Reporte Tecnico - Tarea 1

Edgar Adrian Garcia Villegas - Septiembre 2018

Tarea n. 1, para el curso de programación multinucleo. Modificar los ejemplos de código previos para calcular la multiplicación de matrices. Donde tiene que ser programado de tres maneras:

- En CPU sin hilos
- En CPU con hilos
- En CUDA usando bloques e hilos

Cada programa debe de tener lo siguiente:

Multiplicar 2 matrices de N*N. N tiene que establecerse en 1000, 2000 y 4000. Donde las matrices se llenaran con números naturales;

1. Introducción

Antes que nada se dará una introducción a que s la computación paralela, ya que en lo personal aun no entiendo muy bien la materia con ayuda de un libro de texto planeo complementar lo aprendido en este Tópico es una forma de cómputo en la que muchas instrucciones se ejecutan simultáneamente, operando sobre el principio de que problemas grandes, a menudo se pueden dividir en unos más pequeños, que luego son resueltos simultáneamente (en paralelo). Hay varias formas diferentes de computación paralela: paralelismo a nivel de bit, paralelismo a nivel de instrucción, paralelismo de datos y paralelismo de tarea.

Sabiendo esto tomaremos algunos conceptos clave para el desarrollo de esta tarea entre otros faltantes:

- PARALELISM
- MEMORY

- CACHE
- CUDA
- MEMORY
- UTILIZATION
- ECC
- TEMPERATURE
- POWER
- CLOCK
- COMPUTE
- PIDS
- PERFORMANCE
- SUPPORTED_CLOCKS
- PAGE_RETIREMENT
- ACCOUNTING

2. Desarrollo

Para el desarrollo de estos programas, se va hace una comparación de tiempos donde compararemos 5 programas iterándolos o probándolos entre 50 y 100 veces. Donde uno se correaran en CUDA. Para ello es necesario la siguientes formulas donde compararemos los tiempos

$$Speedup = \frac{Tiempo_{secuencial}}{Tiempo_{paralelo}}$$

Donde T iempo$_{secuencial}$ es el tiempo que le toma ejecutarse a la parte secuencial, y T iempo$_{paralelo}$ es el tiempo que toma ejecutarse en paralelo.

Ley de Amdahl:

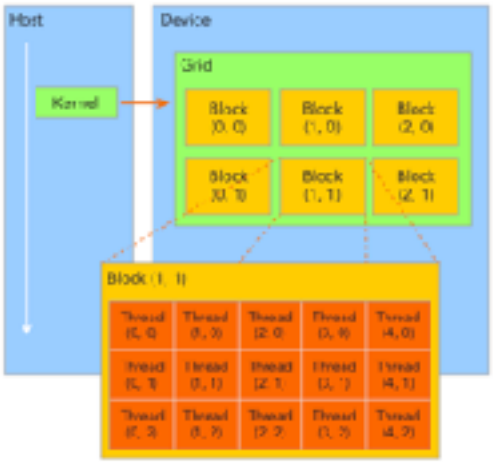
$$Speedup = \frac{1}{S + (1 - S)/n}$$

Donde S es el tiempo que se toma en la ejecución serial, y n es el número de procesadores.

Una ves que sabemos estas formulas, compararemos los tiempos a continuación:

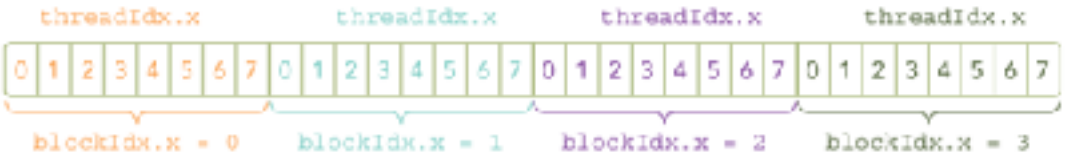
	1000	2000	4000
CPU	5740.45	72048.61	******
CPU+THREAD	6669.4577	72608.480	******
GPU 1D	496.102473	1856.869921	******
GPU 2D_1D	61.45901357	471.7251109	KILLED
GPU 2D_2D	59.69173031	409.73471295	KILLED

3. Ejemplo de referencias, figuras y conclusión



Para este trabajo se uso como guía la clase y un libro llamado Professional CUDA C programming, donde nos habla sobre la programación en CUDA así como alguno ejemplos vistos en clase en este caso a mi se me dificulto la compresión del a division de bloque. Por consiguiente tomo como referencia la siguiente foto de la parte derecha, donde nos muestra un pequeño diagrama donde CUDA expone una abstracción de jerarquía de subprocesos para permitirle organizar sus subprocesos. Esta es una jerarquía de subprocesos de dos niveles descompuesta en bloques de subprocesos y cuadrículas de bloques. Por consiguiente cada bloque se divide en pequeños bloques. Sin embargo me causa un poco de intriga y que aun sigo con la duda la division de tantos bloque. Donde puedes correr N cantidad de proceso que van estar separados de manera igual por así decirlo.

Una parte que también toma mucha importancia en este tipo de programas es el kernel, donde la clave es el mareo de hilos, desde su indice de memoria global, como se muestra acontinuacíon



Tomo como referencia la imagen anterior, ya que hago una relación con las clases dadas al momento de mostrar el ejemplo de la matriz, ya que un principio teníamos una suma de matrices de N*N, donde a cada bloque se le hacia una partición por bloque. En ciertos aspectos que claro como funciona dicha partición para que el kernel tenga una mejor distribución en los hilos.

Sin embargo tome mucho esa parte del libro por que me parecieron muy relacionadas a la clase. Personalmente aun no entiendo del todo muy bien y como funciona el orden de la programación multiproceso. Pienso que con los ejemplo que veremos en el curso tendré las bases comprender un poco más y de que dependen ciertos aspectos

Referencias y links de ayuda

- http://cocinando-tarjetas-graficas.blogspot.com/2013/09/multiplicar-matrices-en-cuda.html
- http://www.hds.bme.hu/~fhegedus/C++/Professional%20CUDA%20C%20Programming.pdf