OpenMP Implementation of the Mandelbrot Set

Marco Tallone

May 13, 2024

DATA SCIENCE & ARTIFICIAL INTELLIGENCE

SCIENTIFIC & DATA-INTENSIVE COMPUTING

Mathematical Background

Mandelbrot Set

The Mandelbrot set is the set of complex numbers c for which the function $f_c(z) = z^2 + c$ does not diverge when iterated from $z = 0$.

$\rightarrow$ elements farther than 2 from the origin $\Rightarrow$ considered divergent

Mathematical Background

Mandelbrot Set

The Mandelbrot set is the set of complex numbers c for which the function $f_c(z) = z^2 + c$ does not diverge when iterated from $z = 0$.

$\rightarrow$ elements farther than 2 from the origin $\Rightarrow$ considered divergent

$\rightarrow \forall c \in S \subseteq \mathbb{C}$ iterate $f_c(z)$ from $z = 0$ until

$$|z_n| > 2 \quad \text{or} \quad n \geq I_{\max}$$

where $S = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ is a subset of the complex plane and $I_{\max}$ is the maximum number of iterations.

Mandelbrot Function Implementation

For a single point $c \in \mathbb{C}$, the previous can be verified with:

Mandelbrot Function

```
short int mandelbrot(const complex double c,
                     const int I_max) {
    complex double z = 0.0;
    unsigned short int i = 0;
    while (cabs(z) < 2.0 && i < I_max) {
        z = z * z + c;
        i++;
    }
    return i;
}
```

Main Loop

⚠️ **Note**: each point (=pixel) can be **computed independently**

Main Loop

⚠️ **Note**: each point (=pixel) can be **computed independently**

$\Downarrow$

- pixels stored in a matrix `M` of `short int`
- each thread computes **one row** at a time independently
- when a thread finishes a row, a new one is assigned

Main Loop

Main Loop

```
#pragma omp parallel for schedule(dynamic)
for (int i = 0; i < n_y; i++) {

  const complex double im_c = (y_l + i * delta_y) * I;

  for (int j = 0; j < n_x; j++) {

    complex double c = (x_l + j * delta_x) + im_c;

    M[i][j] = mandelbrot(c, I_max);

  }
}
```

Strong Scalability Analysis

Strong Scaling

Measuring how the execution time t varies with the number of threads P for a fixed total workload.

Strong Scalability Analysis

Strong Scaling

Measuring how the execution time t varies with the number of threads P for a fixed total workload.

- **EPYC** node: 2 sockets, 64 cores per socket

Strong Scalability Analysis

Strong Scaling

Measuring how the execution time t varies with the number of threads P for a fixed total workload.

- **EPYC** node: 2 sockets, 64 cores per socket
- **Total workload constant**:
  - fixed-size image of $10^3 \times 10^3$ pixels
  - $I_{\max} = 65535$ maximum possible for `short int`

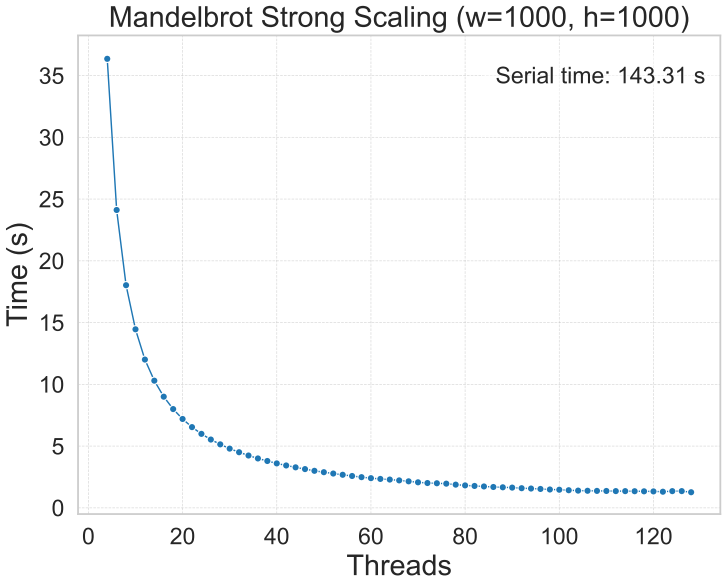
Strong Scalability Analysis

Strong Scaling

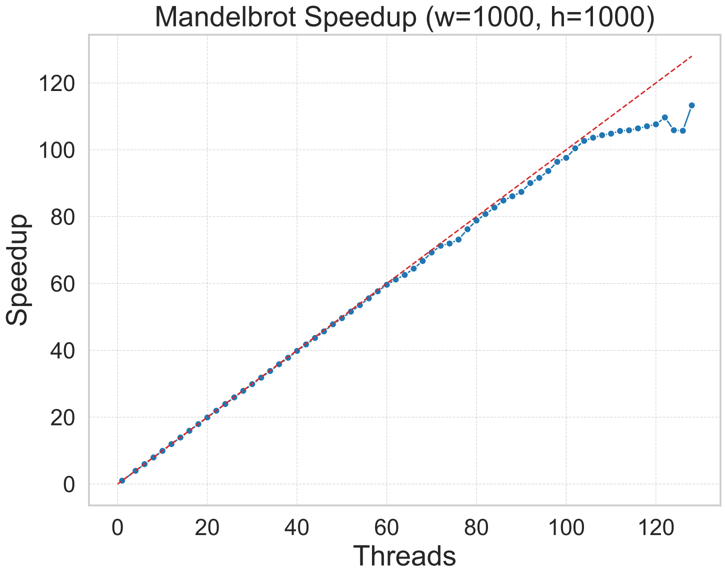
Measuring how the execution time t varies with the number of threads P for a fixed total workload.

- **EPYC** node: 2 sockets, 64 cores per socket
- **Total workload constant**:
  - fixed-size image of $10^3 \times 10^3$ pixels
  - $I_{\max} = 65535$ maximum possible for `short int`
- **Total number of threads**: $1 \to 128$

Strong Scalability Plots



Strong Scalability Plots



References

AMD.
Epyc 7h12 specifications, 2023.

OpenMP Architecture Review Board.
Openmp application program interface, 2023.

AREA Science Park.
Orfeo documentation, 2023.