Práctica número 2
Programación con Pthreads

Ejercicio 1. Desarrolle un algoritmo paralelo que compute una suma de vectores: $A_i = B_i + C_i$. Para ello, considere lo siguiente:

- Identifique las regiones de código que pueden ejecutarse en paralelo y las que no. ¿Existen dependencias?
  El cómputo de la suma de vectores puede hacerse en paralelo, lo que no puede realizarse en paralelo es la inicialización de las estructuras y variables necesarias. no hay dependencias ya que cada hilo trabaja sobre una zona distinta del vector.

- Analice cómo distribuir el trabajo entre los hilos. ¿El cómputo es regular? ¿La arquitectura es homogénea? Mida el tiempo de ejecución para diferentes valores de N. Analice el rendimiento.
  Sí, el cómputo es regular. Todos los hilos reciben la misma carga (aproximadamente) y realizan la misma tarea. Sí, todos los hilos se ejecutan en CPUs con las mismas características. Esto permite suponer que cada hilo tardará un tiempo similar en procesar su parte.

n	paralelo	secuencial
96186620	0.142036	0.759862

$$Speedup = \frac{Tsecuencial}{Tparalelo} = \frac{0{,}759862}{0{,}142036} = 5,3497$$

$$Eficiencia = \frac{TSpeedup}{T(hilos)} = \frac{5{,}3497}{12} = 0,4458$$

Ejercicio 2. Desarrolle un algoritmo paralelo que compute la multiplicación de matrices cuadradas de NxN. Primero, considere a la versión optimizada del ejercicio 6 de la práctica anterior como algoritmo base. Luego, paralelice la versión que computa por bloques. Mida el tiempo de ejecución para N={512, 1024, 2048, 4096} y T={2,4,8}. Analice el rendimiento.(lerolero)

N	T	paralelo	secuencial
512	2		
512	4		
512	8		
1024	2		

1024	4		
1024	8		
2048	2		
2048	4		
2048	8		
4096	2		
4096	4		
4096	8		

$$Speedup = \frac{Tsecuencial}{Tparalelo} = \frac{0{,}759862}{0{,}142036} = 5{,}3497$$

$$Eficiencia = \frac{TSpeedup}{T(hilos)} = \frac{5{,}3497}{12} = 0{,}4458$$

Ejercicio 3. Desarrolle un algoritmo paralelo que cuente la cantidad de veces que un elemento X aparece dentro de un vector de N elementos enteros. Al finalizar, la cantidad de ocurrencias del elemento X debe quedar en una variable llamada ocurrencias. Para la sincronización emplee mutex-locks. Pruebe con diversos tamaños de N y T={2,4,8}. Analice el rendimiento.

N	paralelo	secuencial
96186620	0.025208	0.288655

$$Speedup = \frac{Tsecuencial}{Tparalelo} = \frac{0{,}288655}{0{,}025208} = 11{,}4509$$

$$Eficiencia = \frac{TSpeedup}{T(hilos)} = \frac{11{,}4509}{12} = 0{,}9542$$

Se observa que la implementación paralela supera ampliamente a la secuencial en términos de rendimiento. Al dividir el arreglo en porciones más pequeñas y asignar cada una a un hilo diferente, se logra un procesamiento concurrente que reduce significativamente el tiempo total de ejecución. Esto permite que las lecturas del arreglo se realicen de forma simultánea, mejorando notablemente la eficiencia del sistema.

El *speedup* obtenido (≈11.45) indica que el algoritmo paralelo es más de 11 veces más rápido que el secuencial, mientras que la eficiencia (≈0.954) demuestra un excelente aprovechamiento de los 12 hilos disponibles. Este valor cercano a 1 sugiere que el algoritmo paralelo escala muy bien y logra una distribución casi óptima de la carga de

trabajo. En conclusión, el enfoque paralelo mejora sustancialmente el rendimiento en comparación con la versión secuencial.

Ejercicio 4. Desarrolle un algoritmo paralelo que calcule el valor promedio, mínimo y máximo de los números almacenados en un vector de tamaño N. Para la sincronización emplee semáforos. Pruebe con diversos tamaños de N y T={2,4,8}. Analice el rendimiento.

N	paralelo	secuencial
96186620	0.036587	0.264643

$$Speedup = \frac{Tsecuencial}{Tparalelo} = \frac{0{,}264643}{0{,}036587} = 7{,}233252$$

$$Eficiencia = \frac{TSpeedup}{T(hilos)} = \frac{7{,}233252}{12} = 0{,}602771$$

En este ejercicio se desarrolló un algoritmo paralelo para calcular el valor promedio, mínimo y máximo de los elementos de un arreglo extenso. La paralelización de estas operaciones permite procesar porciones del arreglo de forma concurrente, lo cual reduce considerablemente el tiempo de ejecución total.

El resultado obtenido muestra un *speedup* de aproximadamente 7.23, lo que implica que la versión paralela es más de 7 veces más rápida que la secuencial. La eficiencia, aunque más baja que en el ejercicio anterior (≈0.60), sigue siendo adecuada considerando que se deben realizar tres cálculos distintos con sincronización mediante semáforos, lo cual introduce cierta sobrecarga.

En general, el algoritmo paralelo demuestra ser efectivo, especialmente para grandes volúmenes de datos, aunque el tipo de operación y la complejidad de sincronización influyen en el rendimiento final.