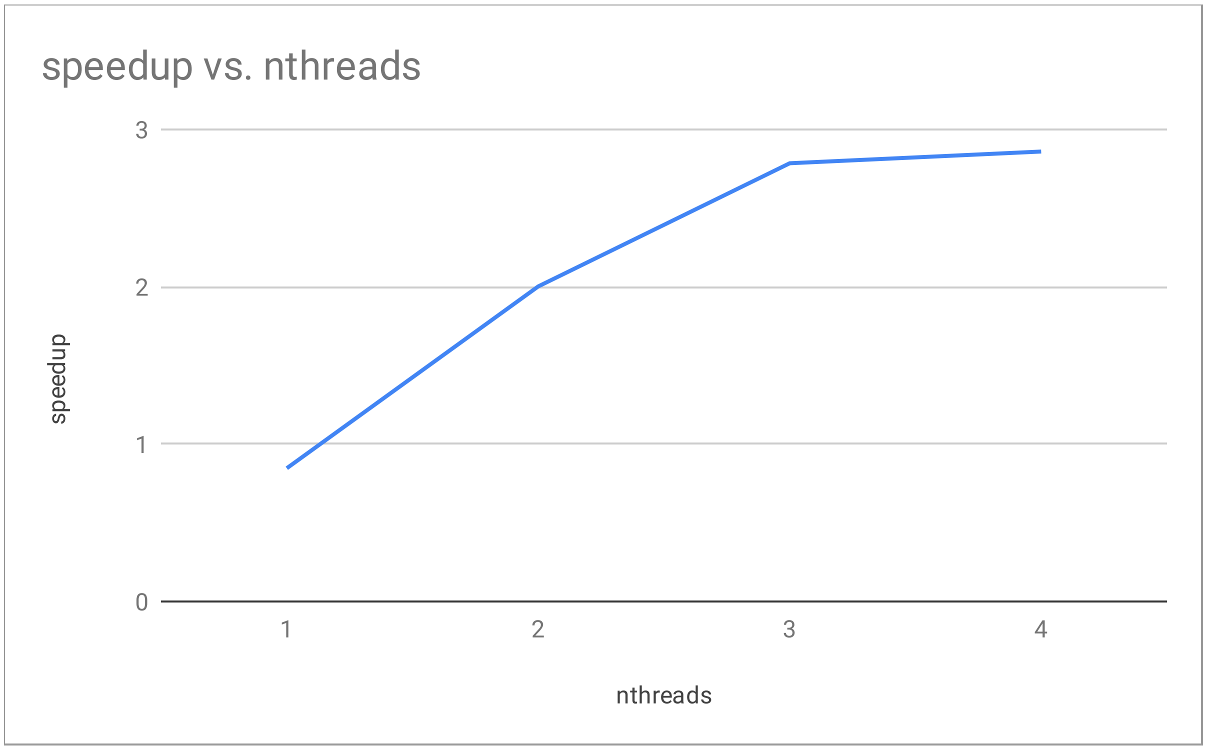
Parallel Prefix speedup with variable amount of threads

nthreads	speedup	work time(sec)
1	0.846231	0.32052
2	2.001758	0.135498
3	2.784375	0.097413
4	2.858823	0.094876



Discussion

From a theoretical perspective, the parallel prefix code should have a smaller speedup than the matrix multiply parallel method. First of all, int the parallel prefix code, due to the synchronization between threads, we have parts of the code that is serialized. This this did not happen in the matrix multiplication, threads did not have to wait for each other nor take turns. We know every time we enter a serialized portion, the speedup is affected. In addition, when creating the parallelized sum_prefixed algorithm, we ended up doing an algorithm that takes more steps (in total, if summing up all threads), than the original algorithm. This did not happen in the case of the matrix multiplication, where the parallel algorithm had no difference with the original algorithm

Nevertheless, data shows that the parallelization in p11 is better than in p4, speedup increases more significantly in p11 as we increase the number of threads. This could be explained due to the fact that the machine is not exclusively running this program, so the speed of each operation is always affected by the workload of the computer at the moment, which varies time to time. To take more rigorous empirical evidence we should work with a dedicated computer for the experiment.