

## **Tema 6: Programación con CUDA**

### **2020/21**

---

15 de enero de 2021

**Grupo 03:** José María Amusquívar Poppe y Prashant Jeswani Teiwani

Universidad de Las Palmas de Gran Canaria

Escuela de Ingeniería en Informática

## Índice

<b>Actividad práctica 1 .....</b>	<b>3</b>
<b>Actividad práctica 2 .....</b>	<b>4</b>
<b>Actividad práctica 3 .....</b>	<b>5</b>
<b>Actividad práctica 4 .....</b>	<b>6</b>
<b>Actividad práctica 5 .....</b>	<b>7</b>
<b>Actividad práctica 6 .....</b>	<b>9</b>
<b>Actividad práctica 7 .....</b>	<b>10</b>
<b>Actividad práctica 8 .....</b>	<b>12</b>
<b>Referencias .....</b>	<b>13</b>

## Actividad práctica 1

Escribe un programa que declare tres vectores de 100 números en coma flotante e inicialice dos de ellos:

- Inicializa cada elemento del primero con el valor de su índice
- Inicializa cada elemento del segundo con el doble del valor de su índice

Crea tres vectores equivalentes en el dispositivo CUDA.

Copia en el dispositivo los dos vectores inicializados.

Lanza 100 núcleos, uno para cada elemento, que calculen el producto de los elementos correspondientes de ambos vectores y pongan el resultado en el tercer vector.

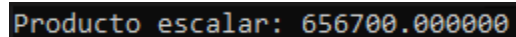
Recupera el vector resultado desde el dispositivo, suma sus elementos para obtener el producto escalar de los dos vectores inicializados y muestra el resultado en pantalla; comprueba que sea correcto.

Libera la memoria reservada en el dispositivo y reinícialo para dar por finalizado el programa.

Al crear un nuevo proyecto "CUDA" en "Visual Studio", se genera automáticamente el código encargado de liberar memoria, así como el de reiniciar para dar por finalizado el programa. Entonces, para este apartado, sólo hubo que editar ciertas líneas de código.

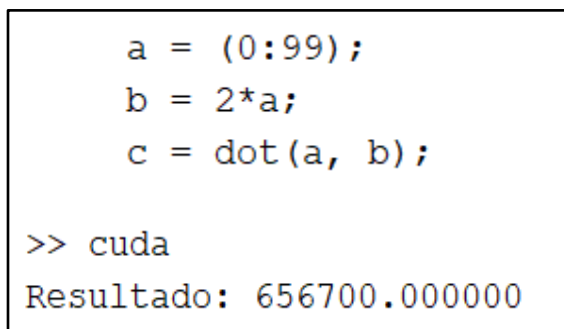
Así pues, una vez se he realizado la respectiva multiplicación de los vectores, elemento a elemento, se realiza la suma del vector resultado "C" para obtener el resultado del producto escalar.

Finalmente, tal como se puede apreciar en las siguientes figuras (1, 2), se comprueba que el resultado obtenido con "CUDA" es correcto.



Producto escalar: 656700.000000

*Figura 1. Resultado del producto escalar usando "CUDA".*



```
a = (0:99);  
b = 2*a;  
c = dot(a, b);  
  
>> cuda  
Resultado: 656700.000000
```

*Figura 2. Comprobación del producto escalar en "MATLAB".*

## Actividad práctica 2

Modifica el programa que calcula el producto escalar de dos vectores para que cada hilo muestre por pantalla el índice tridimensional del bloque y su índice tridimensional de hilo junto con el elemento que está calculando.

Modifica el programa para que el trabajo se divida en 10 bloques y comprueba que el resultado sigue siendo el correcto.

Verifica si el comportamiento del programa varía entre distintas ejecuciones.

Se ha modificado el código anterior para que muestre lo solicitado, y también se ha comprobado que el resultado no varía al dividir la ejecución en 10 bloques, ya que se obtiene el mismo valor que se obtenía en la figura 1.

Finalmente, se ha verificado que el comportamiento del programa varía entre ejecuciones, dado que cambia el orden en el que se ejecutan los distintos bloques y, al haber 10 bloques, cada uno de ellos ejecuta únicamente 10 elementos del total.

Indice del hilo: (6, 0, 0)	Indice del bloque: (5, 0, 0)	Calculando el producto: {56.000000} * {112.000000}
Indice del hilo: (7, 0, 0)	Indice del bloque: (5, 0, 0)	Calculando el producto: {57.000000} * {114.000000}
Indice del hilo: (8, 0, 0)	Indice del bloque: (5, 0, 0)	Calculando el producto: {58.000000} * {116.000000}
Indice del hilo: (9, 0, 0)	Indice del bloque: (5, 0, 0)	Calculando el producto: {59.000000} * {118.000000}
Indice del hilo: (0, 0, 0)	Indice del bloque: (1, 0, 0)	Calculando el producto: {10.000000} * {20.000000}
Indice del hilo: (1, 0, 0)	Indice del bloque: (1, 0, 0)	Calculando el producto: {11.000000} * {22.000000}
Indice del hilo: (2, 0, 0)	Indice del bloque: (1, 0, 0)	Calculando el producto: {12.000000} * {24.000000}

Figura 3. Operaciones para los bloques 1 y 5, de la ejecución con 10 bloques.

Indice del hilo: (6, 0, 0)	Indice del bloque: (3, 0, 0)	Calculando el producto: {36.000000} * {72.000000}
Indice del hilo: (7, 0, 0)	Indice del bloque: (3, 0, 0)	Calculando el producto: {37.000000} * {74.000000}
Indice del hilo: (8, 0, 0)	Indice del bloque: (3, 0, 0)	Calculando el producto: {38.000000} * {76.000000}
Indice del hilo: (9, 0, 0)	Indice del bloque: (3, 0, 0)	Calculando el producto: {39.000000} * {78.000000}
Indice del hilo: (0, 0, 0)	Indice del bloque: (4, 0, 0)	Calculando el producto: {40.000000} * {80.000000}
Indice del hilo: (1, 0, 0)	Indice del bloque: (4, 0, 0)	Calculando el producto: {41.000000} * {82.000000}
Indice del hilo: (2, 0, 0)	Indice del bloque: (4, 0, 0)	Calculando el producto: {42.000000} * {84.000000}

Figura 4. Operaciones para los bloques 3 y 4, de la ejecución con 10 bloques.

Indice del hilo: (6, 0, 0)	Indice del bloque: (6, 0, 0)	Calculando el producto: {66.000000} * {132.000000}
Indice del hilo: (7, 0, 0)	Indice del bloque: (6, 0, 0)	Calculando el producto: {67.000000} * {134.000000}
Indice del hilo: (8, 0, 0)	Indice del bloque: (6, 0, 0)	Calculando el producto: {68.000000} * {136.000000}
Indice del hilo: (9, 0, 0)	Indice del bloque: (6, 0, 0)	Calculando el producto: {69.000000} * {138.000000}
Indice del hilo: (0, 0, 0)	Indice del bloque: (2, 0, 0)	Calculando el producto: {20.000000} * {40.000000}
Indice del hilo: (1, 0, 0)	Indice del bloque: (2, 0, 0)	Calculando el producto: {21.000000} * {42.000000}
Indice del hilo: (2, 0, 0)	Indice del bloque: (2, 0, 0)	Calculando el producto: {22.000000} * {44.000000}

Figura 5. Operaciones para los bloques 2 y 6, de la ejecución con 10 bloques.

## Actividad práctica 3

Escribe un programa que multiplique dos matrices cuadradas de números en coma flotante utilizando la función "MatrixMultiplication".

- Las matrices deben declararse como vectores de  $N \times N$  elementos
- El valor de cada elemento de la primera matriz será la suma de su número de fila más su número de columna
- El valor de cada elemento de la segunda matriz será la resta de su número de fila menos su número de columna

Muestra por pantalla el resultado de una multiplicación con  $N = 3$  para comprobar que el programa funciona correctamente.

0.0	1.0	2.0	X	0.0	-1.0	-2.0	=	5.0	2.0	-1.0
1.0	2.0	3.0		1.0	0.0	-1.0		8.0	2.0	-4.0
2.0	3.0	4.0		2.0	1.0	0.0		11.0	2.0	-7.0

A continuación, se muestra el resultado obtenido en este apartado, comprobando que es análogo al resultado proporcionado en este mismo ejercicio.

```
Matriz A (suma de su numero de fila mas su numero de columna):
    0.00    1.00    2.00
    1.00    2.00    3.00
    2.00    3.00    4.00

Matriz B (resta de su numero de fila menos su numero de columna):
    0.00   -1.0   -2.0
    1.00    0.0   -1.0
    2.00    1.0    0.0

Resultado de la multiplicacion de A * B:
    5.00    2.00   -1.0
    8.00    2.00   -4.0
   11.00    2.00   -7.0
```

Figura 6. Resultado obtenido, idéntico al presentado en el enunciado.

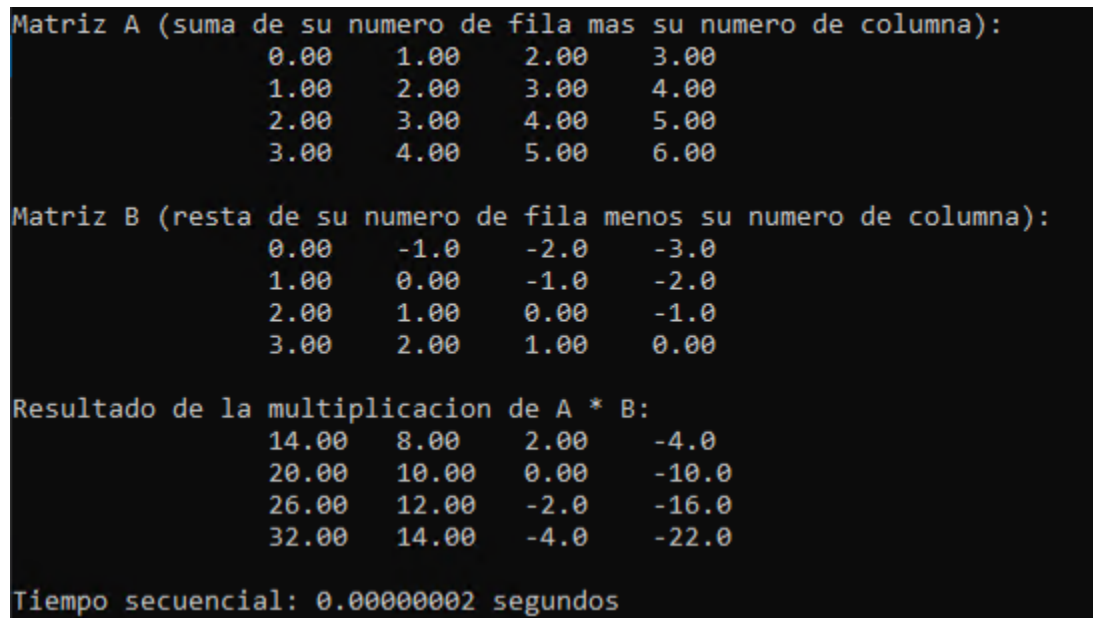
## Actividad práctica 4

Mide el tiempo que tarda en ejecutarse la multiplicación de matrices para valores crecientes del tamaño de la matriz que sean potencia de 2.

- Mide solo el tiempo que tarda la función “MatrixMultiplication”.
- Dibuja una gráfica con el tiempo que tarda para los tamaños 1, 2, 4, 8, 16 y 32.
- La precisión de los contadores no es suficiente, así que tendrás que ejecutar la función al menos 10000 veces para cada tamaño y calcular la media.

```
#include <time.h>
...
#define N 32
...
clock_t start, stop;
...
start = clock();
for (int i = 0; i < 10000; i++) MatrixMultiplication(A, B, C, N);
stop = clock();
printf("Tiempo secuencial: %f segundos\n",
      (float)(stop - start) / CLOCKS_PER_SEC / 10000);
```

El tiempo obtenido para una matriz de 4 por 4 se presenta en la siguiente figura, al igual que la matriz resultado.



```
Matriz A (suma de su numero de fila mas su numero de columna):
    0.00    1.00    2.00    3.00
    1.00    2.00    3.00    4.00
    2.00    3.00    4.00    5.00
    3.00    4.00    5.00    6.00

Matriz B (resta de su numero de fila menos su numero de columna):
    0.00   -1.0   -2.0   -3.0
    1.00    0.0   -1.0   -2.0
    2.00    1.0    0.0   -1.0
    3.00    2.0    1.0    0.0

Resultado de la multiplicacion de A * B:
    14.00    8.00    2.00   -4.0
    20.00   10.00    0.00  -10.0
    26.00   12.00   -2.00  -16.0
    32.00   14.00   -4.00  -22.0

Tiempo secuencial: 0.00000002 segundos
```

Figura 7. Mensaje de consola para una matriz de 4x4, junto a su tiempo de ejecución.

Para concluir, se presenta la respectiva gráfica que representa todos estos tiempos.

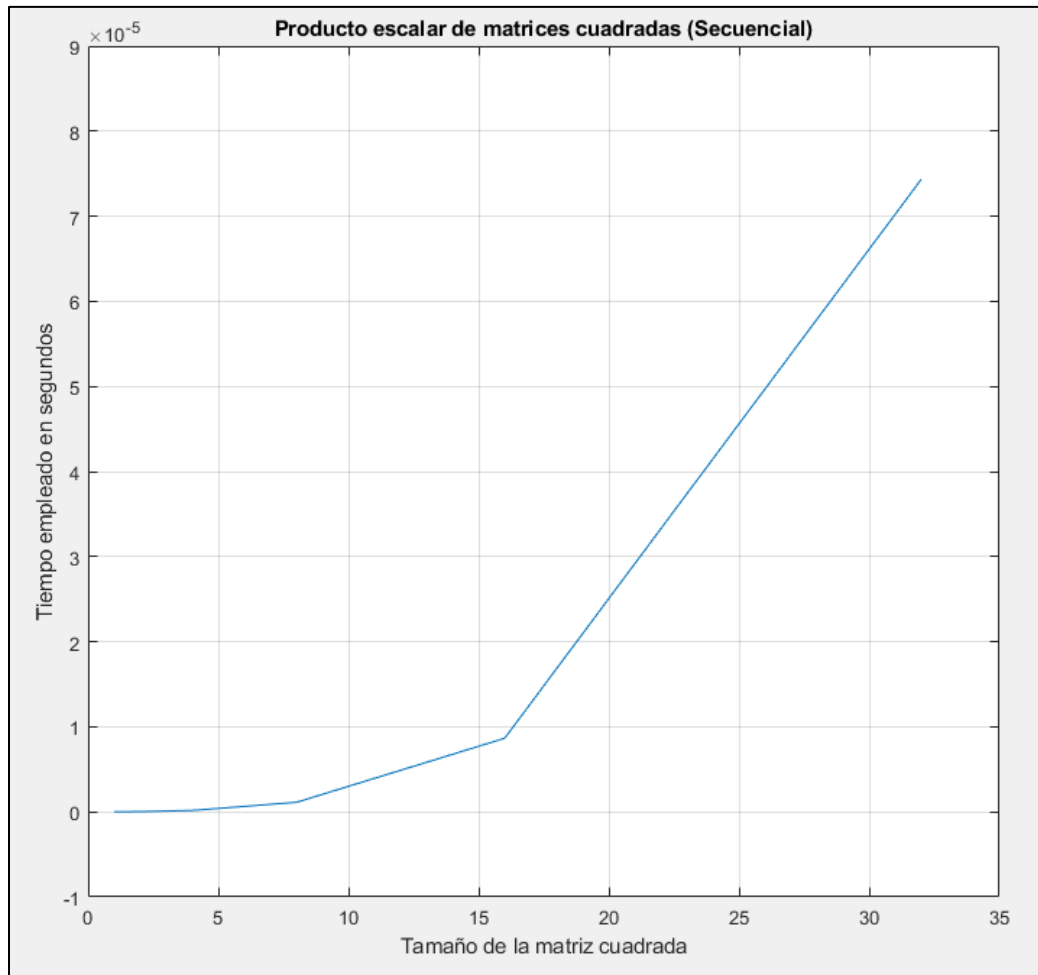


Figura 8. Gráfica para dimensiones de potencias de 2, de 1 hasta 32.

## Actividad práctica 5

Escribe un programa que multiplique dos matrices cuadradas de números en coma flotante utilizando CUDA.

- Transforma la función “MatrixMultiplication” en un núcleo.
  - Sustituye la variable  $i$  por el índice de hilo “threadIdx.x”.
  - Sustituye la variable  $j$  por el índice de hilo “threadIdx.y”.
- El núcleo debe lanzarse usando un único bloque con sus hilos estructurados bidimensionalmente para que coincidan con el tamaño de las matrices.

Muestra por pantalla el resultado de una multiplicación con  $N = 3$  para comprobar que el programa funciona correctamente.

0.0	1.0	2.0	X	0.0	-1.0	-2.0	=	5.0	2.0	-1.0
1.0	2.0	3.0		1.0	0.0	-1.0		8.0	2.0	-4.0
2.0	3.0	4.0		2.0	1.0	0.0		11.0	2.0	-7.0

Para realización de este apartado, se ha creado un núcleo que ejecuta el producto escalar de las matrices. Apoyándose de “threadIdx”, es posible obviar los dos bucles encargados de realizar el recorrido de las matrices, ahorrando así tiempo en ejecución. Además, puesto que se tratan de matrices, es necesario redefinir la reservación de memoria, ya que ahora será de “ $N*N$ ”, donde “ $N$ ” es la dimensión de la matriz. Finalmente, el resultado obtenido en este apartado es el mismo que se ha obtenido en el apartado 3.

```
Matriz A (suma de su numero de fila mas su numero de columna):
    0.00    1.00    2.00
    1.00    2.00    3.00
    2.00    3.00    4.00

Matriz B (resta de su numero de fila menos su numero de columna):
    0.00    -1.0    -2.0
    1.00     0.00   -1.0
    2.00     1.00    0.00

Resultado de la multiplicacion de A * B:
    5.00    2.00   -1.0
    8.00    2.00   -4.0
   11.00    2.00   -7.0
```

Figura 9. Resultado obtenido con CUDA, idéntico al secuencial.



## Actividad práctica 6

Mide el tiempo que tarda en ejecutarse la multiplicación de matrices para valores crecientes del tamaño de la matriz que sean potencia de 2.

- Mide el tiempo que tarda la copia de datos y la ejecución del núcleo.
- Ejecuta el núcleo al menos 10 veces para cada matriz y calcula el tiempo medio.
- Dibuja una gráfica con el tiempo que tarda para los tamaños 1, 2, 4, 8, 16 y 32.

```
cudaEvent_t start, stop;
cudaEventCreate(&start);
cudaEventCreate(&stop);
...
cudaEventRecord(start);
...
cudaEventRecord(stop);
...
cudaEventSynchronize(stop);
float milliseconds = 0;
cudaEventElapsedTime(&milliseconds, start, stop);
```

Compara la gráfica con la que obtuviste para la ejecución secuencial y razona las diferencias entre los dos comportamientos.

Comprueba qué ocurre si intentas ejecutar el programa con tamaños mayores de matriz y determina la causa.

Para resolver este apartado se empleó la misma dinámica que en el apartado 4, obteniendo por consola las dos matrices a operar “A” y “B”, y la matriz resultado “C”, además del tiempo que ha tardado en ejecutar este código. Analizando los tiempos obtenidos con “CUDA” se puede apreciar que estos tiempos son superiores a los obtenidos en secuencial, esto se debe a que en la programación con “CUDA” existen instrucciones adicionales que en secuencial no, como por ejemplo la copia de los vectores desde la memoria principal del ordenador a la memoria de la tarjeta gráfica, y viceversa.

Sin embargo, a medida que el tamaño de las matrices es mayor, resultará más eficiente la programación por “CUDA” que el modo secuencial.

A continuación, se presenta la gráfica comparativa entre el modo secuencial y con “CUDA”, para dimensiones que van desde 1 hasta 32.

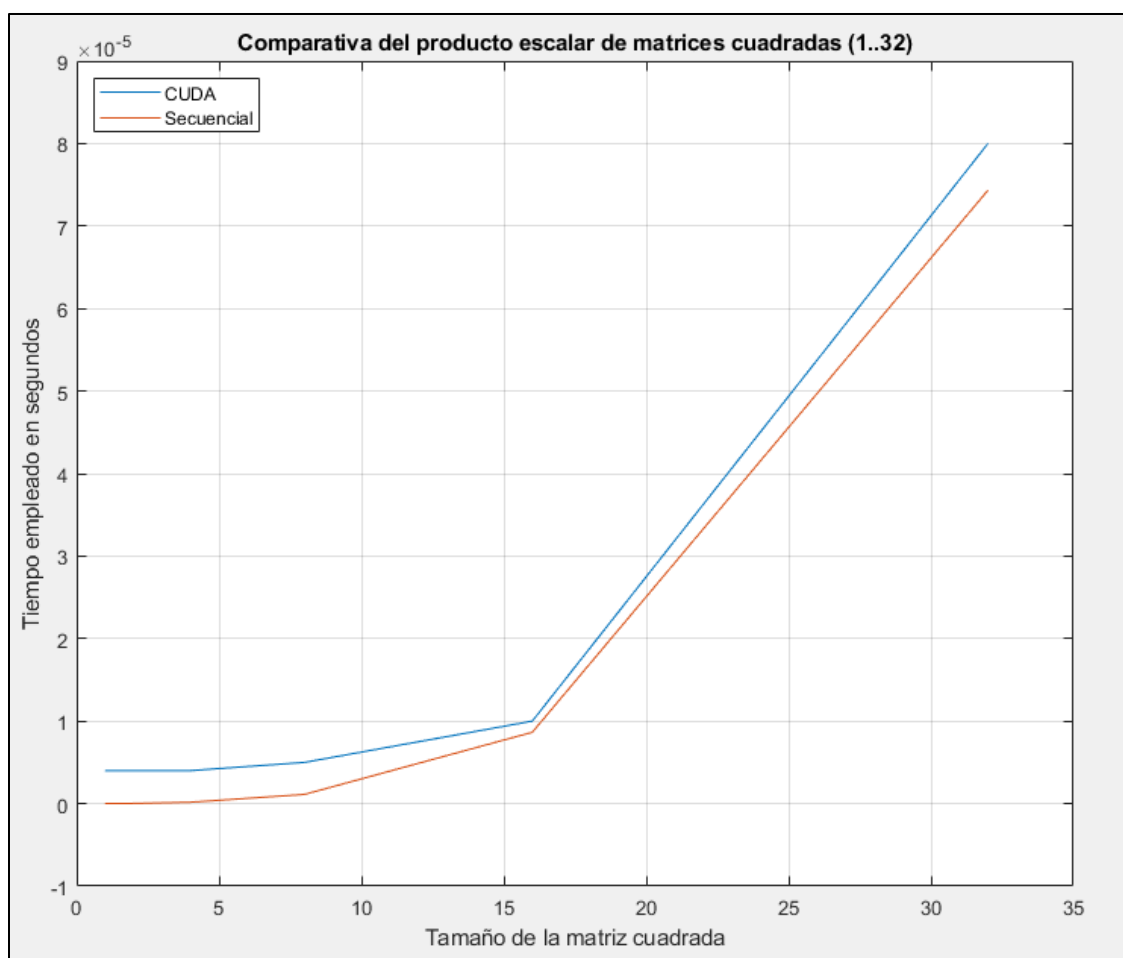


Figura 10. Comparativa de tiempos entre secuencial y "CUDA".

## Actividad práctica 7

Modifica el programa que multiplica matrices cuadradas en CUDA para que las matrices grandes sean divididas en bloques.

- El tamaño de los bloques será siempre 32x32.
- Asumiremos que las dimensiones de las matrices serán potencias de dos.
- Consideraremos matriz grande aquella de dimensiones superiores a 32x32.

Para comprobar que el programa funciona correctamente para matrices grandes, imprime los valores de la diagonal principal de la matriz y compáralos con los valores que da la versión secuencial del programa.

En el apartado anterior se ha visto que no se ha podido resolver matrices con dimensiones superiores a 32, debido a que el tamaño máximo de un bloque es de 1024 hilos (32x32). Por este motivo, para dimensiones superiores se ha ajustado el número de bloques según el tamaño de matriz.

A continuación, se presentan los primeros valores de la diagonal, tanto en secuencial como con “CUDA”, para una matriz de dimensión 128.

```
Resultado de la multiplicacion de A * B (secuencial):  
690880.000000  
690752.000000  
690368.000000  
689728.000000  
688832.000000  
687680.000000  
686272.000000  
684608.000000  
682688.000000  
680512.000000  
678080.000000  
675392.000000  
672448.000000  
669248.000000  
665792.000000
```

Figura 11. Primeros valores de la diagonal principal para N=128. (Secuencial).

```
Resultado de la multiplicacion de A * B(CUDA) :  
690880.000000  
690752.000000  
690368.000000  
689728.000000  
688832.000000  
687680.000000  
686272.000000  
684608.000000  
682688.000000  
680512.000000  
678080.000000  
675392.000000  
672448.000000  
669248.000000  
665792.000000
```

Figura 12. Primeros valores de la diagonal principal para N=128. (CUDA).

## Actividad práctica 8

Extiende la gráfica que compara el tiempo de ejecución de las versiones secuencial y CUDA de la multiplicación de matrices.

- Añade al menos el tiempo para los tamaños 64, 128, 256, 512 y 1024.
- La versión secuencial solo debe ejecutarse 10 veces para estos tamaños.
- Asegúrate de que las matrices están declaradas utilizando memoria dinámica.

Razona las diferencias entre la evolución de las dos líneas de la gráfica a la vista de la nueva información añadida.

Se ha observado que, para dimensiones de matriz pequeños, la ejecución en modo secuencial obtiene un menor tiempo que con “CUDA” (figura 10). Sin embargo, teniendo en cuenta la jerarquía de los hilos y aprovechando el potencial del paralelismo en “CUDA”, a medida que las dimensiones de las matrices aumentan, el tiempo obtenido con “CUDA” es mucho menor que el que se obtiene en modo secuencial, tal como se aprecia en la siguiente figura.

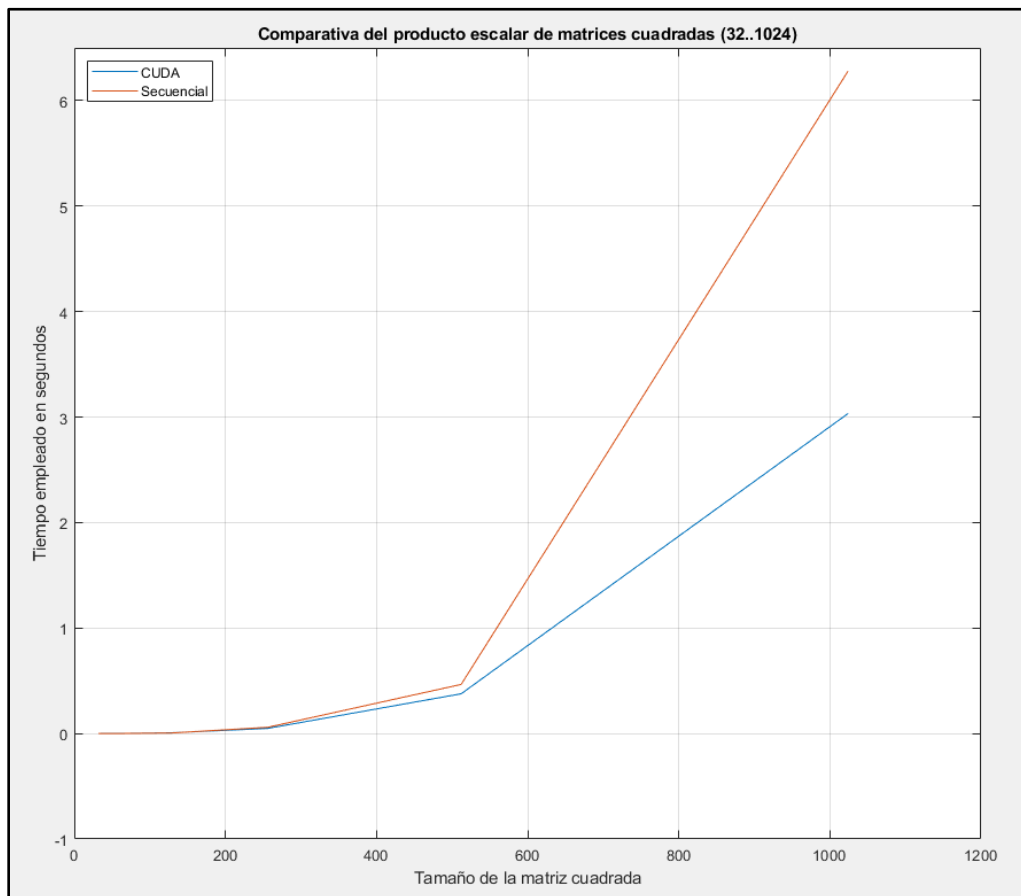


Figura 13. Comparación de tiempos entre CUDA y secuencial para  $N \geq 32$ .

## Referencias

ULPGC. (s.f.). Obtenido de [https://ncvt-aep.ulpgc.es/cv/ulpgctp21/pluginfile.php/412009/mod\\_resource/content/0/9.%20Introducci%C3%B3n%20a%20la%20Programaci%C3%B3n%20CUDA.pdf](https://ncvt-aep.ulpgc.es/cv/ulpgctp21/pluginfile.php/412009/mod_resource/content/0/9.%20Introducci%C3%B3n%20a%20la%20Programaci%C3%B3n%20CUDA.pdf)