ETH zürich

High-Performance Computing Lab for CSE **2020**

Student: Konrad Handrick Discussed with: -

Solution for Project 2

Due date: 23.03.2020, 12:00pm

This project will introduce you to parallel programming using OpenMP.
In general, I used Intel's icc compiler with -fopenmp -std=c++17 (if possible) -O2

To do the testing I used the settings provided with the gcc compiler.

1. Parallel reduction operations using OpenMP [10 points]

It is very evident from back of the envelope calculations that an implementation using a critical section will not be very beneficial for either several threads or large problem sizes: the context switch overhead sums up to make the calculation highly inefficient, up to several orders of magnitude. For rather small problem sizes, a sequential version of the dot product implementation will suffice since the overhead that for example cache evictions introduce will degrade performance. The plot underlines these findings that could be anticipated before even starting the implementation.

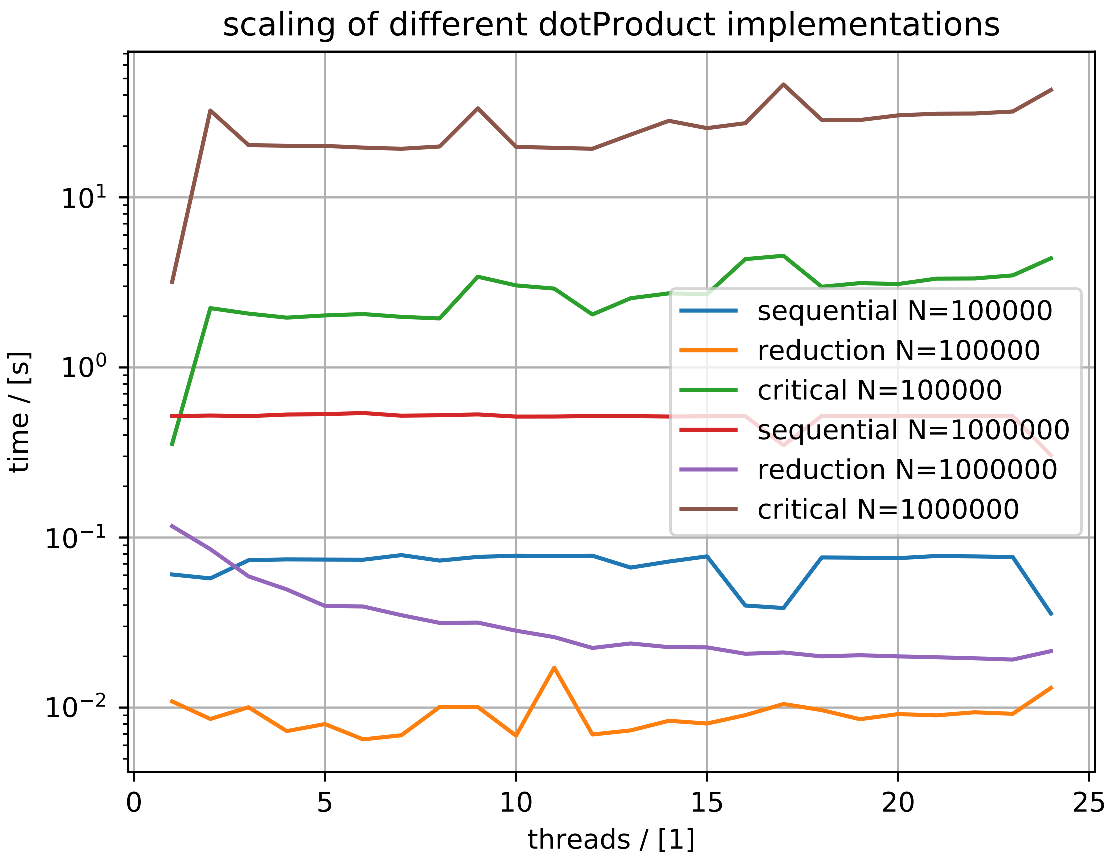

Figure 1: Dot product scaling

2. The Mandelbrot set using OpenMP [30 points]

The seemingly trivial parallelization of the Mandelbrot iteration took me quite some time and even though it seems correct, the iterations of the sequential and the parallelized versions do not match - there seems to have been a data race introduced. Apart from the little incorrectness hickup the parallelization seems to work as expected.

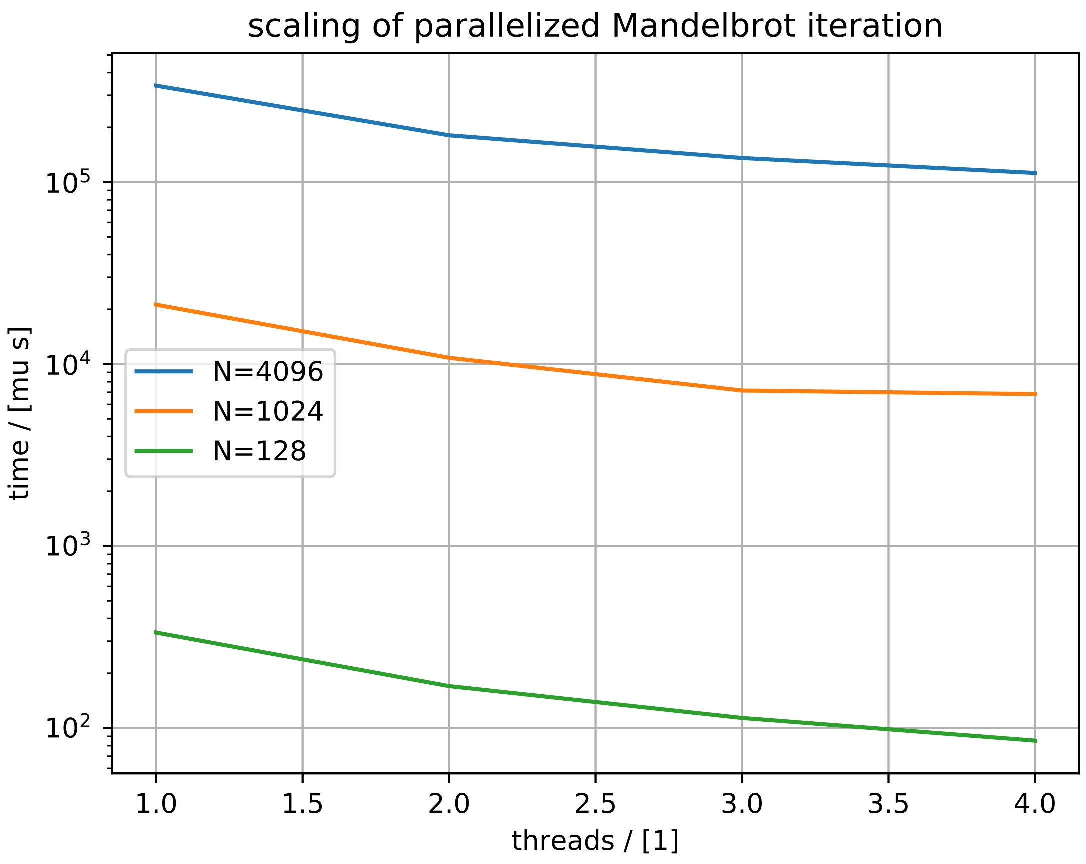

Figure 2: Mandelbrot iteration sclaing

3. Bug hunt [20 points]

Bug 1: tid needs to be put inside the parallel section as the given function call is only valid inside an OMP section.
Bug 2: tid needs to be put as private, else it does not do the job.
Bug 3: The nowait needs to be commented out. Also, the barrier inside the function is one barrier too much which yields a deadlock.
Bug 4:The per default allocated stack for OMP environments is not sufficient and thus needs to be changed manually, for example with ulimit -s ”larger value” - this is a sensitive task that changes enviroment variables such that some commands may not work / work differently anymore.
Bug 5: To respect locking order the lock on (original file) line 60 (locka) needs to be put before the lockb several lines above. Else, we arrive at a deadlock situation.

4. Parallel histogram calculation using OpenMP [20 points]

Just putting a "parallel for" around the loop yields a **false sharing** situation. That's why we need to introduce a local bin array that does bookkeeping and needs to be reduced in the final iteration.

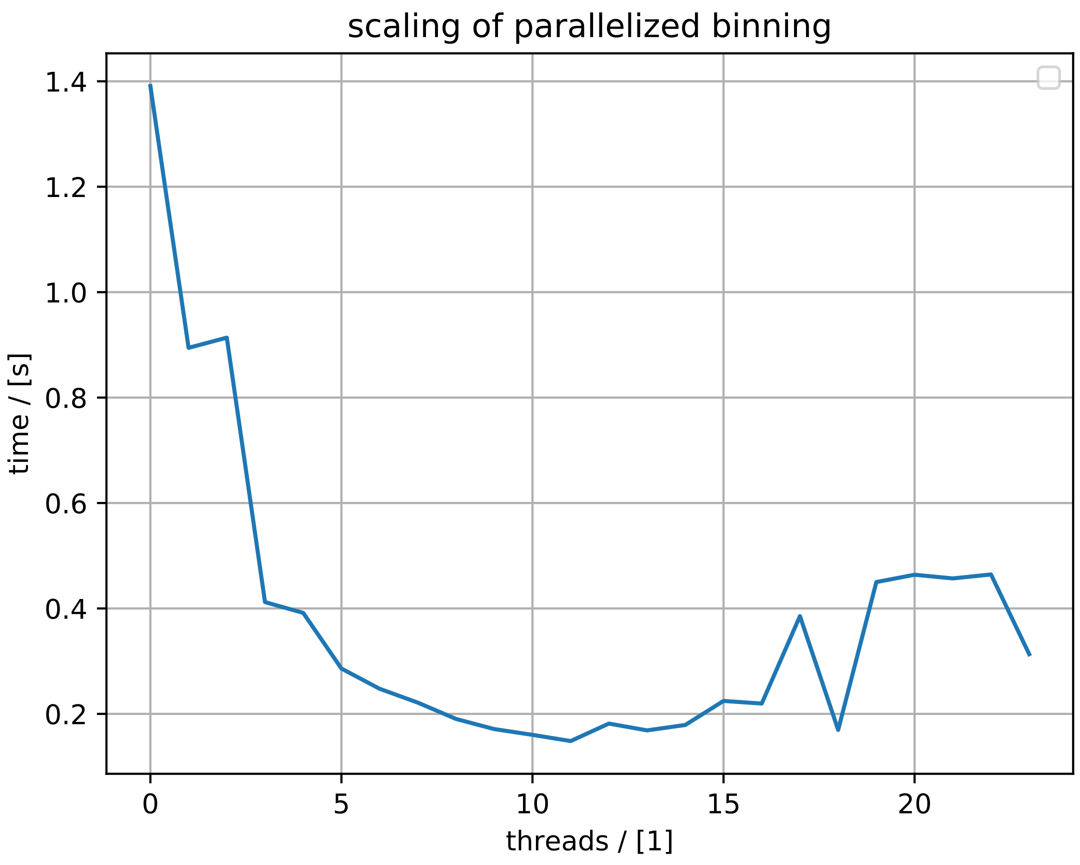

Figure 3: binning scaling test

5. Parallel loop dependencies with OpenMP [20 points]

In this iteration we can exploit the closed form expression and just rearrange the iteration such that we introduce an offset that is applied iteratively. This is probably not the most general and good solution but in this situation it is an adequate measure to take since it scales well and is correct.