Oblig 5, IN3030

What is this rapport about?

This rapport will describe how I implemented the parallel solution to the convex hull, and what measurements it had against the sequential solution.

User guide

To compile/build the program use javac *java.

To run the program run the Main file with <N> <SEED> and <0> or <1>.

If the last argument is 0 no output will be written to file and the convex hull will not be graphically shown on screen. If its 1 the output will be recorded to files, one for parallel and one for sequential solution, and the graphs will be shows on the screen (if n < 10.000).

Example without recording: java Main 1000 777 0

Example with recording: java Main 1000 777 1

Note that if you supply a large N, the generation of the points can take a really long time, before the actual tests of the solutions starts.

Parallel solution

My sequential solution is very similar to the precode/livecode sessions code. My parallel solution is a modified version. At first the main thread launches as many threads as the system has, to find the min and max x values (they search a small part of the point array each). Afterwards the threads find the two other points furthest away, in their part of the points array, splitting the convex into 4 parts. At this point 4 points have been found. The threads terminate. The main thread now launches 4 threads to start the recursion (they get assigned one of the four parts of the graph). The threads will start looking for points. Once a point is found, the thread will see if there is any free threads (threads alive < max threads), if this is the case a new thread will continue the left recursion while the first thread will continue the starts on the right side recursion. If there are no free threads, the thread will first do the left side recursion then the right-side recursion.

To record the points that are found, the program will create two trees. A left side and a right side. The first 4 threads will be assigned either the left or the right node of one of these trees, and will assigned new right and left pointers every time a new thread is launched. At first I tried to just add the points to the convex envelope (the points found). This recorded the correct points, but in the wrong order. So the graphical representation of the convex hull was wrong. By using a tree, I can go trough step by step, and adding the points to the final convex envelope in the correct order.

Tests

To test the program, I simply tested many N and SEED values, and compared the output of the sequential and parallel solution. However, for some reason the first run of my parallel implementation is significantly slower then the 6 iterations after (I run 7 tests, and record the median).

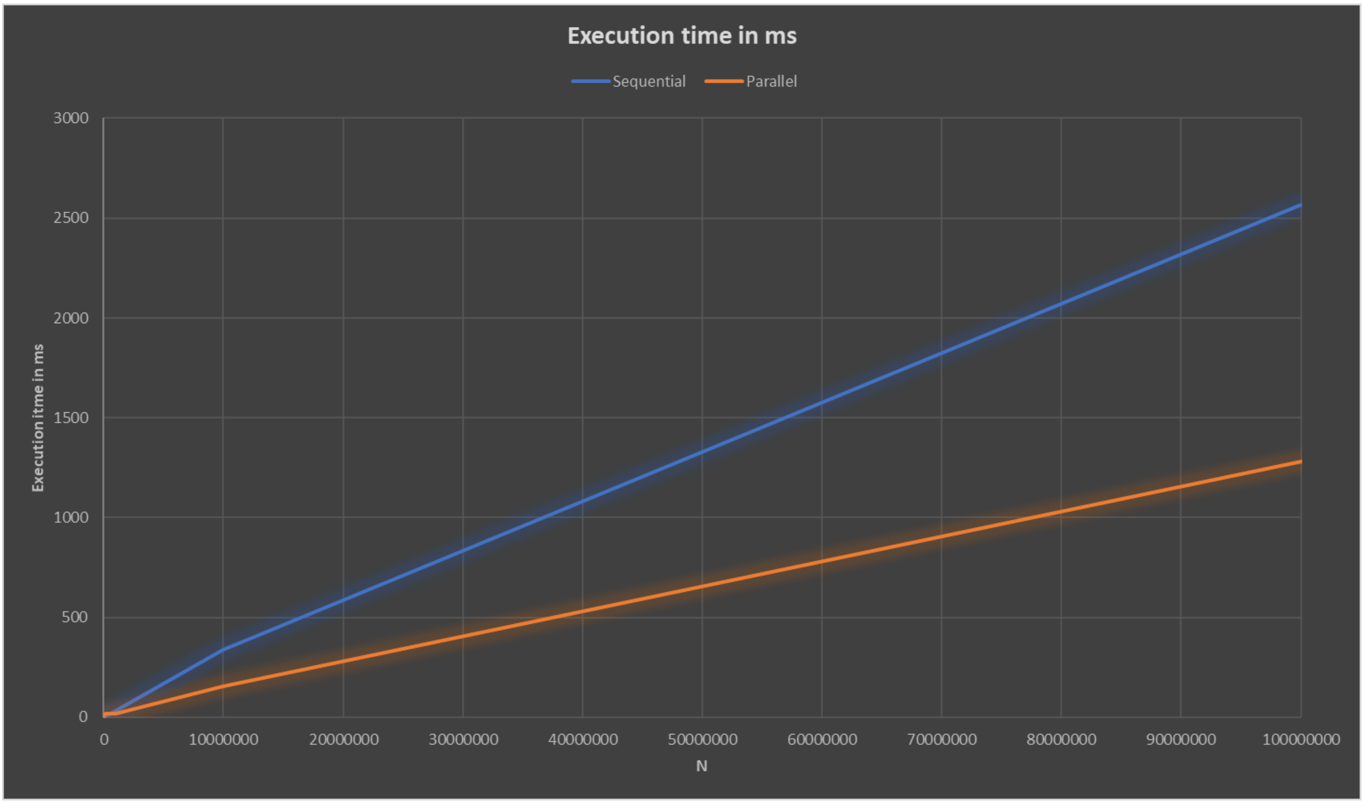
Measurements

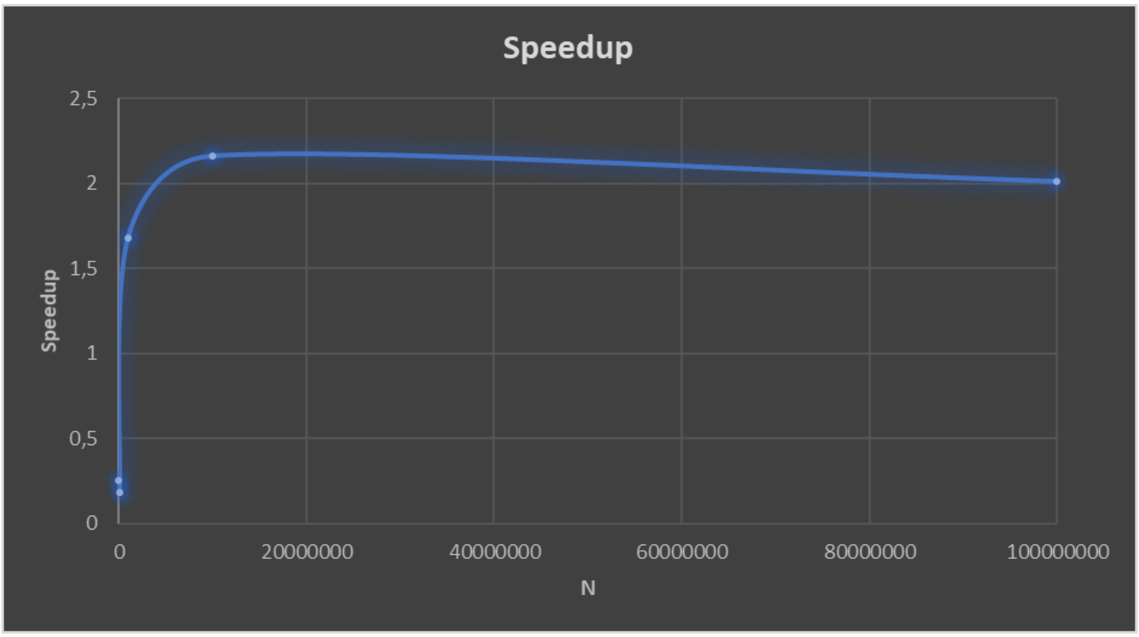
These results are the median values for 7 runs of the sequential and parallel solution with seed = 777. The timings are all in ms.

These results are recorded on a 6 core laptop with: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz and 32 GB on ram. I did however see larger speedups on my 16 core desktop, with 64GB ram. As you can see here, the speedups maxed out at around 2.2. On my desktop however, I saw speedups up to 3.4, but again that machine has much more and much faster cores.

Again, when you test these extremely large values, the N-point generation will take a really long time. As you can see from the table and graphs below, the speedup for the parallel solution is stable and maxes out at around 2, when N is sufficiently large enough.

N	Sequential	Parallel	Speedup
10000	1ms	4ms	0.25
100000	3ms	17ms	0.18
1000000	32ms	19ms	1.68
10000000	339ms	157ms	2.16
100000000	2567ms	1280ms	2.01





Conclusion

To summarize, I achieved speedups of around 2. It's a decent speedup, but I don't think my solution is very efficient. But I don't see any other way to solve it. It was way faster when the threads just added the points, they found to the final point list, but the order was all wrong.

Sources for implementation:

https://github.com/DSharifi/IN3030-Effective-Parallel-Programming/tree/master/oblig%205

https://github.uio.no/shielak/in3030/tree/master/oblig_sequential_solutions/oblig5

https://github.uio.no/shielak/in3030/tree/master/livecoding/week14

https://www.uio.no/studier/emner/matnat/ifi/IN3030/v19/lecture-material/uke-14/uke14-v19.pdf

http://www.nik.no/2010/01-Maus.pdf