High Performance Computing
Exercise 2

Alessandro Minutolo

Mandelbrot Set

- The Mandelbrot set M is generated on the complex plane $\mathbb{C}$ by iterating the complex function $f_c(z) = \mathrm{z}^2 + \mathrm{c}.$
- M is defined as the set of complex points c for which the sequence

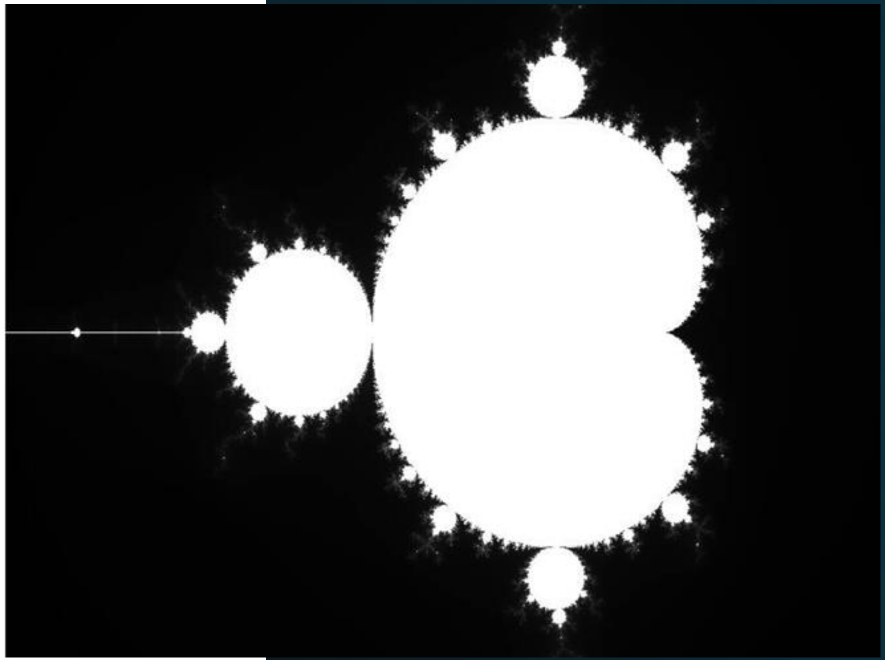
$$z_0 = 0, z_1 = f_c(0), f_c(z_1), \dots, f_c^n(z_{n-1})$$

is bounded.

- Condition to determine whether a point c is in the set M is:

$$|z_n = f_c^n(0)| < 2 \; or \; n > I_{max}$$

Where I_{max} is a parameter that sets the maximum number of iterations after which we can consider c a point in the set M



Problem Statement

- Implement a hybrid code MPI-OpenMP that computes the Mandelbrot set and generate an image of it.

- Evaluate the performance of the code performing:
    - Strong scaling (both for MPI and OpenMP)
    - Weak scaling (both for MPI and OpenMP)

- Asses limitations, problems and bottlenecks of the code, hence possible solutions.

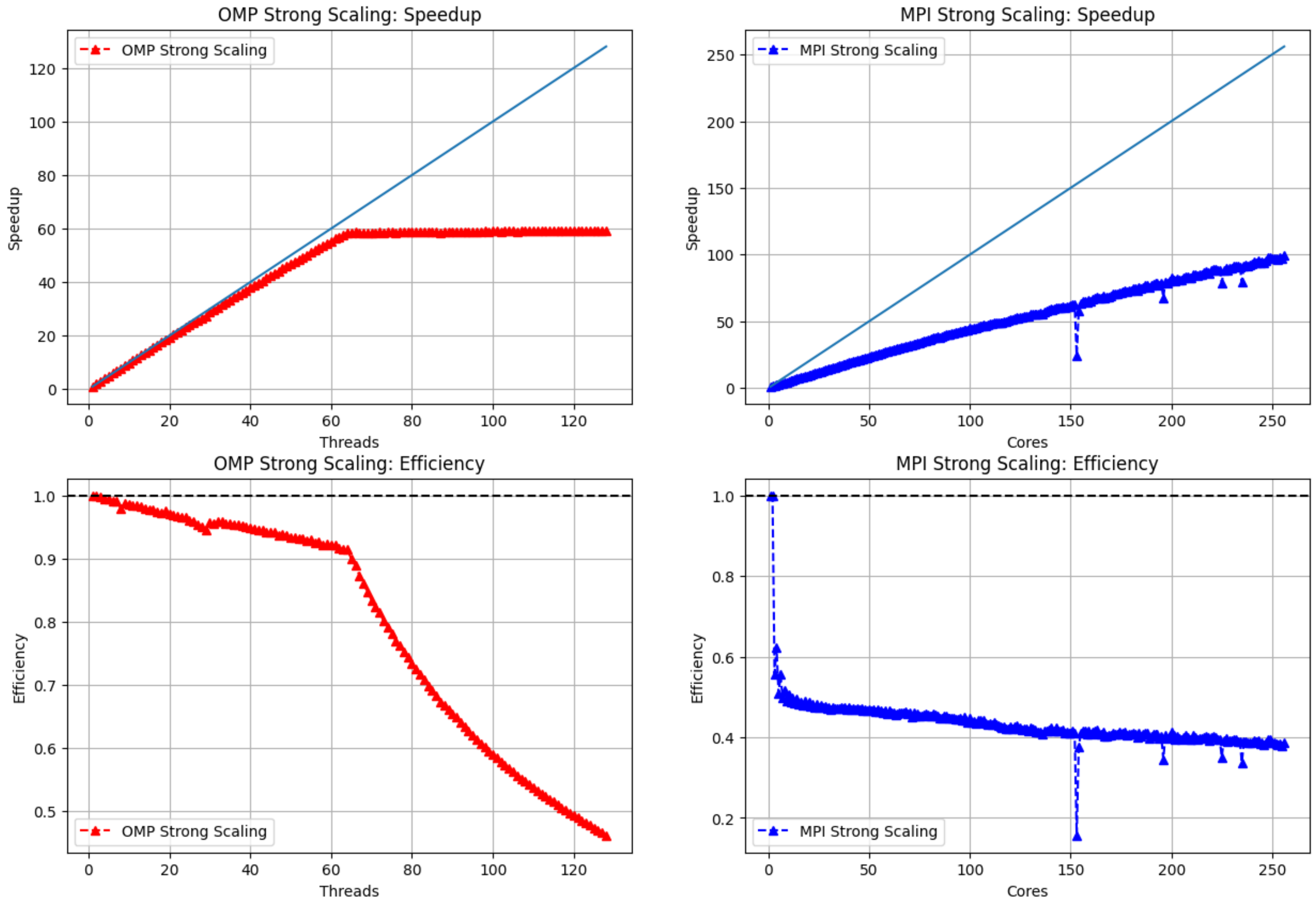
Implementation

- MPI (distributed memory parallelism) part:
  - Each process has its static workload assigned before computing the set.
  - The computation for each core is stored in a local buffer.
  - At the end all the buffers are gathered back into the root process through a **MPI_Gatherv** and generates the image.
- OpenMP (shared memory parallelism) part:
  - Number of threads specified before the parallel region.
  - Rows are dynamically assigned to threads to balance the workload through ***#pragma omp parallel for schedule (dynamic)***
  - Each thread works on its part of the local buffer

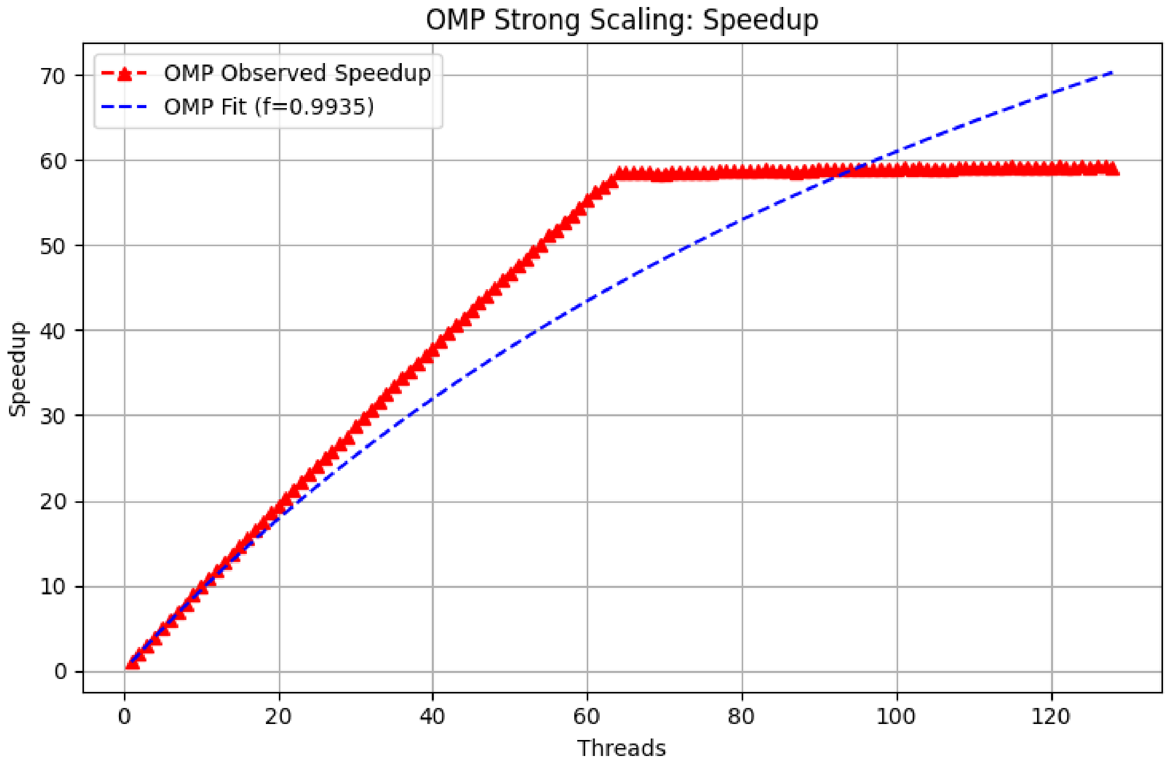
Scaling

- Strong scaling:
  - Increasing number of MPI processes/OpenMP threads
  - Fixed problem size of 10000x10000 pixels
- Weak scaling:
  - Increasing problem size of $\sqrt{C \cdot N}$, where C is a constant of 1MB and N is the number of MPI processes/OpenMP threads.

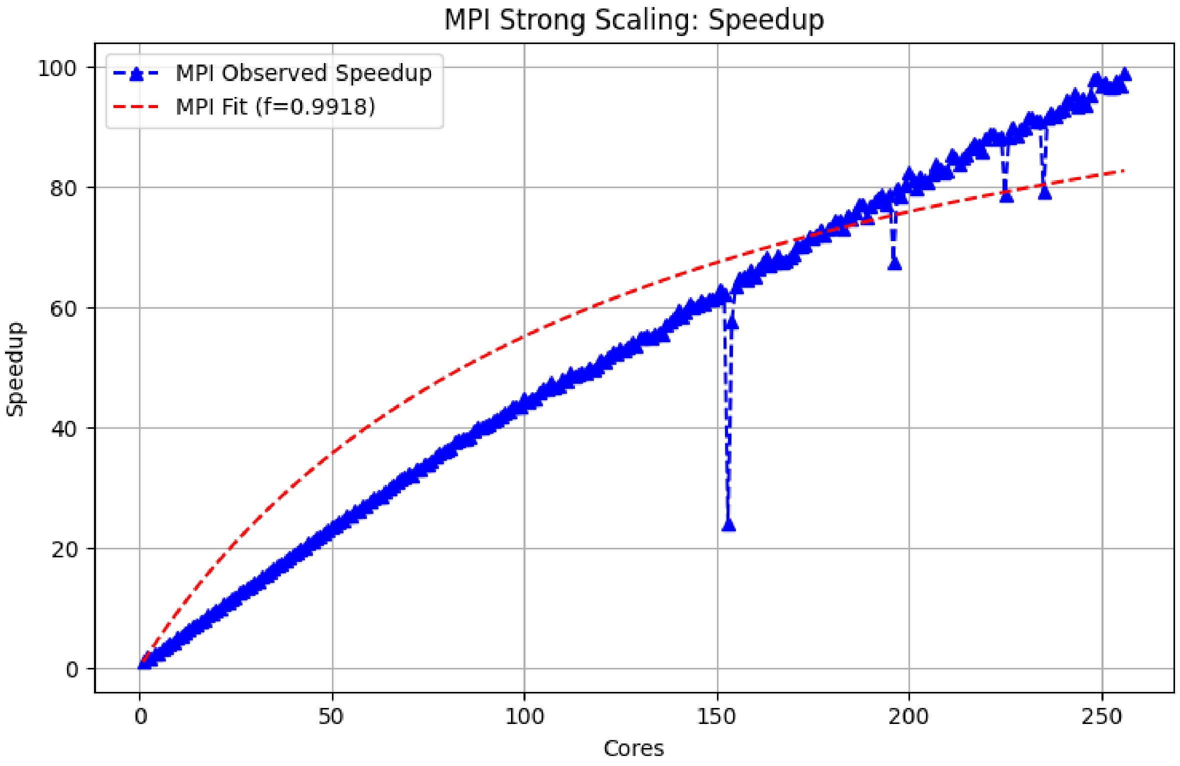
Strong scaling



Amdahl’s Law

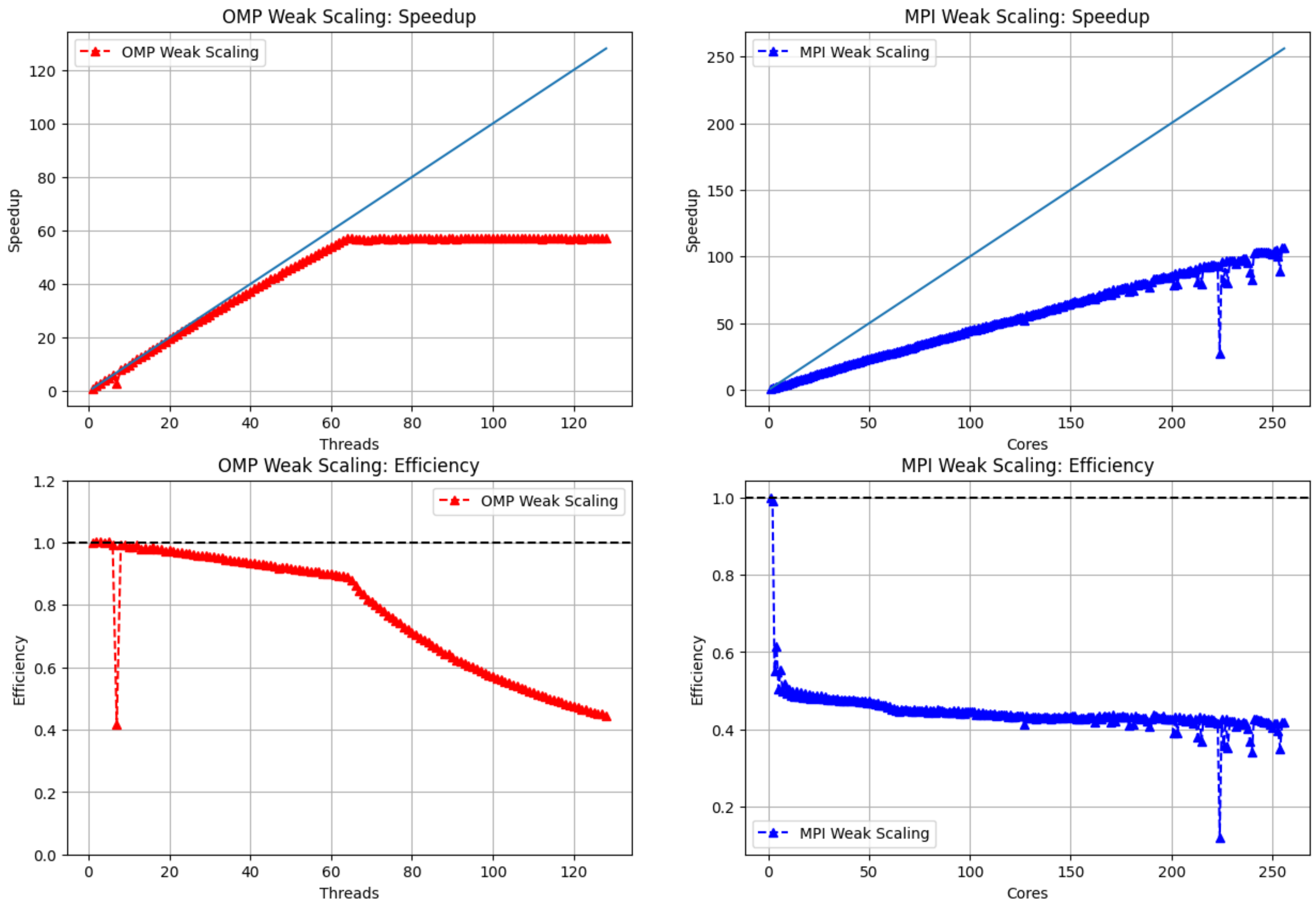


OMP Strong Scaling: Speedup
OMP Observed Speedup
OMP Fit (f=0.9935)
Speedup
Threads
0
10
20
30
40
50
60
70
0
20
40
60
80
100
120

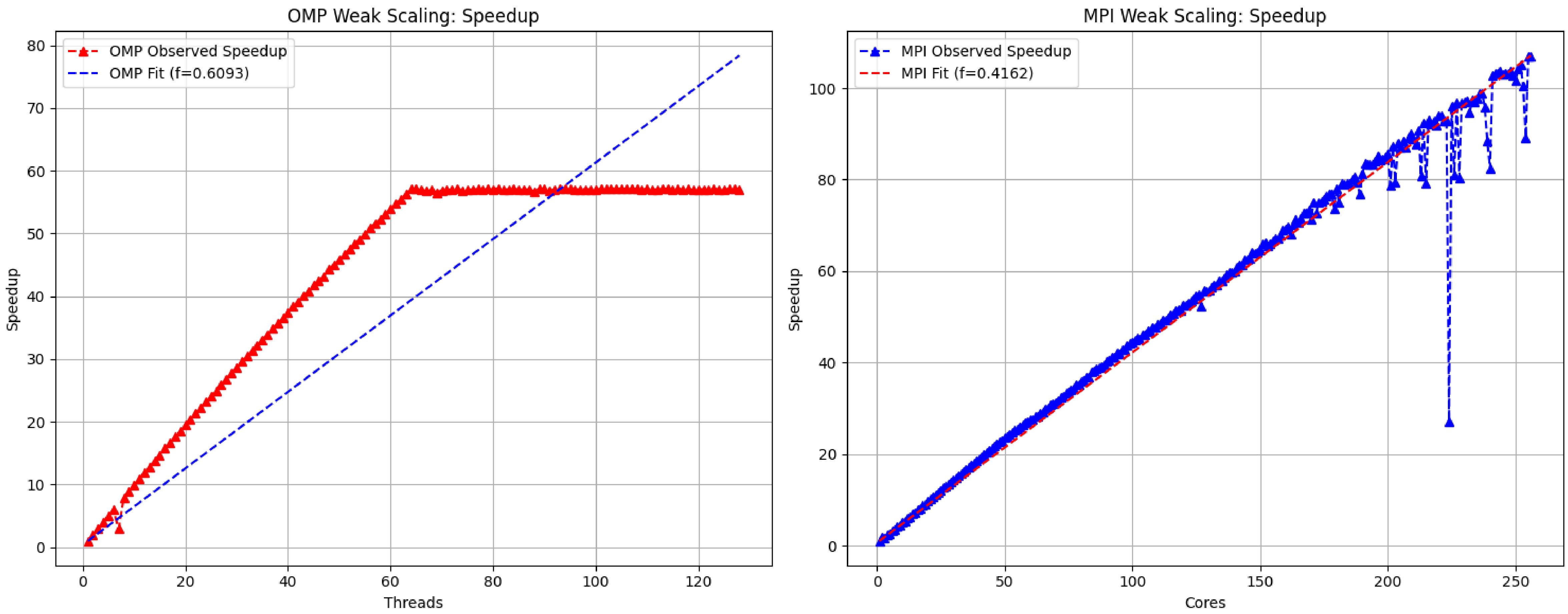


MPI Strong Scaling: Speedup
MPI Observed Speedup
MPI Fit (f=0.9918)
Speedup
Cores
0
20
40
60
80
100
0
50
100
150
200
250

Weak Scaling



Gustafson’s Law



Final Conclusions

- MPI:
  - Worst performances, it is clear that the workload is not very balanced due to the not homogeneus nature of the problem.
  - A solution could be implementing a dynamical division of the workload, balancing the points with more iterations among cores.
- OpenMP:
  - Scales well, we are quite satisfied.
  - Apparently the use of ***scadule(dynamic)*** is already enough to balance the workload among threads.