High Performance Computing

Exercise 2c

Annalisa Paladino

Mandelbrot set

- Generated on the complex plane by iterating the complex function $f_c(z) = z^2 + c$
- Composed of all the complex numbers c such that the sequence $z_0 = 0, z_1 = f_c(z_0), z_2 = f_c(z_1)$, ... is bounded.
- Condition: $|z_n = f_c^n(0)| < 2$ or $n > I_{max}$
- Represented as a fractal shape, which exhibits self similarity.
- Each point c can be calculated independently of each other → problem is well suited for being computed efficiently in parallel.

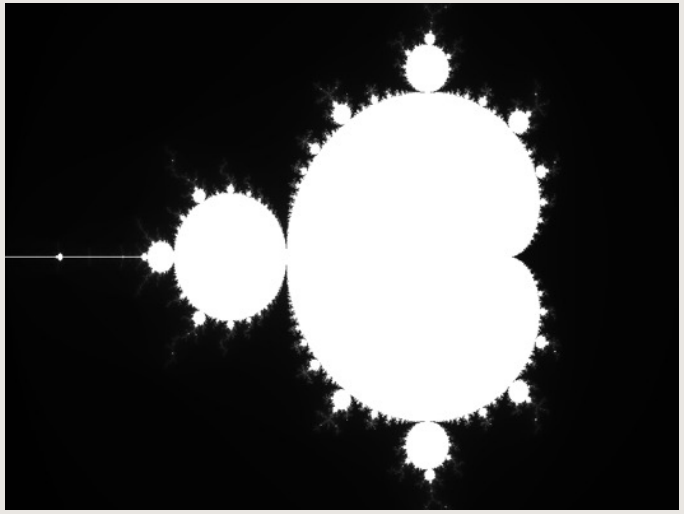
Task: Develop a C hybrid code to compute the Mandelbrot set using both **MPI** and **OMP**; determine the *strong* and *weak* **scalings** of the code.

Implementation - Mandelbrot

Determine the number of iterations it takes for a point (x_0, y_0) to escape a boundary when iterated through the Mandelbrot set equation.

The function returns this iteration count as an unsigned char, which can be used to visualize the Mandelbrot set by mapping iteration counts to colors.

```
unsigned char mandelbrot(double x0, double y0, int max_iter) {
    double x = 0, y = 0, xtemp;
    int iter = 0;
    while (x*x + y*y <= 4 && iter < max_iter) {
        xtemp = x*x - y*y + x0;
        y = 2*x*y + y0;
        x = xtemp;
        iter++;
    }
    return iter;
}
```



Setup:

- 2 **THIN** nodes of the **ORFEO** cluster
- 2 sockets per node
- 12 cores per socket

Implementation – MPI

- **MPI_Init_thread**: set up the MPI environment with thread support specified to `MPI_THREAD_FUNNELED` → while the program may be multithreaded, only the main thread will make MPI calls
- **MPI COMM_WORLD**: communicator that retrieves the total count of processes and the respective identifier (ID) → each process determines the specific segments of the Mandelbrot image it is responsible for generating, utilizing its unique ID and the total number of image rows
- **MPI_Gatherv**: collect partial results from all processes and assemble them into a single image array on the root process; this root process then writes the complete image to a PGM file
- **MPI_Finalize**: shut down the MPI environment

Implementation - OMP

- Further subdivide the computational work assigned to each MPI task among **OMP** threads
- Each **thread** have to compute one row at a time
- Parallel region introduced with the **#pragma omp parallel for** directive
- Use of **dynamic scheduling** help balance the load among threads

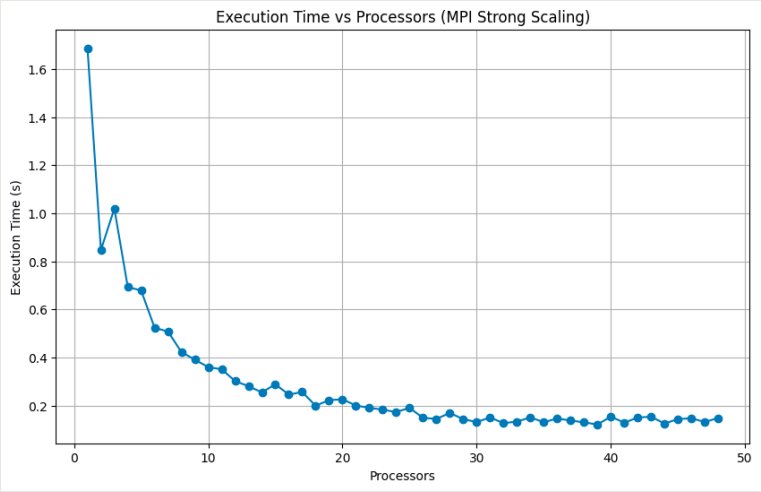
```
#pragma omp parallel for schedule(dynamic)
for (int j = start_row; j < end_row; j++) {
    for (int i = 0; i < nx; i++) {
        double x = xl + i * (xr - xl) / nx;
        double y = yl + j * (yr - yl) / ny;
        local_image[(j - start_row) * nx + i] = mandelbrot(x, y, max_iter);
    }
}
```

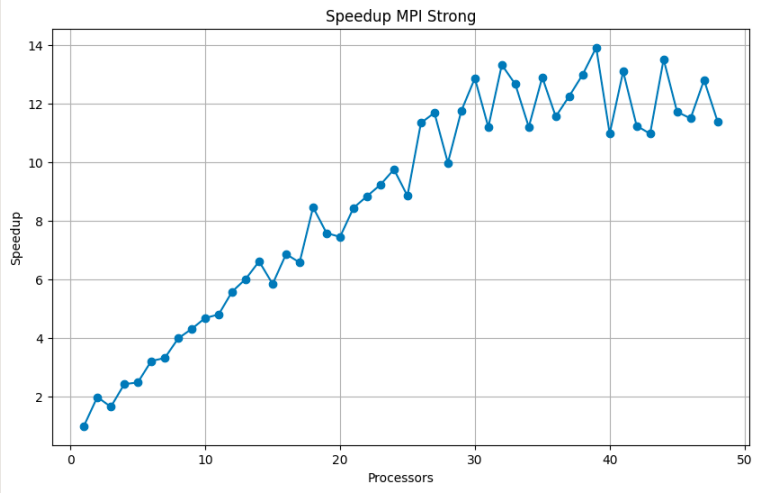
Strong Scaling

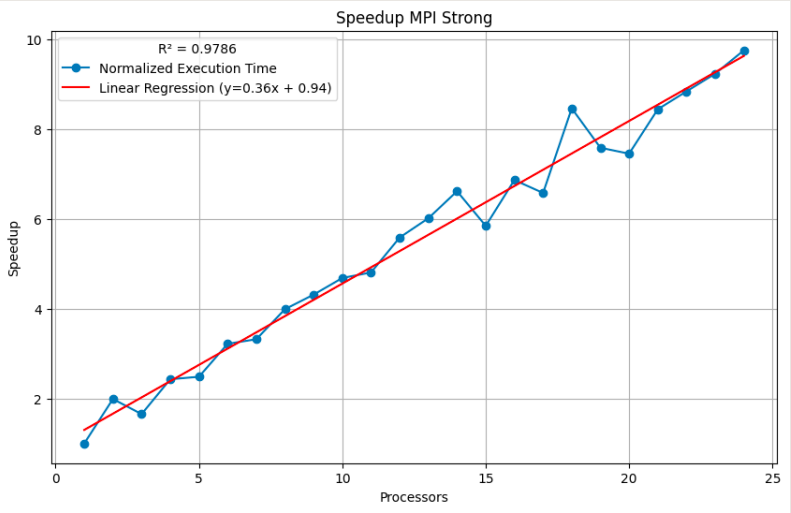
Strong scaling measures the efficiency of a system when solving a *fixed-size problem* with an *increasing number of processors.*

Goal: doubling the number of processors halves the computation time

Strong Scaling – MPI

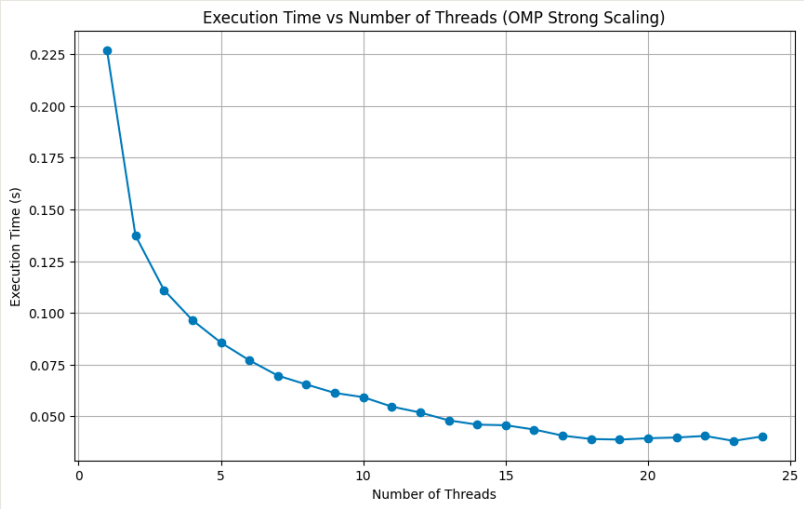


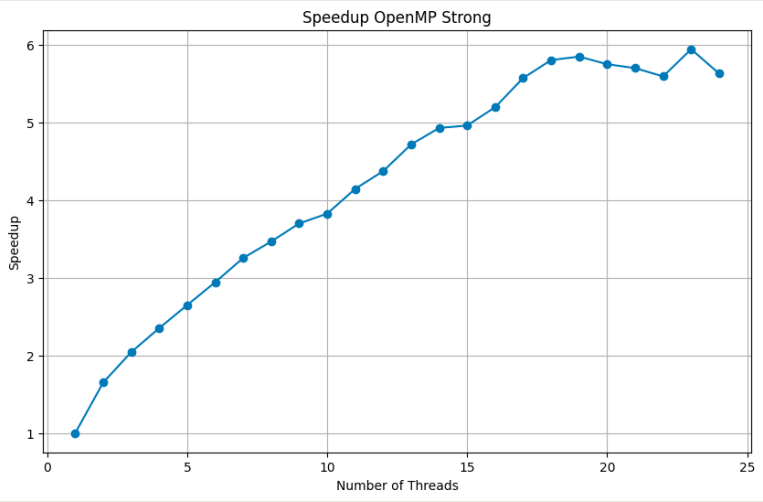


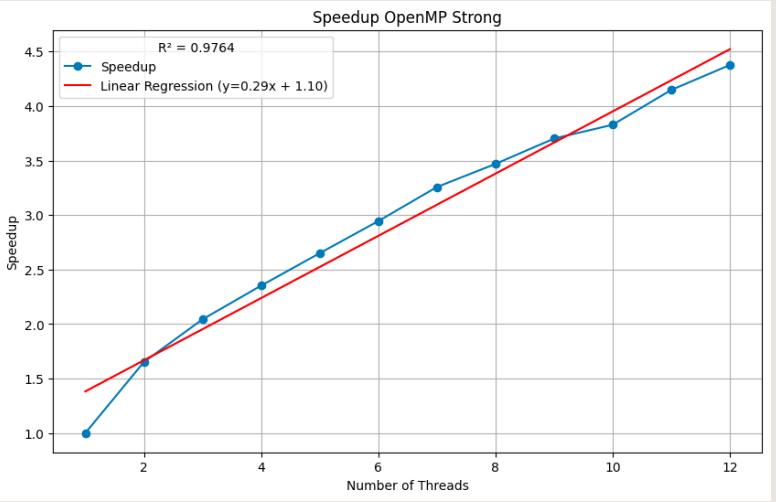


the dimensions of the image, specifically the number of rows and columns, were set at 1600 and 2400, respectively, while employing a single OMP thread

Strong Scaling - OMP







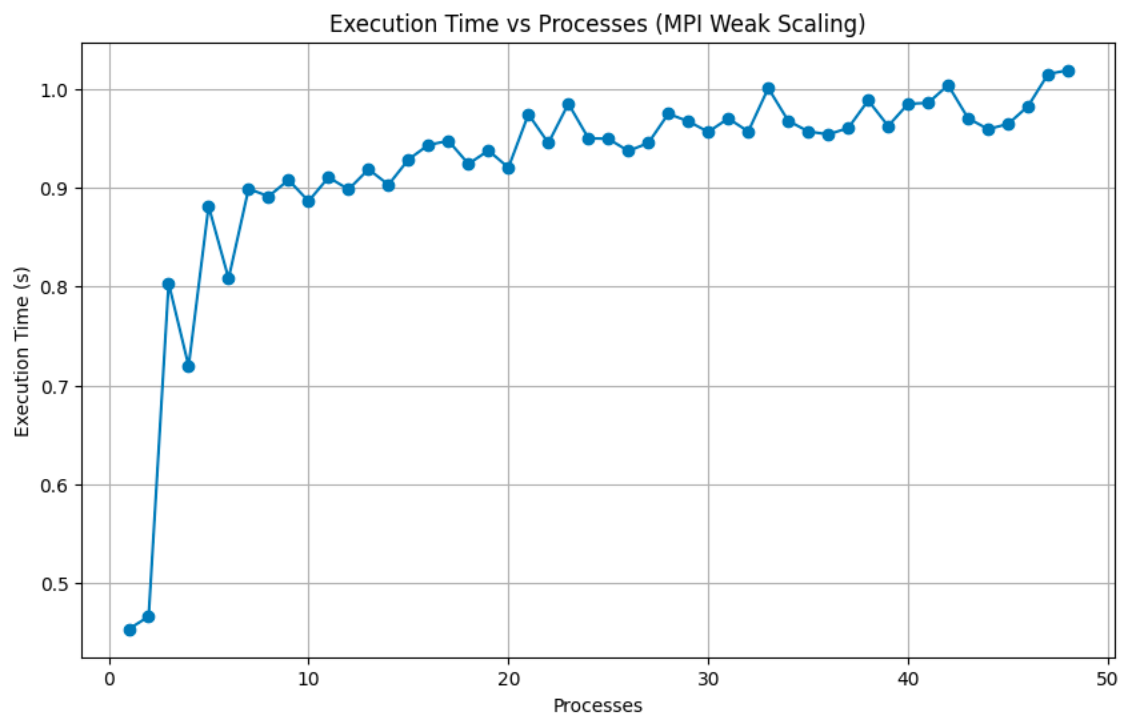
the image dimensions, specifically the number of rows and columns, were maintained at 600 and 800, respectively, with a single MPI process held constant

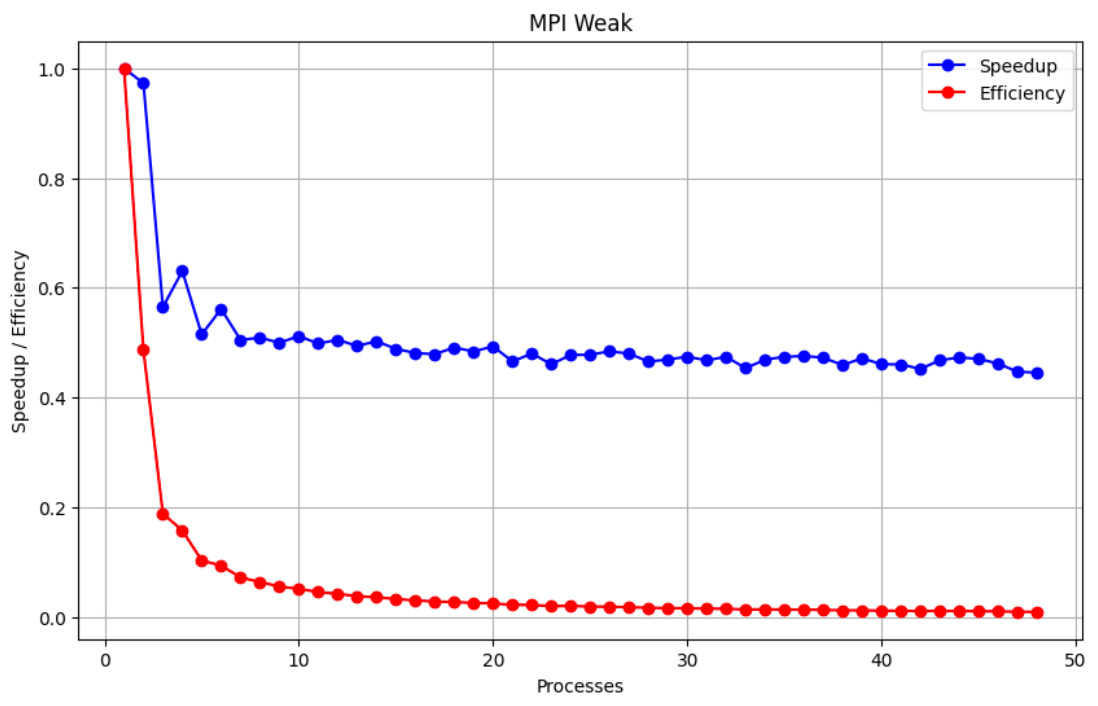
Weak Scaling

Weak scaling measures the efficiency of a system when *increasing the size of the problem* proportionally to the number of *processors*.

Gustafson's law: weak scaling evaluates how the execution time of a parallel application remains nearly constant as the number of processing elements increases in direct proportion to the problem size

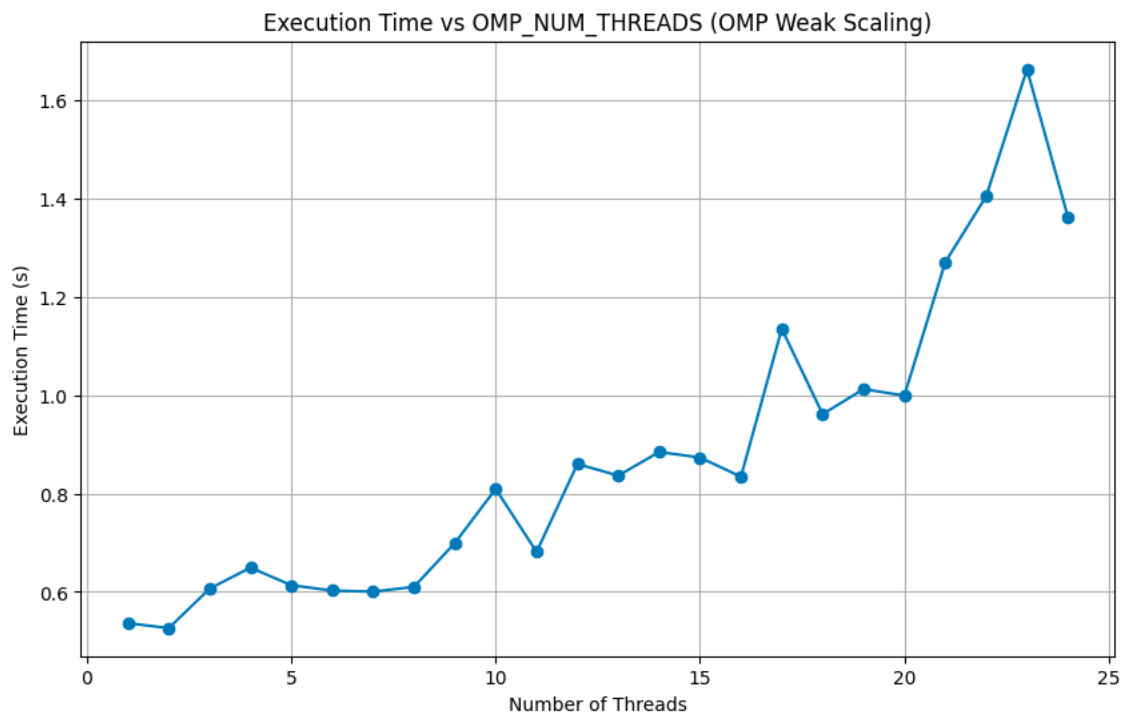
Weak Scaling - MPI

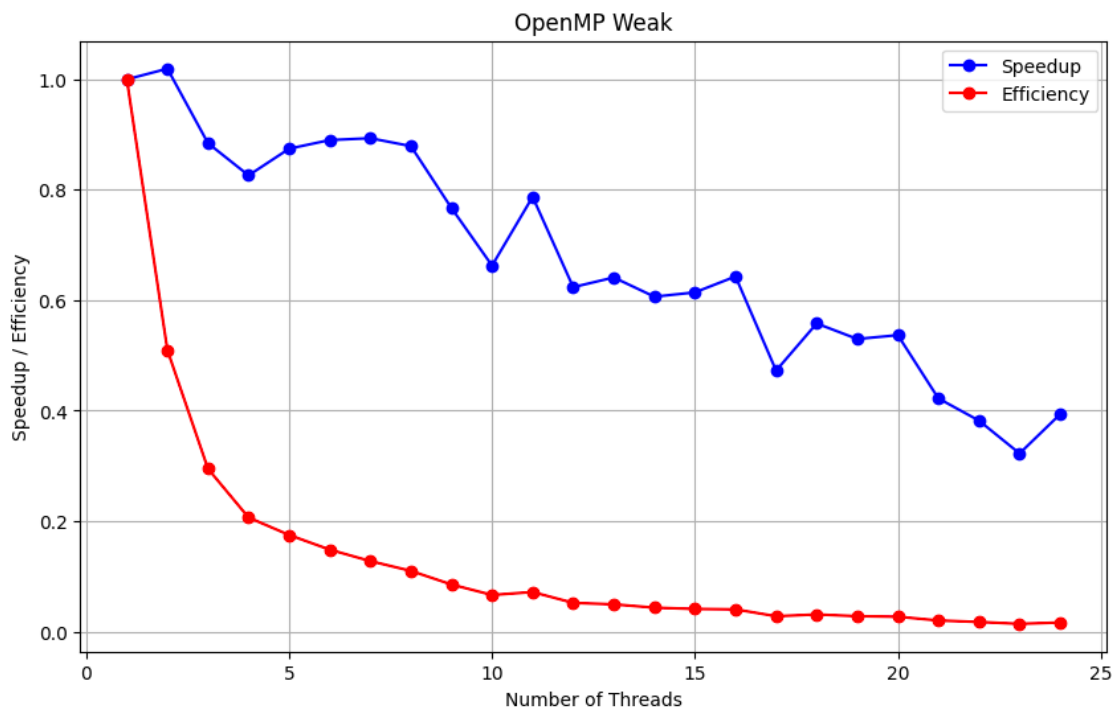




the dimensions of the image were systematically adjusted, with the number of rows and columns starting at 1000 by 1000 and incrementally expanding up to 48000 by 1000

Weak Scaling - OMP





the image size was also varied, with the number of rows and columns starting from 1000 by 1000 and extending up to 48000 by 1000