High Performance Computing 2023 - Exercise 2c

Marco Zampar - SM3800032

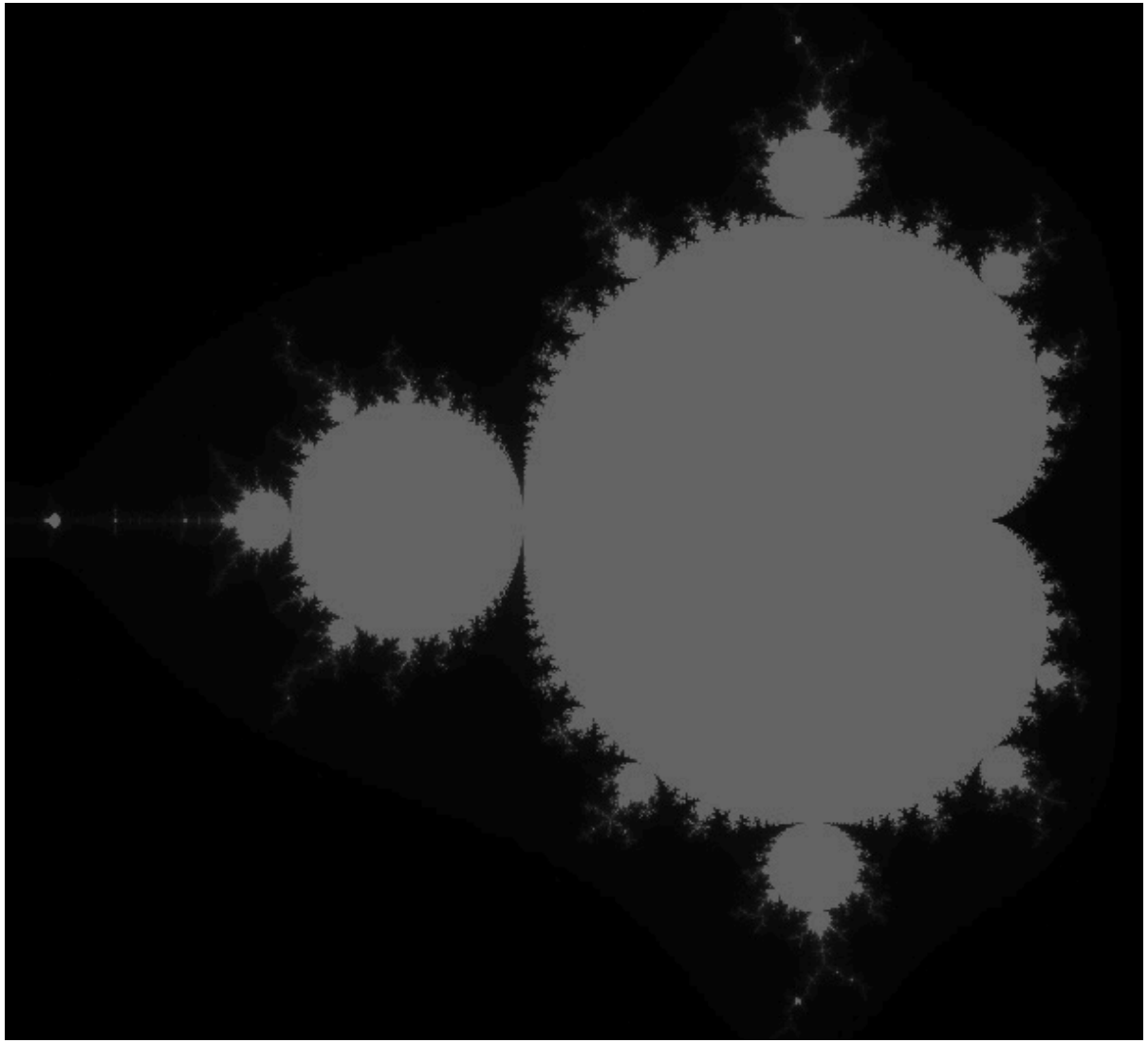
October 01, 2024

Problem Statement

- The **Mandelbrot Set** is defined as the set of points c such that the sequence $z_{n+1} = z_n^2 + c$, with $z_0 = 0$, is bounded. The simplest condition to guarantee that is $|z_n| < 2$.
- The goal is to parallelize the computation of the Mandelbrot Set with an hybrid `C` code using:
  - **MPI** (Message Passing Interface) for distributed memory parallelism;
  - **OpenMP** (Open MultiProcessing) for shared memory parallelism.

Mandelbrot Set Visualization

- The Mandelbrot set is represented as a 2D array.
- Each pixel is checked with the condition $|z_n| < 2$ or $n \leq I_{max}$.



Implementation Details

- `char` 2D array to store the matrix, `I_max=255`.
- **Symmetry** along the x-axis can be leveraged for optimization.
- **OpenMP**: `schedule(dynamic)` to handle workload imbalance across rows.
- **MPI**: Rows distributed based on process rank to balance the workload.
- **I/O Strategy**:
    - Tried MPI parallel I/O.
    - Used **MPI_Gatherv** for gathering data on rank 0, then serial writing to the PGM file.

Computational Resources

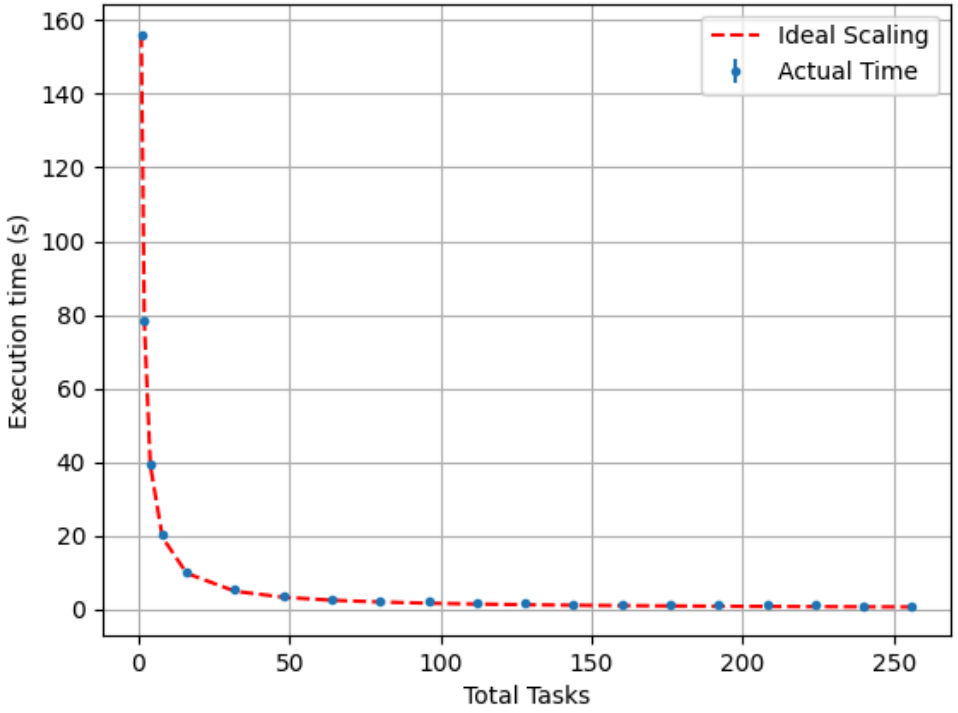
- Computations and testings were performed on **2 EPYC nodes** of the **ORFEO cluster**.
  - **OpenMP**: 64 cores (1 socket with 4 NUMA regions).
  - **MPI**: 2 nodes (256 total cores).
- Resources were allocated:
  - **MPI process binding**: Each process bound to a core.
  - **OpenMP thread binding**: Threads were placed using `OMP_PLACES=cores` and `OMP_PROC_BIND=close`.

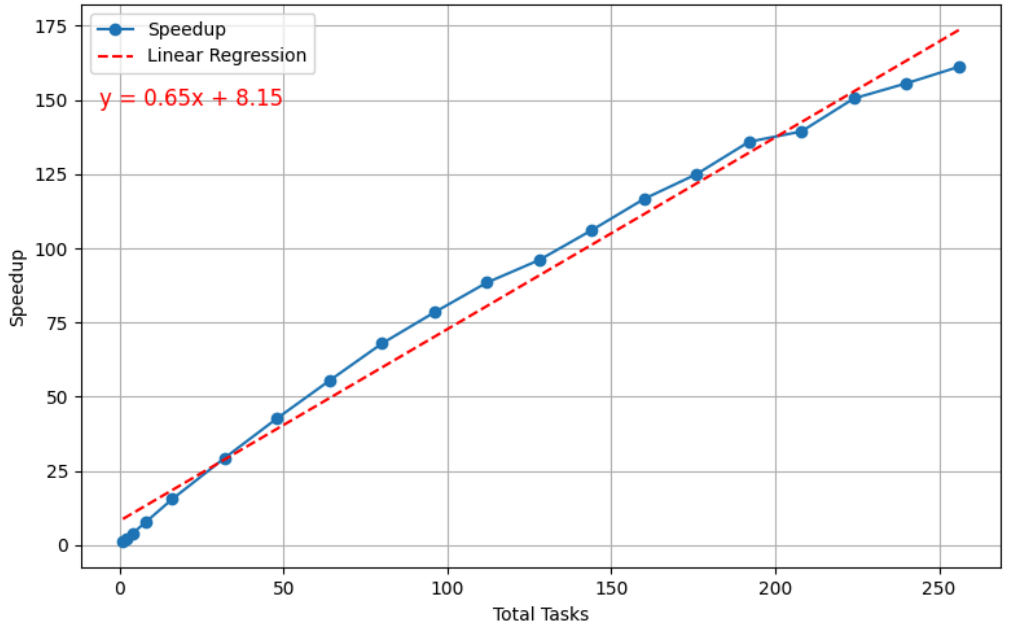
Scaling Performance

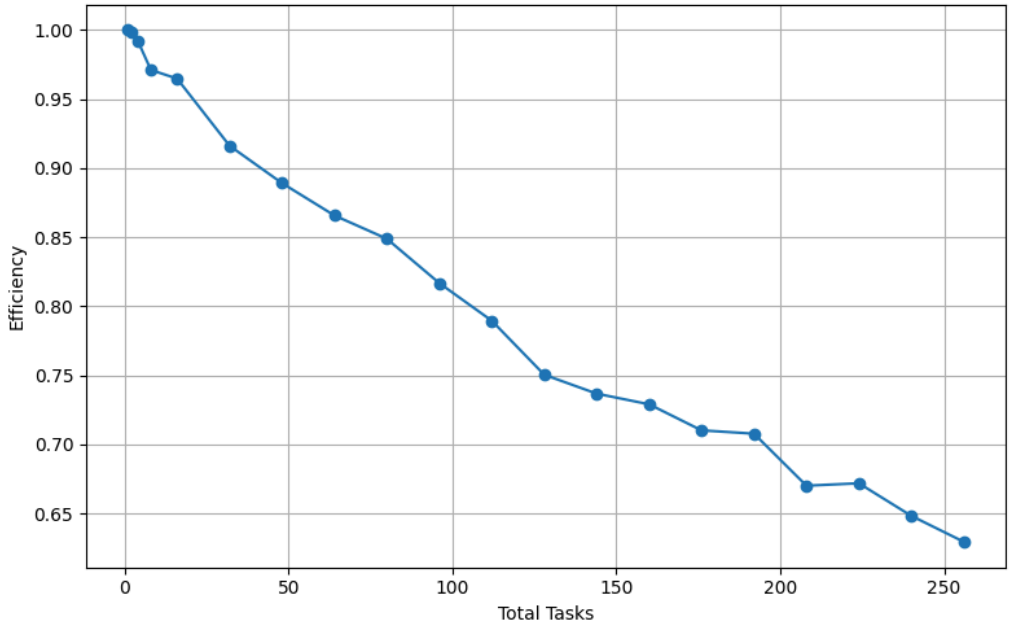
- We evaluated **strong scaling** (increasing workers at constant work) and **weak scaling** (workers proportional to the amount of work) for both **MPI** and **OpenMP**.
- Strong scaling performed with `nx=ny=25000`.
- Weak scaling performed with $C = W/P = (2000/\sqrt{P}) * (2000/\sqrt{P})$, with W, P amount of work and number of processes.

MPI Strong Scaling

- **Serial execution time**: 156.17 s
- **256-process execution time**: 0.97 s
- **Speedup**: 161.18 (99% reduction in time)

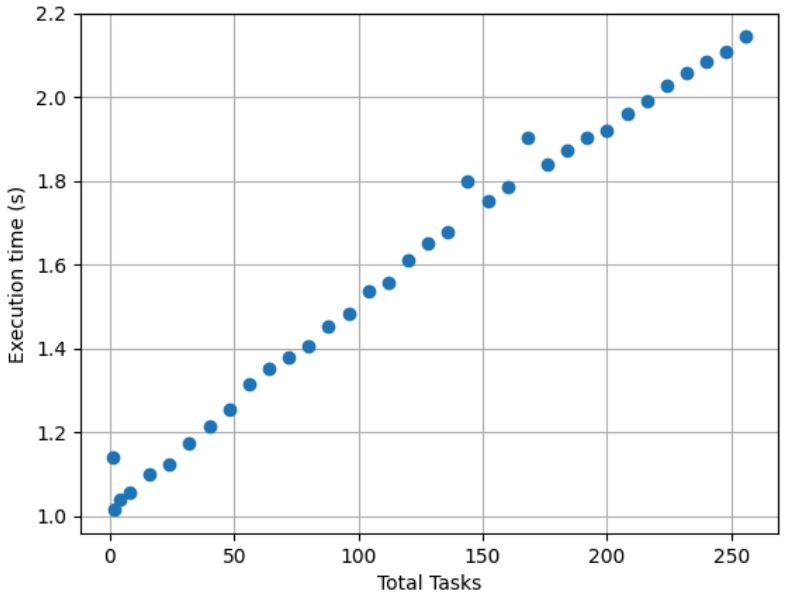


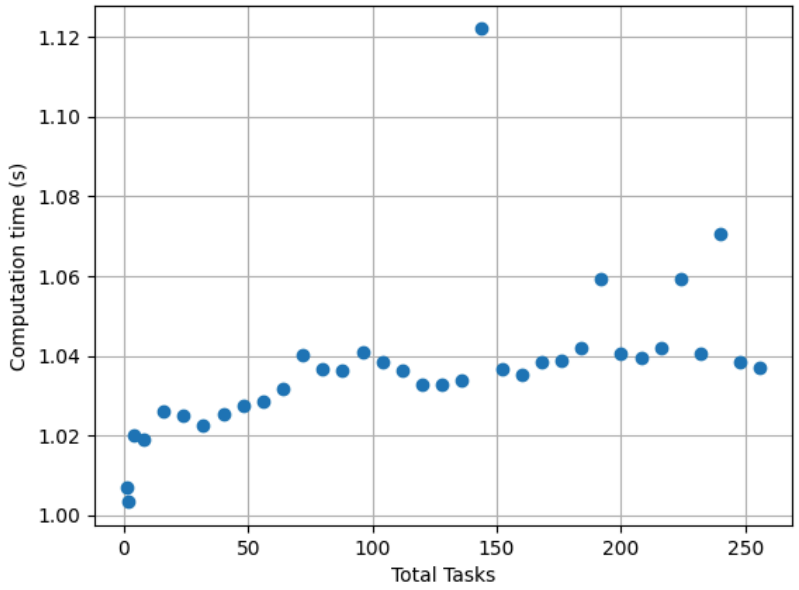


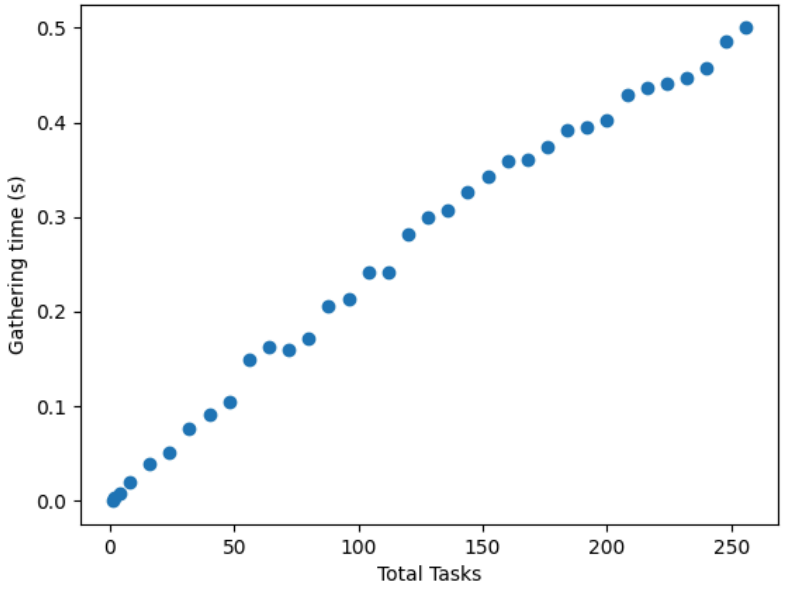


MPI Weak Scaling

- **MPI** weak scaling shows linear increase in time due to the overhead of `MPI_Gatherv`, but the computation time stays almost constant at 1 second.

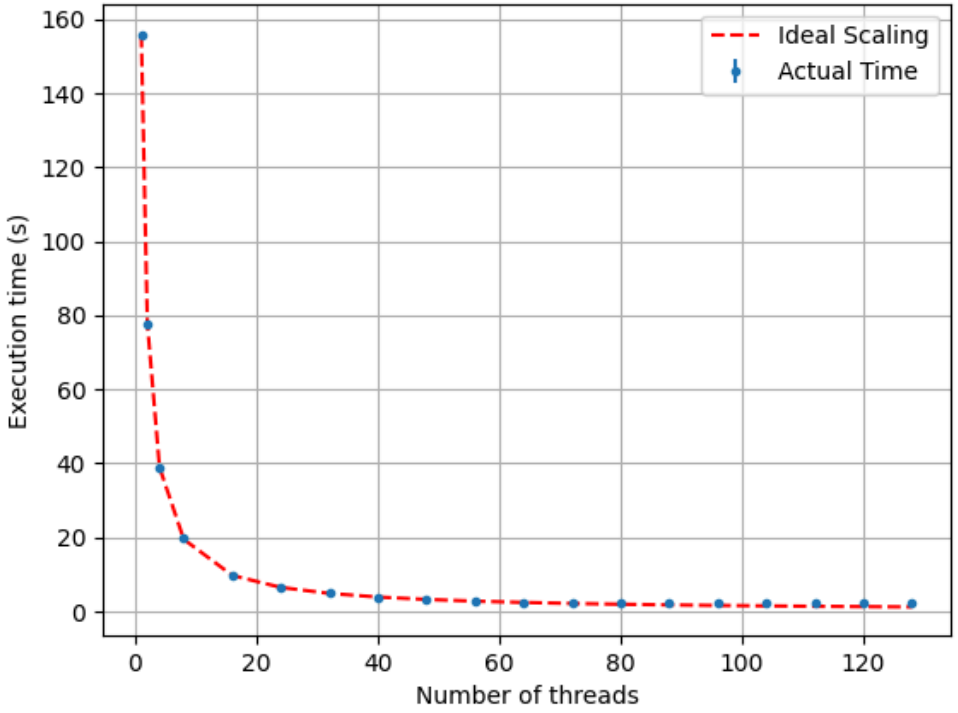


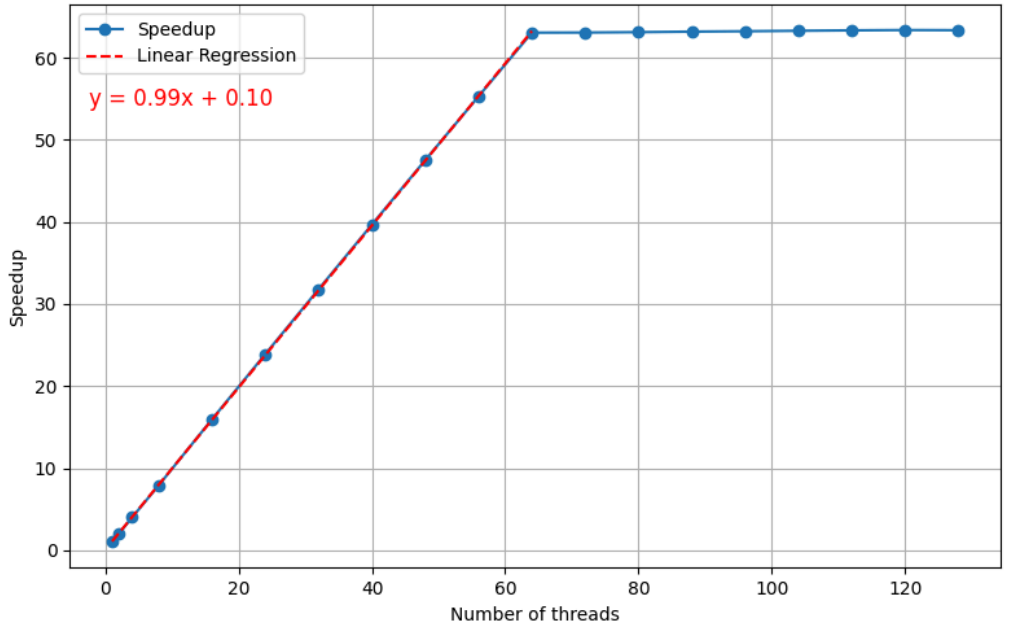


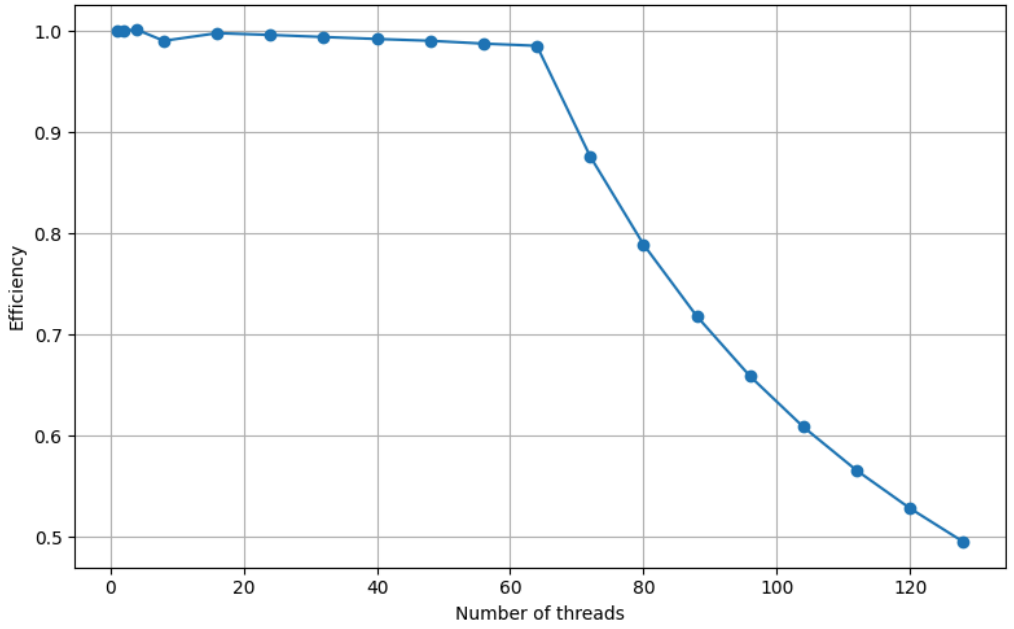


OpenMP Strong Scaling

- **Serial execution time**: 156.74 s
- **64-thread execution time**: 2.44 s
- **Speedup**: 64.2 (98.5% reduction in time)

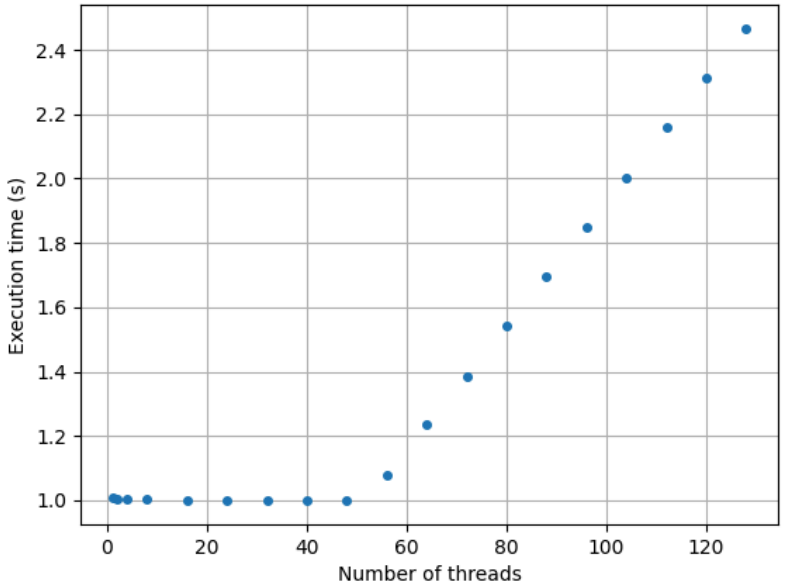


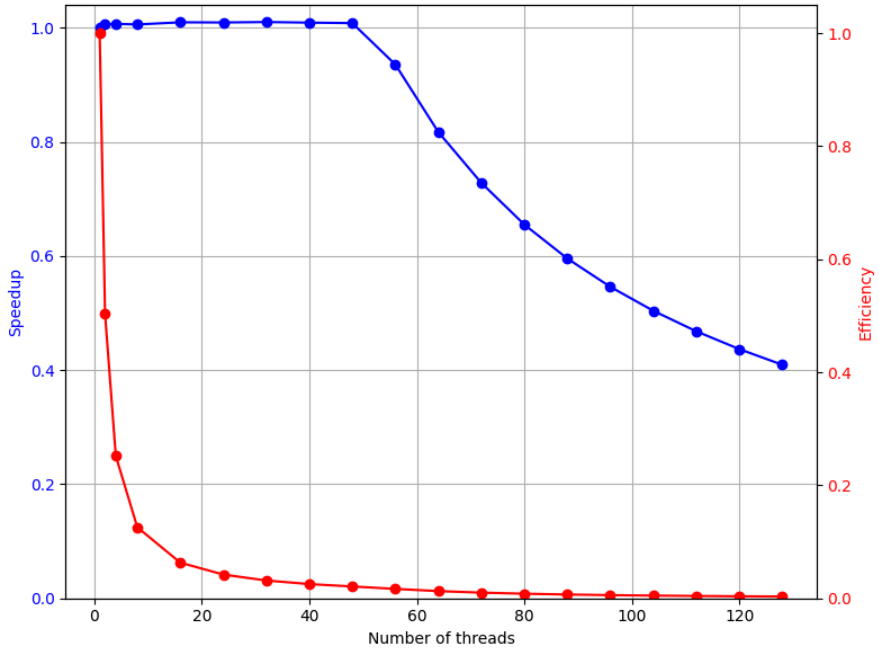




OpenMP Weak Scaling

- **OpenMP** weak scaling is efficient up to 64 threads, after that execution time increases probably because of resource contention between threads.





Final Considerations

- **OpenMP** scales efficiently up to 64 threads but struggles beyond that due to resource contention.
- **MPI** can handle larger-scale problems but incurs in communication overhead.

Hybrid Scaling

- We tested **Hybrid MPI+OpenMP scaling** using both processes and threads.
- The best configuration achieved an execution time of 1.61 s (99% reduction).
- The issues noticed in the MPI weak scaling were found also in the Hybrid scaling, but fixing the number of processes and increasing the number of threads we got almost constant execution times.

Thanks for your attention!