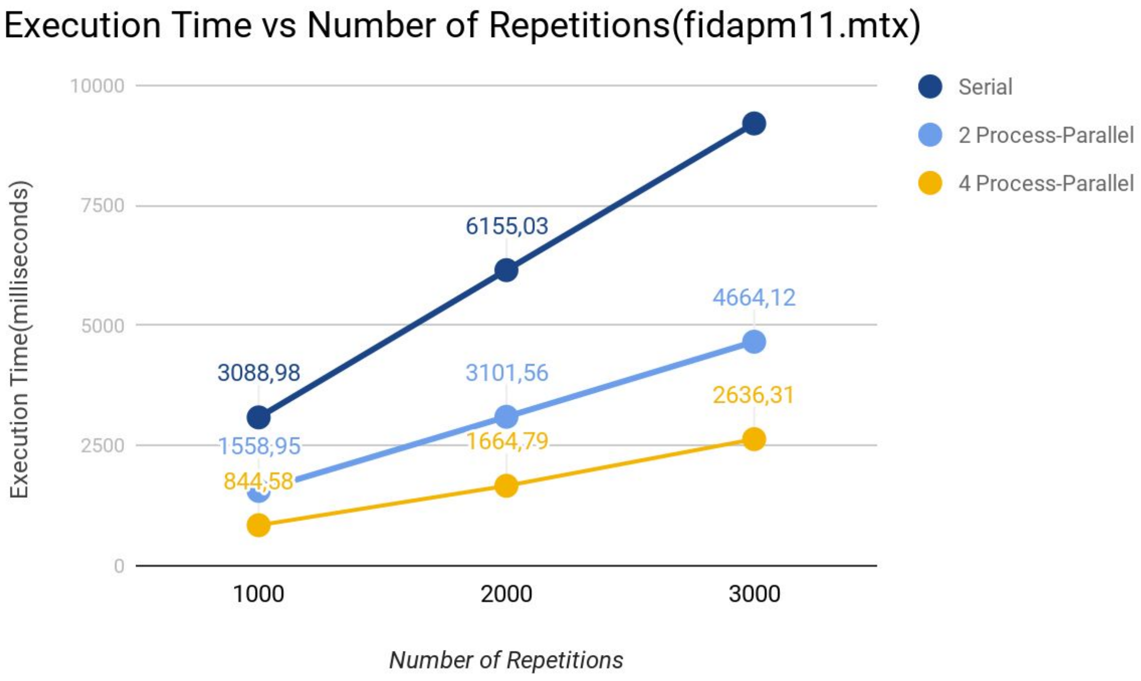

Figure 7: Execution Time vs Number of Repetitions(fidapm11.mtx)

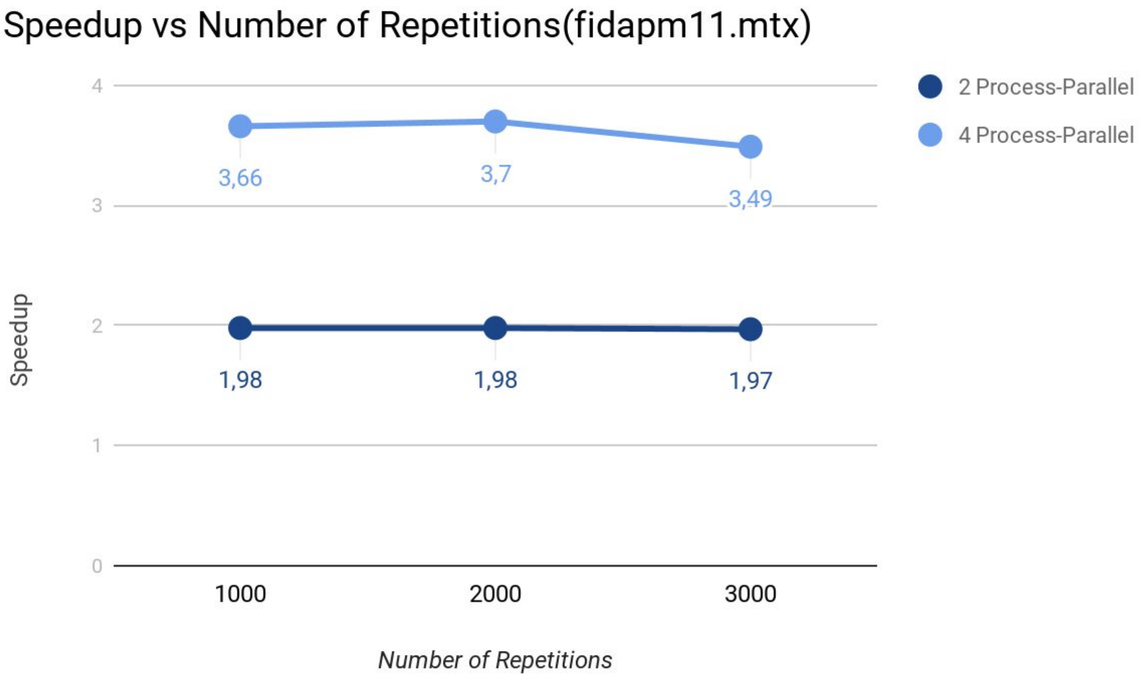

Figure 8: Speedup vs Number of Repetitions(fidapm11.mtx)

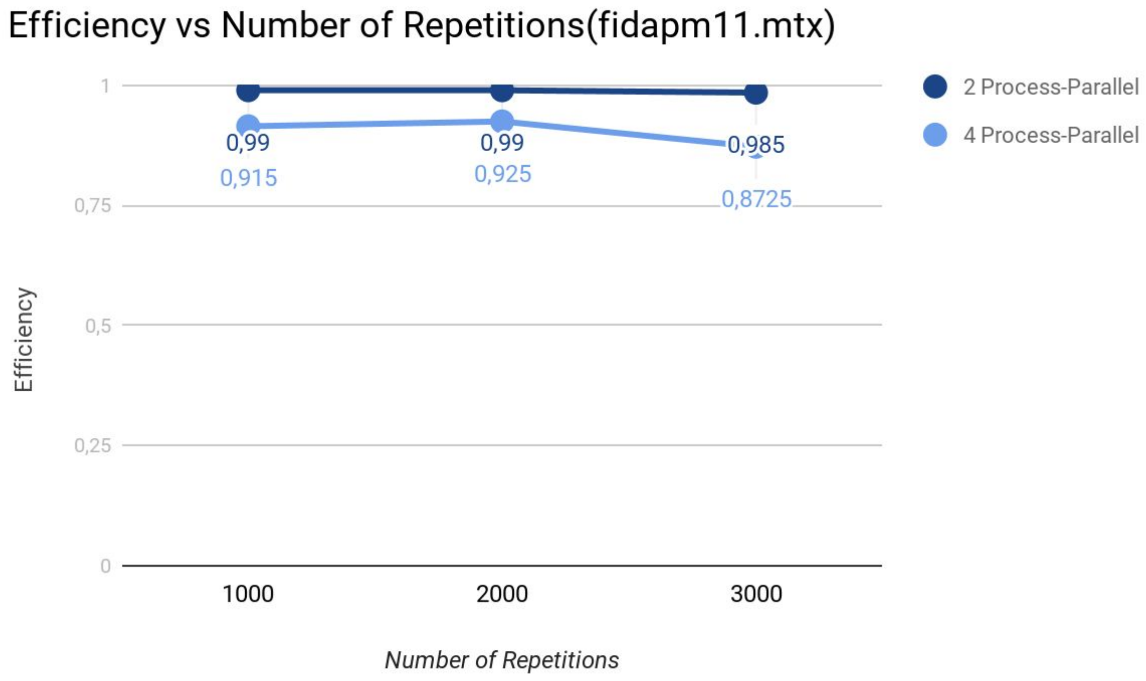

Figure 9: Efficiency vs Number of Repetitions(fidapm11.mtx)

3. Discussion of Experiments

After I made an investigation of the experiment results, I see that fidapm08.mtx and fidapm11.mtx input files gives expected results for the uniform distribution since their speedup values are really closed to their respectful process count. But cavity02.mtx input file gives poor results. I analyze the input file and see that rows in the one chunk contains a lot of non zero values while rows in the other chunks contains small amount of non zero values as I mentioned in the parallelization strategy and design choice part. For this input, dynamic scheduling can perform better but we can’t guarantee that since dynamic scheduling ruins data locality.