Curso 2022-2023

PRÁCTICA 4. PROYECTO: SECUENCIAL Y OPENMP

04/11/2022
Prácticas de Programación Concurrente y Paralela
Pablo Revuelta Sanz – José Ranilla

Tipo de Práctica: Abierta.
Duración: 2 Sesiones de Prácticas.
Entrega Documentación: Ver secciones "*¿Qué, dónde y cuándo entregar la práctica?*"

ÍNDICE

1. Introducción

Primera práctica de una serie que resuelve, en secuencial y en paralelo, una versión adaptada y simplificada de un cálculo real. Este conjunto de prácticas conforma el proyecto planteado este curso para la asignatura.

Este enunciado tiene dos partes: a) presentación del problema y de técnicas habituales para su resolución y b) lo que el alumnado debe hacer.

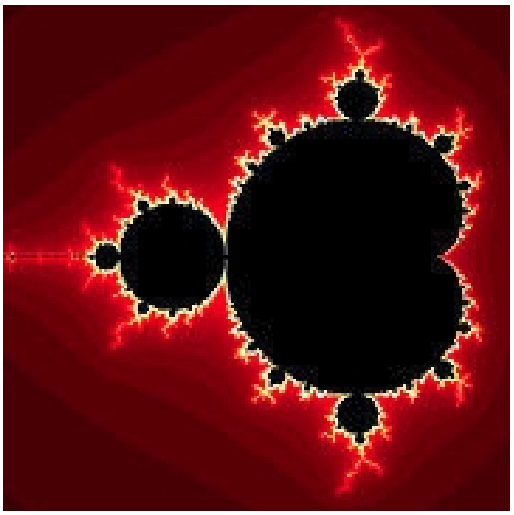
2. El Problema

En 1975, el matemático Benoît Mandelbrot acuñó el término “fractal” para unas estructuras matemáticas ya conocidas desde décadas anteriores.

La definición más básica que se puede dar de un fractal es que son estructuras autosímiles, es decir, replican la misma estructura a distintas escalas. En términos algorítimicos, se construyen mediante aplicación iterativa de alguna función.

Dicho cálculo, aplicado a estructuras bi-, tri- o n-dimensionales, genera complejidades temporales muy altas, lo que requiere la puesta en acción de mecanismos y estrategias de paralelización y computación de altas prestaciones.

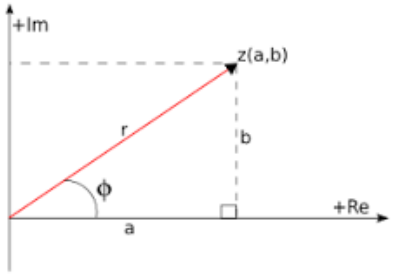
En esta práctica, nos centraremos sobre un subconjunto de la familia de “fractales de Julia” llamado conjunto de Mandelbrot (https://es.wikipedia.org/wiki/Conjunto_de_Mandelbrot):



Dicho conjunto se caracteriza por la función de iteración:

$$z_0 = 0$$
$$z_{k+1} = z_k^2 + c, c \in \mathbb{C} \quad (1)$$

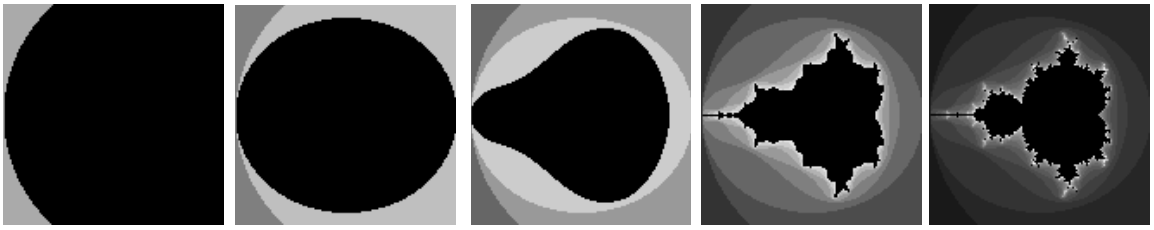
En esta expresión, el valor complejo *c* representa cada coordenada del plano complejo, cuya parte real representa la magnitud en el eje de abscisas, y el imaginario en el de ordenadas.



Se define el fractal como los puntos del plano $\mathbb{C}$ que, ante un número dado de iteraciones (que llamaremos *maxiter*) de la ecuación (1), dicha función está acotada para un cierto radio (para el presente problema, pondremos como radio límite el valor 2.0).

El papel de las iteraciones que ejecutar para comprobar el acotamiento de la serie es determinar el rango de grises que un píxel puede tener cuando no pertenece al conjunto, así como modificar el número de puntos que pertenecen a él (a menor número de iteraciones, más fácil es que la serie sea acotada):

maxiter=3 maxiter=4 maxiter=5 maxiter=10 maxiter=20



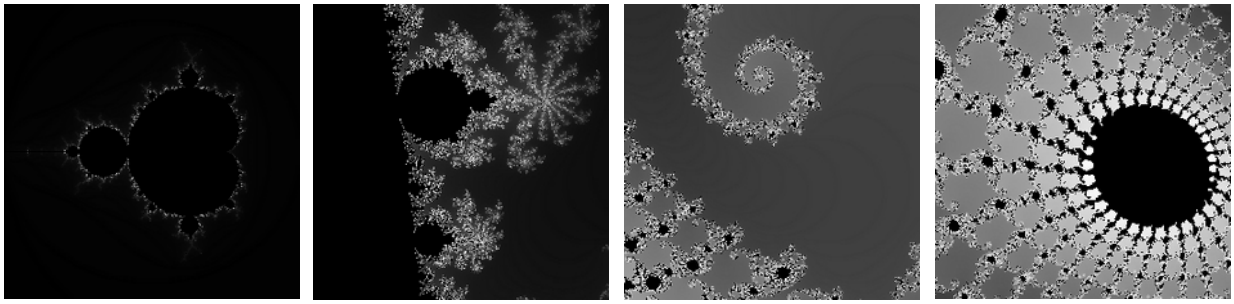
Como vamos a usar el tipo *double*, necesitaremos dos variables para representar el valor complejo de z_n y de c, es decir, $z_n = u + vi$, donde u y v son reales (tipo *double*), e i la constante imaginaria.

Así, el siguiente pseudocódigo se aplica a cada coordenada en la que se desea calcular el valor:

1. Inicializar $z=0$, es decir, sus dos componentes que llamaremos u (parte real) y v (parte imaginaria) a 0.
2. Iterar k desde $1 \rightarrow$ *maxiter*
3. Si $|z_k|^2 < 2^2$:
4. Calcular: $z_{k+1} = z_k^2 + c$
5. Si no: *terminar bucle*
6. Si $k = maxiter$: valor de c=*0* (el punto pertenece al conjunto, pues la serie está acotada)
7. Si no: valor del punto c=*k* (el punto no pertenece al conjunto)

La lógica del cálculo es, por tanto, sencilla: Sobre un área determinada del plano complejo, y para un tamaño de píxel dado (es decir, estableciendo una resolución en filas y columnas de dicha área), calculamos para cada píxel las iteraciones siguiendo el pseudocódigo dado un número máximo de veces (valor dado por parámetro). Si al final de las iteraciones el valor de la serie permanece por debajo de un umbral, pertenece al conjunto (y se almacena con valor 0). Y viceversa. Además, según el número de iteraciones que hace falta para que un píxel sea excluido del conjunto, asignaremos a dicho pixel un nivel de gris determinado (el proporcional a la k). Así, eligiendo el área sobre la que realizamos los cálculos y el número de iteraciones, podremos obtener figuras variadas y pintorescas (coordenadas del área a dibujar expresadas como *xmin*, *xmax*, *ymin* (asumiendo área cuadrada) y siempre 255 iteraciones):

(-2, 1, -1.5) (0.36, 0.4, -0.28) (-0.7479 , -0.7485, 0.101) (-0.7489, -0.74925, 0.1007)



(Si alguien quiere consultar otras zonas del fractal, se puede consultar esta guía: https://math.hws.edu/eck/js/mandelbrot/MB-info.html y las coordenadas en el XML de cada imagen generada).

3. Filtros

La práctica trabaja sobre matrices bidimensionales (con independencia de la forma en que las almacenamos), y por tanto, se pueden aplicar sobre ellas técnicas de procesado de imagen.

Lo que se pretende es implementar algunos procedimientos sencillos y habituales en tratamiento de imágenes.

En la presente práctica se proponen dos operaciones sobre las imágenes generadas: el cálculo del valor promedio y el binarizado (ver apartado 7 de esta memoria).

4. El formato de las Imágenes

Por simplicidad, el formato de imagen usado es **BMP**. También por razones de operatividad, compatibilidad y espacio, el número de *bytes* por *pixel* usado es 1. Un *byte* por *pixel* corresponde con una paleta de 256 colores/tonos de grises, es decir, el valor para cada pixel es $0 \leq pixel \leq 255$. En estas prácticas las imágenes serán almacenadas en memoria principal usando el tipo *double* para dotar al problema de un mayor peso computacional. Obviamente, al escribir las imágenes en fichero se hace la conversión del tipo *double/char*. Esto hará que si queremos grabar a disco imágenes generadas con más de 255 iteraciones, habrá que escalar el valor de *k* a los representables por un byte, es decir, valor de *c=(char)k*255.0/maxiter.* Sin embargo, esto no es relevante para las fases de depurado, producción y corrección, ya que la comparación y cálculo del error se realiza sobre variables *double* sin escalar. En otras palabras: **el código del profesor no lo hace**.

5. Los prototipos suministrados

En ***/opt/PracticasPCP2022_2023/Practica4*** el alumnado dispone del archivo "*PRAC04.tgz*" con los prototipos suministrados. Estos prototipos permiten leer y escribir las imágenes, reservan memoria para almacenar los datos usando 1D *row-major*, comparan variables de imágenes (al objeto de determinar el error cometido), etc.
El fichero también incluye una librería precompilada llamada *mandelProf*, que tiene todas las funciones necesarias para resolver el problema y sirven como herramienta para ayudar al alumnado a comprobar la correcta evolución de su trabajo. En el script de Python, llamado *Fractal.py*, se encontrará también una llamada que compara dos imágenes, siendo su resultado 0 si son iguales.

6. ¿Qué debe hacer el alumnado?

Hipótesis de partida:

- Las imágenes se almacenan como vectores usando 1D *row-major*.
- **Es obligatorio el uso del gestor de colas en la fase de producción o de desarrollo en remoto.**
- Se usará una combinación de Python (gestión de ficheros, llamadas a funciones, cronómetro) y C (cálculos).
- No almacenar permanentemente los resultados (imágenes), por problemas de espacio. **En la fase de producción no se debe guardar ninguna imagen en disco** (la comparativa de error se realiza sobre variables).

El alumnado debe completar el script *Fractal.py*, siguiendo las indicaciones incrustadas en él y *Funciones.c* con los cuerpos de las funciones cuyos prototipos están especificados en *Prototipos.h*. No está permitido cambiar el nombre de las funciones descritas en *Funciones*/*Prototipos*, ni su número o tipo de argumentos.

Obviamente, se pueden añadir más funciones auxiliares a *Funciones.c*, pero no pueden ser llamadas/utilizadas desde *Fractal.py*, serán usadas internamente (por funciones de *Funciones.c*).

Concretamente, se debe implementar:

1 **mandel:** Una función que calcule el fractal en el área dada (se generarán imágenes cuadradas) y con los umbrales y resoluciones recibidos igualmente por parámetros. Versión en python y en C. En cuanto a la versión en C, **secuencial y paralela** con OpenMP. No es necesario codificar dos funciones, simplemente con una función bien diseñada y modificando el valor de la variable de entorno *OMP_NUM_THREADS*, se ejecuta el mismo código en secuencial o paralelo. La función también recibe la estructura matricial sobre la que hacer los cálculos vacía, para ser rellenada. La funcion es *void*. Qué mecanismos OpenMP use cada cual queda a su elección.

2 **Promedio:** Una función que calcule el gris promedio de la imagen. **Secuencial y paralela** con OpenMP.

3 **Binarizado:** Una función, **secuencial y paralela** con OpenMP, para realizar el binarizado de la imagen: asignar a todos y cada uno de los píxeles de la imagen:
   - 255 si el valor del pixel es > que el valor de la media (resultado de *promedio*).
   - 0 si el valor del pixel es ≤ que la media.

En *Fractal.py* se reserva espacio para almacenar distintas instancias de la imagen a generar/procesar. Su objetivo es simplificar el trabajo a realizar por parte del alumnado. El alumnado es libre de usar dichos espacios como mejor le convenga, optar por variantes óptimas en memoria, etc.

8 Comparativas, Conclusiones, etc.

Finalizada la codificación y depuración de las funciones indicadas en la sección anterior, se debe pasar a la etapa de producción, usando el gestor de colas para completar la Tabla 1 tantas veces como tipos de procesador se usen, llamando a las tres funciones seguidas. En la fase de producción se pueden usar tanto los procesadores Intel I3 (**ColaI3**) como el Intel Xeon (**ColaXeon**) o el AMD Ryzen 7 (**ColaGPU**).

Todas las pruebas se harán sobre la función mandel. Los parámetros para dicha función serán *maxiter*=1000 y sobre las coordenadas *xmin*=-0.7489, *xmax*=-0.74925, *ymin*= 0.1007.

yres	t_teórico secuencial	t_teórico paralelo	t_python	t_C_sec	t_C_paralelo 1	t_C_paralelo2	...
256							
512							
1024							
2048							
4096			N/A				
8192			N/A				

Tabla 1 Experimentación mínima a realizar. “N/A” significa que no se deben realizar esos cálculos concretos. Tabla a rellenar para cada función.

La comparativa debe realizarse en dos planos/niveles. Por un lado, el alumnado debe realizar el análisis teórico de la complejidad temporal y espacial, tanto secuencial como paralela, calculando, también, la eficiencia, todo ello para el caso peor. Por el otro, comparar el plano teórico con el empírico de cada solución (secuencial y paralela) por separado.

Además, se debe explicar por qué se han seleccionado los mecanismos OpenMP usados en cada función. Nótese que el algoritmo presentado tiene costes distintos según el píxel, de forma que el uso de técnicas de *scheduling* u otras puede modificar los tiempos. Obviamente, se pueden añadir todas aquellas comparativas adicionales que cada cual considere conveniente. Finalmente, presentar las conclusiones argumentadas.

7. ¿Qué y dónde entregar?

El fichero ***Funciones.c*** modificado por cada alumnx (el resto no son necesarios) debe estar en la carpeta ***$HOME/PRAC04***. El profesor usará una herramienta que recorrerá las carpetas *PRAC04* de cada carpeta personal y:

1 Copiará dicho fichero a un lugar de acceso restringido. Si la fecha de la última modificación, en el momento de la copia, es posterior a la establecida para cada grupo, la práctica se considerará no entregada.

2 Ejecutará *make* (con el *Makefile* original). Construirá un fichero de órdenes adecuado para OGE/SGE y enviará el trabajo al gestor. Comparará la solución alcanzada con la del profesor.

Si alguno de los pasos anteriores falla, o la comparativa es inconsistente, la práctica será considerada incorrecta y no se corregirá la documentación.

El fichero (en *$HOME/PRAC04*) será copiado automáticamente a las **23:55:00** horas del día:

- **20 de noviembre** de 2022 para el grupo de los lunes.
- **22 de noviembre** de 2022 para los grupos de los miércoles.
- **17 de noviembre** de 2022 para los grupos de los viernes.

Después de esta fecha, se prohíbe cualquier modificación del mismo, para poder realizar una corrección consistente de los datos aportados en la memoria. Si se quiere seguir trasteando pasada esa fecha, es obligatorio hacerlo sobre una copia en otra carpeta personal diferente.

Para la documentación, el alumnado debe subir al Campus Virtual un único fichero en formato PDF resolviendo todas las cuestiones planteadas en la sección 8 de este enunciado, además de las aportaciones voluntarias que cada cual estime oportuno. Esta documentación será integrada con la

documentación de la **práctica 6**, cuya fecha definitiva de entrega son las **23:55:00** horas del **21 de diciembre** de 2022.

La tarea del CV para subir la documentación es "**Actividades de Evaluación → Práctica 6**"