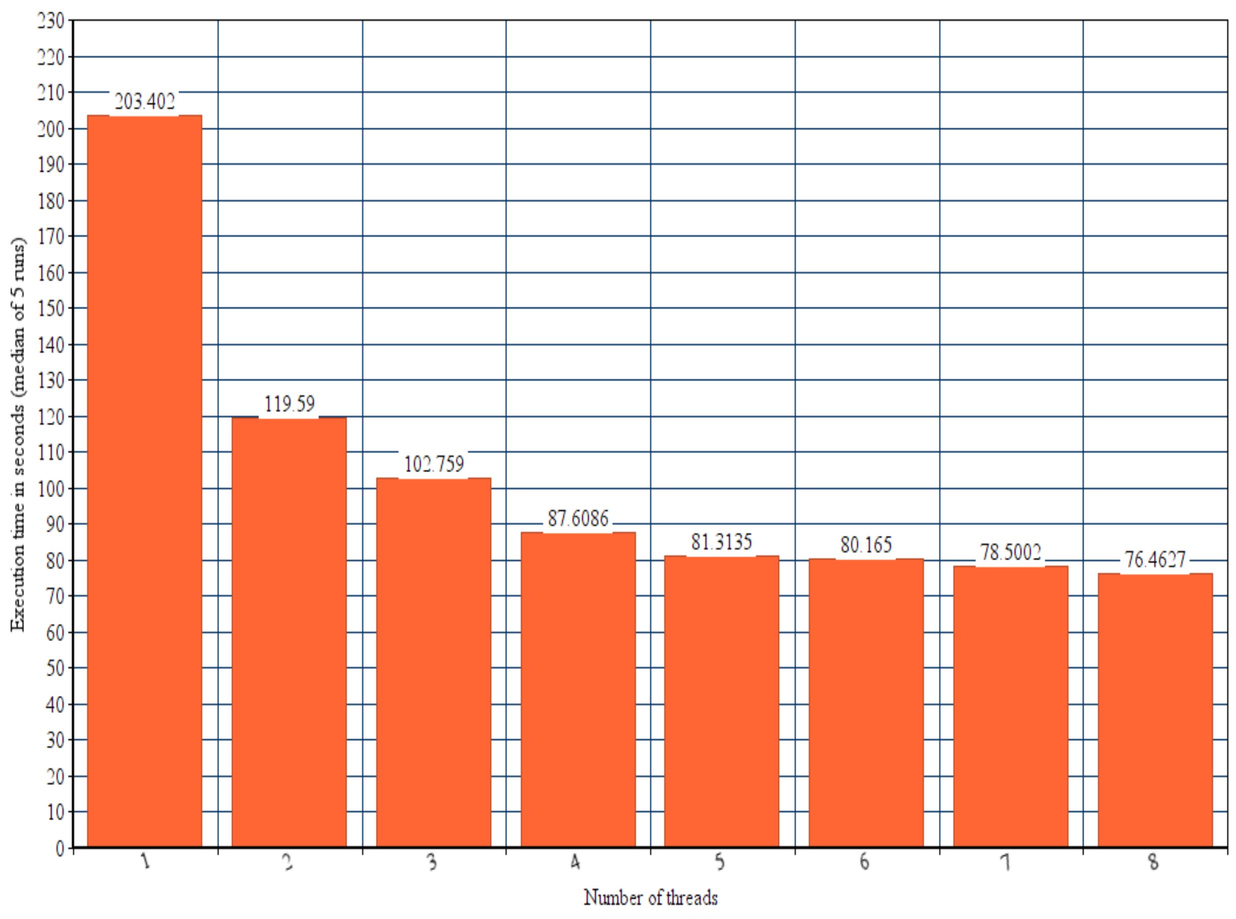
Shiva Besharat Pour
970922-0884
shivabp@kth.se
Homework 2

Report for parallel implementation of matrix sum, min and max with OpenMP:

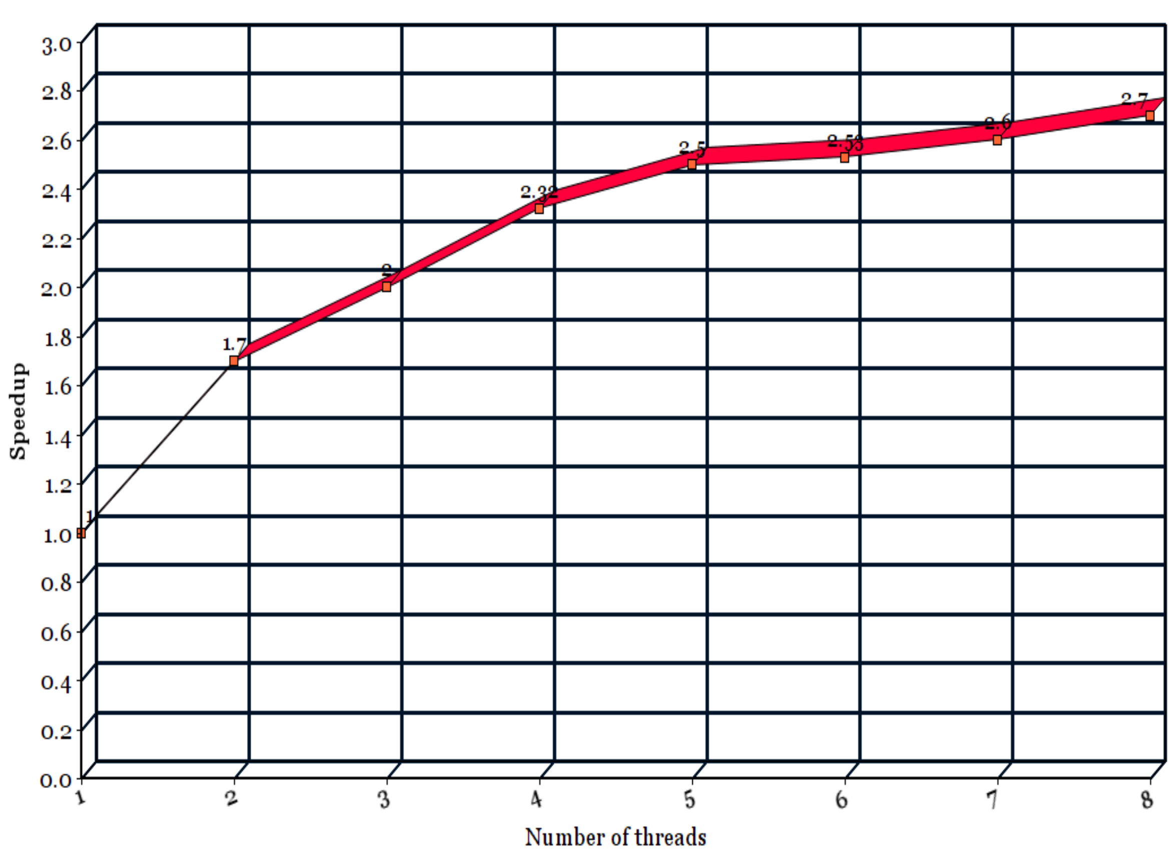
Measurements were taken on an intel core i5 processor totalling 8 hardware threads.
The implementation is highly parallelized and critical sections are minimized as much and possible therefore, taking into account measurement and calculation uncertainties, speedup is roughly linear.
However, every program has its upper bounds on the speedup it can achieve by increasing number of threads on a given problem size. This is because the probability of threads reaching critical sections simultaneously increases. Therefore, as shown in the graph below, as the number of threads closes 8, the difference in parallel execution time becomes less.

Execution time per thread for a matrix size of 100,000x100,000



Shiva Besharat Pour
970922-0884
shivabp@kth.se
Homework 2

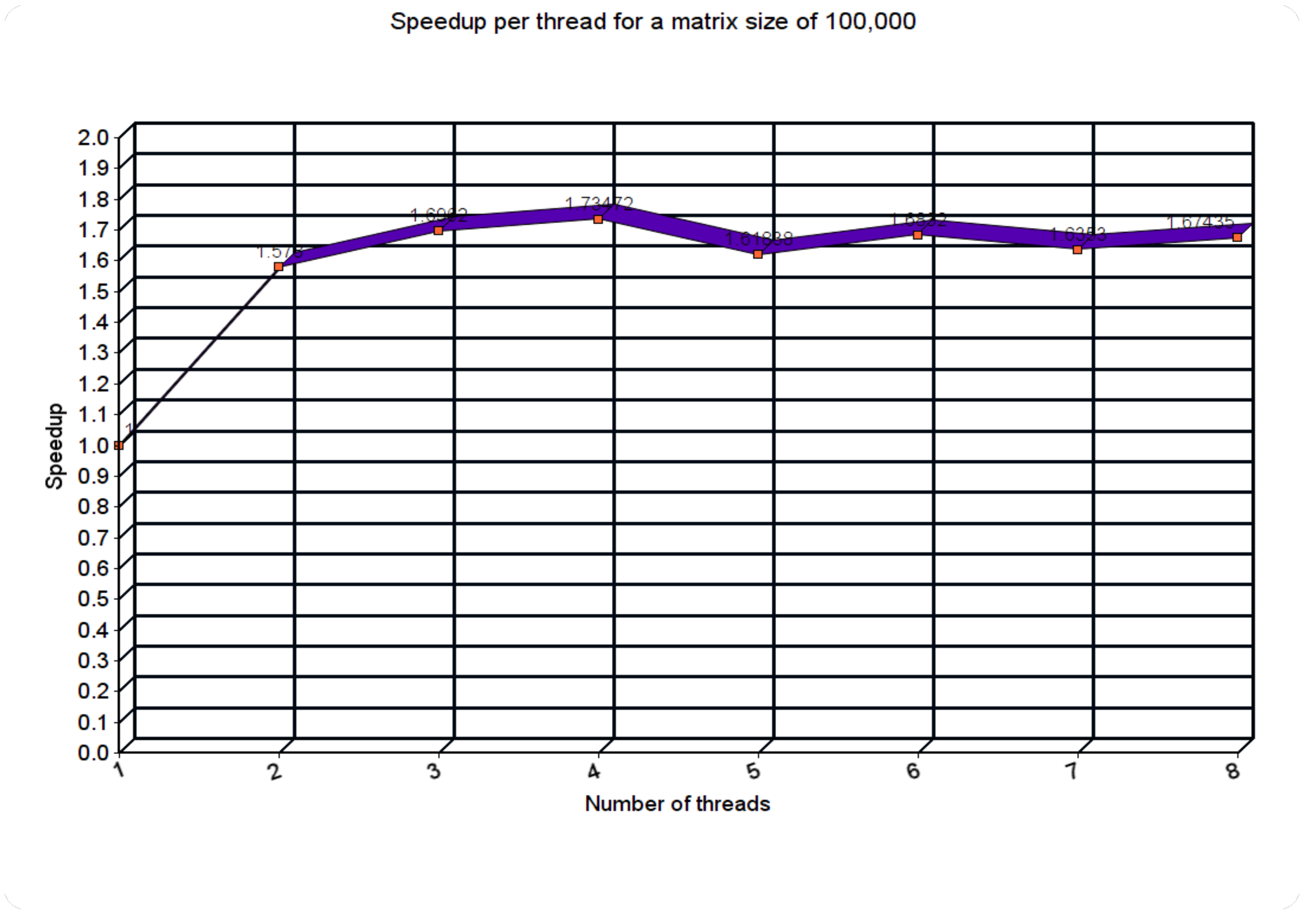
Speedup per number of threads used compared to the sequential execution



Shiva Besharat Pour
970922-0884
shivabp@kth.se
Homework 2

Report for parallel implementation of quicksort with OpenMP:

Measurements were taken on an intel core i5 processor totalling 8 hardware threads.
The implementation is highly parallelized and critical sections are minimized as much and possible therefore, taking into account measurement and calculation uncertainties, speedup is roughly linear.
However, every program has its upper bounds on the speedup it can achieve by increasing number of threads on a given problem size. This is because the probability of threads reaching critical sections simultaneously increases. Therefore, as shown in the graph below, once the parallel speedup has been achieved after the sequential execution with 2 threads, the speedup remains more or less constant.


Speedup per thread for a matrix size of 100,000
Speedup
Number of threads