

### Hoja de trabajo #3 - Computación Paralela y Distribuida

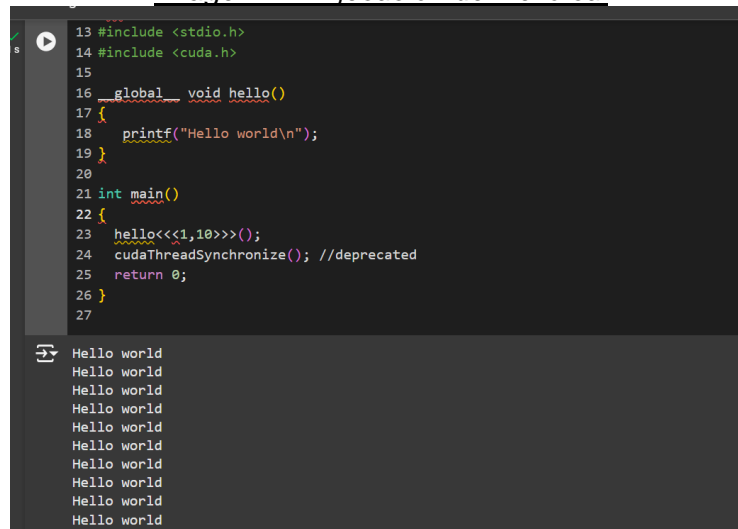
Repositorio de GitHub: [erickguerra22/HDT3\\_Paralela](https://github.com/erickguerra22/HDT3_Paralela)

#### Ejercicio 1

El programa hello.cu ilustra la forma básica del modelo de ejecución para CUDA. Realice las siguientes acciones para comprender el efecto de la configuración del kernel y su relación con el Compute Capability de una tarjeta.

1. Compile el programa (ignore la advertencia sobre código deprecado en caso le salga):
2. Ejecute el programa. Observe cuántas veces se imprime el mensaje y su conexión con la configuración de la llamada al kernel – `hello<<<g,b>>>()`:

*Imagen 1.1: Ejecución de 'hello.cu'*



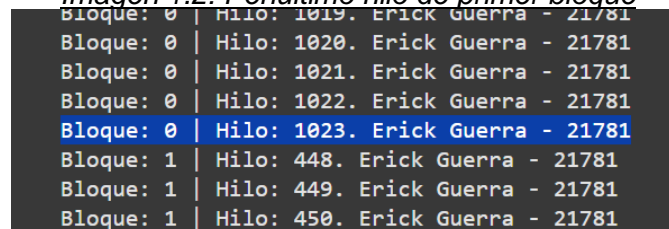
```
13 #include <stdio.h>
14 #include <cuda.h>
15
16 __global__ void hello()
17 {
18     printf("Hello world\n");
19 }
20
21 int main()
22 {
23     hello<<<1,10>>>();
24     cudaThreadSynchronize(); //deprecated
25     return 0;
26 }
27
```

Hello world  
Hello world  
Hello world  
Hello world  
Hello world  
Hello world  
Hello world  
Hello world  
Hello world  
Hello world

**R:** El mensaje se ha impreso un total de 10 veces, lo cual va de acuerdo con la instrucción `hello<<<1,10>>>()`, en donde se indica que se ejecute la función en 1 bloque, dividido en 10 hilos.

3. Modifique el programa para correr 2 bloques de 1024 hilos. Modificarlo también para que imprima su nombre y carnet. Busque en el despliegue de consola el mensaje del último hilo de la serie (1023).

*Imagen 1.2: Penúltimo hilo de primer bloque*




```
Bloque: 0 | Hilo: 1019. Erick Guerra - 21781
Bloque: 0 | Hilo: 1020. Erick Guerra - 21781
Bloque: 0 | Hilo: 1021. Erick Guerra - 21781
Bloque: 0 | Hilo: 1022. Erick Guerra - 21781
Bloque: 0 | Hilo: 1023. Erick Guerra - 21781
Bloque: 1 | Hilo: 448. Erick Guerra - 21781
Bloque: 1 | Hilo: 449. Erick Guerra - 21781
Bloque: 1 | Hilo: 450. Erick Guerra - 21781
```

*Imagen 1.3: Penúltimo hilo de segundo bloque*

```
Bloque: 1 | Hilo: 1018. Erick Guerra - 21781
Bloque: 1 | Hilo: 1019. Erick Guerra - 21781
Bloque: 1 | Hilo: 1020. Erick Guerra - 21781
Bloque: 1 | Hilo: 1021. Erick Guerra - 21781
Bloque: 1 | Hilo: 1022. Erick Guerra - 21781
Bloque: 1 | Hilo: 1023. Erick Guerra - 21781
Bloque: 0 | Hilo: 928. Erick Guerra - 21781
Bloque: 0 | Hilo: 929. Erick Guerra - 21781
```

4. Busque en el sitio de Nvidia el Compute Capability de la tarjeta que poseen las máquinas del Laboratorio (o de la computadora que está utilizando). Escriba acá el valor de CC y busque la tabla resumen con las características técnicas del CC:

**R:** Para este laboratorio se está utilizando Google Colaboratory, cuya máquina utiliza la tarjeta gráfica NVIDIA Tesla T4, cuya Compute Capability es de 7.5. Las especificaciones técnicas básicas para este tipo de Compute Capability se muestran a continuación:

Table 21: Technical Specifications per Compute Capability 

	Compute Capability														
Technical Specifications	5.0	5.2	5.3	6.0	6.1	6.2	7.0	7.2	7.5	8.0	8.6	8.7	8.9	9.0	
Maximum number of resident grids per device (Concurrent Kernel Execution)	32		16	128	32	16	128	16	128						
Maximum dimensionality of grid of thread blocks	3														
Maximum x -dimension of a grid of thread blocks [thread blocks]	2 <sup>31</sup> -1														
Maximum y- or z-dimension of a grid of thread blocks	65535														
Maximum dimensionality of thread block	3														
Maximum x- or y-dimensionality of a block	1024														
Maximum z-dimension of a block	64														
Maximum number of threads per block	1024														
Warp size	32														
Maximum number of resident blocks per SM	32									16	32	16	24	32	
Maximum number of resident warps per SM	64									32	64	48	64		
Maximum number of resident threads per SM	2048									1024	2048	1536	2048		
Number of 32-bit registers per SM	64 K														

Para ver la tabla completa e información más detallada, dirigirse a: [CUDA C++ Programming Guide/Compute Capabilities/Features and Technical Specifications](https://docs.nvidia.com/cuda/cuda-c-programming-guide/#compute-capabilities)

5. Modifique el programa para correr 1 bloque de 2048 hilos.

*Imagen 1.4: Ejecución con 1 bloque y 2048 hilos*



```
13 #include <stdio.h>
14 #include <cuda.h>
15
16 global void hello()
17 {
18     printf("Bloque: %d | Hilo: %d. Erick Guerra - 21781\n", blockIdx.x, threadIdx.x);
19 }
20
21 int main()
22 {
23     hello<<1,2048>>>();
24     cudaThreadSynchronize(); //deprecated
25     return 0;
26 }
27
```

**R:** Como se aprecia en la imagen, el código no fue capaz de imprimir ningún resultado, ya que tal como se muestra en la tabla anterior con las especificaciones técnicas para la Compute Compatibility de 7.5, el máximo de hilos soportados por bloque es de 1024. Por lo tanto, la tarjeta gráfica no fue capaz de procesar la solicitud de 1 bloque con 2048 hilos.

6. Busque en la tabla de CC los siguientes datos de la GPU que está utilizando:

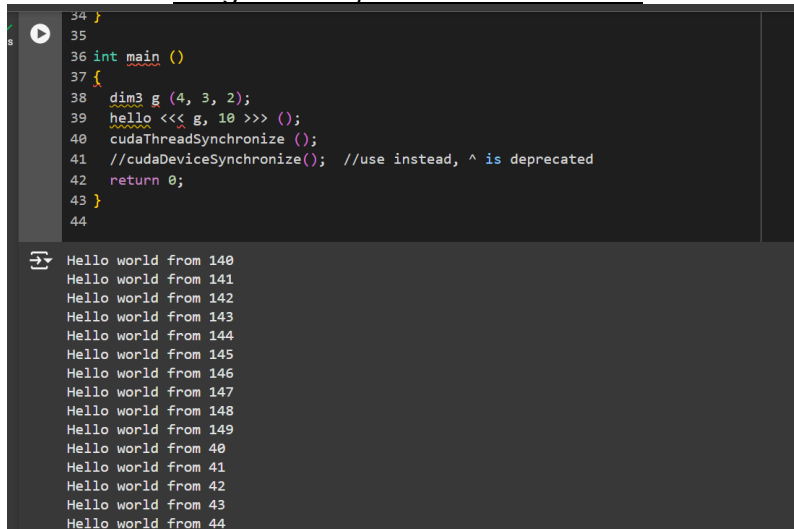
- Warp size: **32**
- Maximum number of threads per block: **1024**
- Maximum dimensionality of a grid of thread blocks: **3**
- Maximum size per grid dimension:
  - X:  **$2^{31}-1$**
  - Y, Z: **65535**
- Maximum dimensionality of a thread block: **3**
- Maximum size per block dimension:
  - X,Y: **1024**
  - Z: **64**

## Ejercicio 2

El programa hello2.cu ilustra la forma para calcular un identificador global al momento de usar hilos que pertenecen a bloques diferentes. Realice las siguientes acciones para comprender el efecto de la configuración del kernel y su relación con la forma de calcular el ID único de los hilos.

1. Descargue, compile y ejecute hello2.cu. Observe la relación de la configuración de la llamada al kernel con la geometría de los hilos y el resultado. Escriba la respuesta a los dos enunciados:

*Imagen 2.1: Ejecución de 'hello2.cu'*



```
34 }
35
36 int main ()
37 {
38     dim3 g (4, 3, 2);
39     hello <<< g, 10 >>> ();
40     cudaThreadSynchronize ();
41     //cudaDeviceSynchronize(); //use instead, ^ is deprecated
42     return 0;
43 }
44
```

```
Hello world from 140
Hello world from 141
Hello world from 142
Hello world from 143
Hello world from 144
Hello world from 145
Hello world from 146
Hello world from 147
Hello world from 148
Hello world from 149
Hello world from 40
Hello world from 41
Hello world from 42
Hello world from 43
Hello world from 44
```

- a. Máximo ID de los hilos: **239**
  - b. Ejecución de los hilos en orden: **Si bien la impresión de los hilos no va en orden secuencial desde 0 hasta 239, sí siguen secuencias de decenas (0-9), esto se debe a que el cálculo del ID del hilo depende estrictamente del bloque en el que se están ejecutando. Al asignarse 10 hilos para cada bloque, los IDs de un mismo bloque respetarán la secuencia de 0 a 9.**
2. Observe que la fórmula genérica para cálculo del ID global está en los comentarios. Modifique el programa para que imprima también su nombre y carné. Luego, realice la siguiente modificación al programa (al inicio del main) y use la fórmula genérica para derivar el nuevo cálculo de ID:

```
dim3 g (4,2);
dim3 b (32,16);
hello <<>>();
```

*Imagen 2.2: Fórmula para calcular el ID global*

```
16 __global__ void hello ()
17 {
18     int myID = ( blockIdx.z * gridDim.x * gridDim.y +
19                 blockIdx.y * gridDim.x +
20                 blockIdx.x ) * blockDim.x * blockDim.y * blockDim.z +
21                 threadIdx.z * blockDim.x * blockDim.y +
22                 threadIdx.y * blockDim.x +
23                 threadIdx.x;
24
25 // Simplification of above
26 //grid: 3D --- z,y,x: all dims and blockids
27 //block: 1D -- x
28 // int myID = ( blockIdx.z * gridDim.x * gridDim.y +
29 //             blockIdx.y * gridDim.x +
30 //             blockIdx.x ) * blockDim.x +
31 //             threadIdx.x;
```

*Imagen 2.3: Salida de pantalla con fórmula descomentada*

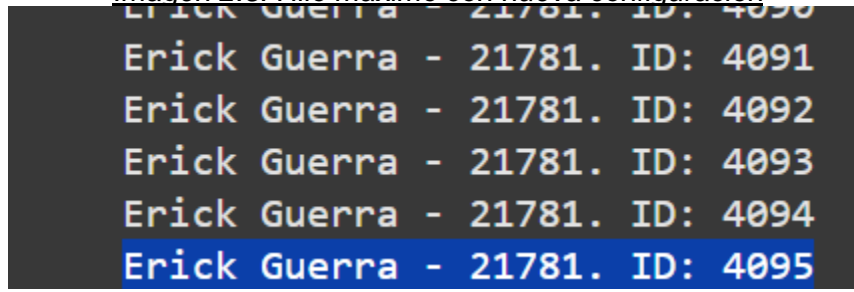
```
+ Código + Texto
2s
Erick Guerra - 21781. ID: 2464
Erick Guerra - 21781. ID: 2465
Erick Guerra - 21781. ID: 2466
Erick Guerra - 21781. ID: 2467
Erick Guerra - 21781. ID: 2468
Erick Guerra - 21781. ID: 2469
Erick Guerra - 21781. ID: 2470
Erick Guerra - 21781. ID: 2471
Erick Guerra - 21781. ID: 2472
Erick Guerra - 21781. ID: 2473
Erick Guerra - 21781. ID: 2474
Erick Guerra - 21781. ID: 2475
Erick Guerra - 21781. ID: 2476
Erick Guerra - 21781. ID: 2477
Erick Guerra - 21781. ID: 2478
Erick Guerra - 21781. ID: 2479
Erick Guerra - 21781. ID: 2480
Erick Guerra - 21781. ID: 2481
Erick Guerra - 21781. ID: 2482
Erick Guerra - 21781. ID: 2483
Erick Guerra - 21781. ID: 2484
Erick Guerra - 21781. ID: 2485
Erick Guerra - 21781. ID: 2486
Erick Guerra - 21781. ID: 2487
Erick Guerra - 21781. ID: 2488
Erick Guerra - 21781. ID: 2489
Erick Guerra - 21781. ID: 2490
Erick Guerra - 21781. ID: 2491
```

*Imagen 2.4: Nueva configuración*

```
32
33 printf ("Erick Guerra - 21781. ID: %i\n", myID);
34 }
35
36 int main ()
37 {
38     dim3 g (4,2);
39     dim3 b (32,16);
40     hello <<<g, b>>>();
41     cudaThreadSynchronize ();
42     //cudaDeviceSynchronize(); //use instead, ^ is deprecated
43     return 0;
44 }
45
```

```
Erick Guerra - 21781. ID: 327
Erick Guerra - 21781. ID: 328
Erick Guerra - 21781. ID: 329
Erick Guerra - 21781. ID: 330
Erick Guerra - 21781. ID: 331
Erick Guerra - 21781. ID: 332
Erick Guerra - 21781. ID: 333
```

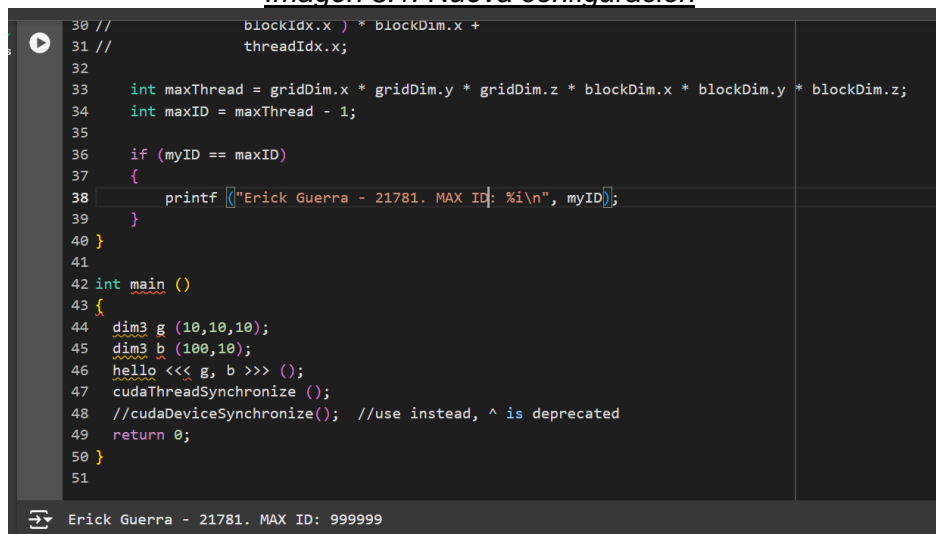
*Imagen 2.5: Hilo máximo con nueva configuración*



```
Erick Guerra - 21781. ID: 4090
Erick Guerra - 21781. ID: 4091
Erick Guerra - 21781. ID: 4092
Erick Guerra - 21781. ID: 4093
Erick Guerra - 21781. ID: 4094
Erick Guerra - 21781. ID: 4095
```

3. Revise nuevamente la información del Compute Capability respecto a las dimensiones máximas de hilos-bloque en x, y, & z para una grilla. Cree una configuración para lanzar exitosamente el kernel para procesar 100,000 datos. (Sugerencia: busque una configuración que lance como mínimo 100,000 hilos. Modifique el kernel para que imprima el mensaje únicamente si es el ID global máximo)

*Imagen 3.1: Nueva configuración*



```
30 //      blockIdx.x ) * blockDim.x +
31 //      threadIdx.x;
32
33 int maxThread = gridDim.x * gridDim.y * gridDim.z * blockDim.x * blockDim.y * blockDim.z;
34 int maxID = maxThread - 1;
35
36 if (myID == maxID)
37 {
38     printf ("Erick Guerra - 21781. MAX ID: %i\n", myID);
39 }
40
41
42 int main ()
43 {
44     dim3 g (10,10,10);
45     dim3 b (100,10);
46     hello <<< g, b >>> ();
47     cudaThreadSynchronize ();
48     //cudaDeviceSynchronize(); //use instead, ^ is deprecated
49     return 0;
50 }
51
```

Erick Guerra - 21781. MAX ID: 999999