High Performance Computing
Departamento de Ingeniería en Informática
LAB2

1 Objetivo

El objetivo de este laboratorio es repasar técnicas de programación multiproceso o multihebras usando semáforos, memoria compartida y hebras. Para ello, construiremos una aplicación paralela simple para la generación de conjuntos Mandelbrot.

Su labor cosiste en:

1. Documentarse en profundidad sobre qué y cómo se generan los conjuntos Mandelbrot
2. Construir un programa C secuencial (mono-tarea) para generar imágenes de los conjuntos
3. Construir un programa paralelo usando una de las tecnologías en memoria compartida, que realice la misma tarea anterior
4. Medir los tiempos de ejecución de ambos programas
5. Usar **octave** (o Matlab si tiene licencia) para visualizar los resultados

2 El conjunto Mandelbrot

El siguiente material fue redactado usando información de la wiki (`http://en.wikipedia.org/wiki/Mandelbrot_set`).

El **Conjunto Mandelbrot** es un conjunto de números complejos c para los cuales la órbita de 0 bajo la iteración

$$z_{n+1} = z_n^2 + c, \qquad z_0 = 0$$

permanece acotada (bounded). Es decir, c pertenece al conjunto Mandelbrot si el valor absoluto de z_n permance acotado, no importa cuán grande sea n. Formalmente, el conjunto Mandelbrot M es

$$M = \left\{c \in C, \exists s \in R, \forall n \in N, |P_c^n(0)| \leq s\right\} \tag{1}$$

Note que no es lo mismo que una secuencia sea acotada a que converga. Por ejemplo, si $c = -1 + i0$, los z_n son 0, -1, 0, -1, 0, etc. Claramente la secuencia no converge, pero está acotada, pues no crece. Luego, $c = -1$ pertenece al cojunto Mandelbrot. Si $c = 1$, la secuencia es 0, 1, 2, 5, 26,..., y continúa creciendo hacia infinito, es decir $c = 1$ no pertenece al conjunto.

La figura 1 (a) muestra una imagen que repesenta puntos del conjunto Mandelbrot. Los puntos que pertenecen al cojunto son los de color negro. En la figura 1 (b), los puntos se han coloreado de acuerdo al número de iteraciones necesarias para que la secuencia deje de ser acotada.

3 Generación de imágenes Mandelbrot

3.1 Detección de escape

El algoritmo más simple para generar imágenes de los conjuntos Mandelbrot es el algoritmo *tiempo de escape*, el cual asigna un nivel de gris (o color) al punto c de acuerdo al número de iteraciones necesarias para que la secuencia se escape o deje de estar acotada.

Dada una grilla (2D) en el plano complejo, cada posición $x + iy$ es testeada si pertenece al conjunto o no. Es decir, cada posición es un potencial c. Recuerde que siempre la iteración comienza con $z_0 = 0$, y cada iteración chequea si z_{n+1} se ha escapado o no.

En este lab. usaremos la siguiente propiedad para chequear condición de escape: *Ningún punto complejo con parte real o imaginaria mayor que 2 pertenece al conjunto.* Computacionalmente, chequeando si la distancia desde el origen es mayor que 2, detecta escape más tempranamente. Luego, en cada iteración, si $|z_{n+1}| > 2$, entonces la iteración termina

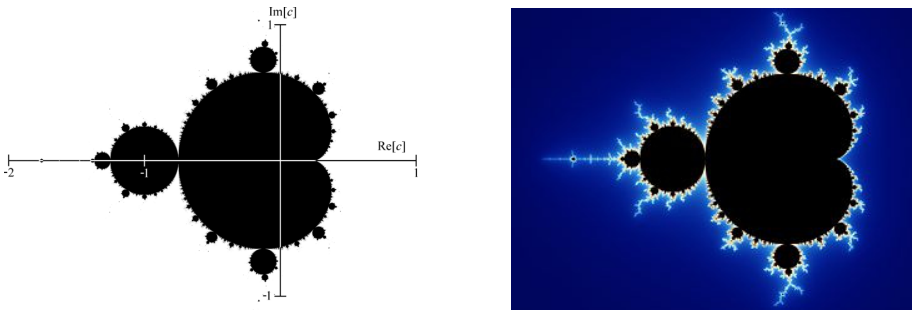

Figure 1: Ejemplo del conjunto mandelbrot. (a) Los puntos color negro pertenecen a M y los puntos blancos no. (b) Se colorean los puntos de acuerdo al "tiempo de escape".

y c no pertenece al conjunto. Para generar imágenes con colores, todos los puntos de la grilla serán coloreados con el número de iteraciones necesarias para escape.

Es posible que un punto nunca escape o necesite muchas iteraciones para escapar. Para controlar dicha situación, usaremos un límite en el número de iteraciones, llamado *depth*.

3.2 Algoritmo

El siguiente es un algoritmo para la generación de imágenes Mandelbrot. Este algoritmo genera una imagen M que luego debe ser visualizada en Octave.

---

for all $c = (x, y)$ **do**
  $z_0 = 0$
  $n = 1$
  $z_n = z_0 + c$
  while $|z_n| < 2$ **and** $n < depth$ **do**
    $z_{n+1} = z_n^2 + c$
    $n = n + 1$
  end while
  $M(x, y) = \log(n) + 1$
end for

Para implementar este algoritmo hay que tener en cuenta las siguientes consideraciones.

Las dimensiones y espaciado de la grilla dan origen a los puntos $c = (x, y)$. Por ejemplo, supongamos que se desea generar el conjunto Mandelbrot en el espacio complejo $[-1, -1] \times [1, 1]$, con un muestreo de 0.001. Luego, comenzando por el límite inferior izquierdo del plano complejo, el primer punto es $c = (-1, -1)$. Si nos movemos primero por la parte imaginaria, el segundo punto es $c = (-1, -1 + 0.001)$; el siguiente es $c = (-1, -1 + 2 \times 0.001)$ y así sucesivamante hasta llegar al límite superior izquierdo, que sería $c = (-1, 1)$. Luego, nos movemos un espacio hacia el lado positivo de los reales y muestreamos $c = (-1 + 0.001, -1)$, $c = (-1 + 0.001, -1 + 0.001)$, etc. Note que este muestreo genera una grilla de 2001×2001 puntos.

En el algoritmo se ha usado $c = (x, y)$ tanto como número complejo, como posición en la grilla. Esto no es posible en un programa computacional. Debemos usar índices enteros para acceder las entradas de la matriz M. Es decir, cada posición (i, j) en la matriz corresponde a un punto complejo (x, y).

El programa recibe como argumentos de entrada, los límites del plano complejo, y el intervalo de muestreo.

3.3 Despliegue de la imagen

Una vez que se ha generado la grilla, ésta se debe guardar en una archivo en formato binario de `float`. Para desplegar la imagen, usamos los siguientes comando Octave:

```
>> f = fopen('nombrearchivo.raw', 'r')
```

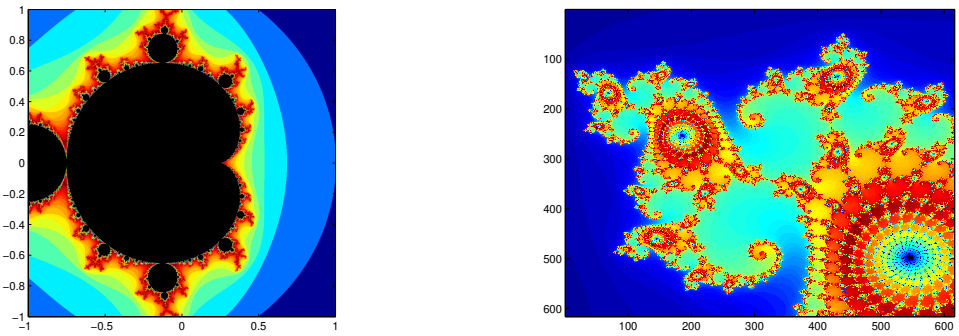

Figure 2: (a) Resultado ejecución programa con opciones indicadas en ejemplo. (b) Resultado con los siguientes parámetros: 500, -0.748766713922161, 0.123640844894862, -0.748766707771757, 0.123640851045266, 1e-11.

```
>> M = fread(f, 'float');
>> M = reshape(M, P, Q);
>> imagesc(M)
>> colormap( [jet();flipud( jet() );0 0 0] );)
```

Donde P y Q son las dimensiones de la grilla.

4 Los programas

4.1 El secuencial

Usted debe contruir un programa secuencial (mono-tarea) que implemente el algoritmo de escape para cojuntos Mandelbrot y un programa paralelo.

El programa secuencial debe ejecutarse de la siguiente forma:

```
$ time ./mandelbrot -i 500 -a -1 -b -1 -c 1 -d 1 -s 0.001 -f salida.raw
```

donde las opciones indican lo siguiente:

- `-i`: depth, o número máximo de iteraciones
- `-a`: limite inferior del componente real del plano complejo
- `-b`: límite inferior componente imaginario
- `-c`: límite superior componente real
- `-d`: límite superior componente imaginario
- `-s`: muestreo
- `-f`: archivo de salida

Como validación, la imagen de la figura 2 muestra el resultado de la ejecución del programa con las opciones del ejemplo.

Se usa el comando `time` para medir el tiempo de ejecución del programa. Siempre registre el tiempo real.

4.2 El paralelo

El programa paralelo debe ejecutarse igual que el secuencial, agregando el número de tareas (procesos o hebras) que deben trabajar en paralelo. Este número no incluye al proceso o hebra padre.

```
$ time ./mandelbrotp -i 500 -a -1 -b -1 -c 1 -d 1 -s 0.001 -f salida.raw -t 12
```

El trabajo debe ser divido entre las tareas por bloques de filas de la grilla. Los límites del plano complejo y el muestreo dan origen a un tamaño (P, Q) de la grilla. Claramente, estas dimensiones deben ser calculadas antes de comenzar el algoritmo. Como algoritmo sencillo de división, simplemente asignaremos un número igual de filas a cada tarea. Si la divisón no es entera, entonces se debe asignar un número parecido a cada tarea.

Cada tarea recorrerá todos los punto c del plano complejo correspondientes a su conjunto de filas. Note que la grilla es compartida y que no es necesario establecer secciones críticas, pues las tareas sólo accederán a sus posiciones. Lo que si debe sincronizar es el término de todas las tareas para que la tarea padre escriba el archivo de salida.

Para la ejecución del proceso paralelo usaremos un nodo del cluster. Ejecute y mida los tiempos para 1, 2, 3, 4,..., 24 tareas. Registre los tiempos y calcule el speedup de la siguiente forma:

$$S(n) = \frac{T_s}{T_p(n)}$$

donde T_s es el tiempo real secuencial y $T_p(n)$ es el tiempo real, paralelo para n tareas.

5 Entregables

Envíe a `fernando.rannou@usach.cl` al menos los siguientes archivos:

1. `Makefile`: archivo para make que compila los programas
2. `mandelbrot.c` archivo con el código secuencial
3. `mandelbrotp.c` archivo con el código paralelo
4. `Speedup.pdf` archivo con un gráfico que muestre el speedup del programa paralelo en función del número de tareas.

Fecha de entrega: 14 de Octubre antes de las 24:00 hrs.