Tarea 3: Introducción a la computación de alto rendimiento (10%)

Marlon Brenes*

El propósito de esta tarea es practicar los conceptos de paralelismo de memoria compartida y distribuida. La tarea involucra entregar código fuente de C++ (utilizando bibliotecas especializadas) con su respectivo reporte detallando el análisis. La tarea se puede realizar en parejas, pero **cada estudiante debe entregar sus propios archivos**. Usted debe entregar los siguientes archivos para ser evaluados:

- `mandelbrot.cpp`
- `comunicación_circular.cpp`

PARTE I: MEMORIA COMPARTIDA (6%)

El conjunto de Mandelbrot es un conjunto de dos dimensiones que exhibe comportamiento caótico y fractal. A pesar de la sencillez de la regla que genera el conjunto, la complejidad asociada al conjunto es altamente magnificada. El conjunto está definido en el plano complejo como el conjunto de números c tales que la función

$$f_c(z) = z^2 + c \tag{1}$$

no diverge al infinito cuando se inicia la iteración con $z = 0$. Esto es, cuando la secuencia $[f_c(0), f_c(f_c(0)), \cdots]$ permanece acotada en su valor absoluto.

Al visualizar los números c que satisfacen esta condición en el plano complejo (Fig. 1), se obtiene un **fractal**. Note que $\mathrm{Re}[c] \in [-2, 1]$ y $\mathrm{Im}[c] = [-1, 1]$.

- Se le ha proporcionado una aplicación en `C++` que genera el conjunto de Mandelbrot e imprime los resultados en formato `ASCII` en la terminal
- La aplicación es muy rudimentaria, dado que no se imprimen distintos símbolos dependiendo del valor de acotación mencionado anteriormente. Lo que hace es imprimir '#' o '.' dependiendo si c da lugar a iteraciones acotadas después de cierto valor de corte. Para visualizar el conjunto en la terminal, asegúrese de que la terminal tenga el tamaño de al menos 155x50 caracteres

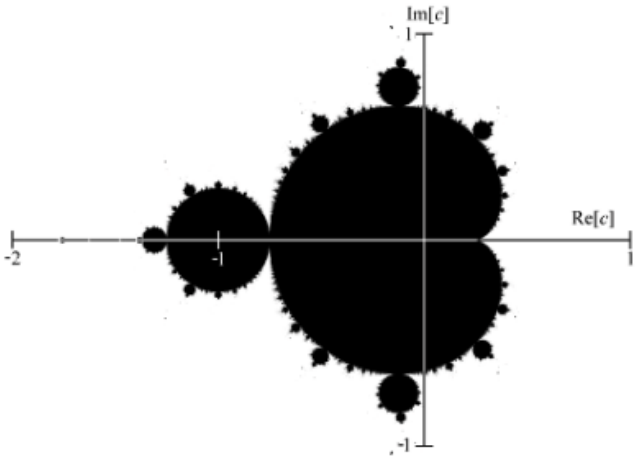

FIG. 1. Conjunto de Mandelbrot

* marlon.brenes@utoronto.ca

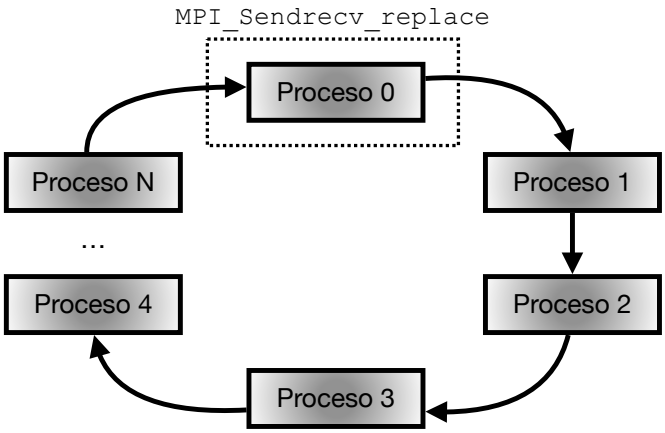

FIG. 2. Patrón de comunicación circular

- No obstante, la aplicación es suficiente para observar los detalles mínimos del set de Mandelbrot
- **Su tarea** es encontrar una metodología para paralelizar el procedimiento que genera el conjunto utilizando OpenMP en el paradigma de memoria compartida y medir su aceleración (gráfico de speed-up) utilizando hasta 8 procesadores [6%]
- Algunos detalles a considerar:
  - Una aplicación en paralelo debe dar lugar a los mismos resultados que una aplicación en serie: su aplicación acelerada con OpenMP **debe generar el mismo gráfico en la terminal al que genera la aplicación en serie**.
  - No modifique los parámetros de ejecución: las modificaciones deben realizarse al código base solo para asegurarse de que el conjunto de Mandelbrot se imprime de manera correcta en la terminal y se acelera el procedimiento que lo genera

PARTE II: MEMORIA DISTRIBUIDA (4%)

Su tarea es implementar un **patrón de comunicación circular** utilizando N procesos.

- Cada proceso p le envía un mensaje al proceso $p+1$ y recibe un mensaje del proceso $p-1$
- Inicialmente, cada proceso tiene un buffer en el cual almacena su rango
- Cada proceso envía su propio rango (p) al proceso siguiente ($p+1$) y recibe el rango del proceso emisor ($p-1$). Cada proceso guarda en el buffer inicial el mensaje recibido
- En la siguiente iteración, el buffer se comparte de manera circular
- Utilice comunicación bloqueada (*blocking calls*) y el mismo buffer de memoria para enviar y recibir el mensaje
- En cada iteración, el proceso $p=0$ escribe en terminal el contenido de su buffer
- Después de N iteraciones, la comunicación circular se completa y el proceso $p=0$ recibe en su buffer de nuevo el valor 0

Utilice Fig. 2 como una guía para realizar el patrón de comunicación. **El reporte debe incluir como se compila el código, como se ejecuta y un ejemplo de ejecución entregando los resultados esperados**.