Exercise 3

1. Implement a parallel version of the Hadamard product using OpenMP. Use the snippet below as sequential implementation.

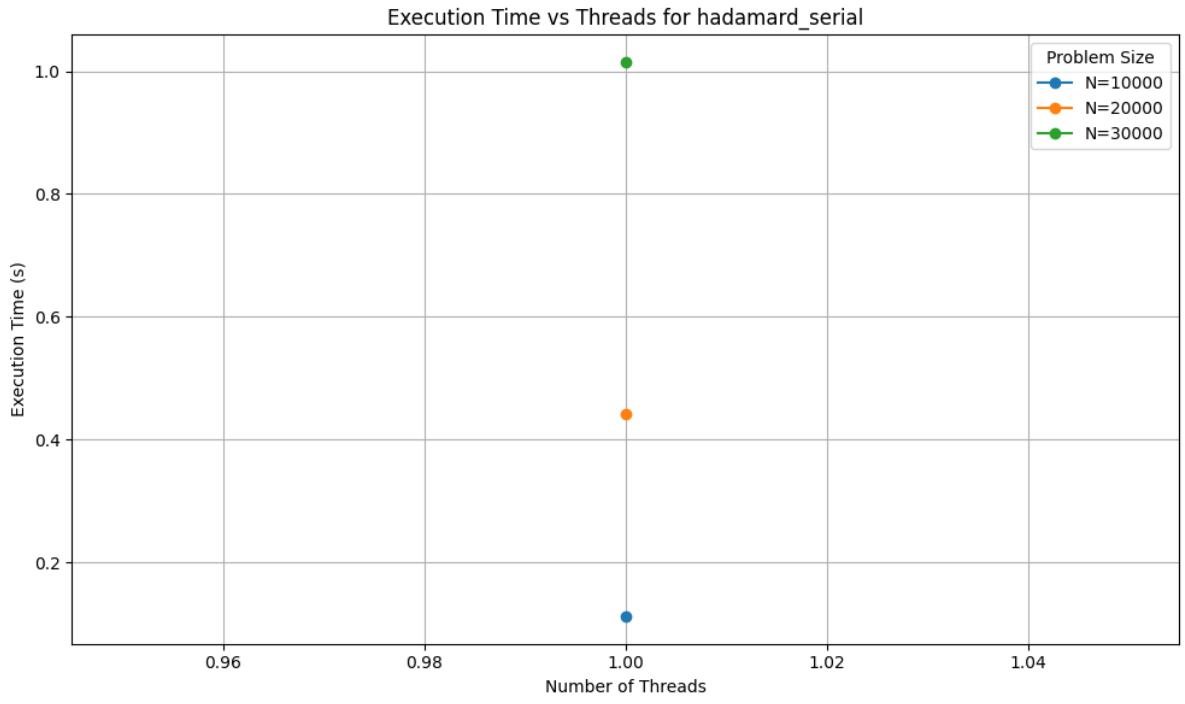
```
for (size_t i = 0; i < n; ++i) {
  for (size_t j = 0; j < n; ++j) {
    c[i][j] = a[i][j] * b[i][j];
  }
}
```

2. Use the loop scheduling methods discussed in the lecture, static, dynamic, guided and auto. Explain their differences and compare their performance for both the Hadamard and Mandelbrot OpenMP implementations. What can you observe?

For the Mandelbrot implementation look at my answer for exercise 2.

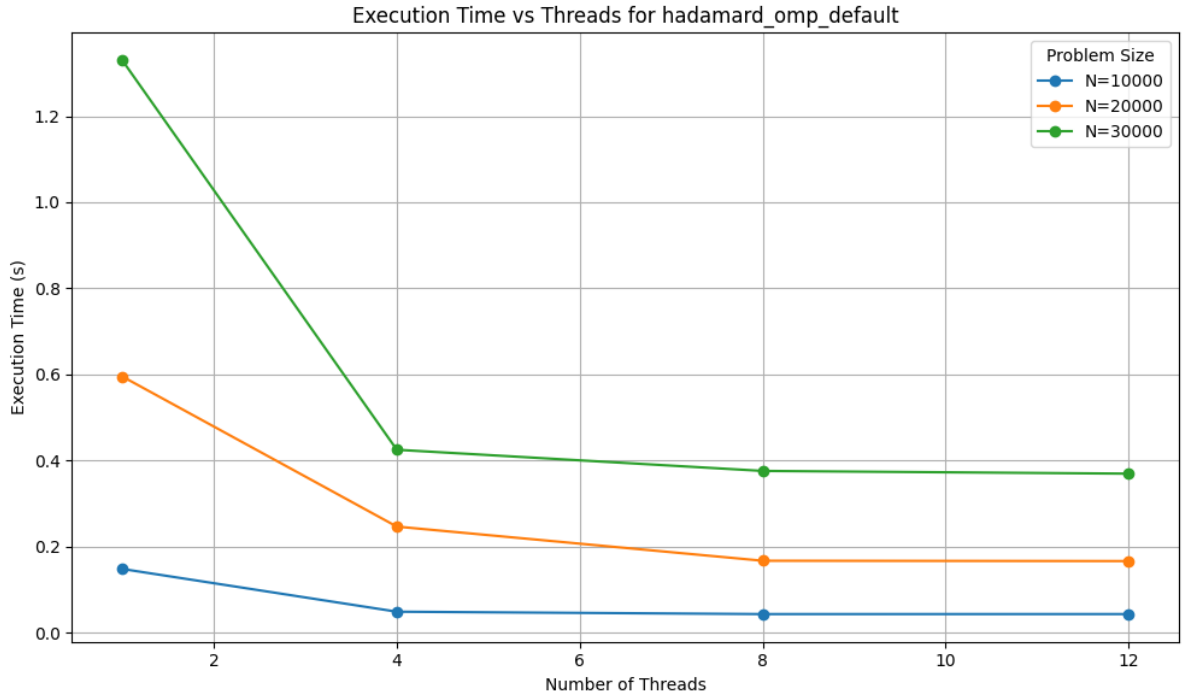
The assumption is, as this task should be highly predictable as the multiplication should always take approximately the same time, static scheduling should be the fastest

serial



Default (schedule(static,1))

-assigns 1 iteration to each thread compile-time until no iterations are left

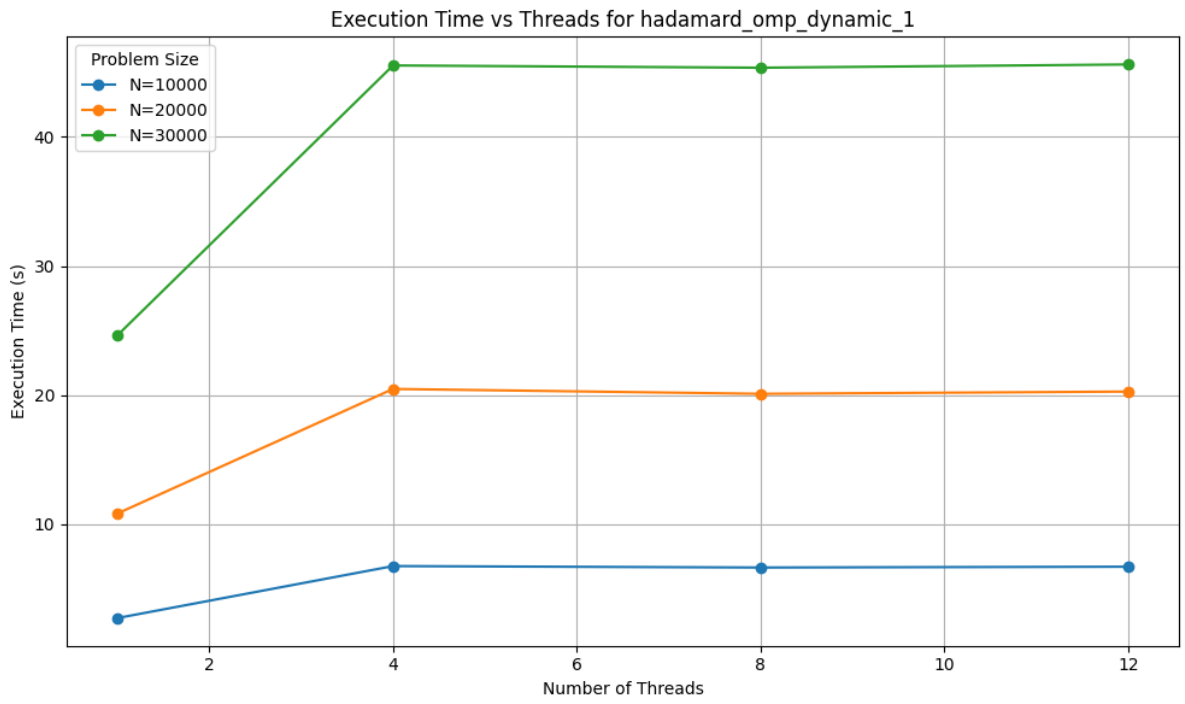


Schedule(static); equivalent to schedule(static, loop_count/number_of_threads)

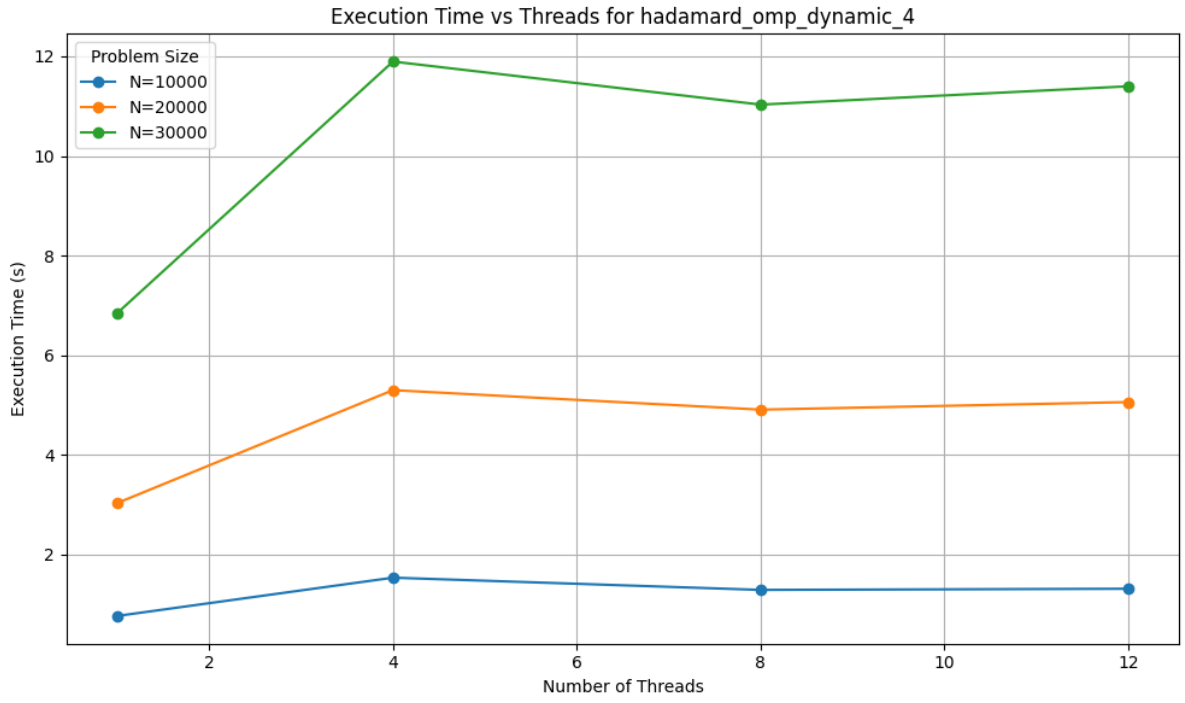
Schedule(dynamic [size])

-assigns size iterations to each thread available at runtime until no iterations are left

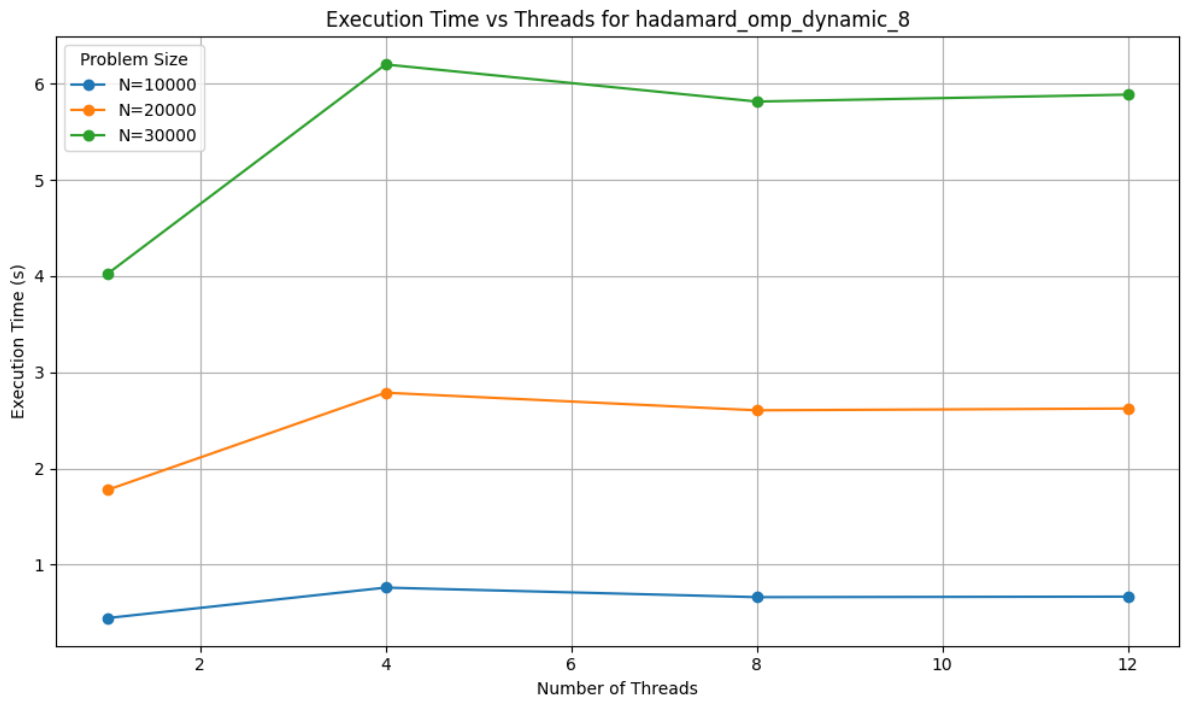
Dynamic,1



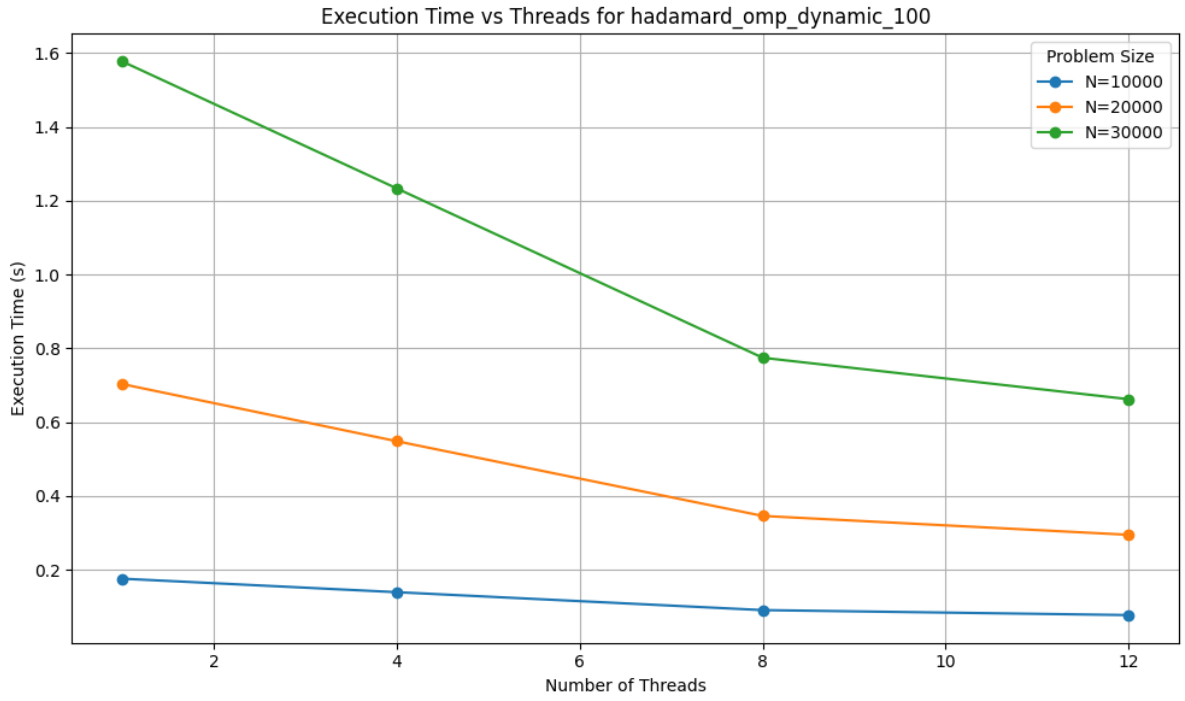
Dynamic, 4



Dynamic, 8

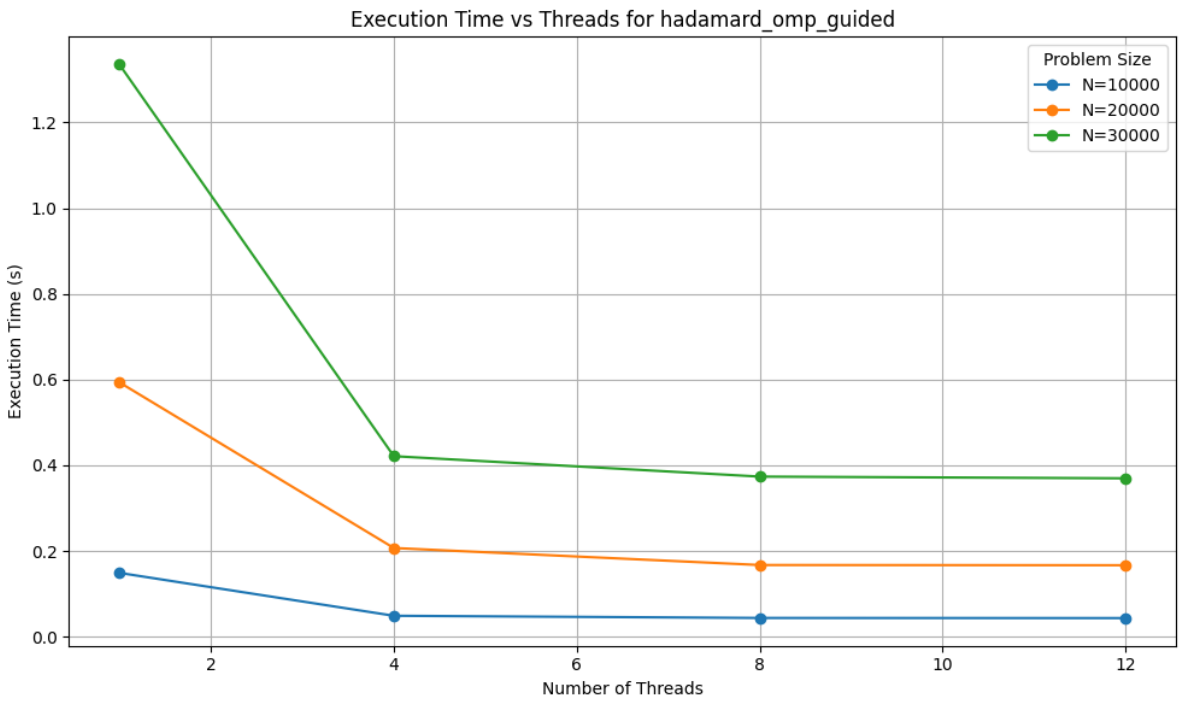


Dynamic, 100



Schedule(guided)

-special case of dynamic scheduling which assigns more iterations in the beginning and then decreases the amount to reduce overhead

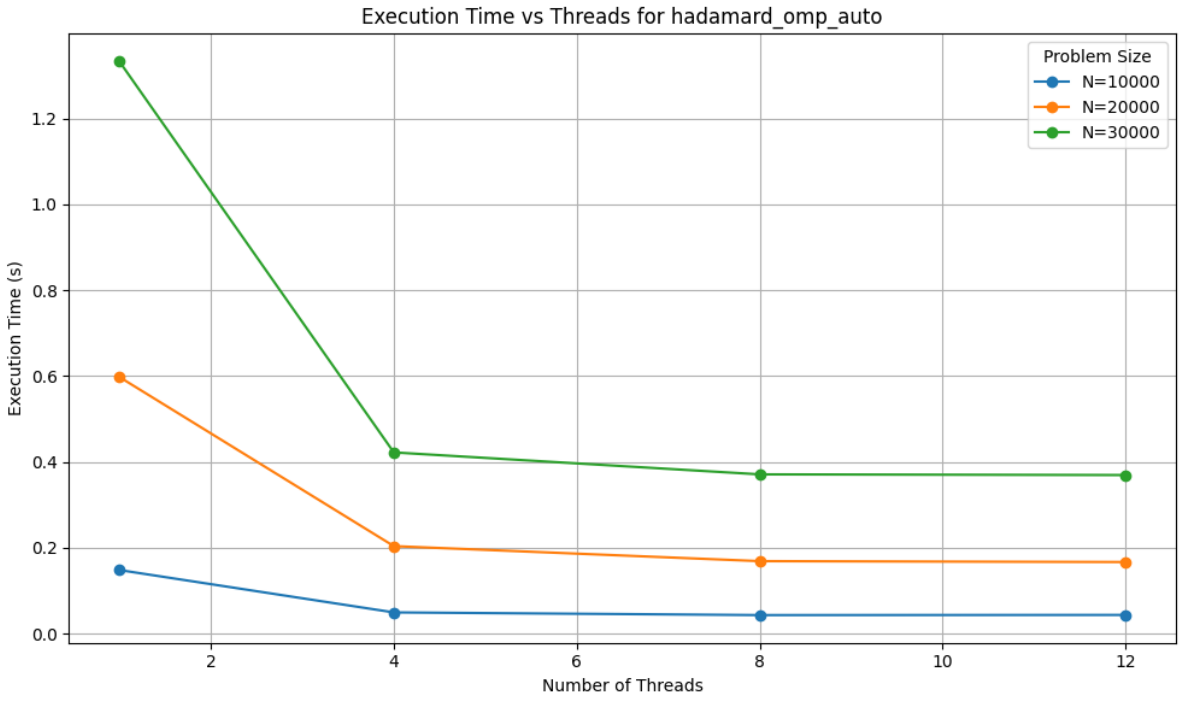


Schedule(auto)

-runtime-based: learns from previous iterations of the same loop / up too the runtime system to choose scheduling policy, but doesn't have to be one of the above

-"auto: The decision regarding scheduling is delegated to the compiler and/or runtime system."

(https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf)



3. In addition, try the loop scheduling methods auto and runtime. What do they do, what can you observe?

Auto is explained in (2.),

Runtime scheduling schedules according to env variable OMP_SCHEDULE:

-OMP_SCHEDULE=”dynamic”,
-OMP_SCHEDULE=”GUIDED,4“

“[4.1] [4.1] OMP_SCHEDULE type[,chunk] Sets the run-sched-var ICV for the runtime schedule type and chunk size. Valid OpenMP schedule types are static, dynamic, guided, or auto.”
(later versions also support monotonic/nonmonotonic as modifier)

(https://www.openmp.org/wp-content/uploads/OpenMP-4.0-C.pdf)

“-monotonic: Each thread executes its assigned chunks in increasing logical iteration order. Clauses schedule (static) or order imply monotonic.
-nonmonotonic: Chunks are assigned to threads in any order. Behavior of an application that depends on execution order of the chunks is unspecified.”

(https://www.openmp.org/wp-content/uploads/OpenMPRef-5.0-0519-web.pdf)

4. Benchmark your parallel implementations with 1, 4, 8 and 12 threads on LCC3 using N=10.000^2, 20.000^2 and 30.000^2. Use OpenMP's time measurement function to measure only the computational loop.

See (3.)

5. Enter the time for N=32.768^2 into the comparison spreadsheet.

Sorry, too much for my int32_t