Diego Barreiro Pérez 49403711-X

Práctica 4

Programación de GPUs

Obtención de Información sobre las GPUs

NVIDIA V100

1. Computer Capability (major.minor): 7.0
2. Número de multiprocesadores (SMs) en el dispositivo: 80
3. Máximo número de CUDA threads residentes por SM: 2048
4. Tamaño máximo de cada dimensión de un Grid: (2147483647, 65535, 65535)
5. Máximo número de hilos por bloque: 1024
6. Tamaño máximo de cada dimensión de un bloque: (1024, 1024, 64)
7. Número de registros de 32 bits disponibles por SM: 65536
8. Número de registros de 32 bits disponibles por bloque: 65536
9. Tamaño de la memoria compartida disponible por procesador (en KiB): 96
10. Tamaño de la memoria compartida disponible por Bloque (en KiB): 48
11. Tamaño de la memoria global disponible en el dispositivo (en GiB): 15.781738
12. Frecuencia pico de la memoria (en MHz): 877
13. Ancho del bus de memoria global (en bits): 4096
14. Ancho de banda pico de memoria (en GiB/s): 0.898048
15. Número de CUDA cores: 5120

NVIDIA K2

1. Computer Capability (major.minor): 3.0
2. Número de multiprocesadores (SMs) en el dispositivo: 8
3. Máximo número de CUDA threads residentes por SM: 2048
4. Tamaño máximo de cada dimensión de un Grid: (2147483647, 65535, 65535)
5. Máximo número de hilos por bloque: 1024
6. Tamaño máximo de cada dimensión de un bloque: (1024, 1024, 64)
7. Número de registros de 32 bits disponibles por SM: 65536
8. Número de registros de 32 bits disponibles por bloque: 65536
9. Tamaño de la memoria compartida disponible por procesador (en KiB): 48
10. Tamaño de la memoria compartida disponible por Bloque (en KiB): 48
11. Tamaño de la memoria global disponible en el dispositivo (en GiB): 3.941895
12. Frecuencia pico de la memoria (en MHz): 2500
13. Ancho del bus de memoria global (en bits): 256
14. Ancho de banda pico de memoria (en GiB/s): 0.160000
15. Número de CUDA cores: 1536

NVIDIA K80

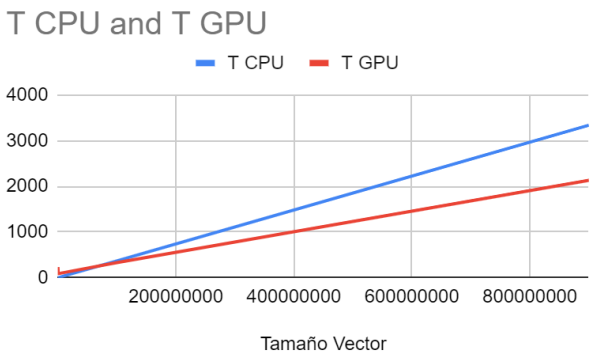
1. Computer Capability (major.minor): 3.7
2. Número de multiprocesadores (SMs) en el dispositivo: 13
3. Máximo número de CUDA threads residentes por SM: 2048
4. Tamaño máximo de cada dimensión de un Grid: (2147483647, 65535, 65535)
5. Máximo número de hilos por bloque: 1024
6. Tamaño máximo de cada dimensión de un bloque: (1024, 1024, 64)
7. Número de registros de 32 bits disponibles por SM: 131072
8. Número de registros de 32 bits disponibles por bloque: 65536

9. Tamaño de la memoria compartida disponible por procesador (en KiB): 112
10. Tamaño de la memoria compartida disponible por Bloque (en KiB): 48
11. Tamaño de la memoria global disponible en el dispositivo (en GiB): 11.173035
12. Frecuencia pico de la memoria (en MHz): 2505
13. Ancho del bus de memoria global (en bits): 384
14. Ancho de banda pico de memoria (en GiB/s): 0.240480
15. Número de CUDA cores: 2496

Medida de rendimiento de un código CUDA simple

Distinto tamaño de vector

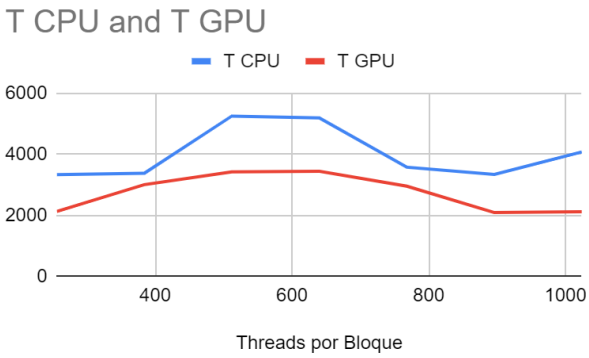
Tam. Vector	T CPU	T GPU
1000	0.003094	99.218847
10000	0.038932	122.104381
100000	0.536465	234.655114
1000000	4.547046	93.19904
10000000	41.208498	114.897046
100000000	370.40762	334.001779
900000000	3335.491958	2133.38073



Distintos "threads" por bloque

- **Tamaño del Vector:** 900.000.000

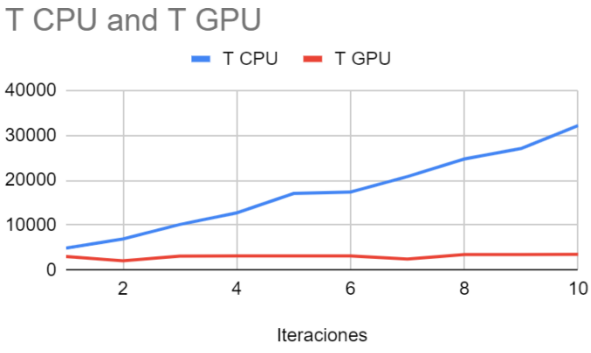
Th. por Bloque	T CPU	T GPU
256	3335.491958	2133.38073
384	3380.957039	3007.306363
512	5249.430578	3423.455602
640	5189.521492	3447.357216
768	3577.311665	2958.288918
896	3342.619337	2094.445496
1024	4075.539131	2122.262682



Distinto número de iteraciones

- **Tamaño del Vector:** 900.000.000
- **Threads por Bloque:** 1024

Iteraciones	T CPU	T GPU
1	4927.413225	3091.010436
2	6977.852692	2128.684769
3	10196.79349	3167.075092
4	12769.82827	3200.684201
5	17068.11148	3218.011827
6	17388.09984	3210.510526
7	20830.16136	2537.543988
8	24734.82931	3488.87478
9	27071.54594	3524.922987
10	32155.62204	3558.492754



Medida de las etapas

- **Tamaño del Vector:** 900.000.000
- **Threads por Bloque:** 1024
- **Número de Iteraciones:** 10

Se ha ejecutado el código 5 veces, y los resultados han sido los siguientes (se muestra el valor en la mediana):

- **Tiempo en CPU:** `29568.36256`
- **Tiempo total en GPU:** `4014.22322`
  - Reserva de memoria: `187.295007`
  - Copia de Host a Dispositivo: `1162.637537`
  - Ejecución en kernel: `652.940907`
  - Copia del Dispositivo al Host: `2011.349766`

Como bien se aprecia, las etapas más costosas son las de copia entre dispositivo y host, es decir, la transferencia de datos entre ambos.

Si solo se comparase el tiempo de ejecución en el kernel de la GPU con el tiempo de CPU, el tiempo de ejecución es x45 veces menor en la GPU que en la CPU.

Memoria unificada

Se ha modificado el programa para que trabaje sólo con memoria unificada, de tal forma que se trabaja sólo con las variables `h_X`, siendo estas variables reservadas con `cudaMallocManaged`. Sin embargo, al realizar toda la reserva de 4 vectores (en vez de 3), la tarjeta se queda sin memoria, por lo que, en vez de probar con un vector de 900.000.000 elementos, se procede a probar con un vector de 700.000.000 elementos.

La principal diferencia en este programa con los anteriores está en la reserva de memoria. En los programas anteriores era necesario reservar memoria en la RAM y en la GPU por separado, ahora se reserva sólo una vez para ambas. En cuanto a tiempos de ejecución, si se toma sólo el tiempo de ejecución, utilizando la memoria normal se requiere de 510,965259 ms de ejecución, mientras que con memoria unificada es de 1430,217009 ms.

Este incremento se puede deber a que la memoria unificada no es tan rápida como la memoria “normal” de la GPU. Es decir, esta memoria es accesible tanto por la CPU como por la GPU, por lo que no es tan rápida como la otra, obligando a la GPU a necesitar más tiempo de ejecución para acceder a esta memoria unificada que está “más lejos”.