Rendimiento del uso de threads en C++ en multiplicación de matrices bidimensionales

Integrantes:

- Ariana Villegas
- Renato Bacigalupo
- José Ignacio Huby

Introducción:

Hoy en día, el software secuencial tiene un límite marcado. El hardware del cual depende no logra mejorar el rendimiento de un único CPU; la cantidad de transistores en un mismo CPU está sujeto a leyes físicas de dispersión de temperatura.

Ante ello, se empezó a implementar diseños arquitectónicos que incluyen varios cores. Es decir, en lugar de intentar mejorar a un solo “trabajador”, se contratan a más “trabajadores”. Gracias a ello, la capacidad de un computador sigue aumentando a lo largo de los años.

Sin embargo, esta implementación en hardware representa un problema al momento de programar. En consecuencia, para hacerlo de manera eficiente se tendrá que utilizar la presencia de múltiples cores y dividir el algoritmo en partes “independientes” que puedan ejecutarse de manera paralela.

En vista de la evolución de la programación, este informe busca medir el rendimiento de multiplicar dos matrices en C++ con threads.

Metodología:

Primero, el rendimiento del proceso con un determinado número de threads dependerá del tiempo. Para contar con mayor efectividad, se medirá 3 veces cada caso particular y se obtendrá el promedio. De esta manera, menor tiempo, significa mayor rendimiento.

Segundo, el intervalo de threads que cubrirá el experimento será desde 0 (secuencial) hasta el máximo de cores de la computadora. En este estudio no se medirán los casos en los que el número de hilos sobrepasa al número de cores.

Tercero, se realizarán las mediciones con tres tamaños de matrices diferentes: 10x10, 100x100, 1000x1000. El criterio para elegir el tamaño es el orden y el tiempo que consume en medir. Es decir, mientras que no demande una gran cantidad de tiempo, la diferencia de tamaños entre un caso y otro serán considerables (10 a 100, 100 a 1000).

Cuarto, puesto que una matriz de dos dimensiones puede ser representada por muchas estructuras y contenedores, se eligirá las dos más comunes: arreglo dinámico (punteros) y vectores, Con ambas se repetirá el experimento.

Especificaciones de computadora:

Processador: i7-8550 U

4 cores - 8 threads

1.80Ghz ~ 2.00Ghz

RAM: 8 GB

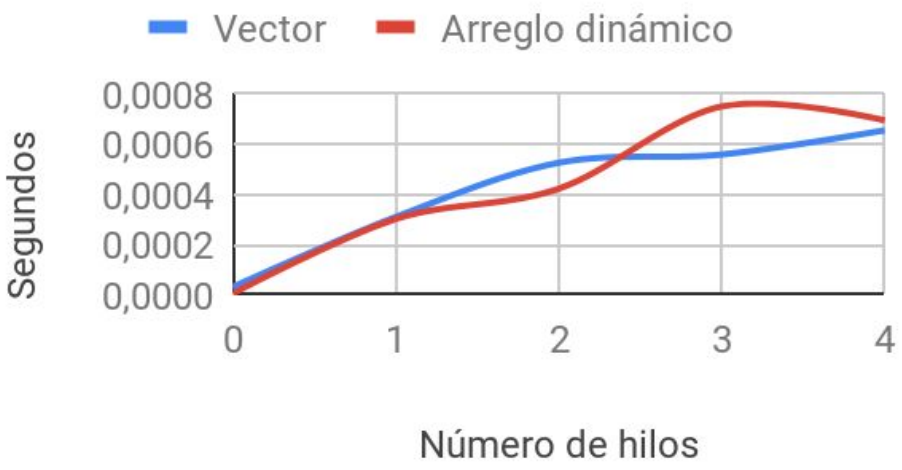
GPU: Radeon 530

HDD: 1 TB

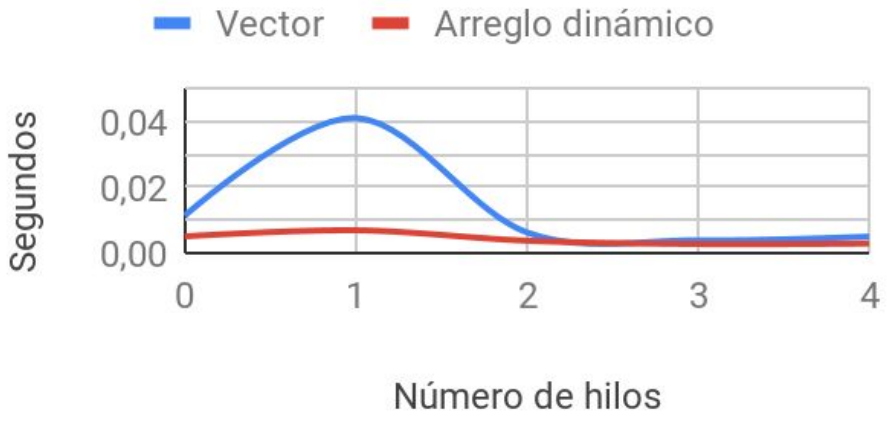
OS: Windows 10 Pro 60-bit

Resultados de la medición:

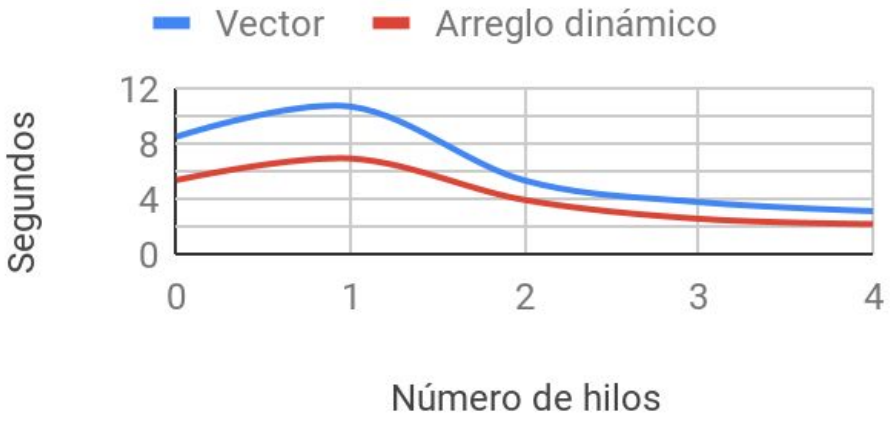
Multiplicación de dos matrices 10x10 rellenas con 10 con hilos



Multiplicación de dos matrices 100x100 rellenas con 10 con hilos



Multiplicación de dos matrices 1000x1000 rellenas con 10 con hilos



Todo con relleno de 10

Rendimiento de threads en multiplicación de matrices con vectores

10x10	Threads	Vector
	0	0,00003130
	1	0,00031067
	2	0,00052677
	3	0,00055963
	4	0,00065810

Mediciones	1	2	3
	0,00000490	0,00004900	0,00004000
	0,00031620	0,00027190	0,00034390
	0,00047450	0,00051150	0,00059430
	0,00054540	0,00053670	0,00059680
	0,00065180	0,00065330	0,00066320

100x100	Threads	Vector
	0	0,011446
	1	0,0409816
	2	0,00635866667
	3	0,00397366667
	4	0,0051156

	1	2	3
	0,010681	0,010405	0,013252
	0,102981	0,0099655	0,0099989
	0,0061364	0,0051842	0,0057554
	0,0040066	0,0039473	0,0039752
	0,0049997	0,004789	0,0055581

1000x1000	Threads	Vector
	0	8,499946667
	1	10,6762
	2	5,339533333
	3	3,783566667
	4	3,132396667

	1	2	3
	8,48527	8,50625	8,50832
	10,5178	10,4329	11,0749
	5,43516	5,24307	5,34037
	3,93366	3,70575	3,71135
	3,134	3,07451	3,18868

Rendimiento de threads en multiplicación de matrices con arreglos dinámicos

10x10	Threads	Arreglo dinámico
	0	0,000005666666
	1	0,000302666666
	2	0,000423333333
	3	0,000751333333
	4	0,000695666666

Mediciones	1	2	3
	0,000005	0,000005	0,000006
	0,000324	0,000249	0,000335
	0,000469	0,000391	0,00041
	0,001083	0,000582	0,000589
	0,000644	0,000686	0,000757

100x100	Threads	Arreglo dinámico
	0	0,0051903
	1	0,0070310
	2	0,0038723
	3	0,0028873
	4	0,0030073

	1	2	3
	0,0049410	0,0053630	0,0052470
	0,0072960	0,0066520	0,0069450
	0,0039070	0,0038610	0,0038290
	0,0027920	0,0029400	0,0028700
	0,0033500	0,0025030	0,0031690

1000x1000	Threads	Arreglo dinámico
	0	5,3690000
	1	6,9276633
	2	3,9362733
	3	2,5784967
	4	2,1967700

	1	2	3
	5,3581000	5,3871400	5,3617800
	6,8355200	7,0909200	6,8565500
	4,2003000	4,1496200	3,4590000
	2,5869200	2,5845700	2,5840000
	2,2576300	2,1559900	2,1766900

Conclusiones:

A partir de los datos obtenidos llegamos a dos conclusiones importantes:

- Primero, en la matriz de 10x10 el menor tiempo de ejecución corresponde a la multiplicación sin hilos; en la matriz de 100x100 el menor tiempo es aproximadamente con 3 hilos; y en la matriz de 1000x1000 el menor tiempo es con 4 hilos. Esto se debe a que en el tiempo de ejecución de las matrices pequeñas la creación de los hilos significa un mayor esfuerzo relativo que en las matrices grandes, por lo cual su creación es contraproducente. Teniendo en cuenta esto, los hilos son recomendables solo si la matriz tiene el tamaño suficiente para que el tiempo de creación de los hilos pueda ser insignificante. Los resultados anteriores se cumplen para el caso de arrays y el de vectores.

- Segundo, en todos los casos de matrices de diferentes dimensiones obtuvimos que al construir la matriz con arrays dinámicos el tiempo es menor a que si usamos vectores. Esto se debe a que en la construcción de los vectores se tiene que gestionar el espacio en la memoria, lo cual, tiene un precio adicional, por lo que el tiempo aumenta y el rendimiento disminuye.