Here we will have an overview about my solution for the parallelization of the Sieve of Eratosthenes, to find the prime numbers to n, and the factorization in prime numbers of a large number.

To run the program input [number] [threadNumber] where:

- *number* means “prime numbers from 0 to number” and “factorise from number*number to number*number-100”
  This number must be greater than 16
- *threadNumber* is the number of threads that the program will use
  If the number is less than 1 it will be assigned the number of cores of the pc.

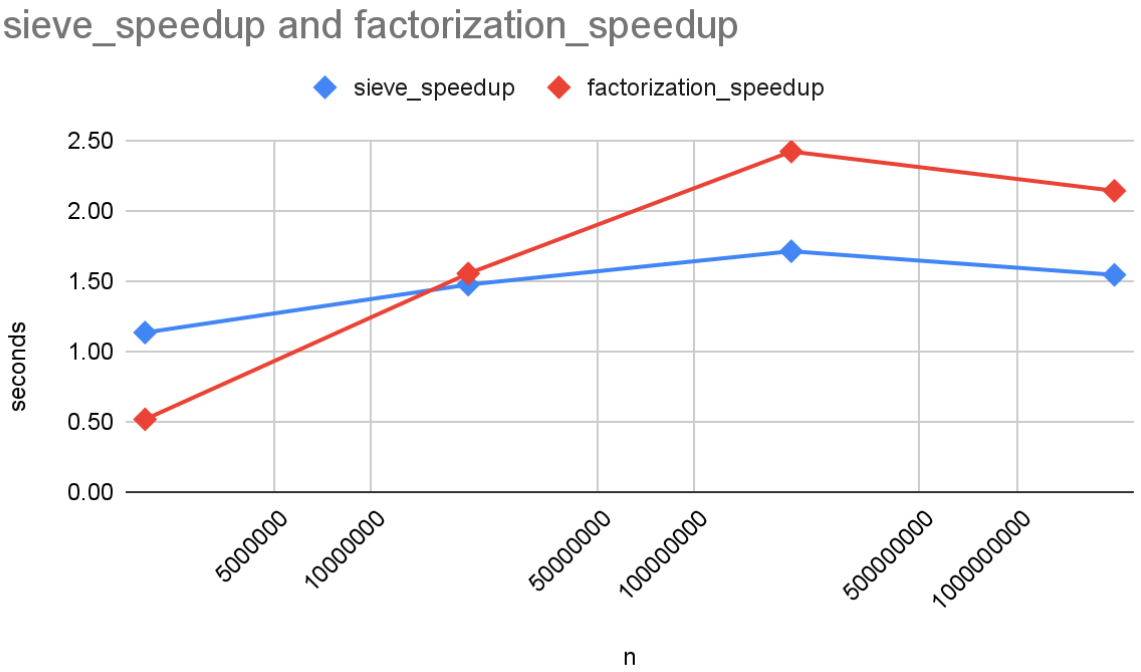
About the parallelisation of the sieve, the first step is to find all the primes to sqrt(number) sequentially, then I divide the oddNumbers array of bytes by the number of threads to get some subarrays of oddNumber. Foreach of these arrays I cross out all the non-prime numbers, using all the primes computed before.
You don’t need to manage the concurrency because each part of the oddNumber array is scanned only by a single thread.

In the parallel factorization I begin finding all the primes to sqrt(n*n). I assign some of this number to each thread (primes.length/nThreads) and then I do the division for those numbers in each thread.
At the end I check whether the numbers found are right, if not I divide the initial number by the product and I add that number as prime.
This is because I only look for the primes to sqrt(n*n) so I have to add it if it is greater than that.

These are the results of my program

n	sequential_sieve	parallel_sieve	sieve_speedup
2000000	0.009	0.008	1.13
20000000	0.081	0.055	1.47
200000000	0.940	0.549	1.71
2000000000	12.594	8.157	1.54
n	sequential_factorization	parallel_factorization	factorization_speedup
2000000	0.049	0.095	0.52
20000000	0.365	0.235	1.55
200000000	3.530	1.459	2.42
2000000000	35.032	16.362	2.14

Times are expressed in seconds



I used my machine with an Intel 4 cores/8 threads.
The sieve is always better with the parallelisation but the parallel factorization makes sense only after n > 6*10^6.
We have to notice that the factorisation speeding up includes, in some way, alse the sieve one: we use the parallel sieve in order to factorise a number in parallel.

In conclusion it is possible to parallelise the sieve and the factorization but it is convenient only with large Ns.

I attach to this document the .txt files containing the result of the program.