Poor Man's Parallelism

Jesper Lundin

February 22, 2016

Abstract

This is the documentation of the project *Poor Man's Parallelism*.

1 Assignment

The assignment is to create two applications, a server and a client, which together render parts of the Mandelbrot set and saves it into multiple .pgm files. The server should accept requests on a TCP port and the client should spread the the workload over a set of servers.

2 Background

The Mandelbrot set is a set of complex numbers $c \in Z$ for which the function $f_c(z) = z^2 + c$ does not diverge when iterated, i.e. the sequence $f(0), f(f(0)), ...$ It has been shown that if the sequence contains a number larger than 2 then the sequence diverge. In this application we iterate until we have reach a specific number of iterations (we assume that the sequence converge) or until a number is larger than 2 (which means that the sequence diverge). For $c = a + bi$ we take the number of iterations modulo 256 which then corresponds to the color of the pixel at position (a, b). The colors has values between 0-255 in a grey scale.

3 Compile and run the applications

Use a command prompt to compile the application with

```
javac Myserver.java
javac MyCLient.java
```

To run a server enter the following line

```
java MyServer serverAdress:portNumber
```

where serverAdress is the adress and portNumber is the port number. For example,

```
java MyServer localhost:1234
```

To run the client use the following line.

```
java MyClient minReC maxReC minImC maxImC
maxNoIterations width height noXPictures
noYPictures noServers serverAdress1:portnumber1
serverAdress2:portnumber2 ...
```

The arguments are stored with the same name and are described in the next section. For example,

```
java MyClient -1.5 0.5 -0.5 0.5 256 900 800 5 2 3
localhost:1234 localhost:2345 192.168.33.3:4444
```

This will produce 10 images in a 5x2 grid which together creates an image of size 900x800 pixels. The workload is spread to the three servers: localhost:1234 localhost:2345 and 192.168.33.3:4444.

4 Client

4.1 Input

The client has the following 12 input arguments.

minReC, maxReC, minImC, maxImC	The minimal and maximal value of the real and imaginary axis in the complex plane. $$\text{minReC} \leq Re(z) \leq \text{maxReC}$$ $$\text{minImC} \leq Im(z) \leq \text{maxImC}$$
maxNoIterations	The maximal number of iterations before we assume that the sequence $f(z), f(f(z)), ...$ does converge for a value of c.
noXPictures noYPictures	The output files creates a $A \cdot B$ grid where A and B are noXPictures and noYPictures respectively.
width, height	The width and height of the output image.
noServers	The number of servers.
servers	A list of the servers addresses.
portNumbers	A list of the servers port numbers.

The applications assume that noXPicture divides width, noYPicture divides height and that noServers divides width.

4.2 Procedure:

1. We begin by reading, storing and printing the input.

2. We create a socket, buffered reader, print writer and a input stream reader to create a connection to the servers. For each server we send minReC, minImC, maxReC, maxImC, maxNoIterations, width, height, server and noServers. Where server is the number of the current server.
   The idea is to let server i calculate the pixel value of all pixels in the columns $i + Sn$ for $0 \leq n \leq \frac{w-i}{S}$ where S is the number of services and w is the width of the output image.

3. When all data pixel values has been calculated in the servers we read the values from the servers and store them in a 2D array which then will create our images.

4. Finally for each file we loop through the pixel values that are contained in that sub-image. Let $p_{(x,y)}$ be the sub picture at position (x, y) in the output grid then $p_{(x,y)}$ contains the pixels (i, j) for

$$y \cdot \text{subHeight} \leq j \leq (y+1) \cdot \text{subHeight}$$

$$x \cdot \text{subWidth} \leq i < (x+1) \cdot \text{subWidth}$$

   where

$$\text{subWidth} = \frac{\text{width}}{\text{noXPictures}}$$

$$\text{subHeight} = \frac{\text{height}}{\text{noYPictures}}$$

   At the top of each file we write the magic number for the pgm file, a header, the width, the height and the maximal value of the colors (255).

5 Server

The server creates a new thread each time a client connects which means that the client (and other clients) can make several connections to the same port number at the same time (not recommended).

5.1 Procedure

1. The server stores the input from the client.

2. Then it calculates the start value of c

$$c = (\text{minReC+server*dx}) + (\text{minImC}) \cdot i$$

   where

$$\text{dx} = \frac{\text{maxReC} - \text{minReC}}{\text{width} - 1}$$

   Remember that server is the number of the current server.

3. For each row we iterate over the columns and we increase the real part with

$$\text{dxServer} = \text{noServers} \cdot \text{dx}$$

in each step. Between the rows we increase the imaginary part with

$$\text{dy} = \frac{\text{maxImC} - \text{minImC}}{\text{height} - 1}$$

We have a function named mandelbrot which calculates how many iteration it takes before the sequence diverge (reaches a value > 2). After each calculation is done we send it directly back to the client.

6 Contact

For questions contact Jesper Lundin at jesper.l.lundin@gmail.com