UNIVERSIDAD DEL VALLE DE GUATEMALA

CC3069 - Computación Paralela y Distribuida

Sección 21

Ing. Miguel Novella Linares

Proyecto #1

Programación paralela con OpenMP

Davis Alvarez, 15842

Juan Solorzano, 18151

Mario Perdomo, 18029

GUATEMALA, 18 de marzo de 2022

Tabla de contenidos

Introducción

Para simular la forma en que se propaga el calor en un barra de metal se utiliza una Ecuación Diferencial Parcial (PDE). En esta ecuación se utiliza una metodología unidimensional, la que asigna un ancho 0 a la barra. Con ayuda de aproximaciones diferenciales finitas de las derivadas en el tiempo y la distancia se puede discretizar esta PDE con intervalos de tiempo iguales, y luego utilizando el método explícito de Euler para integraciones en el dominio del tiempo se obtiene el algoritmo:

$$T_j(t_i + 1) = T_j(t_i) + \frac{c\Delta t}{(\Delta x)^2}(T_{j-1}(t_i) - 2T_j(t_i) + T_{j+1}(t_i)$$

Este algoritmo simplificado para el cálculo de la nueva temperatura, basado en temperaturas anteriores, se utilizará en este proyecto para realizar la transformación de un programa secuencial en uno paralelo con ayuda de la herramienta OpenMP.

Objetivos

- Implementar y diseñar un programa para la paralelización de procesos con memoria compartida usando OpenMP.
- Aplicar el método PCAM y los conceptos de patrones de partición y programa para modificar un programa secuencial y volverlo paralelo.

Disipación de calor

La ecuación de difusión, generalmente conocida como ecuación de calor es:

$$\frac{\partial T(x,t)}{\partial t} = c\frac{\partial^2 T(x,t)}{\partial x^2}$$

donde $T(x,t)$ es la función que buscamos resolver y donde x es la coordenada en el espacio y t es el tiempo. La constante c es el coeficiente de difusión que determina qué tan rápido cambia $T(x,t)$ en el tiempo. Los problemas de difusión de calor suelen experimentar cambios drásticos al principio, pero luego la evolución de $T(x,t)$ se vuelve más lenta cada vez.

Es posible resolver $T(x,t)$ usando un esquema explícito, pero las restricciones del paso del tiempo se convierten en menos favorables, que por ejemplo, para la ecuación de ola. Y más importante, la solución de $T(x,t)$ es bastante suave y cambia lentamente, pequeños cambios de tiempo no son convenientes y no se requiere exactitud, ya que el proceso de difusión va convergiendo a un estado estacionario.
Para obtener una solución equivalente de la ecuación de difusión y para aplicar métodos numéricos, necesitamos condiciones iniciales y de límites. La ecuación de difusión va con una condición inicial $T(x,0) = I(x)$, donede I es una función prescrita. Un límite es necesario a cada uno de los puntos de lados, lo que significa que en una dimensión T debe ser conocida, T debe ser conocida o alguna combinación de estas.

Iniciando con la condición de límite más simple: $T = 0$. El problema de difusión en una dimensión puede ser especificado como:

$$\frac{\partial T(x,t)}{\partial t} = c\frac{\partial^2 T(x,t)}{\partial x^2},\ \ x \in (0,\ L),\ t \in (0,\ T] \qquad (1)$$

$$T(x,\ 0) = \ I(x),\ x \in [0,\ L] \qquad (2)$$

La ecuación (1) es conocida como ecuación de una dimensión de difusión, y generalmente llamada ecuación de calor. Con solo la derivada de primer orden una condición inicial es necesaria, mientras que, la derivada de segundo orden lleva a la demanda de dos condiciones de límite. El parámetro c debe ser dado y es llamado coeficiente de difusión.

Para discretizar esta ecuación, el primer paso que debemos realizar es reemplazar el dominio $[0,\ L]$ X $[0,\ T]$por un conjunto de puntos:

$$x_i = i\Delta x,\ i = 0,\ \ldots,\ N_x,$$
$$t_n = n\Delta x,\ n = 0,\ \ldots,\ N_t$$

Así, T_i^n denota la función que aproxima $T(x_i,\ t_n)$ para $i = 0,\ \ldots,\ N_x$ y $n = 0,\ \ldots,\ N_t$. Exigiendo que la PDE (1) sea cumplida en el punto $(x_i,\ t_n)$ nos lleva a la ecuación:

$$\frac{\partial T(x_i, t_n)}{\partial t} = c\frac{\partial^2 T(x_i, t_n)}{\partial x^2}$$

El siguiente paso es reemplazar las derivadas por aproximaciones de diferencias finitas. El método más simple es utilizar el método Forward Euler en el tiempo y una diferencia central en el espacio. Con esto logramos transformar PDE a una ecuación algebraica:

$$[D_t^+ T = \alpha D_x D_x T]_i^n$$

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = c\frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2}$$

La característica clave de esta ecuación es que es algebraica, que lo hace más simple de resolver. Asumimos que T_i^n es conocida y que T_{i+1}^nes la única incógnita para resolver:

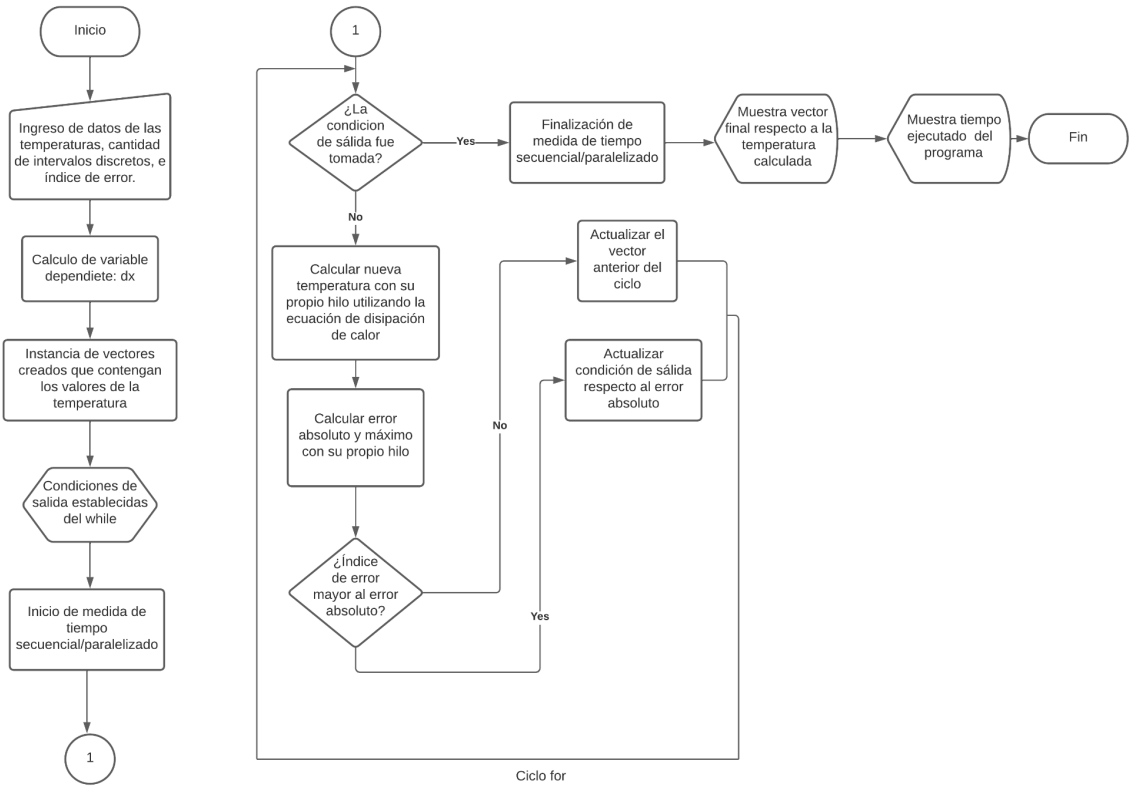
$$T_i^{n+1} = T_i^n + \beta\,(T_{i+1}^n - 2T_i^n + T_{i-1}^n)$$

Donde β no tiene dimensiones, que contiene a la coeficiente de difusión y las discretizaciones del cambio del tiempo Δt y el cambio en el espacio Δx.

Planificación de actividades

No.	Semana	Tarea	Descripción
1	28 feb - 4 mar	Plantilla Reporte	Se crea el documento compartido para el reporte.
2		Algoritmo de disipación de calor	Se tiene el algoritmo base de cálculo de disipación de calor para iniciar a trabajar en su optimización
3	7 mar - 11 mar	Partición de los datos	Se planea la forma en que se van a dividir los datos para tu paralelización y se implementa
4		planificación de variables	Se identifica las variables privadas y compartidas que se utilizaran en el algoritmo
5		paralelizar algoritmo	se inicia con la paralelización del algoritmo
5	14 mar - 18 mar	paralelizar algoritmo	se inicia con la paralelización del algoritmo
6		realizar pruebas	Se pone a prueba el algoritmo paralelo y se calcula el speedup
7		Interfaz amigable de ingreso de datos	Se realiza una interfaz amigable para el ingreso de los datos al algoritmo
8		Completar reporte	Se complementa el reporte, discutiendo los resultados, colocando las conclusiones y recomendaciones.
9		Entregar/Presentar	Realizar entrega en canvas

Diagrama de flujo



Resultados

```
Ingrese la cantidad de threads: 3
Ingrese el numero de intervalos: 10
Ingrese la longitud: 100
Ingrese la temperatura inicial de la barra: 10
Ingrese la temperatura de la frontera izquierda: 10
Ingrese la temperatura de la frontera derecha: 100
Calculando con 10 intervalos discretos

Iteracion 67149

 Vector solucion
2.369250 4.613116 6.617978 8.278532 9.477085 10.062053 9.853036 8.691886 6.530769 3.514354

Error = 0.000100
Tiempo =  3.117000 s
```

Resultado #1: Paralelo

```
Ingrese el numero de intervalos: 10
Ingrese la longitud: 100
Ingrese la temperatura inicial de la barra: 10
Ingrese la temperatura de la frontera izquierda: 10
Ingrese la temperatura de la frontera derecha: 100

Calculando con 10 intervalos discretos

Iteracion 67149

 Vector solucion
2.369250 4.613116 6.617978 8.278532 9.477085 10.062053 9.853036 8.691886 6.530769 3.514354

Error = 0.000100
Tiempo =  0.060000 s
```

Resultado #2: Secuencial

Discusión

Durante las pruebas que se realizaron al proyecto, el programa paralelizado mostraba tiempos de ejecución tardados, incluso al agregar la directiva omp for dentro del programa. La directiva omp single era una parte vital del código del proyecto, debido a que de todos los threads, uno se debía encargar de identificar los errores absoluto dentro de cada iteración de una nueva temperatura.

Los otros hilos del equipo, que no ejecutan el bloque de código de la evaluación de errores, deben esperar en una barrera implícita al final de la construcción única de la directiva single. Esto es una limitación del programa debido a que afectaba al equipo de threads esperar al thread responsable de comparar errores entre las temperaturas procesadas. Podemos observar los resultados, donde los tiempos de ejecución secuenciales fueron más rápidos que los tiempos paralelos, gracias a la limitación de la directiva single.

Conclusiones

1. El tiempo secuencial fue mucho más rápido que el tiempo paraleziado con directivas de omp for y single.
2. La limitación más impactante fue la directiva single, debido a que obligaba la espera de su proceso hacía los thread loops asignados.

Recomendaciones

1. Observando los resultados, se pudo haber creado funciones en el proceso de obtener los errores absolutos y convertir los en procesos con directiva task, con el fin de mejorar el rendimiento del programa.

Bibliografía

Pacheco, P. (2011). “An Introduction to Parallel Programming. ISBN 978-0-12-374260-5 (hardback)

Bitácora de pruebas

Threads = 6
Intervalos = 10
Longitud= 100
Temp inicial= 10
Temp izquierda = 10
Temp derecha= 100

Secuencial

Prueba	Tiempo
1	0.060000 s
2	0.060000 s
3	0.060000 s

Paralelo

Prueba	Tiempo
1	5.794000 s
2	5.802000 s
3	5.760000 s

Threads = 3
Intervalos = 10
Longitud= 100
Temp inicial= 10
Temp izquierda = 10
Temp derecha= 100

Secuencial

Prueba	Tiempo
1	0.060000 s
2	0.060000 s
3	0.060000 s

Paralelo

Prueba	Tiempo
1	3.179000 s
2	3.121000 s
3	3.033000 s

Threads = 3
Intervalos = 10
Longitud= 100
Temp inicial= 60
Temp izquierda = 20
Temp derecha= 150

Secuencial

Prueba	Tiempo
1	0.089000 s
2	0.088000 s
3	0.089000 s

Paralelo

Prueba	Tiempo
1	4.577000 s
2	4.634000 s
3	4.585000 s

inconvenientes