MULTIPLICACIÓN DE MATRICES USANDO TRES DIFERENTES MÉTODOS DE CODIFICACIÓN:

SECUENCIAL, PARALELO Y PARALELO CON TILING

HIGH PERFORMANCE COMPUTING (HPC)

PRIMER PARCIAL

ESTUDIANTES:
Diego Alejandro Ramirez Ramirez
Carlos Eduardo Zuleta Uribe

PROFESOR:
John J. Osorio

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

30/09/2015

INTRODUCCIÓN

Los últimos años durante la evolución de la tecnología se ha buscado aumentar la velocidad computacional de las máquinas con el fin de alcanzar el menor tiempo de ejecución en aplicaciones e incluso en los calculos cientificos que tienen mucha carga de procesamiento. Todo eso consiste en utilizar un conjunto de técnicas para proporcionar tareas simultáneas de procesamiento de datos, aprovechando los miles de núcleos de procesamiento que poseen las GPU (Graphics processing unit) que permiten distribuir las cargas de trabajo de forma paralela y muy eficiente.

El objetivo de este documento es hacer una comparación de tiempos y de aceleración de la computación de algoritmos implementados en el lenguaje de programación C y Cuda C creado por la compañía NVIDIA, utilizando diferentes técnicas de optimización, mediante el uso de la GPU y la CPU.

ESPECIFICACIONES DEL SISTEMA

La máquina en la cual se corrió el algoritmo tiene las siguientes características:

CPU:

processor : 1
vendor_id : GenuineIntel
cpu family : 6
model : 58
model name : Intel® Core™ i7-3770K CPU @ 3.50GHz
stepping : 9
cpu MHz : 1600.000
cache size : 8192 KB
cpu cores : 4

GPU:

Tesla K40c: 3.5
Global memory: 11519mb
Shared memory: 48kb
Constant memory: 64kb
Block registers: 65536
Warp size: 32
Threads per block: 1024
Max block dimensions: [1024, 1024, 64]
Max grid dimensions: [2147483647, 65535, 65535]

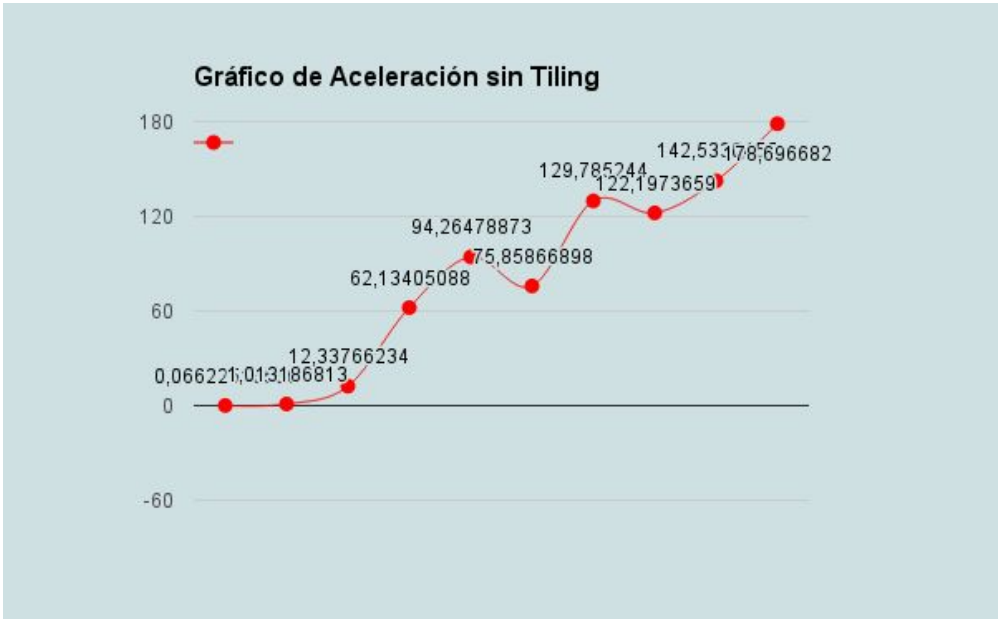
PROCESO APLICADO

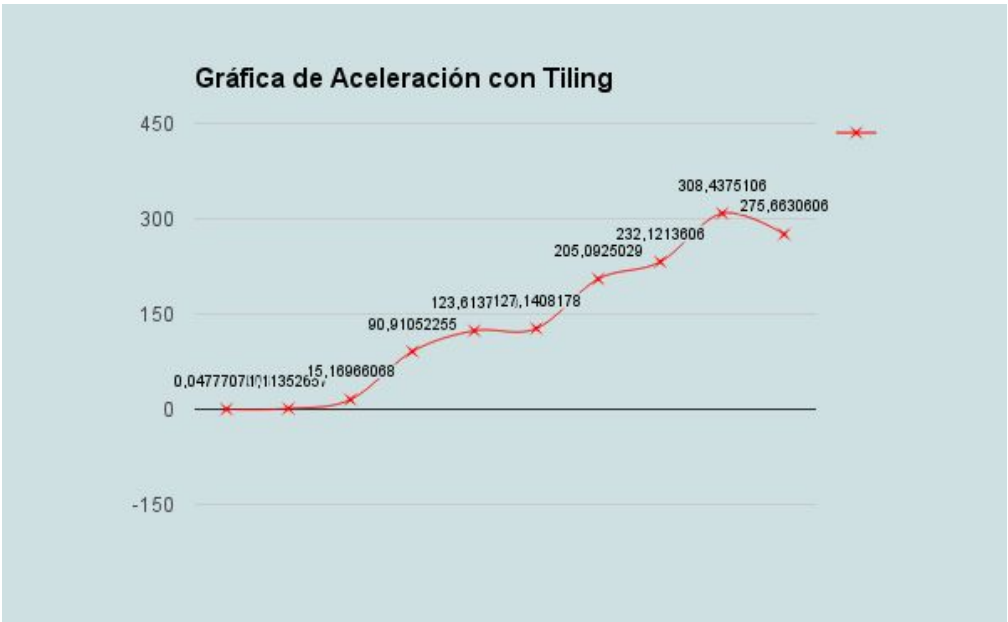
Se tomaron 10 datos para cada uno de los tiempos con los tres diferentes métodos: secuencial, paralelo y paralelo con tiling teniendo en cuenta que se debía ir incrementando el tamaño de las filas y columnas para cada matriz, luego se calculó una media más precisa. De la misma manera se halló la aceleración y la aceleración con tiling, tomando el valor medio de cada tiempo Secuencial y dividiéndolo entre cada valor de tiempo medio del Paralelo respectivamente, del mismo modo para la Aceleración con Tiling en este caso, tiempo Secuencial dividido entre tiempo Paralelo con tiling.

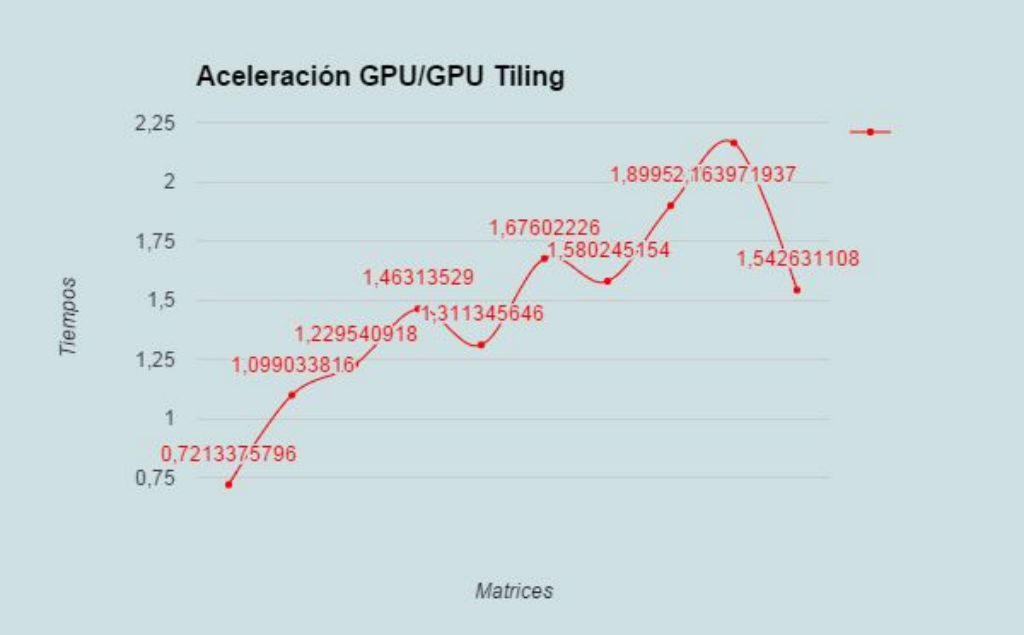
En la siguiente tabla se muestra con más detalle lo mencionado anteriormente.

Dim matrices			Time			Aceleraciones		
N	M	O	Secuencial	Parallel	Parallel with Tiling	CPU vs GPU	CPU vs GPU Tiling	GPU Tiling vs CPU
4	8	10	0,000003	0,0000453	0,0000628	0,06622516556	0,047770700 64	0,7213375796
32	16	32	0,0000461	0,0000455	0,0000414	1,013186813	1,11352657	1,099033816
64	32	128	0,00076	0,0000616	0,0000501	12,33766234	15,16966068	1,229540918
128	128	256	0,0127002	0,0002044	0,0001397	62,13405088	90,91052255	1,46313529
256	128	160	0,0234248	0,0002485	0,0001895	94,26478873	123,6137203	1,311345646
320	256	240	0,0525473	0,0006927	0,0004133	75,85866898	127,1408178	1,67602226
400	200	600	0,1438929	0,0011087	0,0007016	129,785244	205,0925029	1,580245154
512	360	600	0,3173099	0,0025967	0,001367	122,1973659	232,1213606	1,899561083
1.024	512	960	1,6310484	0,0114433	0,0052881	142,5330455	308,4375106	2,163971937
2.048	1.024	1.200	9,7669352	0,0546565	0,0354307	178,696682	275,6630606	1,542631108

A continuación se mostrarán las gráficas para cada una de las aceleraciones obtenidas, sin Tiling y con Tiling.

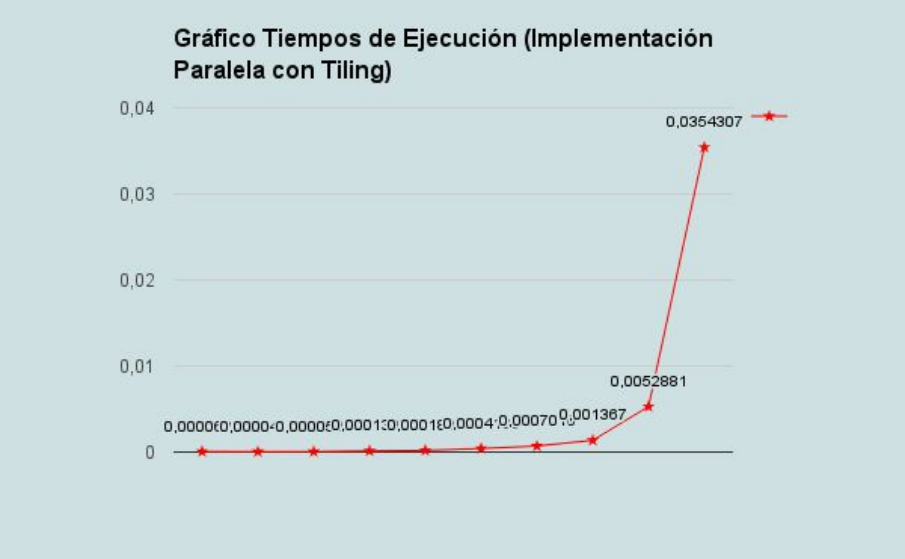






Si hacemos una comparativa entre ambas gráficas podemos notar que en la segunda que es la de Aceleración con Tiling existe una notable diferencia de incremento en los datos con respecto a la otra, es decir, la ejecución de los algoritmos tienen una aceleración mucho mayor cuando se hace de manera paralela sin tiling o con tiling que cuando se ejecuta de manera secuencial. Aún así la implementación Paralela con Tiling sigue siendo más alta en su aceleración.

Gráficos de tiempos con cada una de las implementaciones.

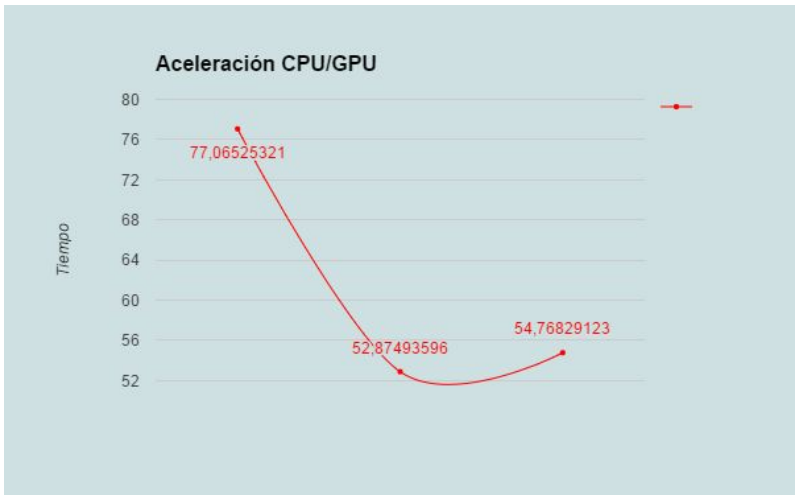


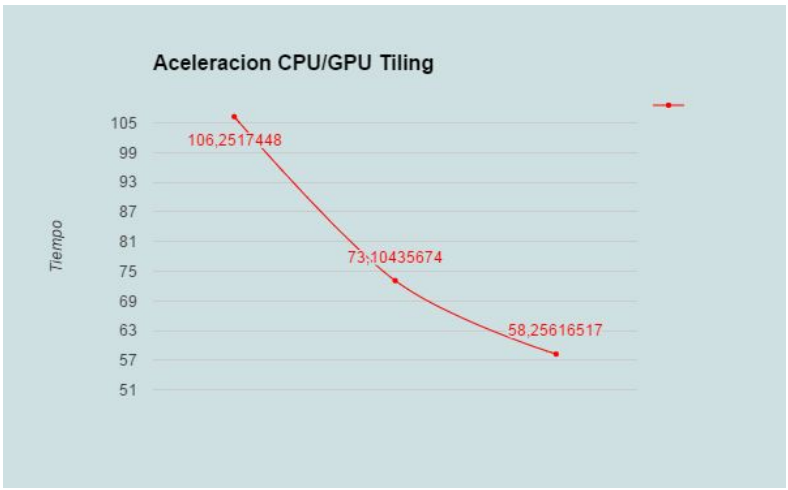
Tomar tiempos de ejecución para diferentes tamaños de bloque (4x4, 16x16, 32x32), manteniendo el mismo número de datos, verificar el funcionamiento del algoritmo.

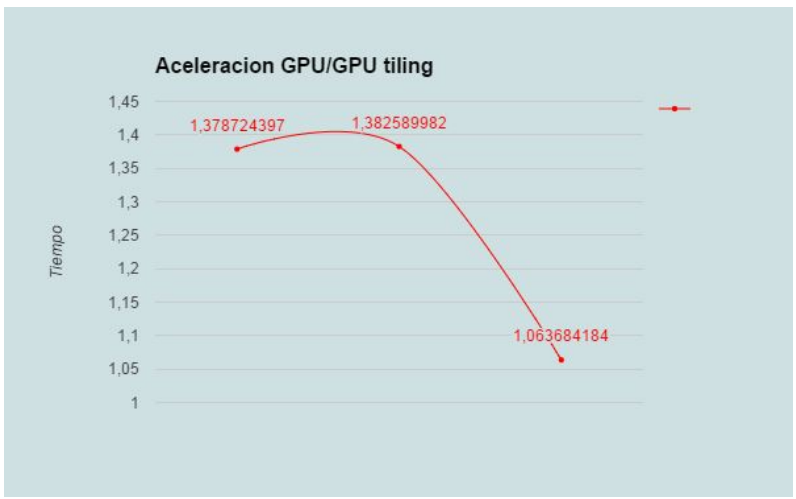
Tiempos de ejecución para un tamaño de bloque 4x4.

Dim matrices			Tiempos			Aceleraciones		
N	M	O	Secuencial	Parallel	Parallel with Tiling	CPU/GPU	CPU/GPU Tiling	GPU/GPU-Tiling
512	1024	960	1,8664819	0,0242195	0,0175666	77,06525321	106,2517448	1,378724397
1024	2048	1200	10,000464	0,1891343	0,1367971	52,87493596	73,10435674	1,382589982
2048	2048	1500	30,0204447	0,5481355	0,5153179	54,76829123	58,25616517	1,063684184

A continuación se mostrarán las gráficas resultantes de las aceleraciones de la tabla anterior.



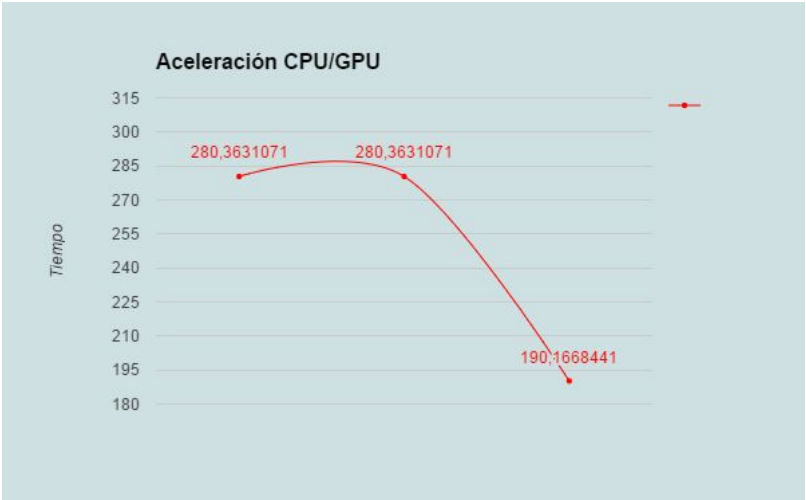


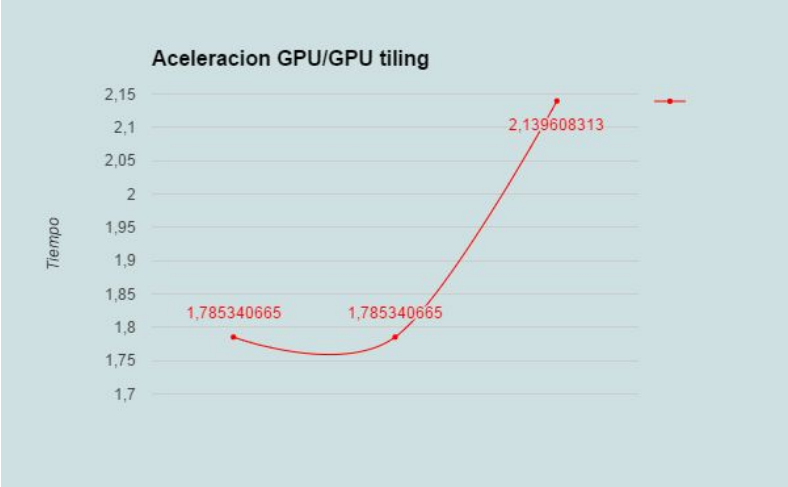


Tiempos de ejecución para tamaño de bloque de 16x16.

Dim matrices			Tiempos			Aceleraciones		
N	M	O	Secuencial	Parallel	Parallel with Tiling	CPU/GPU	CPU/GPU Tiling	GPU/GPU-Tiling
512	1024	960	2,2392321	0,0079869	0,0044736	280,3631071	500,5436561	1,785340665
1024	2048	1200	9,1621815	0,0481797	0,022518	280,3631071	500,5436561	1,785340665
2048	2048	1500	30,4936391	0,1444242	0,081842	190,1668441	406,8825606	2,139608313

A continuación se mostrarán las gráficas resultantes de las aceleraciones de la tabla anterior.

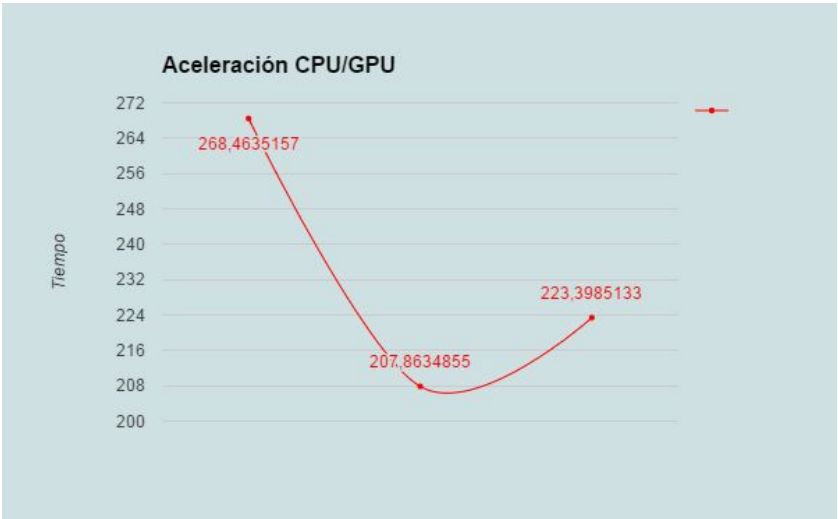


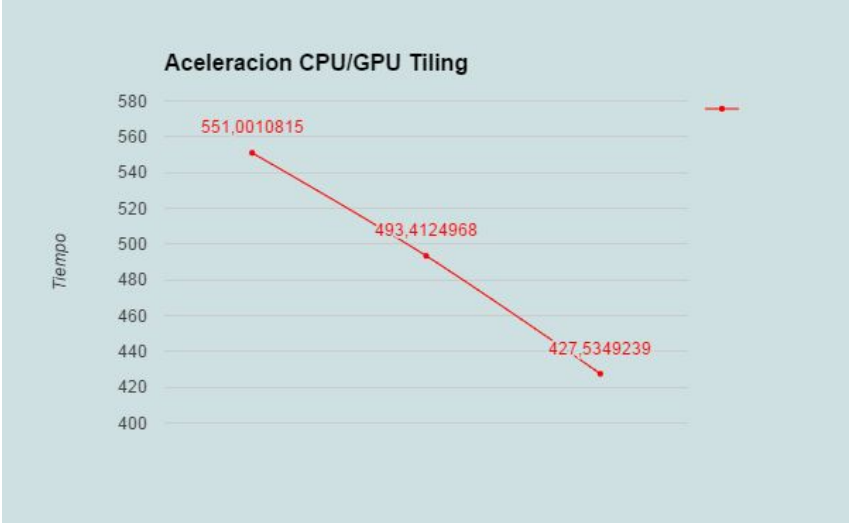


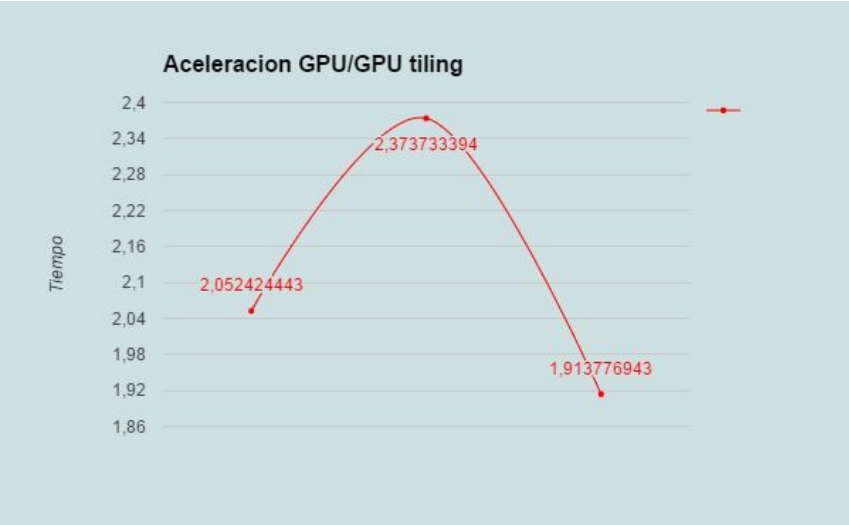
Tiempos de ejecución para un tamaño de bloque de 32x32.

Dim matrices			Tiempos			Aceleraciones		
N	**M**	**O**	**Secuencial**	**Parallel**	**Parallel with Tiling**	**CPU/GPU**	**CPU/GPU Tiling**	**GPU/GPU-Tiling**
512	1024	960	1,8340622	0,0068317	0,0033286	268,4635157	551,0010815	2,052424443
1024	2048	1200	9,7680872	0,0469928	0,019797	207,8634855	493,4124968	2,373733394
2048	2048	1500	30,16178725	0,135013375	0,070548125	223,3985133	427,5349239	1,913776943

A continuación se mostrarán las gráficas resultantes de las aceleraciones de la tabla anterior.




Aceleracion CPU/GPU Tiling
Tiempo
580
560
540
520
500
480
460
440
420
400
551,0010815
493,4124968
427,5349239


Aceleracion GPU/GPU tiling
Tiempo
2,4
2,34
2,28
2,22
2,16
2,1
2,04
1,98
1,92
1,86
2,052424443
2,373733394
1,913776943

CONCLUSIONES

Analizando el trabajo realizado se encontró que:

- La implementación de multiplicación secuencial es eficiente solamente con matrices relativamente pequeñas.
- Pudimos notar que al hacer una comparación entre las gráficas realizadas con cada una de las implementaciones, obviamente la implementación secuencial tarda muchísimo más a medida que se aumenta el valor de las dimensiones en las matrices.
- Se puede apreciar claramente que al comparar las gráficas de las respectivas aceleraciones, notamos un cambio bastante grande en cuanto a la aceleración de GPU con respecto a CPU, GPU-Tiling con respecto a CPU y GPU-tiling con respecto a GPU; aunque todas son ligeramente decrecientes, la última es creciente con dimensiones de bloque 16x16, a mayores datos más aceleración.
- Al trabajar con memoria compartida para la ejecución de algoritmos podemos comprobar que es una excelente herramienta, pero cuando se llena debido a la gran cantidad de datos ejecutándose, empieza a perder velocidad de aceleración.