Presentación Proyecto

Resolución de la ecuación de calor mediante diferencias finitas

Ethiel Carmona
Joaquín Castillo
María Paz Morales
Axel Reyes

Taller de Modelos y Métodos Cuantitativos
Departamento de Informática
Universidad Técnica Federico Santa María

27 de marzo de 2021

Índice

Índice

Presentación del Problema: Ecuación del Calor

La ecuación del calor es una ecuación diferencial que describe el intercambio de temperatura a lo largo del tiempo entre un cuerpo y el medio en donde se encuentra.

Para una función de dos variables $(x,\ y) \in \mathbb{R}^2$ y la variable temporal t, el enunciado clásico en que no existe una fuente de energía es:

$$\frac{\partial U}{\partial t} = \alpha \left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} \right) \tag{1}$$

Donde α es la difusividad térmica, que es un índice que expresa la velocidad de cambio y flujo de temperaturas en un material hasta que alcanza el equilibrio térmico.

Presentación del Problema: Diferencias Finitas

El método de las diferencias finitas es utilizado para calcular de manera numérica las soluciones a las ecuaciones diferenciales, usando **ecuaciones diferenciales finitas para aproximar derivadas.**

El método consiste en una aproximación de las derivadas parciales por expresiones algebraicas con los valores de la variable dependiente en un limitado número de puntos seleccionados.

Como resultado de la aproximación, la ecuación diferencial parcial que describe el problema es reemplazada por un número finito de ecuaciones algebraicas, en términos de los valores de la variable dependiente en puntos seleccionados.

Resolución de la Ecuación del Calor mediante Diferencias Finitas

A través del método de Diferencias Finitas se puede llevar la ecuación del calor (1) a una expresión computable:

$$\frac{\partial U}{\partial t} = \left(\frac{U(x, y, t) - U(x, y, t-1)}{\Delta t} \right) \tag{2}$$

Donde se puede despejar $U(x, y, t)$ como una ecuación de recursión:

$$U(x, y, t) = U(x, y, t-1) + \alpha \left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} \right) \Delta t \tag{3}$$

Resolución de la Ecuación del Calor mediante Diferencias Finitas

Para transformar el Laplaciano de la ecuación (1) a una función computable, se tienen que llevar todas las derivadas parciales a una expresión algebraica. De este modo, se puede representar la segunda derivada parcial de U respecto a x como:

$$\frac{\partial^2 U}{\partial x^2} = \frac{U(x-1,y) - 2U(x,y) + U(x+1,y)}{\Delta x^2} \tag{4}$$

Análogamente para y:

$$\frac{\partial^2 U}{\partial y^2} = \frac{U(x,y-1) - 2U(x,y) + U(x,y+1)}{\Delta y^2} \tag{5}$$

Resolución de la Ecuación del Calor mediante Diferencias Finitas

Así, de (3), (4) y (5), se obtiene la ecuación de recurrencia para un instante t como:

$$\mathrm{U(x,y,t)=U(x,y,t\text{-}1)}+\alpha\left(\frac{U(x-1,y,t-1)-2U(x,y,t-1)+U(x+1,y,t-1)}{\Delta x^2}+\frac{U(x,y-1,t-1)-2U(x,y,t-1)+U(x,y+1,t-1)}{\Delta y^2}\right)\Delta t \qquad (6)$$

Para el caso de esta implementación, se utilizó $\Delta x = 1$, $\Delta y = 1$ y $\Delta t = 1$, ya que se consideraron instantes y espacios consecutivos, y $\alpha = 0{,}1$.

Índice

Soluciones Implementadas: CPU

Para resolver el problema, inicialmente se realizó un programa en el lenguaje C que trabaja con la CPU:

- Primero, se realizaron archivos con casos de pruebas pequeños, y las funciones de leer y escribir archivos.
- Luego, se implementó la función que simularía las etapas por las que pasaría el objeto, llamada `CPU_Calor`. Esta calcula el valor de cada pixel de la iteración usando la fórmula de diferencias finitas para la ecuación del calor (6), y la información de la iteración anterior.
- La información de cada iteración se guarda en un gran arreglo conjunto de tamaño $size * T$ (donde T es la cantidad de iteraciones y $size$ el tamaño de la matriz). Esta representación se conservó también para ambas implementaciones con GPU.

Soluciones Implementadas: GPU

Similar a lo desarrollado en CPU, se desarrolló la función `GPU_Calor` que implementa la resolución de la ecuación del calor con diferencias finitas (6).

Esta vez, para cada casilla del arreglo se define una hebra que trabaja sobre el arreglo de la iteración anterior, de donde extrae la información y escribe el resultado actual correspondiente.

Soluciones Implementadas: GPU con memoria compartida

La memoria compartida tiene un tiempo de acceso a memoria muy inferior al de la memoria global de la GPU. Ya que cada dato de la iteración anterior se accede 5 veces (excepto los bordes), se propuso implementar memoria compartida para ver si hay una reducción significativa en los tiempos. Para esto:

- Se divide el arreglo de entrada en cuadrados de 16x16 para emular los bloques de hebras y aprovechar esta ventaja.
- Cada hebra copia los datos desde memoria global de la GPU a la memoria compartida de su bloque y luego implementa la resolución de la ecuación para el pixel.
- Para los casos borde de los bloques de 16x16, los datos necesarios que no están dentro de la memoria compartida del bloque se leen desde memoria global.

* Todos los archivos probados en esta implementación eran de dimensiones múltiplos de 16.

Índice

Desafíos y Complicaciones: Índices, The Big Boss

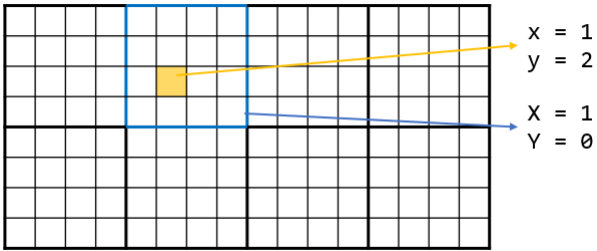
El principal desafío fueron los índices y las condiciones borde:

- En la implementación de CPU, el manejo de índices fue simple (`y = i/N` y `x = i%N`), y para manejar las condiciones borde se decidió calcular el valor de cada casilla por partes (cada parte manejaba la sección de la ecuación que necesitaba una casilla diferente).
- En la implementación de GPU se realizó algo similar (`y = tId/N` y `x = tId%N`).
- Sin embargo, en la implementación con memoria compartida surgió una nueva complicación, ya que se debía dividir la matriz completa de la iteración en cuadrados de 16x16. En este sentido, se aumentaban los casos bordes, y el manejo de índices se volvía mucho más complicado.

Desafíos y Complicaciones: Índices, The Big Boss

Para solucionar el problema de manejo de índices presentado en memoria compartida, se definieron 4 índices: `x`, `y`, `X` e `Y`.

- x e y definen la posición de la casilla que se quiere calcular dentro de la matriz de 16x16. Se calculan como: `x = threadIdx.x%16` e `y = threadIdx.x/16`.
- X e Y definen la posición de la matriz de 16x16 que contiene a la casilla que se quiere calcular, dentro de la matriz completa de la iteración. Se calculan como: `X = blockIdx.x%(N/16)` e `Y = blockIdx.x/(N/16)`.



Índice

Resultados

Tiempo ejecución [ms]	10 iteraciones	20 iteraciones	50 iteraciones
CPU	1061.0	2359.0	5761.0
GPU	7.173728	15.142016	39.061119
GPU Memoria compartida	8.892416	18.428928	45.148930

Tabla 1: Resultados de los tiempos obtenidos con un archivo inicial de 1600x3200

Resultados

Tiempo ejecución [ms]	10 iteraciones	50 iteraciones	1000 iteraciones
CPU	10.0	58.0	1241.0
GPU	0.087040	0.461152	10.182048
GPU Memoria compartida	0.094208	0.508928	10.712064

Tabla 2: Resultados de los tiempos obtenidos con un archivo inicial de 160x320

Resultados

Tiempo ejecución [ms]	80x160	160x320	800x1600	1600x3200
CPU	15.0	58.0	1450.0	5761.0
GPU	0.238112	0.461152	9.829024	39.061119
GPU Memoria compartida	0.249312	0.508928	11.855872	45.148930

Tabla 3: Resultados de los tiempos obtenidos con 50 iteraciones para distintos tamaños de la imagen inicial

Resultados

Tiempo ejecución [ms]	16x32	80x160	160x320	800x1600
CPU	4.0	122.0	350.0	9554.0
GPU	0.885760	1.520224	2.916800	60.888065
GPU Memoria compartida	0.818176	1.480704	3.074752	72.476318

Tabla 4: Resultados de los tiempos obtenidos con 300 iteraciones para distintos tamaños de la imagen inicial

Índice

Análisis de tiempos y rendimiento

Todos los resultados presentados indican que las implementaciones usando GPU son más rápidas que la implementación usando CPU. Esto se esperaba, ya que el uso de hebras permite trabajar el cálculo de todos los pixeles a la vez en cada iteración.

Sin embargo, los resultados al comparar GPU y GPU con memoria compartida sorprendieron: se esperaba que el uso de memoria compartida redujera en todos los experimentos los tiempos porque se agilizaría la lectura de los datos, pero no fue así.

Análisis de tiempos y rendimiento

- Para archivos muy pequeños (por ejemplo, 16x32 o 80x160 de las tablas 3 y 4), al volver a realizar los experimentos una y otra vez, a veces la GPU con memoria compartida se completaba en menor tiempo que GPU con memoria global, y a veces pasaba lo contrario. De hecho, la diferencia de tiempos es muy pequeña en estos casos.
- Además, cuando el archivo de entrada aumentaba en tamaño (por ejemplo, archivos de 160x320 en adelante en todas las tablas), independiente a la cantidad de iteraciones, la GPU con memoria global obtenía los menores tiempos.
- Esto se debe a que a mayor tamaño de archivo de entrada, más bloques son generados en memoria compartida, lo que implica pérdida de tiempo en escritura y en sincronización, que no se ve compensada con la rápida lectura.
- Por otro lado, la diferencia entre los tiempos de GPU es directamente proporcional a la cantidad de iteraciones. Esto se debe a que, si ya en cada iteración la memoria compartida retrasa, al aumentar las iteraciones el tiempo de retraso acumulado va aumentando también.

Índice

Conclusiones

Tras los resultados obtenidos, se puede concluir que utilizar GPU, en cualquiera de los escenarios probados, es mucho mejor que utilizar CPU. Además, utilizar memoria compartida para GPU en este problema no significó una ventaja, ya que sólo en archivos pequeños había opción de que mejorara el resultado. Sin embargo, el resultado dependía de la ejecución y la diferencia no era mucha.

Por lo tanto, los tiempos que se demoraba en leer la información de memoria global, escribir en memoria compartida, y escribir el resultado final en memoria global nuevamente, no se compensaban con los tiempos rápidos de lectura desde memoria compartida.

Gracias por su atención!

Presentación Proyecto

Resolución de la ecuación de calor mediante diferencias finitas

Ethiel Carmona
Joaquín Castillo
María Paz Morales
Axel Reyes

Taller de Modelos y Métodos Cuantitativos
Departamento de Informática
Universidad Técnica Federico Santa María

27 de marzo de 2021