Parallel Programming
Exercise 2
Mandelbrot

Student 1: Thomas Bründl

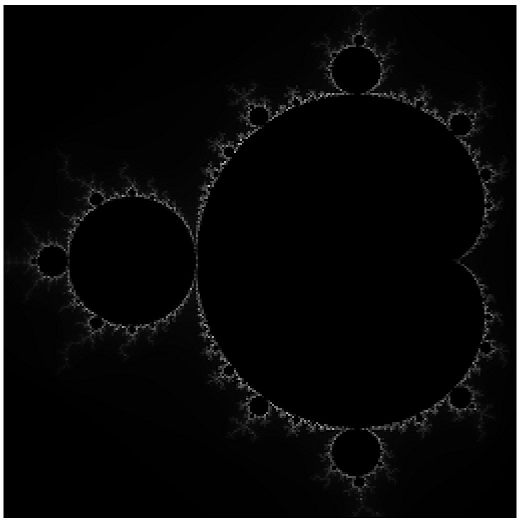
Student 2: Thomas Stummer

Approach

We measured the time required to perform the necessary calculations for the Mandelbrot set.

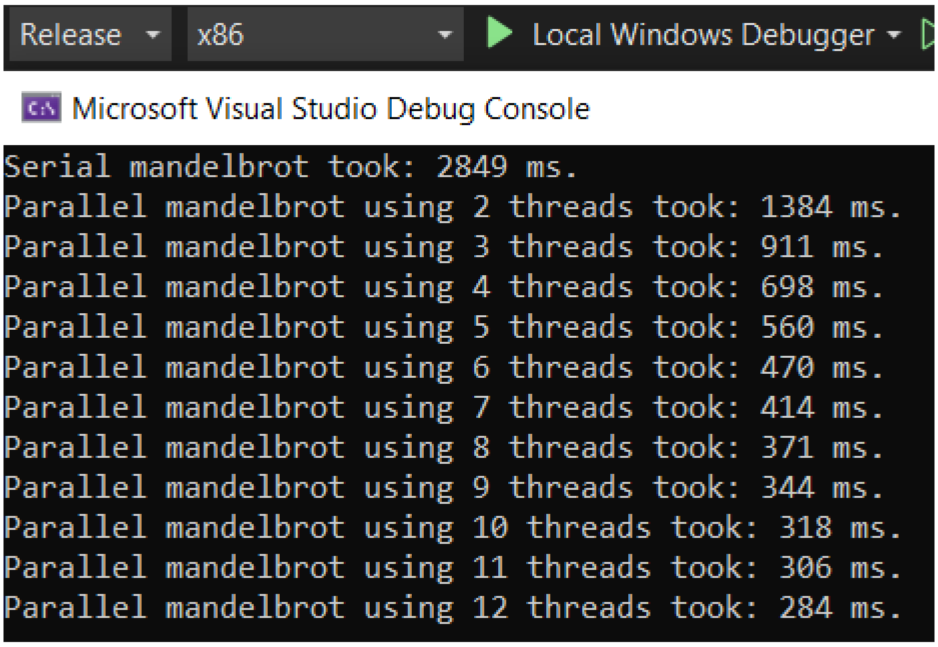
For the serial base line we implemented two nested for loops iterating over each row (outer for loop) and each column (inner for loop). In every iteration the color of one pixel is calculated.

For the parallel implementation the generall general strucutre with nested for loops remained unchanged. We decided to to assign each row to one thread in round robin style. So if e.g. 6 threads were used, thread 0 got rows 0, 6, 12, 18, etc. and thread 1 got rows 1, 7, 13, 19, etc. and so on. We took this approach in the assumption to split the work that needs to be done amount the different threads in an even way (in contrast to assigning thread 0 the first x rows, thread 1 the next x rows and so on). To distribute the work, we used the *omp parallel* pragma because for us it seemed the most straight forward way to assign a sub set of fields of the shared vector v to each thread that can be filled intependently. With this approach no manual merging of data was necessary.

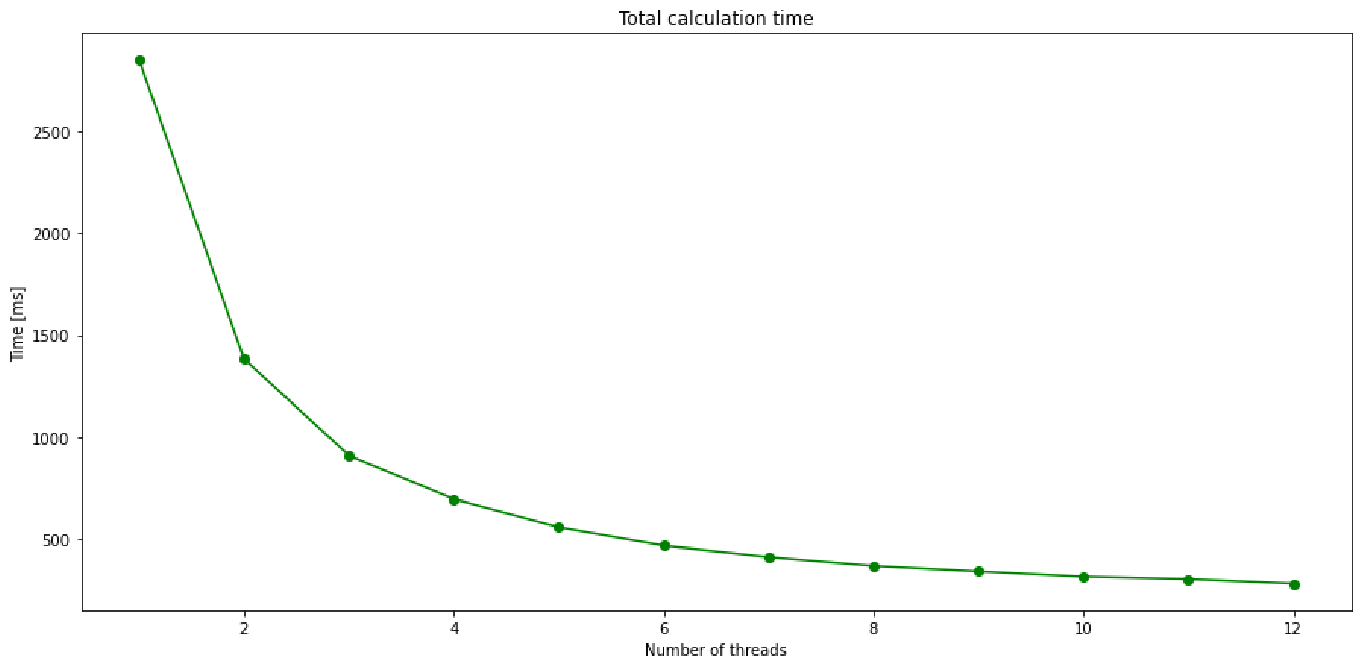


Speedup Analysis

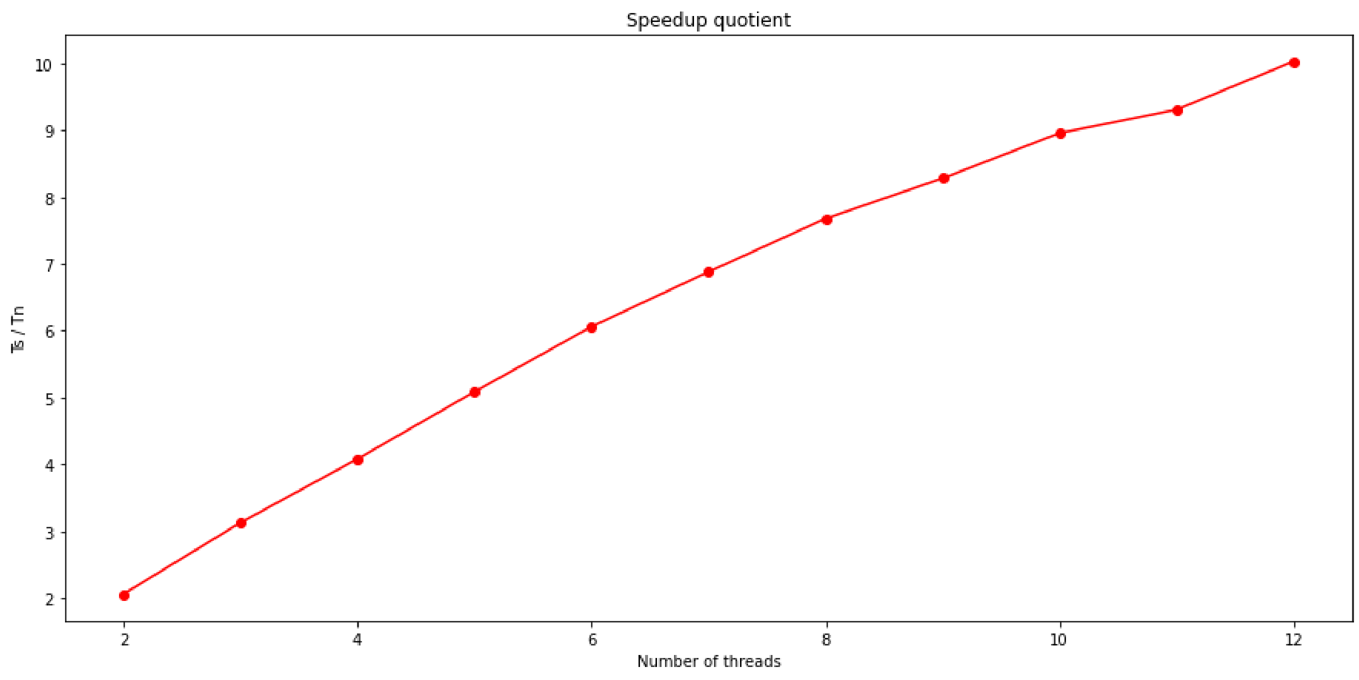
The following benchmark was performed with an image resolution of 1024 x 1024 pixel on a machine with 12 logical CPU cores:



```
Serial mandelbrot took: 2849 ms.
Parallel mandelbrot using 2 threads took: 1384 ms.
Parallel mandelbrot using 3 threads took: 911 ms.
Parallel mandelbrot using 4 threads took: 698 ms.
Parallel mandelbrot using 5 threads took: 560 ms.
Parallel mandelbrot using 6 threads took: 470 ms.
Parallel mandelbrot using 7 threads took: 414 ms.
Parallel mandelbrot using 8 threads took: 371 ms.
Parallel mandelbrot using 9 threads took: 344 ms.
Parallel mandelbrot using 10 threads took: 318 ms.
Parallel mandelbrot using 11 threads took: 306 ms.
Parallel mandelbrot using 12 threads took: 284 ms.
```



This leads to the following speedup quotient:



It can be seen, that the speedup quotient increased very linear until about 6 threads. From 6 to 8 threads a very slight decrease in the slope can be observed. At 10 threads another decrease in the slope can be seen. When using 12 threads, the slope (surprisingly) increased again. In summary, the speedup quotient is quite quite linear until the number of threads used reaches nearly the maximum number of available (logical) CPU cores.