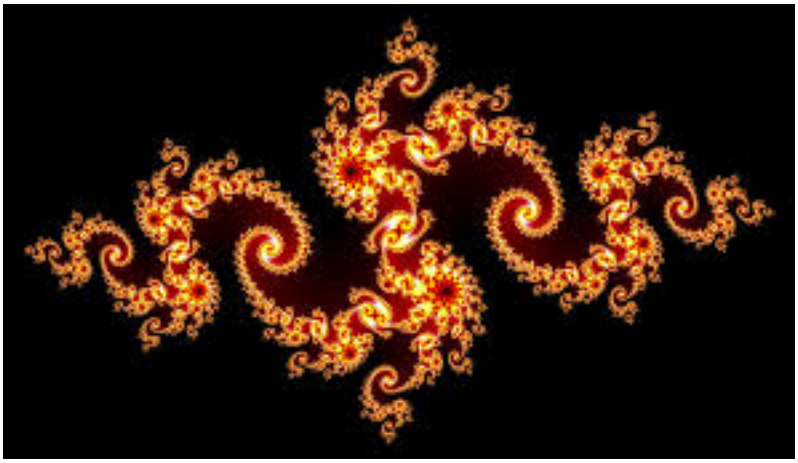
Calcul numérique de Fractales - Architecture

Rodolphe de Schaetzen et Pierre Verstraeten

30 Mars 2018

1 Introduction

Dans le cadre de notre projet pour le cours de Systèmes informatiques, nous allons créer un programme permettant de calculer des fractales et de comparer plusieurs fractales. Une fractale est un objet mathématique, ayant une structure particulière, et que l'on retrouve dans la nature. Pour calculer une fractale, il faut faire des calculs mathématiques sur chaque pixel d'une image pour obtenir un nombre qui est ensuite représenté avec une couleur sur l'image. C'est ainsi qu'on peut visualiser ces structures. Chaque fractale est définie par un nombre complexe, qui est utilisé pour effectuer les calculs. Notre but est donc de créer un programme permettant d'effectuer ces calculs et de comparer la valeur moyenne des pixels de chaque fractale, et de décider laquelle a une moyenne la plus élevée.



2 Structures de données

Il nous est demandé dans l'énoncé de représenter les fractales sous forme de structure que nous nommerons *struct Fractal*.

```
struct Fractal{
    char name[64];
    unsigned int height;
    unsigned int width;
    double a;
    double b;
    double values[height*width];
}
```

Dans cette structure, nous retrouvons:

- *name*: Le nom de la fractale
- *height*: La hauteur de l'image en pixels

- *width*: La largeur de l'image en pixels
- a et b: La partie imaginaire et réelle de l'argument
- *values*: Le tableau de la valeur de chaque point de la fractale

3 Type de threads

Nous pouvons diviser notre programme en 3 parties, chacune avec un nombre de threads déterminé. La première partie consiste à charger les descriptions de fractales. Nous utiliserons un seul thread pour cette partie-là qui se chargera de créer un buffer de la même taille que le nombre fractales à calculer. La deuxième partie est la partie dans laquelle le programme calcule les valeurs de chaque fractale. Nous assignerons un thread par calcul de fractale tant qu'il y en a moins que le nombre de threads maximal, sans quoi une fractale pourrait calculer plusieurs fractales de manière consécutive. Les threads créent un nouveau buffer dans lequel ils stockent les fractales une fois calculées. La troisième partie consiste à déterminer la fractale dont la moyenne des valeurs de chaque point est la plus élevée. Pour ce faire nous utiliserons un thread pour qu'il parcourt les fractales et renvoie la fractale avec la moyenne la plus élevée.

4 Méthodes de communication

Les threads doivent pouvoir communiquer entre eux. En effet le thread qui charge les descriptions doit pouvoir les passer aux threads de calculs. Les threads de calcul, eux, doivent pouvoir passer les fractales calculées au thread qui détermine la fractale avec la moyenne de la valeur des points la plus élevée. Ceci peut être réalisé en stockant des buffers dans le heap commun à tout les threads.

5 Informations communiquées entre les threads

Le premier thread, qui charge les descriptions de fractales, doit pouvoir passer chaque description aux threads de calcul. Les threads de calcul doivent pouvoir communiquer entre eux pour savoir qui calculera quel thread dans le cas où le nombre maximal de threads est inférieur au nombre de fractales à calculer. Le dernier thread utilisera les fractales calculées par les threads de calcul pour déterminer la fractale dont la valeur moyenne est la plus haute et renverra celle-ci.

6 Conclusion

Notre architecture est donc basée sur une structure de fractale simple et 3 types de thread différents, qui permettent selon nous d'effectuer la tâche de manière efficace.