Universidad Del Valle de Guatemala
Computación Paralela
Miguel Novella

Proyecto # 1

18/03/2022
Augusto Alonso #181085
Esteban Cabrera #17781
Mario Sarmientos #17055

Índice

Antecedentes:

Se presenta a continuación una breve revisión de datos de trabajos acerca del mismo tema o que tienen un enfoque parecido al expuesto en este proyecto:

1. A continuación se presenta un trabajo relacionado con la disipación de calor utilizando la técnica del punto material (MPM), su metodología fue trasladar el algoritmo a un lenguaje de programación, y una vez ahí jugar con los diferentes parámetros para observar y graficar el comportamiento.

   (John A. Nairn, n.d.) Modeling heat flow across material interfaces and cracks using the material point method.

2. Así mismo, hay precedente de trabajos anteriores que han utilizado tecnologías similares de la mano de CUDA para simular la disipación de calor en distintos materiales.
   (Vinaya Sivanandan, 2015) *Designing a parallel algorithm for Heat conduction using MPI, OpenMP and CUDA*.

3. El el trabajo “Parallel algorithm for solving the problems of heat and mass transfer in the open geothermal system” se explora la posibilidad de mejorar el algoritmo de disipación de calor en tres dimensiones, al igual que en el presente proyecto se propone la utilización de de openMP para la paralelización del algoritmo.

   (Vladimir E. Misilov, 2020) *Parallel algorithm for solving the problems of heat and mass transfer in the open geothermal system*

4. Se de han visto proyectos de alcance similar como “Heat transfer simulation of heat storage unit with nanoparticles and fins through a heat exchanger” En el que utilizan un algoritmo de dispersión de calor en superficies planas para crear un algoritmo mayor de transferencia de calor multicapa.

   (M.Sheikholeslami, 2019) *Heat transfer simulation of heat storage unit with nanoparticles and fins through a heat exchanger*.

Objetivos

- Implementa y diseña programas 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

Catálogo de funciones

double maxFromArray(double data[], int start, int end)
double data[] valor máximo de las diferencias de temperaturas.
int start es el inicio del tiempo de ejecución
int end es el final del tiempo de ejecución

Devuelve el máximo valor de las diferencias de temperaturas.

void resolveInitialValues()

Resuelve los valores iniciales de los deltas basados en la ecuación diferencial unidimensional de disipación de calor.

void sendTask(int start, int end, int num_threads, double temperatures2[], double temperatures1[], double diff[])

int startes el inicio del tiempo de ejecución
int end es el final del tiempo de ejecución
int num_threads cantidad de threads que se ejecutan
double temperatures2[] set de temperaturas actuales
double temperatures1[] set de temperaturas pasadas
double diff[] set de diferencias temperaturas

Catálogo de variables

#define ERR 0.00001
Precisión o diferencia requerida

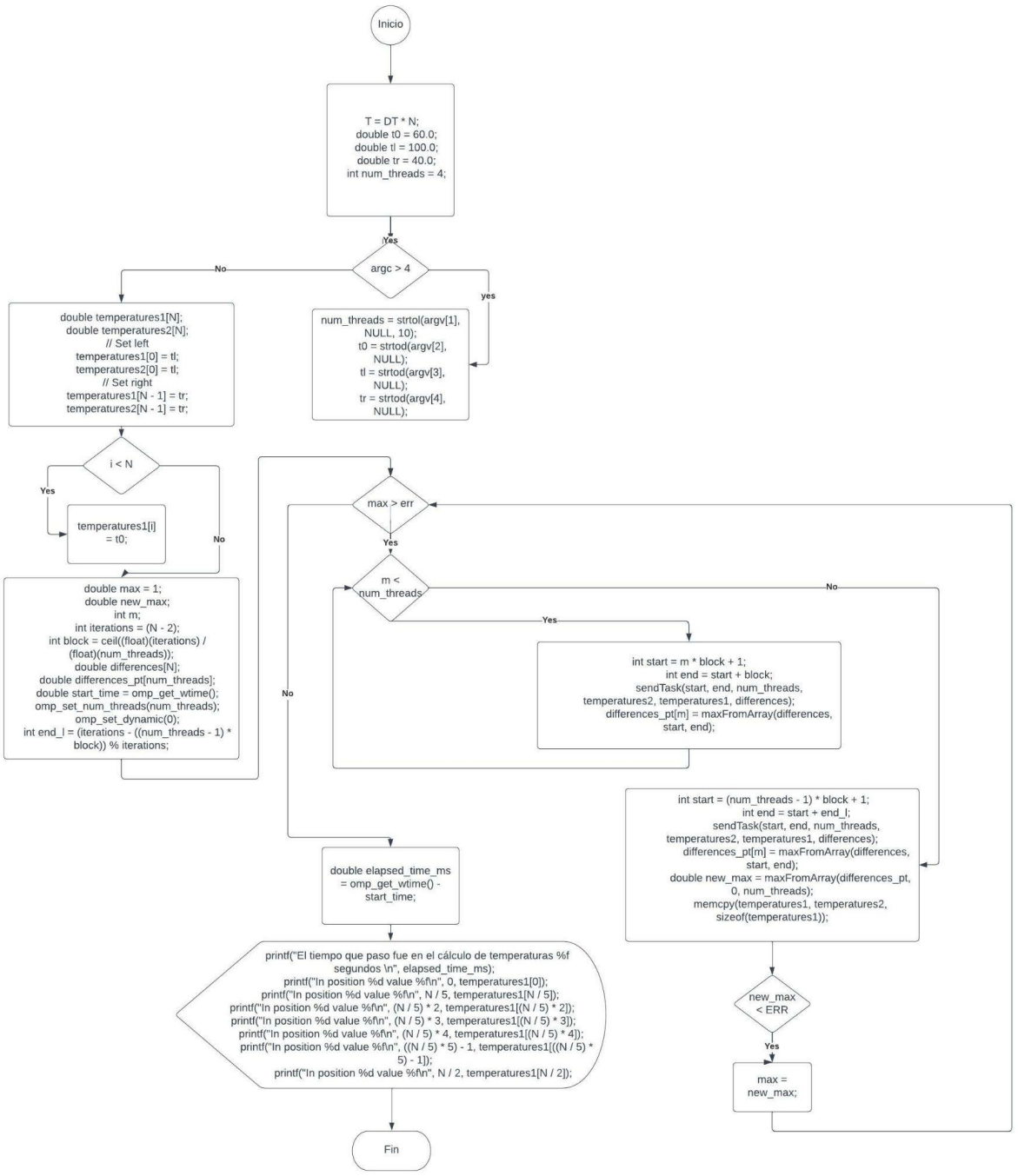
#define DIFUSIVE_VAL 10e-4
Es el número de intervalos discretos.

double temperatures1[N];

double temperatures2[N];
Las temperaturas iniciales respectivamente.

double elapsed_time_ms = omp_get_wtime() - start_time;
Tiempo transcurrido de la ejecución del programa.

Diagrama de flujo



Resultados

con 4 threads N=5000

```
→  proyect git:(master) ✗ ./proyectParallel
El tiempo que paso fue en el cálculo de temperaturas 46.305729 segundos
In position 0 value 100.000000
In position 1000 value 78.811306
In position 2000 value 65.400775
In position 3000 value 58.226930
In position 4000 value 50.706127
In position 4999 value 40.000000
In position 2500 value 61.444882
```

Imagen 1. Ejecución 4 threads, N=5000 en paralelo.

Secuencial

```
→  proyect git:(master) ✗ ./proyect
El tiempo que paso fue en el cálculo de temperaturas 56.033104 segundos
In position 0 value 100.000000
In position 1000 value 78.811306
In position 2000 value 65.400775
In position 3000 value 58.226930
In position 4000 value 50.706127
In position 4999 value 40.000000
In position 2500 value 61.444882
→  proyect git:(master) ✗ ./proyect
```

Imagen 2. Ejecución en secuencial

```
→  proyect git:(master) ✗ ./proyect
El tiempo que paso fue en el cálculo de temperaturas 63.797415 segundos
In position 0 value 100.000000
In position 1000 value 78.811306
In position 2000 value 65.400775
In position 3000 value 58.226930
In position 4000 value 50.706127
In position 5000 value 40.000000
In position 2500 value 61.444882
```

Imagen 3. Ejecución en paralelo con el tiempo más tardado obtenido.

Nombre	Tiempo
TSeq	54.43
Tiempo parallelo 6 cores	28.18

Tabla 1. Tiempo en paralelo 6 cores.

SpeedUp	
Tpar	1.93151171

Tabla 2. Speedup.

Efficiency	N = 20
Tpar	0.09657558552

Tabla 3. Eficiencia con N=20 y tiempo paralelo.

Límite superior teórico	
Tpar	0.04902693213

Discusión

Definitivamente la rapidez del cálculo paralelo fue mayor a la del cálculo secuencial. Como se observa en las imágenes 1 y 2, para el cálculo en paralelo se obtuvo una media de 46.305 s y para el secuencial una media de 56.033, dando como resultado una rapidez del 17.361% mayor. Cabe aclarar que estas medias fueron basadas en cálculos haciendo uso de 4 threads, lo cual significa que con mayor cantidad de threads se dará un resultado significativamente incrementado para el porcentaje mencionado. En términos de exactitud en los cálculos, los valores siguen siendo exactamente iguales, lo único que varió fue el tiempo de ejecución, cosa que se pretendía lograr paralelizando el programa.

En la imagen 1 y 2, se observa que en cada iteración, la disipación del calor incrementa, por lo que la temperatura se reduce respectivamente. Esto es debido al método de euler se está aplicando, el cual sirve para la integración del dominio del tiempo tomando en cuenta una fuente externa de calor para una barra L. Sin embargo, el tiempo de ejecución del programa tanto secuencial como paralelo no tiene nada que ver con los resultados del dominio del tiempo del método de euler.

Basado en la ley de Amdahl, existe un límite superior teórico de la aceleración de un programa para añadir más unidades de ejecuciones de modo paralelo. En el caso del proyecto, el resultado del límite superior teórico fue de 0.096. Este no supera la unidad, por lo que es correcto decir que no se puede añadir más ejecuciones paralelas para brindar más aceleración al programa y obtener un resultado aún más rápido que el actual.

El speedup que se obtuvo fue de 1.931, el cual es mayor a, lo cual significa que el cálculo paralelo se hizo en menos de la mitad del tiempo del cálculo secuencial. Esto verifica el funcionamiento de la paralelización y que la implementación y manejo de la librería OMP se hizo correctamente. Tomando de referencia el caso de éxito con la paralelización, se puede afirmar que esta brinda buenos resultados al momento de implementar ecuaciones diferenciales dentro de los cálculos que realiza el programa.

Conclusiones

- Las mediciones de temperatura se hicieron 17 % más rápidas en el programa paralelizado.
- El speedup fue de 1.931, confirmando que el cálculo paralelo se hizo en menos de la mitad del tiempo del cálculo secuencial.
- No se pueden añadir más unidades de ejecución paralela debido a que el límite de ejecución teórico no lo permite.

Recomendaciones

Se tienen como recomendaciones las siguientes:

1. Se debe tomar en cuenta que el último thread del bloque gastará tiempo de ejecución del cálculo debido a que este se debe ejecutar independientemente de las iteraciones que se hagan. Es por esto que las iteraciones tienen que tomar en cuenta la cantidad de threads menos uno.
2. La paralelización funciona perfectamente para disminuir el tiempo de ejecución de las implementaciones de ecuaciones diferenciales como las que involucran el método de euler.

Bibliografía

John A. Nairn. (n.d.). Modeling heat flow across material interfaces and cracks using the material point method. Retrieved March 19, 2022, from http://www.cof.orst.edu/cof/wse/faculty/Nairn/papers/InterfaceHeat.pdf

M.Sheikholeslami. (2019, 02 03). *Heat transfer simulation of heat storage unit with nanoparticles and fins through a heat exchanger*. ScienceDirect. https://www.researchgate.net/publication/297613897_Designing_a_parallel_algorithm_for_Heat_conduction_using_MPI_OpenMP_and_CUDA

Vinaya Sivanandan. (2015, 02 01). *Designing a parallel algorithm for Heat conduction using MPI, OpenMP and CUDA*. Research Gate. https://www.researchgate.net/publication/297613897_Designing_a_parallel_algorithm_for_Heat_conduction_using_MPI_OpenMP_and_CUDA

Vladimir E. Misilov. (2020, 04 02). *Parallel algorithm for solving the problems of heat and mass transfer in the open geothermal system*. Parallel algorithm for solving the

problems of heat and mass transfer in the open geothermal system.

https://aip.scitation.org/doi/10.1063/5.0035531

Apéndice

Formula1: Eficiencia

$$E(n) = \frac{S(n)}{n} = \frac{T(1)}{n * T(n)}$$

Formula 2: Ley de Ahmdal

$$T_m = T_a \cdot \left((1 - F_m) + \frac{F_m}{A_m} \right)$$

formula 3: Disipación De calor una barra

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