Práctica Nro. 2

Programación con Pthreads

Información útil para compilar y ejecutar:

- Para compilar en Linux con gcc+Pthreads: *gcc -pthread –o salidaEjecutable archivoFuente*
- Para ejecutar: *./salidaEjecutable arg1 arg2 … argN*

Pautas generales

- Para obtener el tiempo de ejecución de todos los algoritmos se debe utilizar la función provista por la cátedra (*dwalltime*).
- Por convención sólo deberá tomarse el tiempo de ejecución del procesamiento de datos. Esto significa excluir del tiempo de ejecución:
  - *Reserva y liberación de memoria.*
  - *Inicialización de estructuras de datos.*
  - *Impresión y verificación de resultados.*
  - *Impresión en pantalla (printf)*
- Las pruebas deben realizarse de forma aislada a la ejecución de otras aplicaciones. Se debe ejecutar desde consola, sin otras aplicaciones ejecutándose al mismo tiempo.
- Para todos los ejercicios se debe calcular el speedup y la eficiencia del algoritmo paralelo respecto al secuencial.

Ejercicios

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?
   - 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 y T={2,4,8}. Analice el rendimiento.

2. Desarrolle un algoritmo paralelo que compute la multiplicación de matrices cuadradas de *NxN*. Primero, considere a la versión optimizada del ejercicio 8 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.

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.

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.

5. Desarrolle un algoritmo paralelo que obtenga la intersección de dos conjuntos de números enteros. Pruebe con diversos tamaños de N y T={2,4,8} y analice el rendimiento . Tenga en cuenta la siguiente propiedad:

$$A \cap B = A \cap (B_0 \cup B_1 \cup B_2 \ldots \cup B_n) = (A \cap B_0) \cup (A \cap B_1) \cup (A \cap B_2) \ldots \cup (A \cap B_n)$$