FASE IV

Proyecto de programación con GPGPU

Arquitectura de los Computadores
Lunes 17:00 a 19:00
Director: Pablo Requena González
Auxiliar: Alberto Sapiña Mora
Secretario: Marcos González Verdú
Controlador: Jorge Nuñez González

ÍNDICE

INTRODUCCIÓN

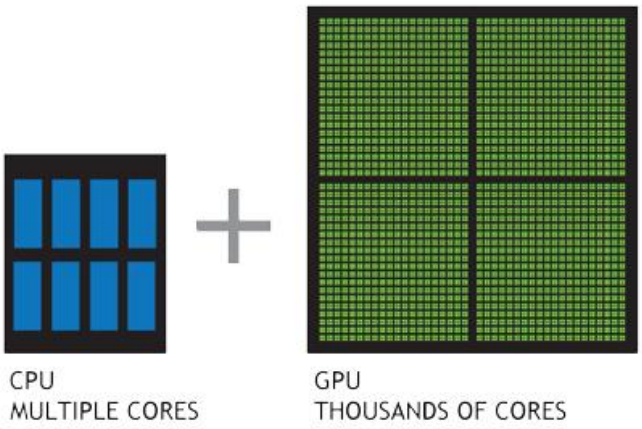
En primer lugar, debemos explicar dos conceptos: CUDA y GPGPU.

GPGPU significa General-Purpose Computing on Graphics Processing Units, es decir, si una GPU está diseñada para tratar, exclusivamente, el procesamiento gráfico, el concepto GPGPU es utilizar la GPU como dispositivo de propósito general para solucionar cualquier tipo de problema.

Para empezar, aunque cualquier algoritmo implementable en una CPU, también se puede implementar en una GPU, hay algunas diferencias en cuanto al rendimiento de este algoritmo.

El acceso a memoria es mucho más restringido en GPU que en CPU, por lo tanto, no se pueden utilizar estructuras de datos complejas de la misma manera que en la CPU.

La GPU se utiliza para optimizar el rendimiento de algoritmos que utilizan operaciones simples muchas veces, susceptibles de ser paralelizadas mediante hilos, gracias a su gran cantidad de núcleos.



Tradicionalmente se ha implementado en lenguaje ensamblador. Actualmente, BrookGPU y Sh también utilizan GPGPU, pero la opción más extendida es CUDA.

CUDA viene de Compute Unified Device Architecture, o sea, Arquitectura de dispositivos de cómputo unificada, y nos permite utilizar una variación de C para programar algoritmos en GPU de NVIDIA, ya que nos ofrecen un compilador y un conjunto de herramientas. CUDA funciona gracias a estos pasos:

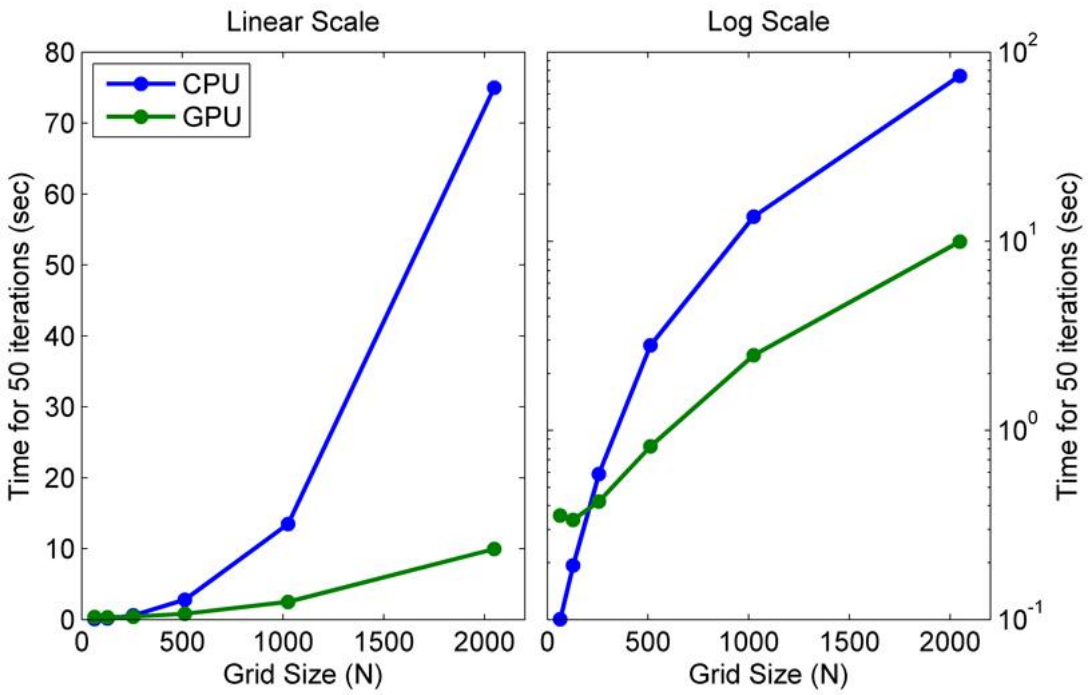
1. Copiar los datos de la memoria principal a la memoria de la GPU.
2. La CPU encarga el proceso a la GPU.
3. La GPU lo ejecuta en paralelo en cada núcleo utilizando hilos (threads).
4. Se copia el resultado del proceso, de la memoria de la GPU a la principal.

Estos son los elementos que constituyen su estructura interna:

1. Kernel: la función que se ejecutará en diferentes hilos (N).
2. Grid: la manera de estructurar los diferentes bloques en el kernel.
3. Bloque: agrupación de hilos. Cada bloque se ejecuta en un solo segmento de memoria, pero cada segmento puede tener varios bloques asignados.

Ahora que sabemos más sobre las características de CUDA, vamos a ver sus ventajas e inconvenientes.

- Ventajas
    1. Lecturas dispersas: nos permite consultar cualquier posición de memoria.
    2. Memoria compartida: CUDA dispone de un área de memoria que puede ser compartida entre hilos.
    3. Permite lecturas más rápidas, tanto de entrada como de salida de la GPU.
    4. Ofrece soporte para enteros y operadores bit a bit.
- Inconvenientes
    1. No se puede utilizar ni punteros a funciones, ni recursividad, ni variables estáticas dentro de funciones, ni funciones con número de parámetros variable.
    2. No soporta el renderizado.
    3. No soporta números NaN (Not a Number) en precisión simple.
    4. El ancho de banda de los buses y sus latencias, a veces generan un cuello de botella entre la CPU y la GPU.
    5. Obligatoriamente, los hilos deben lanzarse en grupos de 32, como mínimo.



Tiempos de ejecución usando CPU y GPU para resolver tandas de ecuaciones de segundo grado

DESARROLLO DE LA PRÁCTICA

La práctica ha consistido en la implementación de un algoritmo clásico de tratamiento de imágenes, en concreto un filtro de mediana:

Un filtro de mediana es un método utilizado para reducir el ruido de una imagen aplicando a cada píxel de la imagen una función para modificar su valor.

La forma de aplicar este algoritmo es la siguiente:

Para cada píxel de la imagen se cogerán todos sus vecinos y se ordenaran de tal forma que nos quedaremos con el valor medio en caso impar y en el caso de que sea par se realizara la media entre los dos valores centrales. Este proceso se realizará para todos los píxeles de la imagen, para así obtener una imagen mucho más suavizada. Ya que esta operación se va a repetir tantas veces como píxeles haya en la imagen es posible acelerar el proceso utilizando arquitecturas GPGPUs.

En el ejercicio se nos da un archivo que tenemos que modificar para poder ejecutar el código (mediana.cu). Modificando los valores de Grid_W, Grid_H, BLOCK_W y BLOCK_H.
Hemos probado con estas combinaciones:

```
#define GRID_W  1024
#define GRID_H  1024
#define BLOCK_W 1
#define BLOCK_H 1
```

```
#define GRID_W  256
#define GRID_H  256
#define BLOCK_W 4
#define BLOCK_H 4
```

```
#define GRID_W  128
#define GRID_H  128
#define BLOCK_W 8
#define BLOCK_H 8
```

```
#define GRID_W  64
#define GRID_H  64
#define BLOCK_W 16
#define BLOCK_H 16
```

```
#define GRID_W  32
#define GRID_H  32
#define BLOCK_W 32
#define BLOCK_H 32
```

Además de modificar estos parámetros también había que calcular la y columna:

```
col = threadIdx.x + blockIdx.x * blockDim.x + 1;
row = threadIdx.y + blockIdx.y * blockDim.y + 1;
```

Así como la función para ordenar los elementos para encontrar la mediana:

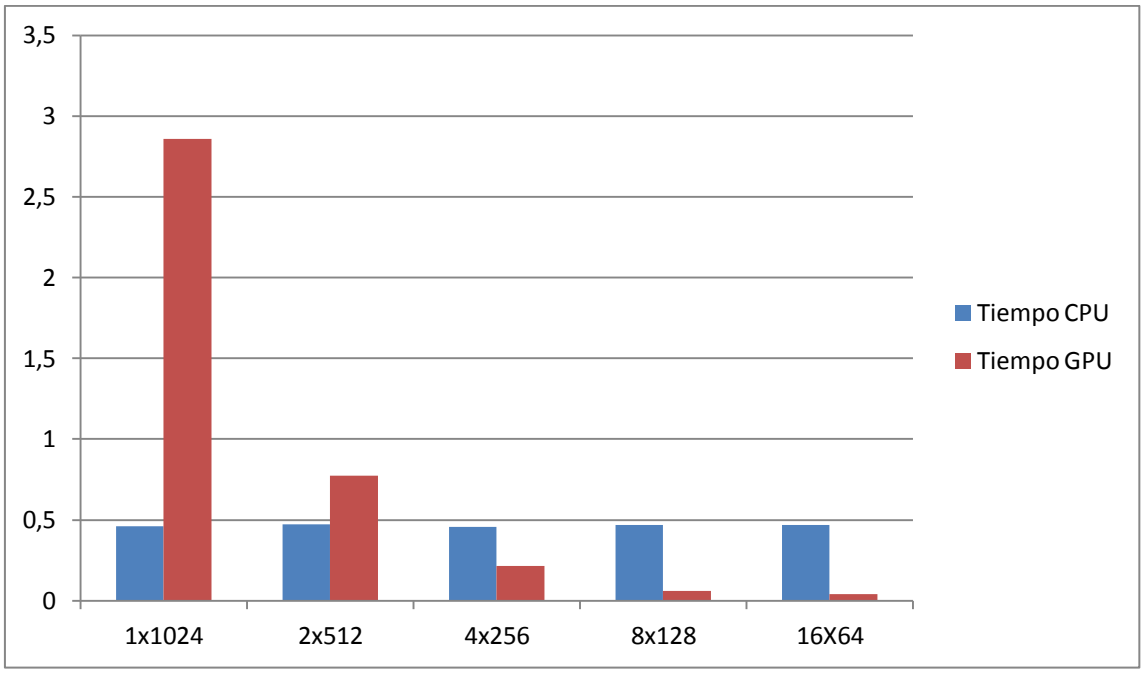
```
        for (unsigned int j=0; j<5; ++j)
        {
            int min=j;
            for (unsigned int i=j+1; i<9; ++i)
                if (neighborhood[i] < neighborhood[min])
                    min=i;
                temp=neighborhood[j];
                neighborhood[j]=neighborhood[min];
                neighborhood[min]=temp;
        }
        d_output[idx]=neighborhood[4];
    }
```

PRESENTACIÓN DE RESULTADOS

Hemos ejecutado en el mismo equipo la aplicación, exactamente en la computadora del laboratorio. Hemos probado con diferentes tamaños de bloques (1024, 512, 256, 128, 64).

Capturas de las ejecuciones

Gráfica con los datos extraídos de las ejecuciones:



En la gráfica anterior se puede observar los tiempos de la aplicación Filtro de Media en una imagen. La diferencia de procesamiento es notable mientras que la CPU siempre tiene un tiempo constante la GPU tiene tiempos mucho mejores cuanto menos sea el bloque y por lo tanto mayor número de hilos ejecuta a la vez. La GPU tiene un gran rendimiento a partir de los 4 hilos de ejecución, por lo tanto cuanto más se divida la carga mejor serán los tiempos.

A continuación se muestra la diferencia de la imagen original a la izquierda y la procesada a la derecha. Apreciando que el proceso se ha efectuado correctamente.