Mauricio Rico Zavala / A01370874

03 - 09 - 18

Programación Multinúcelo

Documentación:
Análisis y Resultados de Multiplicación de Matrices

Los siguientes resultados se basan en los cálculos de multiplicación de matrices tomando en cuenta el cálculo en CPU sin threads, CPU con múltiples threads y finalmente en GPU con distintas configuraciones de bloques y threads. Debido al tamaño de las matrices los programas no imprimen las matrices solamente calculan el resultado y miden el tiempo que toma cada opción. Por lo que el resultado con matrices más pequeñas ya fue revisado en cualquiera de los tres métodos. Lo siguiente es el tiempo en ms de las operaciones.

Especificaciones de la computadora:

Para las pruebas en CPU:

Macbook Pro 13’ early 2011:

- 2.3GHz dual-core Intel Core i5 processor con 3MB shared L3 cache
- Intel HD Graphics 3000 with 384MB of DDR3 SDRAM shared with main memory.

Para las pruebas en GPU:

El GPU utilizado fue el del servidor proporcionado que cuenta con un GeForce GTX 670:

GPU Engine Specs:	
CUDA Cores	1344
Graphics Clock (MHz)	915
Processor Clock (MHz)	980
Texture Fill Rate (billion/sec)	102.5
Memory Specs:	
Memory Clock	6.0 Gbps
Memory Interface	GDDR5
Memory Interface Width	256-bit GDDR5
Memory Bandwidth (GB/sec)	192.2

Los siguientes son los resultados de todas las pruebas con tiempos en ms.

Cálculo en CPU sin Threads:

Tamaño de matriz	1000 x 1000	2000 x 2000	4000 x 4000
Tiempos en ms.	9678.18	78665.6	732513
Tiempos en ms.	9587.87	83790.3	733265
Tiempos en ms.	9841.43	79062.3	731534
Tiempos en ms.	9887.82	79369.9	733543.2
Tiempos en ms.	9840.46	79055	731453.6
Tiempos en ms.	9666.17	79313.6	733153

Tamaño de matriz	1000 x 1000	2000 x 2000	4000 x 4000
Tiempos en ms.	9736.3	79362.5	732666
Tiempos en ms.	9724.88	79514.3	731645.8
Tiempos en ms.	9654.75	78479.4	730423
Tiempos en ms.	9785.33	79484	730131.4
Promedio en ms.	**9740.319**	**79609.69**	**732032.8**

Cálculo en CPU con Threads:

Utilizando 2 threads :

Tamaño de matriz	1000 x 1000	2000 x 2000	4000 x 4000
Tiempos en ms.	4901.69	42901.9	442278
Tiempos en ms.	5145.83	42994.1	442554.6
Tiempos en ms.	4935.94	42479.9	441535
Tiempos en ms.	4898.19	43902.8	44322.4
Tiempos en ms.	5119.45	42807.8	443433
Tiempos en ms.	5698.27	43543.8	442434.2
Tiempos en ms.	5145.14	43636.1	442278.5
Tiempos en ms.	4848.28	44562.9	443433.4
Tiempos en ms.	4978.7	44632.9	441533.1
Tiempos en ms.	5030.83	42475.5	445432
Promedio en ms.	**5070.232**	**43393.77**	**402923.42**

Utilizando 4 threads :

Tamaño de matriz	1000 x 1000	2000 x 2000	4000 x 4000
Tiempos en ms.	4403.27	39288.4	417938
Tiempos en ms.	4426.51	38746.8	416234
Tiempos en ms.	4447.4	39308.5	417434.5
Tiempos en ms.	4392.14	40104.7	416232.7
Tiempos en ms.	4471.75	40598.1	416212.4
Tiempos en ms.	4484.54	41037.7	415231.9
Tiempos en ms.	4404.2	41502.3	416224.1
Tiempos en ms.	4419.3	42994.2	416231
Tiempos en ms.	4413.8	42142.3	416886.6
Tiempos en ms.	4403.64	41197.7	415235.3
Promedio en ms.	**4426.655**	**40692.07**	**416386.05**

Utilizando 6 threads:

Tamaño de matriz	1000 x 1000	2000 x 2000	4000 x 4000
Tiempos en ms.	4295.44	38400.5	422618
Tiempos en ms.	4375.76	38820.6	423531.4
Tiempos en ms.	4336.74	41727.8	422432.5
Tiempos en ms.	4336.05	39248.1	425688.7
Tiempos en ms.	4343.96	40622.3	417894.6
Tiempos en ms.	4324.39	39598.8	421434
Tiempos en ms.	4359.6	39188.8	425321.1
Tiempos en ms.	4324.91	39801.4	424355.9
Tiempos en ms.	4305.15	39985.5	424645.7

Tamaño de matriz	1000 x 1000	2000 x 2000	4000 x 4000
Tiempos en ms.	4365.23	40851.9	422143.5
Promedio en ms.	**4336.723**	**39824.57**	**423006.54**

Cálculo en GPU:

Utilizando 256 threads (16x16):

Tamaño de matriz	1000 x 1000	2000 x 2000	4000 x 4000
Tiempos en ms.	176.281143	1238.489746	-
Tiempos en ms.	177.874207	1192.068115	-
Tiempos en ms.	178.559296	1271.952148	-
Tiempos en ms.	178.177994	1192.567017	-
Tiempos en ms.	176.533875	1190.654663	-
Tiempos en ms.	177.783997	1192.725586	-
Tiempos en ms.	178.502045	1195.877686	-
Tiempos en ms.	177.741486	1192.430420	-
Tiempos en ms.	176.136932	1270.573730	-
Tiempos en ms.	176.707397	1196.406128	-
Promedio en ms.	**177.4298372**	**1213.3745239**	**-**

Utilizando 512 threads (16x32):

Tamaño de matriz	1000 x 1000	2000 x 2000	4000 x 4000
Tiempos en ms.	177.377426	1248.171997	-
Tiempos en ms.	177.135635	1188.035034	-
Tiempos en ms.	177.389938	1186.120605	-

Tamaño de matriz	1000 x 1000	2000 x 2000	4000 x 4000
Tiempos en ms.	177.885986	1186.174561	-
Tiempos en ms.	176.786179	1187.744751	-
Tiempos en ms.	176.008148	1185.936523	-
Tiempos en ms.	179.738663	1248.492554	-
Tiempos en ms.	176.150635	1186.011475	-
Tiempos en ms.	176.754227	1187.084595	-
Tiempos en ms.	178.572556	1189.037109	-
Promedio en ms.	**177.3799393**	**1199.2809204**	**-**

Utilizando 512 threads (2x256):

Tamaño de matriz	1000 x 1000	2000 x 2000	4000 x 4000
Tiempos en ms.	43.945793	309.471283	2157.725342
Tiempos en ms.	44.234467	278.188751	2097.918945
Tiempos en ms.	41.375813	263.389954	2097.399902
Tiempos en ms.	42.965321	277.138916	2153.763428
Tiempos en ms.	41.799175	262.064667	2115.175293
Tiempos en ms.	41.405323	260.797058	2117.463135
Tiempos en ms.	43.379341	262.751038	2117.272461
Tiempos en ms.	42.137852	262.011017	2117.353516
Tiempos en ms.	42.196072	261.916595	2162.425293
Tiempos en ms.	43.361408	262.084930	2117.532959
Promedio en ms.	**42.6800565**	**269.9814209**	**2125.4030274**

Utilizando 1024 threads (32x32):

Tamaño de matriz	1000 x 1000	2000 x 2000	4000 x 4000
Tiempos en ms.	332.275513	2311.748779	-
Tiempos en ms.	294.461853	2304.842773	-
Tiempos en ms.	294.432770	2237.037598	-
Tiempos en ms.	280.292908	2236.554932	-
Tiempos en ms.	282.906464	2237.394287	-
Tiempos en ms.	280.283447	2238.400879	-
Tiempos en ms.	333.878479	2318.905029	-
Tiempos en ms.	322.774780	2234.403076	-
Tiempos en ms.	292.210449	2235.263672	-
Tiempos en ms.	280.294037	2238.442871	-
Promedio en ms.	**299.38107**	**2259.2993896**	**-**

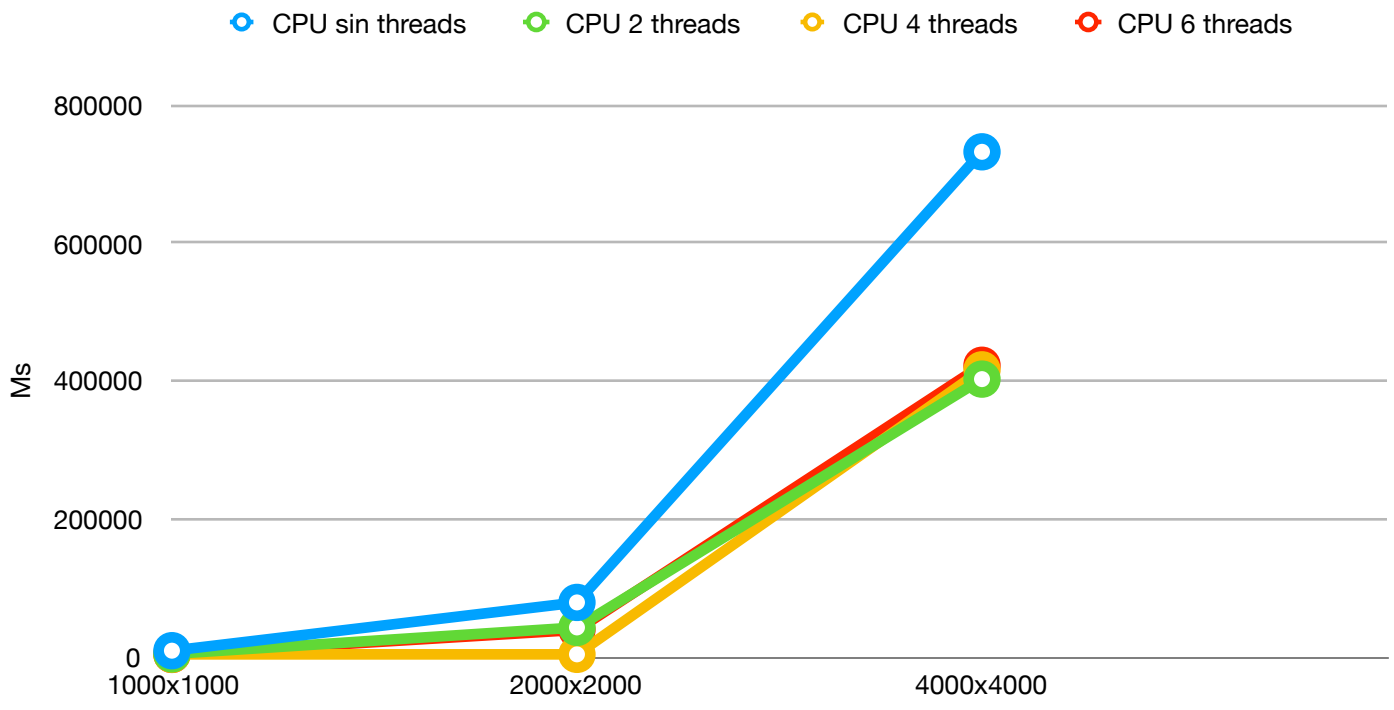
Utilizando 128 threads (1x128):

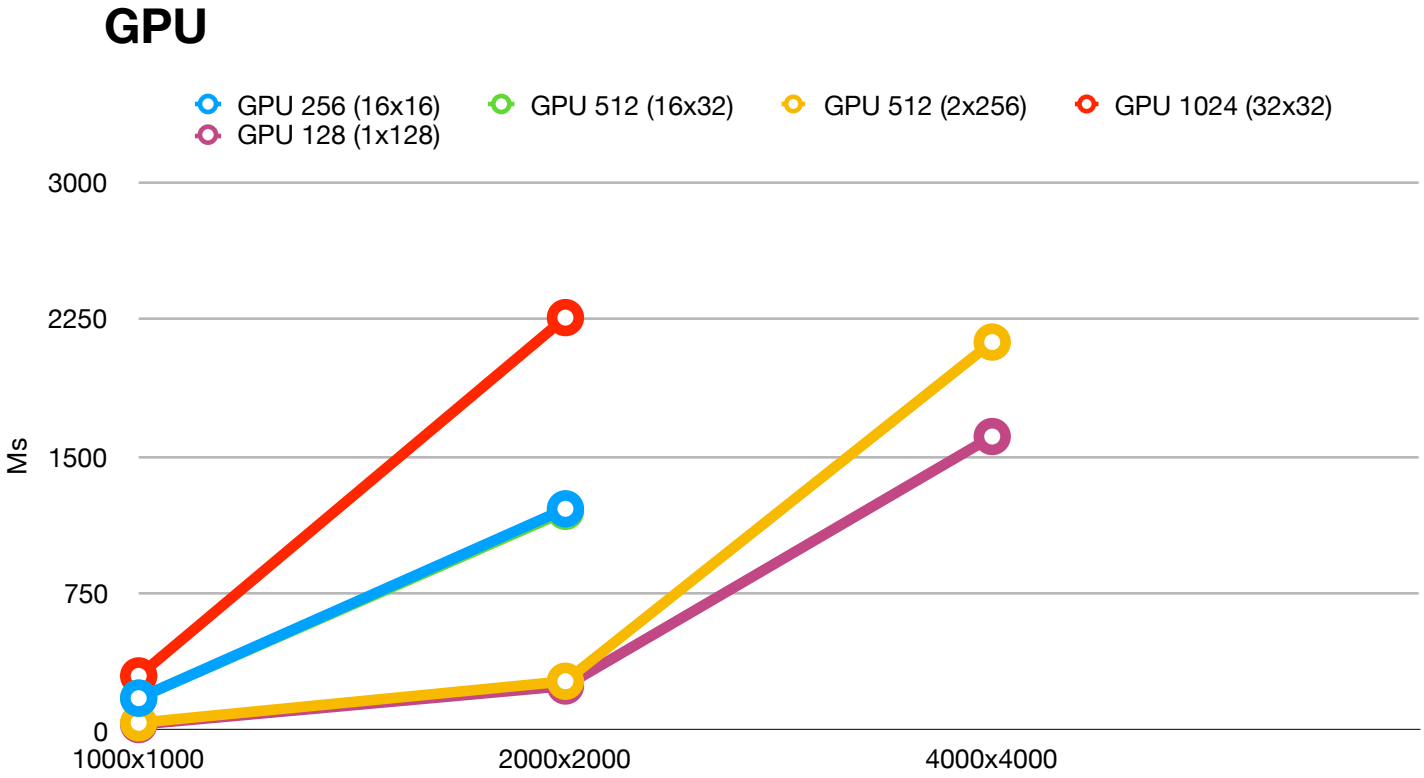
Tamaño de matriz	1000 x 1000	2000 x 2000	4000 x 4000
Tiempos en ms.	32.836853	252.448273	1619.351685
Tiempos en ms.	33.270420	251.220947	1601.134766
Tiempos en ms.	32.877312	244.619278	1602.444336
Tiempos en ms.	32.852352	245.739868	1624.431763
Tiempos en ms.	32.891914	251.101181	1601.748291
Tiempos en ms.	33.725525	244.851913	1600.255615
Tiempos en ms.	32.865929	250.623337	1600.179199
Tiempos en ms.	32.860695	229.936493	1598.073608
Tiempos en ms.	33.146763	219.332642	1601.090820

Tamaño de matriz	1000 x 1000	2000 x 2000	4000 x 4000
Tiempos en ms.	32.878929	250.616287	1638.629028
Promedio en ms.	**33.0206692**	**244.0490219**	**1608.7339111**

Speedup:

Tamaño de matriz	1000 x 1000	2000 x 2000	4000 x 4000
CPU sin threads	9740.319	79609.69	732032.8
CPU 2 threads	5070.232	43393.77	402923.42
CPU 4 threads	4426.655	40692.07	416386.05
CPU 6 threads	4336.723	39824.57	423006.24
GPU 256 (16x16)	177.4298372	1213.3745239	-
GPU 512 (16x32)	177.3799396	1199.2809204	-
GPU 512 (2x256)	42.6800565	269.9814209	2125.4030274
GPU 1024 (32x32)	299.38107	2259.2993896	-
GPU 128 (1x128)	33.02066992	244.0490219	1608.7339111





Análisis y Conclusiones:

Al observar los promedios de cada una de las pruebas se puede obtener una idea clara de lo que sucedió al programar paralelismo (primeramente en CPU con threads), ya que en los tres casos de tamaños de matrices se demuestra un claro beneficio en cuanto a tiempo de operación. En el ejemplo de los threads, solamente tomando el aumento a 2 threads (con 4 y 6 baja más aunque la diferencia entre estos dos últimos no sea tan drástica) de ningún thread varia entre el 52% y 55% del tiempo original; por lo que el cambio si es bastante notable.

Dentro de los threads en la programación de Cuda hay resultados todavía más interesantes. Debido a la lógica de como se van arreglando los indices del array de una dimensión, el indice o medida de x no puede ser demasiado grande o el programa

puede llegar a romperse. Es por esto que se intentaron los últimos experimentos con 128 y 512 threads teniendo como medida en x 1 y 2 respectivamente. Esto proporcionó los tiempos más rápidos en las tres medidas de las matrices. Y si se notó un patrón de cuando el número en la medida x era menor se tenía un mejor tiempo. Esto también puede atribuirse a que el programa llega a su limite de “rapidez” en los 128 threads que fue el caso más rápido. Puede ser que mas threads que estos serían innecesarios y solamente mantener tantos threads causa un aumento en el tiempo de operación total.

Es importante notar que el GPU para esta prueba es de un nivel alto por lo que aguanta el paralelismo pero una hipótesis con el CPU sería que utilizar un procesador que sea quad o six core en lugar de dual core mejoraría los resultados de estas pruebas bastante; ya que el procesador en mi computadora es lento a comparación de modelos más nuevos.

Referencias:

Especificaciones del CPU y GPU (respectivamente):

https://support.apple.com/kb/sp619?locale=en_US

https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-670/specifications