

### PRÁCTICA 2 – BLOQUES, MALLAS Y DIMENSIÓN

FECHA: 05/03/2022

GRUPO: 4CDV1

EQUIPO: NETPOWER

Integrantes:

Alcibar Zubillaga Julián
De Luna Ocampo Yanina

#### OBJETIVOS

**Parte 1:** Correr y analizar el programa 02\_gridBlock.cu

**Parte 2:** Revisar y analizar el programa 03\_dim.cu

#### ESCENARIO

Un bloque de subprocesos es una abstracción de programación que representa un grupo de subprocesos que se pueden ejecutar en serie o en paralelo. Para una mejor asignación de procesos y datos, los subprocesos se agrupan en bloques de subprocesos. La cantidad de subprocesos en un bloque pueden ser de hasta 1024. Los subprocesos en el mismo bloque de subprocesos se ejecutan en el mismo flojo del procesador. Los subprocesos en el mismo bloque pueden comunicarse entre sí a través de la memoria compartida.

La combinación de varios bloques forma una malla (grid). Todos los bloques en la misma malla contienen el mismo número de subprocesos. La cantidad de subprocesos en un bloque es limitada, pero las mallas se pueden usar para cálculos que requieren una gran cantidad de bloques de subprocesos para operar en paralelo y usar todos los multiprocesadores disponibles.

En esta práctica deberás correr un par de programas con el fin de que comprendas el uso de bloques, mallas y la dimensión de éstos.

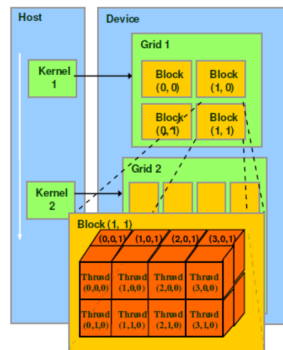
**Nota:** Se utilizará un servidor externo para realizar la práctica. Es necesario contar con un programa cliente ssh y credenciales de acceso al servidor externo.

#### RECURSOS NECESARIOS PARA REALIZAR LA PRÁCTICA

- 1 computadora personal
- Acceso a internet
- 1 programa cliente SSH\* (Putty: <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>)

## INTRODUCCIÓN

La arquitectura de software de CUDA, consta de una cuadrícula (Grid), un bloque de subprocesos (Bloque) y un subproceso (Thread), que equivale a dividir la unidad de computación en la GPU en varias cuadrículas, cada cuadrícula contienen varios de bloques de hilo (65535), cada uno de estos contiene varios hilos (512), su relación es la siguiente [1]:

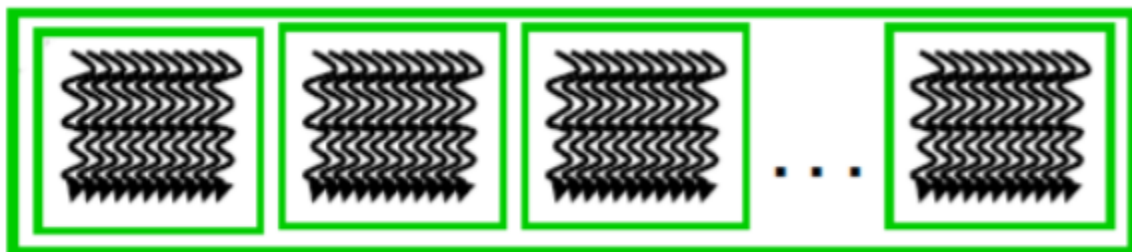


El número de cuadrículas que se pueden crear en esta, está relacionado con la potencia de cálculo de la GPU. En la arquitectura clásica de una tarjeta gráfica presenta dos problemas importantes, por un lado el desequilibrio de carga que aparece entre ambos procesadores y por otro la diferencia entre sus respectivos repertorios de instrucciones. Además de las características generales de ésta, existen otras características que dependen de cada dispositivo en particular. Por ese motivo, es muy importante conocer las características actuales del dispositivo o dispositivos con los que vamos a trabajar, como por ejemplo cuánta memoria y qué capacidad de cómputo posee.

Los hilos (threads), constituyen la base de la ejecución de aplicaciones en paralelo. Cuando se lanza un kernel se crea una malla de hilos donde todos ellos ejecutan el mismo programa. Es decir, el kernel especifica las instrucciones que van a ser ejecutadas por cada hilo individual.

Este es uno de los casos más generales es el lanzamiento de un kernel con M bloques y N hilos por bloque, en cuyo caso tendríamos un total de MN hilos .

Habrán un total de MN hilos ejecutándose en paralelo que para identificarlos será necesario hacer uso conjunto de las dos variables anteriores y de dos nuevas constantes que permitan a la GPU conocer en tiempo de ejecución las dimensiones del kernel que hemos lanzado. Estas dos constantes son: gridDim.x y blockDim.x. La primera nos da el número de bloques (M) y la segunda el número de hilos que tiene cada bloque (N). De este modo, dentro del kernel cada hilo se puede identificar de forma unívoca mediante la expresión:  $\text{int myID} = \text{threadIdx.x} + \text{blockDim.x} * \text{blockIdx.x}$  [2]. La sintaxis es:



### PARTE 1: CORRER Y ANALIZAR EL PROGRAMA 02\_gridBlock.cu

UTILIZA UN CLIENTE SSH PARA CONECTARTE AL SERVIDOR LAMBDA 01.

(TE PUEDES CONECTAR AL SERVIDOR LAMBDA DE LA MISMA FORMA QUE LO HICISTE EN LA PRÁCTICA 01).

Dentro del servidor Lambda. Acceder a la carpeta “hpc\_4AV1

Copiaremos el programa 02\_gridBlock.co a nuestra carpeta de trabajo, ejecuta el siguiente comando:

```
cp ~/hpc_4AV1/unidad01/02_gridBlock.cu ~/hpc_4AV1/estudiantes/<tu_carpeta_de_trabajo>/u01
```

Reemplaza <tu\_carpeta\_de\_trabajo> por el nombre de tu carpeta de trabajo. Este comando copiará el programa “02\_gridBlock.cu” a la subcarpeta u01 de tu carpeta de trabajo. Accede a esa subcarpeta y abre el programa, usa el comando:

```
nano 02_gridBlock.cu
```

Revisa el programa y analízalo.

Compila y ejecuta el programa, usa los siguientes comandos:

```
nvcc 02_gridBlock.cu -o 02_gridBlock.x
```

```
./02_gridBlock.x
```

INCLUYE AQUÍ LA CAPTURA DE PANTALLA CON EL RESULTADO DEL PROGRAMA EJECUTÁNDOSE

#### A) CAPTURAS ALCIBAR ZUBILLAGA JULIÁN



```
GNU nano 4.8 02_gridBlock.cu
#include <cuda_runtime.h>
#include <stdio.h>

/*
Se define la dimensión de un bloque de subprocesos y una malla de bloques desde el CPU
*/

int main(int argc, char **argv)
{
    // Definir el total de elementos
    int nElem = 2048;

    // Definir el tamaño de la malla y del bloque en estructuras
    dim3 block (1024);
    // Se define el tamaño de la malla con la siguiente formula:
    dim3 grid ((nElem + block.x - 1) / block.x);
    // Se imprime el tamaño del bloque y de la malla
    printf("grid.x %d block.x %d \n", grid.x, block.x);

    // Definimos un tamaño de bloque de 512 subprocesos
    block.x = 512;
    grid.x = (nElem + block.x - 1) / block.x;
    printf("grid.x %d block.x %d \n", grid.x, block.x);

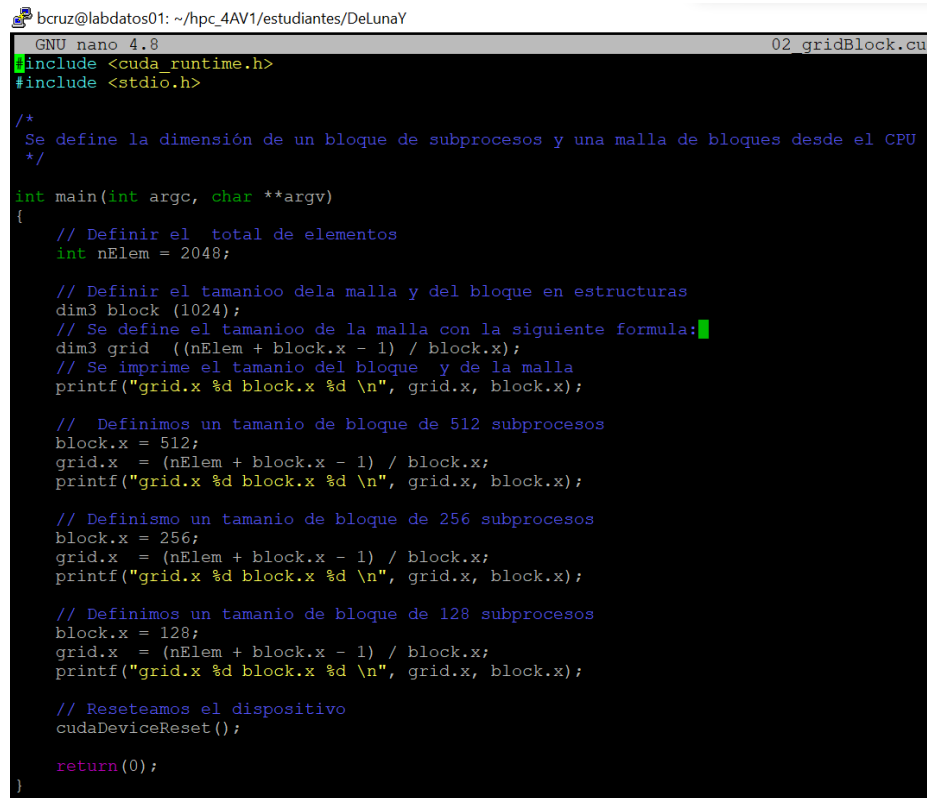
    // Definimos un tamaño de bloque de 256 subprocesos
    block.x = 256;
    grid.x = (nElem + block.x - 1) / block.x;
    printf("grid.x %d block.x %d \n", grid.x, block.x);

    // Definimos un tamaño de bloque de 128 subprocesos
    block.x = 128;
    grid.x = (nElem + block.x - 1) / block.x;
    printf("grid.x %d block.x %d \n", grid.x, block.x);

    // Reseteamos el dispositivo
    cudaDeviceReset();
}
```

Captura de pantalla con la ejecución del programa

## B) CAPTURAS DE LUNA OCAMPO YANINA



```

bcruez@labdatos01: ~/hpc_4AV1/estudiantes/DelunaY
GNU nano 4.8                                02_gridBlock.cu
#include <cuda_runtime.h>
#include <stdio.h>

/*
Se define la dimensión de un bloque de subprocesos y una malla de bloques desde el CPU
*/

int main(int argc, char **argv)
{
    // Definir el total de elementos
    int nElem = 2048;

    // Definir el tamaño de la malla y del bloque en estructuras
    dim3 block(1024);
    // Se define el tamaño de la malla con la siguiente formula:
    dim3 grid ((nElem + block.x - 1) / block.x);
    // Se imprime el tamaño del bloque y de la malla
    printf("grid.x %d block.x %d \n", grid.x, block.x);

    // Definimos un tamaño de bloque de 512 subprocesos
    block.x = 512;
    grid.x = (nElem + block.x - 1) / block.x;
    printf("grid.x %d block.x %d \n", grid.x, block.x);

    // Definimos un tamaño de bloque de 256 subprocesos
    block.x = 256;
    grid.x = (nElem + block.x - 1) / block.x;
    printf("grid.x %d block.x %d \n", grid.x, block.x);

    // Definimos un tamaño de bloque de 128 subprocesos
    block.x = 128;
    grid.x = (nElem + block.x - 1) / block.x;
    printf("grid.x %d block.x %d \n", grid.x, block.x);

    // Reseteamos el dispositivo
    cudaDeviceReset();

    return(0);
}

```

Captura de pantalla con la ejecución del programa

## PARTE 1: CORRER Y ANALIZAR EL PROGRAMA 03\_DIM.CU

Copiaremos el programa 03\_dim.cu a nuestra carpeta de trabajo, ejecuta el siguiente comando:

```
cp ~/hpc_4AV1/unidad01/03_dim.cu ~/hpc_4AV1/estudiantes/<tu_carpeta_de_trabajo>/u01
```

Reemplaza <tu\_carpeta\_de\_trabajo> por el nombre de tu carpeta de trabajo. Este comando copiará el programa "03\_dim.cu" a la subcarpeta u01 de tu carpeta de trabajo. Accede a esa subcarpeta y abre el programa, usa el comando:

```
nano 03_dim.cu
```

Revisa el programa y analízalo.

Compila y ejecuta el programa, usa los siguientes comandos:

```
nvcc 03_dim.cu -o 03_dim.x
```

```
./03_dim.x
```

INCLUYE AQUÍ LA CAPTURA DE PANTALLA CON EL RESULTADO DEL PROGRAMA EJECUTÁNDOSE

## c) CAPTURAS ALCIBAR ZUBILLAGA JULIÁN

```

bacruz@labdatos01: ~/hpc_4AV1/estudiantes/Alcibar/
GNU nano 4.8                                03_dim.cu
#include <cuda_runtime.h>
#include <stdio.h>

/*
 * Muestra la dimensiion de un bloque de subprocessos y mallas desde el host y
 * el dispositivo.
 */

__global__ void checkIndex(void)
{
    printf("threadIdx: (%d, %d, %d)\n", threadIdx.x, threadIdx.y, threadIdx.z);
    printf("blockIdx: (%d, %d, %d) ", blockIdx.x, blockIdx.y, blockIdx.z);

    printf("\nblockDim: (%d, %d, %d)", blockDim.x, blockDim.y, blockDim.z);
    if (threadIdx.x == 0 && blockIdx.x == 0)
        printf("\ngridDim: (%d, %d, %d)\n", gridDim.x, gridDim.y, gridDim.z);
}

int main(int argc, char **argv)
{
    // Definimos el numero total de elementos
    int nElem = 7;

    // Definimos las estructuras del bloque y la malla
    dim3 block(3,1,2);
    dim3 grid((nElem + block.x - 1) / block.x);

    // Revisamos los tamanos del bloque y la malla desde el CPU
    printf("grid.x = %d grid.y = %d grid.z = %d\n", grid.x, grid.y, grid.z);
    printf("block.x = %d block.y = %d block.z = %d\n", block.x, block.y, block.z);

    // Revisamos los tamanos del bloque y la malla desde el dispositivo
    checkIndex<<<grid, block>>>();

    // Resetear el dispositivo antes de terminar
    cudaDeviceReset();

    return(0);
}

```

Captura de pantalla con la ejecución del programa

## d) CAPTURAS DE LUNA OCAMPO YANINA

```

bacruz@labdatos01: ~/hpc_4AV1/estudiantes/DeLunaY
GNU nano 4.8                                03_dim.cu
#include <cuda_runtime.h>
#include <stdio.h>

/*
 * Muestra la dimensiion de un bloque de subprocessos y mallas desde el host y
 * el dispositivo.
 */

__global__ void checkIndex(void)
{
    printf("threadIdx: (%d, %d, %d)\n", threadIdx.x, threadIdx.y, threadIdx.z);
    printf("blockIdx: (%d, %d, %d) ", blockIdx.x, blockIdx.y, blockIdx.z);

    printf("\nblockDim: (%d, %d, %d)", blockDim.x, blockDim.y, blockDim.z);
    if (threadIdx.x == 0 && blockIdx.x == 0)
        printf("\ngridDim: (%d, %d, %d)\n", gridDim.x, gridDim.y, gridDim.z);
}

int main(int argc, char **argv)
{
    // Definimos el numero total de elementos
    int nElem = 7;

    // Definimos las estructuras del bloque y la malla
    dim3 block(3,1,1);
    dim3 grid((nElem + block.x - 1) / block.x);

    // Revisamos los tamanos del bloque y la malla desde el CPU
    printf("grid.x = %d grid.y = %d grid.z = %d\n", grid.x, grid.y, grid.z);
    printf("block.x = %d block.y = %d block.z = %d\n", block.x, block.y, block.z);

    // Revisamos los tamanos del bloque y la malla desde el dispositivo
    checkIndex<<<grid, block>>>();

    // Resetear el dispositivo antes de terminar
    cudaDeviceReset();

    return(0);
}

```

Captura de pantalla con la ejecución del programa

### PARTE 3: CUESTIONARIO

#### EN EL CASO DE ESTE PROGRAMA EN PARTICULAR, ¿QUÉ SE EJECUTÓ EN PARALELO?

EN LA CPU SE REALIZA COPROCESAMIENTO REPARTIDO ENTRE LA CPU Y LA GPU. SE COPIAN LOS DATOS DE LA MEMORIA PRINCIPAL A LA MEMORIA DE LA GPU, EL CPU ENCARGA EL PROCESO A LA GPU, EL GPU LO EJECUTA EN PARALELO EN CADA NÚCLEO Y SE COPIA EL RESULTADO DE LA MEMORIA DE LA GPU A LA MEMORIA PRINCIPAL.

EL ELEMENTO QUE SE LE ASIGNE, LO MANDA AL GRID Y AL BLOCK, ACOPLANDO EL MÚLTIPLO INMEDIATO A LA IMPRESIÓN, IMPRIME LA CANTIDAD DE ELEMENTOS DADOS Y DEPENDIENDO DE LA CANTIDAD DADA LOS THREADS QUE SE NECESITAN SE PONEN EMPEZANDO DESDE 0 Y EN EL BLOCK, SE IMPRIME LA CANTIDAD DE ELEMENTOS PUESTA EN LA DIMENSIÓN DADA. COMO EN LA SIGUIENTE IMAGEN:

```
(base) boruz@labdatos01:~/hpc_4AV1/estudiantes/AlcibarJS ./03_dim.x
grid.x = 3 grid.y = 1 grid.z = 1
block.x = 3 block.y = 1 block.z = 1
threadIdx: (0, 0, 0)
threadIdx: (1, 0, 0)
threadIdx: (2, 0, 0)
threadIdx: (0, 0, 0)
threadIdx: (1, 0, 0)
threadIdx: (2, 0, 0)
threadIdx: (0, 0, 0)
threadIdx: (1, 0, 0)
threadIdx: (2, 0, 0)
blockIdx: (2, 0, 0) blockIdx: (2, 0, 0) blockIdx: (2, 0, 0) blockIdx: (1, 0, 0) blockIdx: (1, 0, 0) blockIdx: (1, 0, 0) blockIdx: (0, 0, 0) blockIdx: (0, 0, 0) blockIdx: (0, 0, 0)
blockDim: (3, 1, 1)
blockDim: (3, 1, 1)
blockDim: (3, 1, 1)
blockDim: (3, 1, 1)
blockDim: (3, 1, 1)
blockDim: (3, 1, 1)
blockDim: (3, 1, 1)
blockDim: (3, 1, 1)
blockDim: (3, 1, 1)
```

CONFORME VAYAMOS CAMBIANDO LOS DATOS, LAS IMPRESIONES VAN ACOPLÁNDOSE Y CAMBIANDO CON LO QUE SE VA PONIENDO.

### CONCLUSIONES

Estos se encargan de tareas gráficas, asistiendo a la CPU. Este tiene diferentes aplicaciones, conforme la tecnología avanza tenemos diferentes cambios importantes y esto no se queda atrás, como imágenes médicas, dinámicas de fluidos computacionales, ciencia ambiental, entre muchos otros.

La tecnología CUDA nos permite desarrollar avances en muchos campos de la ciencia, llevando consigo, una nueva manera para desarrollar aplicaciones de software más eficiente, en donde las operaciones de cálculo científico intensivo que anteriormente tomaba meses y años, se desarrollan en mucho menos tiempo. Permite aprovechar todo el hardware disponible, usando los componentes correctos para ejecutar las tareas de cálculo intensivo que se desean llevar a cabo.

La escalabilidad de esta, proporciona una ventaja en el desarrollo de aplicaciones e impide que el hardware quede rezagado al ejecutar los múltiples hilos de ejecución, proporcionando un alto grado de operabilidad, vital para los investigadores, sin perder calidad.

### BIBLIOGRAFÍA

1. [CUDA] *Relaciones de organización de cuadrículas, bloques y subprocesos y la fórmula de cálculo del índice de subprocesos - programador clic.* (2020). programador clic. <https://programmerclick.com/article/18961834297/>
2. RIUBU Principal. [https://riubu.ubu.es/bitstream/handle/10259/3933/Programacion\\_en\\_CUDA.pdf;jsessionid=0258C8C85F9E17C09F569CEC5C6DE065?sequence=1](https://riubu.ubu.es/bitstream/handle/10259/3933/Programacion_en_CUDA.pdf;jsessionid=0258C8C85F9E17C09F569CEC5C6DE065?sequence=1) (accedido el 6 de marzo de 2022).

### CONSIDERACIONES FINALES

Descarga el documento antes de llenarlo.

Este documento se debe llenar en equipo, aunque la práctica la deben hacer TODOS los integrantes de este.

Después de llenar el documento, guárdalo como PDF y envíalo a través de la plataforma TEAMS, en la pestaña de tareas correspondiente. Solamente lo tiene que subir uno de los integrantes. Pero deben incluir TODOS los nombres de los integrantes del equipo en la primera página.

Queda estrictamente prohibido cualquier tipo de plagio a otros equipos o grupos. En caso de que ocurra, se anulará la práctica y se descontarán dos puntos a los equipos involucrados.