Reporte técnico

Adrián Biller Alcántara
A01018940
Septiembre 4

Resumen

El siguiente documento es una descripción del proyecto realizado como tarea de multiplicación de matrices utilizando varios métodos de programación paralela,

1. Introducción

Dentro de los ejemplos vistos en cuanto a programación paralela, existen múltiples métodos, todos con diferentes características. El principal método en el que cualquier persona programa es desde la aproximación lineal. Sin embargo, como visto en clase, es necesario optimizar ciertos procesos utilizando nuevas herramientas como lo es el cómputo paralelo. En este proyecto se utilizaron dos tipos diferentes de computación paralela: threads con openMP y threads con el uso de una tarjeta de video y cuda toolkit.

Basándose en ambos métodos se hará una comparación del performance entre ellos para concluir que algoritmo es más eficiente para esta tarea e incluso los detalles de cuál es la mejor configuración.

2. Desarrollo

En esta práctica se desarrollaron códigos utilizando 3 diferentes métodos para realizar una multiplicación de matrices. El primero fue con programación lineal utilizando el cpu. Esta es la más básica y la que más tiempo toma al ejecutarla, realiza cada proceso uno después del otro, debido a esto, el tiempo de ejecución es mayor a otros métodos.

Después se multiplicaron las mismas matrices utilizando CPU con OMP el cual permite crear y usar threads. De esta manera se puede observar que el tiempo de ejecución disminuyó considerablemente ya que al realizar varios threads un trabajo por separado el proceso lineal se hizo más corto.

Finalmente se utilizó el servidor proporcionado para el uso del GPU con Cuda Toolkit. En este caso se utilizó el mismo algoritmo de multiplicación de matrices tomando en cuenta el index de los threads del GPU. En este caso se puede notar una mejora considerable en cuanto a tiempo de ejecución al tener mas threads disponibles permite disminuir aún más todo el proceso general de multiplicación. En esta herramienta también se puede variar el modo en el que los threads están configurados en el kernel.

3. Ejemplo de referencias y figuras

En la siguiente tabla se pueden observar los tiempos de ejecución de la multiplicación de matrices dependiendo de la herramienta y en donde se están ejecutando los procesos.

Tiempo de ejecución en segundos

	1000	**2000**	**4000**
CPU Intel core i5 Coffee Lake	**5.625**	**72.9219**	**617.016**
CPU Threads Intel core i5 Coffee Lake	**4.6093**	**62.75**	**528.23**
GPU 1D Servidor con GeForce GTX 670	**1.95196**	**12.5594218**	**-**
GPU 1D 2D Servidor con GeForce GTX 670	**1.27846**	**13.79386**	**-**
GPU 2D Servidor con GeForce GTX 670	**0.52928**	**3.75152**	**-**

*** Los resultados de GPU en la multiplicación de matrices de 4000x4000 no se pudo realizar debido a errores de tiempo de ejecución.**

Analizando la tabla se puede notar que la diferencia de tiempos de procesamiento de CPU y procesamiento de CPU con threads es muy pequeña pero aun así tomando en cuenta que sea una cantidad grande de datos puede llegar a ser relevante disminuir el tiempo. Posteriormente se puede ver que al utilizar el GPU disminuye considerablemente el tiempo en los tres diferentes tamaños de multiplicación. Se puede observar también que entre el uso de GPU en 1D y 1D 2D

la diferencia es muy pequeña, sin embargo a 2D si llega a disminuir en los tres casos.

El speedup según la fórmula $speedup = \frac{Time\ Sequential}{Time\ Parallel}$

y utilizando los números obtenidos salen los siguientes resultados:

	1000	2000	4000
GPU 1D Servidor con GeForce GTX 670	**s = 2.88**	**s = 5.8061**	**-**
GPU 1D 2D Servidor con GeForce GTX 670	**s = 4.3998**	**s = 5. 286547**	**-**
GPU 2D Servidor con GeForce GTX 670	**s = 10.62**	**s = 19.437961**	**-**

D

4. Referencias

Cheng, John, et al. *Professional CUDA C Programming*. John Wiley & Sons, 2014.