Lab2: the Mandelbrot set

October 12, 2022

1 Introduction

The Mandelbrot set is composed of the points c of the complex plane C for which the following sequence:

$$\begin{cases} z_0 = 0 \\ z_{n+1} = z_n^2 + c \end{cases} \tag{1}$$

does not diverge. Setting $z = x + iy$ and $c = a + ib$, equation (1) is rewritten as:

$$\begin{cases} x_{n+1} = x_n^2 - y_n^2 + a \\ y_{n+1} = 2x_n y_n + b \end{cases}$$

with the following initial conditions: $x_0 = y_0 = 0$.

We can show that if there exists an integer n such that $|z_n| \geq 2$ (that is to say: $|z_n|^2 = (x_n^2 + y_n^2) \geq 4$), then the sequence (1) diverges. For more information, see [1, 2].

2 Structure of images in memory

An image is a two-dimensional array. Each element of this array is named a *pixel*, short for *picture element*. Its value is, depending on the image type, a gray level value, a color or a radiance value. This table is organized in memory row by row: we have the first row, then the second, and so on. In particular, we will handle images encoded on one byte (size of a pixel), and the value of each pixel (therefore a integer between 0 and 255) represents an index in a color table.

In C programming, this leads to such codes:

```
{
  unsigned char *Image;
  int w, h;         // width and height of the image
  int i, j;         // indices to browse the image pixels

  Image = (unsigned char *) malloc (sizeof(unsigned char)*w*h);

  for(i=0; i<h; i++)
    for(j=0; j<w; j++)
      Image[j+i*w] = 0;  // access to pixel i,j
}
```

By convention, the pixel with $(0, 0)$ coordinates is the upper left point of an image displayed on the screen, and it is therefore also the first element of the `Image` array in memory.

3 Image format

There is a huge variety of image formats, that is, ways to store an image in a file. We use the Sun Rasterfile format which is very simple to implement and which can be viewed with most image viewing programs[1]. A Rasterfile file consists of a header that describes the characteristics of the image (size, encoding, ...), followed by a series of 1-byte words describing the color table (if necessary). Then comes the image itself, stored as a raw data table.

[1] for example `display` (from the `ImageMagick` software suite) on Linux

4 Sequential algorithm

High-level algorithm:

1. For the center of each pixel in the image:
   (a) compute the number of iterations for which the sequence diverges
   (maximum number of iterations limited to a "depth" set by the user);
   (b) set the value of the corresponding pixel as:
   If depth reached: pixel_color ← 255
   Otherwise: pixel_color ← IterationNumber % 255
2. Save the image.

5 Work description

1. Describe (with an algorithm) a first parallel version (using MPI) of the algorithm presented in section 4, which distributes evenly the pixels on the processors.
2. Implement this parallel algorithm in C+MPI, and present the parallel speedups obtained with (at least) 2, 4 and 8 processors (using the default parameters). Analyze the performance results (one could possibly rely on the analysis of the execution times of each processor).
3. Describe a second parallel algorithm aiming at improving these performance results.

   What is the advantage of this new algorithm within a platform such as Grid'5000 with multiple clusters and multiple concurrent users?
4. Implement this second parallel algorithm in C+MPI, present and analyze its performance results.

References

[1] The Fractal Geometry of the Mandelbrot Set, *Robert L. Devaney* `http://math.bu.edu/DYSYS/FRACGEOM/`

[2] The Spanky Fractal Database, *Noël Giffin*, `http://spanky.triumf.ca/www/welcome1.html`