UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Facultad de Ingeniería

Introducción a CUDA C

Laboratorio de Intel para la Academia y Cómputo de Alto Desempeño

Elaboran: Ariel Ulloa Trejo
Fernando Pérez Franco
Revisión: Ing. Laura Sandoval Montaño

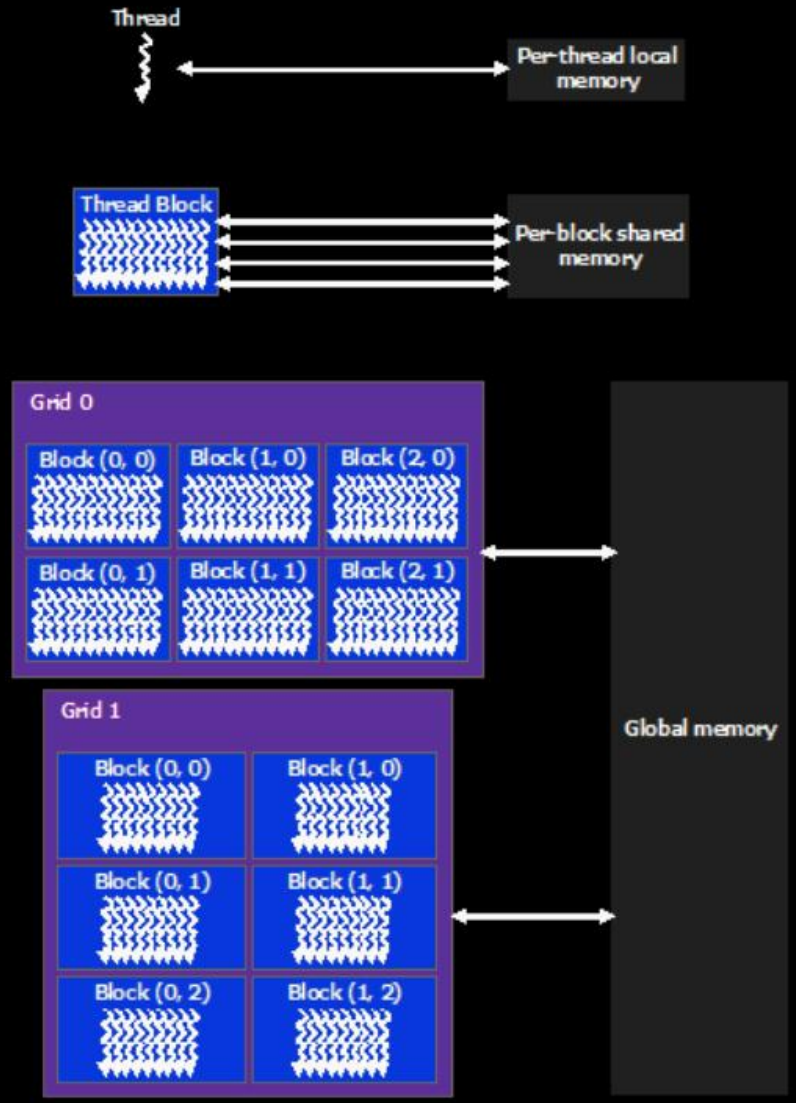
Temario

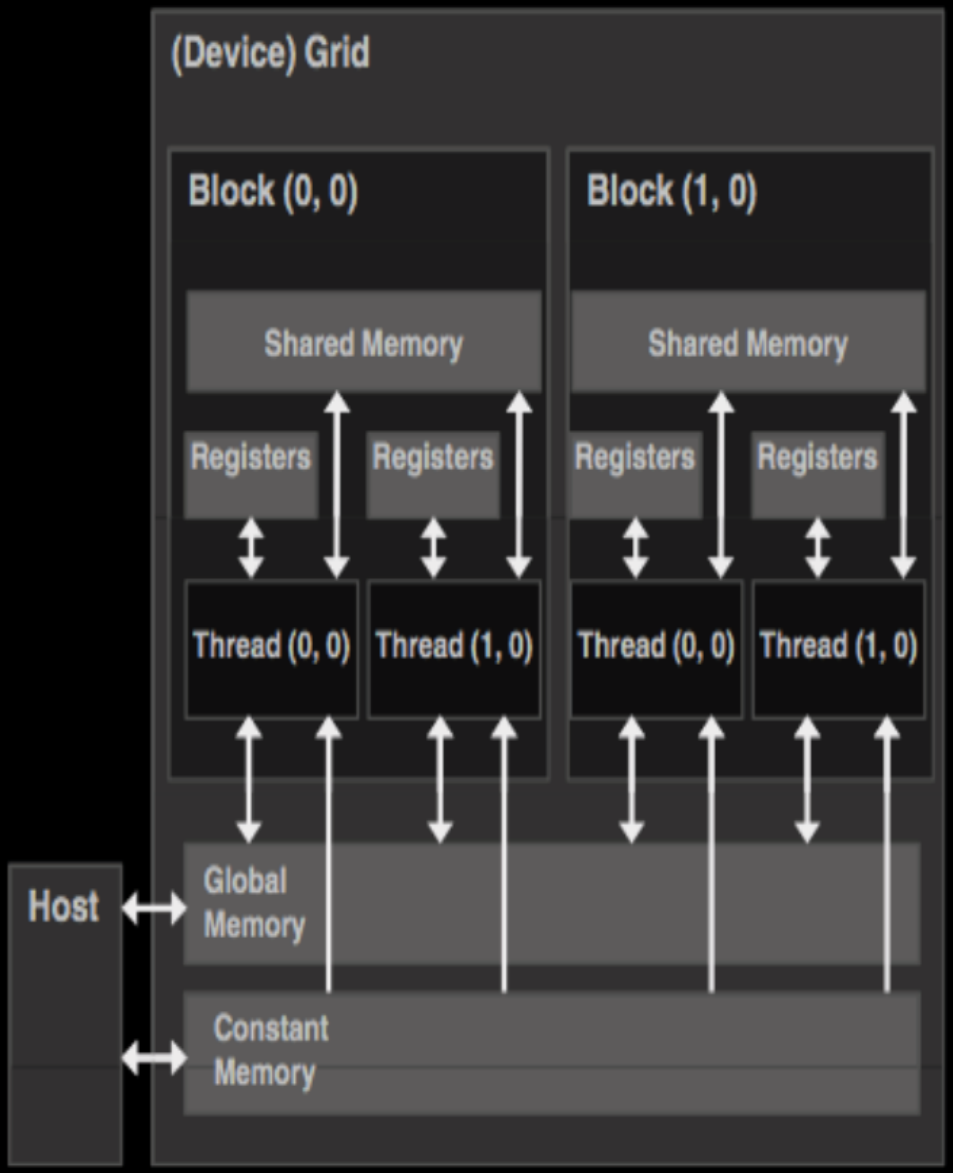
1. Antecedentes
2. El GPU
3. Funciones y vectores
4. Manejo de matrices
5. Memoria compartida

3 Funciones

Jerarquía de la memoria

- Cada hilo tiene memoria local privada.
- Cada bloque tiene memoria compartida visible a todos los hilos del mismo.
- Todos los hilos tienen acceso a la misma memoria global.



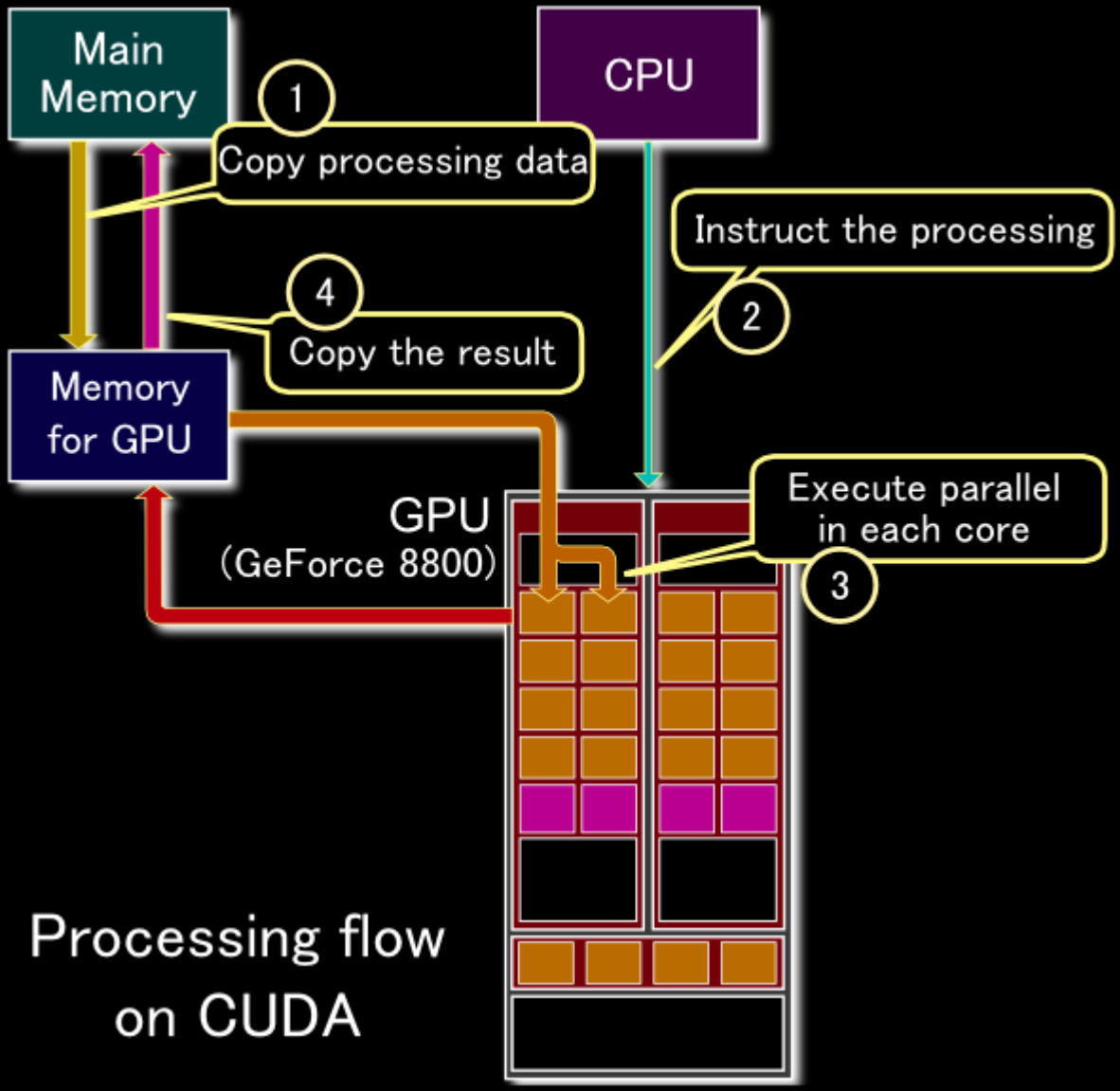


El dispositivo puede:

- L/E registros por hilo.
- L/E memoria local por hilo.
- L/E memoria compartida por bloque.
- L/E memoria global por grid.
- Leer memoria constante por grid.

El host puede:

- Transferir datos a la memoria global y constante y viceversa.


Main Memory
CPU
1
Copy processing data
Instruct the processing
2
4
Copy the result
Memory for GPU
GPU
(GeForce 8800)
Execute parallel in each core
3
Processing flow on CUDA

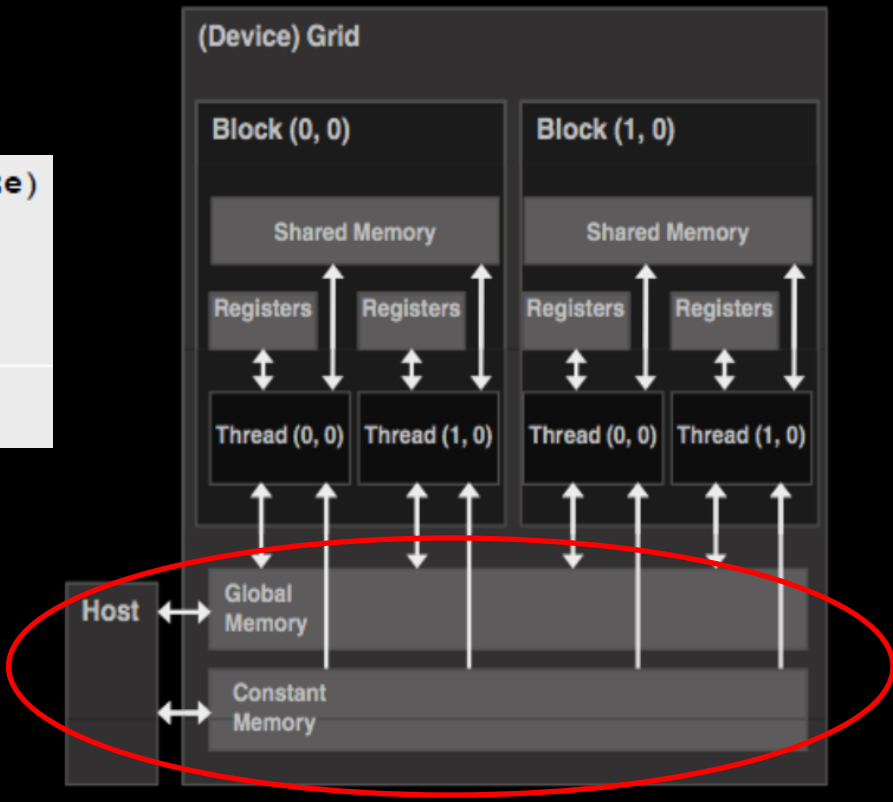
cudaMalloc()

- Además de la función kernel, es necesario asignar memoria en el dispositivo.

cudaFree()

- Es necesario liberar la memoria que ya no se necesita.

```
cudaError_t   cudaMalloc(void** devPtr, size_t size)
where:
devPtr - Pointer to allocated device memory
size – Requested allocation size in bytes
cudaError_t cudaFree (void * devPtr)
```

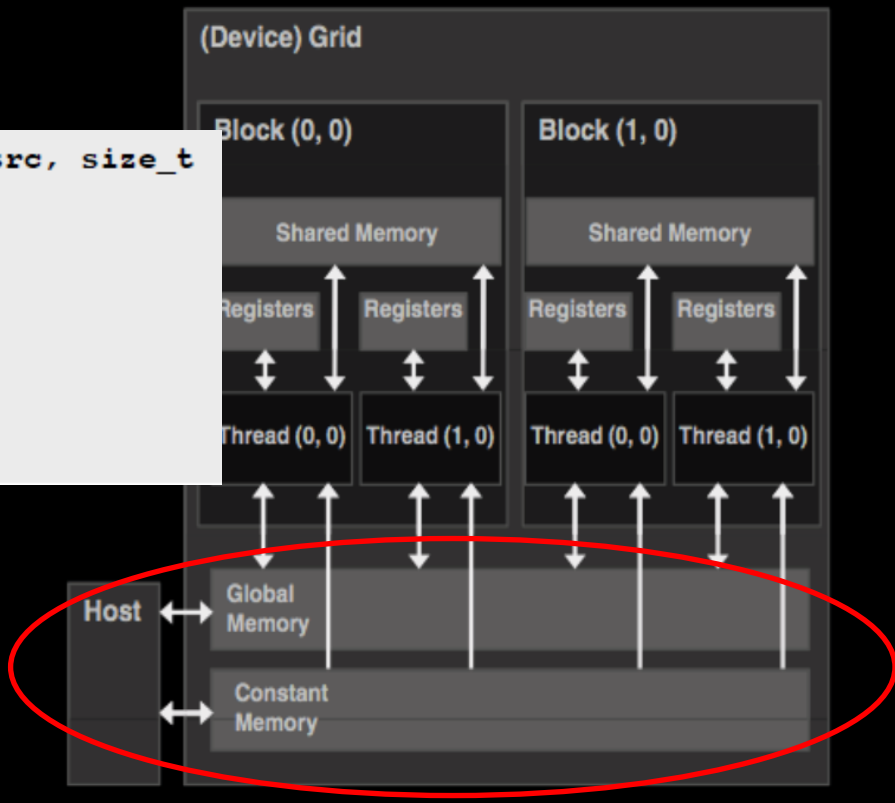


cudaMemcpy()

- Finalmente, para la copia de una memoria a otra:

```
cudaError_t  cudaMemcpy( void * dst, const void * src, size_t
count, enumcudaMemcpyKind kind)
where:
dst - Destination memory address
src – Source memory address
count – Size in bytes to copy
kind – Type of transfer (cudamemcpyHostToDevice,
cudamemcpyDeviceToHost,)
```

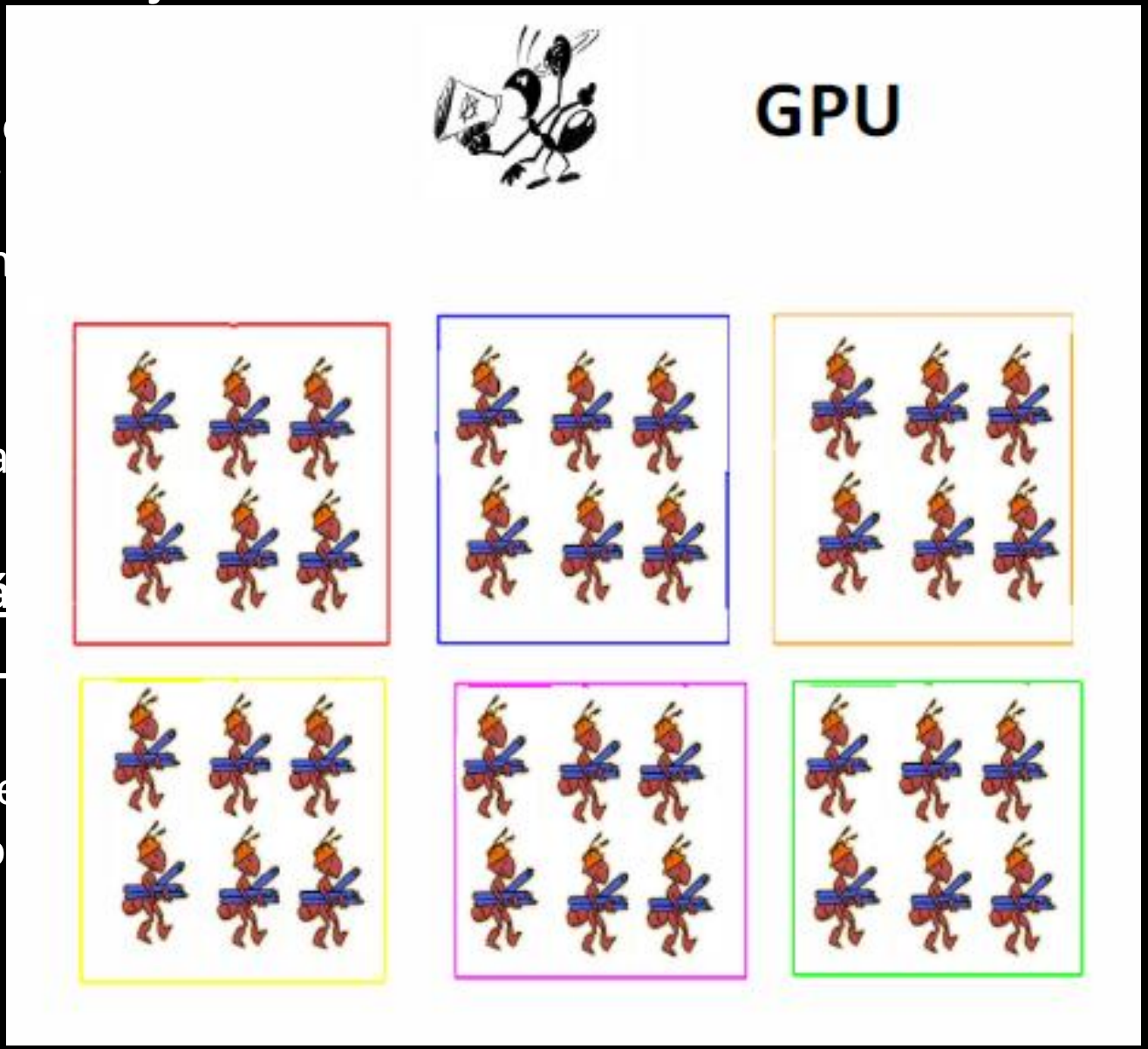
- La transferencia es asíncrona (el programa continúa antes de que la copia sea finalizada)



01_funciones_01.cu

Configuración de la ejecución:

- La ejecución
  a través de hi
- Cada hilo tien
  y registros.
- La cantidad
  pueden crea
  hardware.
- **Un grid está**
  bloques, y a
  hilos.
- El objetivo de
  es usar hilo
  paralelo para
  rendimiento.



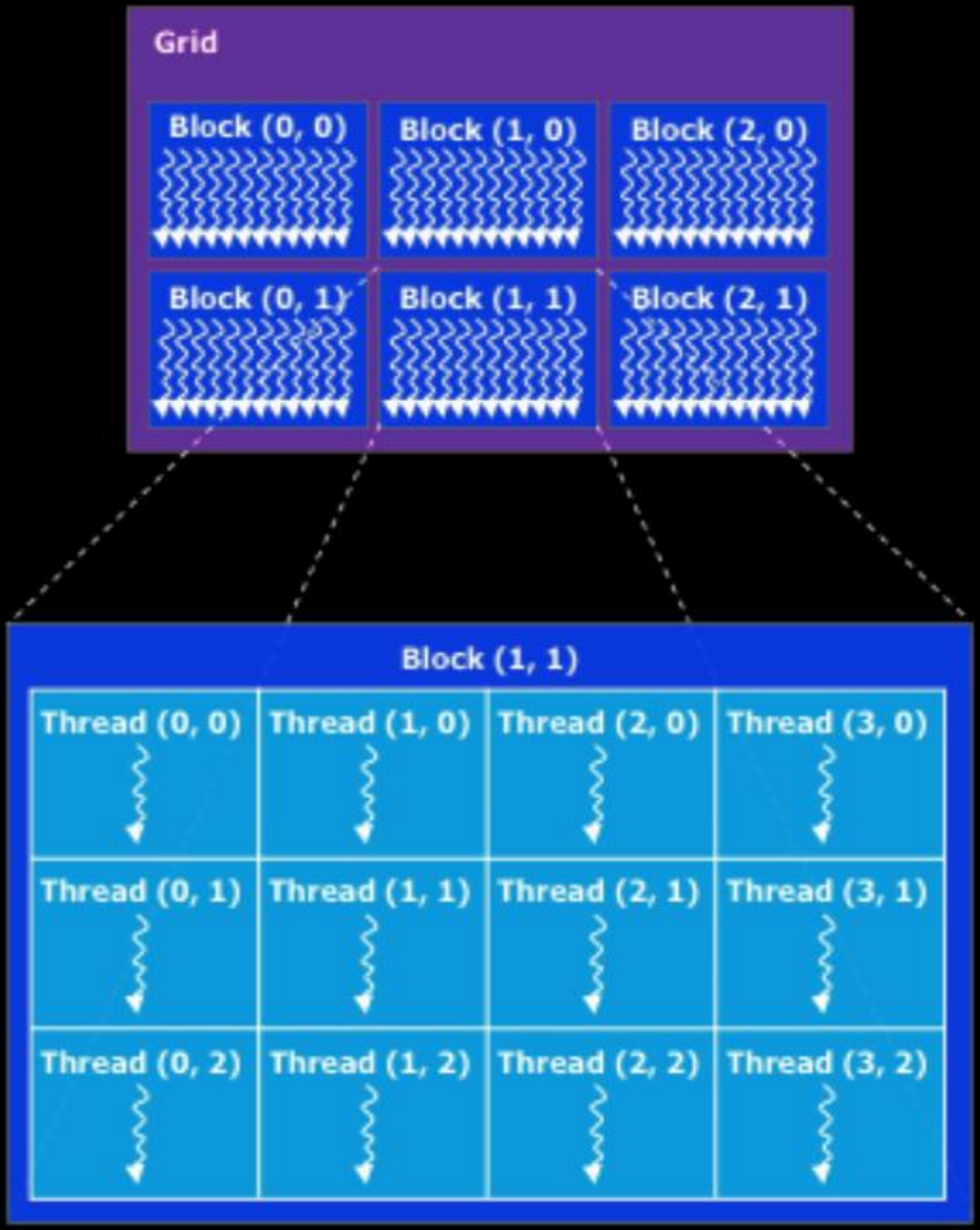
Índices:

Hilos:

- Todos los hilos tienen índices con el fin de calcular las direcciones de memoria y tomas decisiones de control.
- Los hilos dentro del bloque tienen variables para identificarse, ya sea en 1, 2 ó 3 dimensiones (lo que proporciona la facilidad para el manejo de vectores, matrices o volúmenes).

Bloques:

- Tal como los hilos, los bloques (dentro del grid) también tienen variables en 1, 2 ó 3 dimensiones.
- El kernel es copiado en todos los bloques “invocados”.
- El número total de hilos es la cantidad de hilos de cada bloque por la cantidad de bloques.


Grid
Block (0, 0)
Block (1, 0)
Block (2, 0)
Block (0, 1)
Block (1, 1)
Block (2, 1)
Block (1, 1)
Thread (0, 0)
Thread (1, 0)
Thread (2, 0)
Thread (3, 0)
Thread (0, 1)
Thread (1, 1)
Thread (2, 1)
Thread (3, 1)
Thread (0, 2)
Thread (1, 2)
Thread (2, 2)
Thread (3, 2)

Variables (creadas automáticamente):

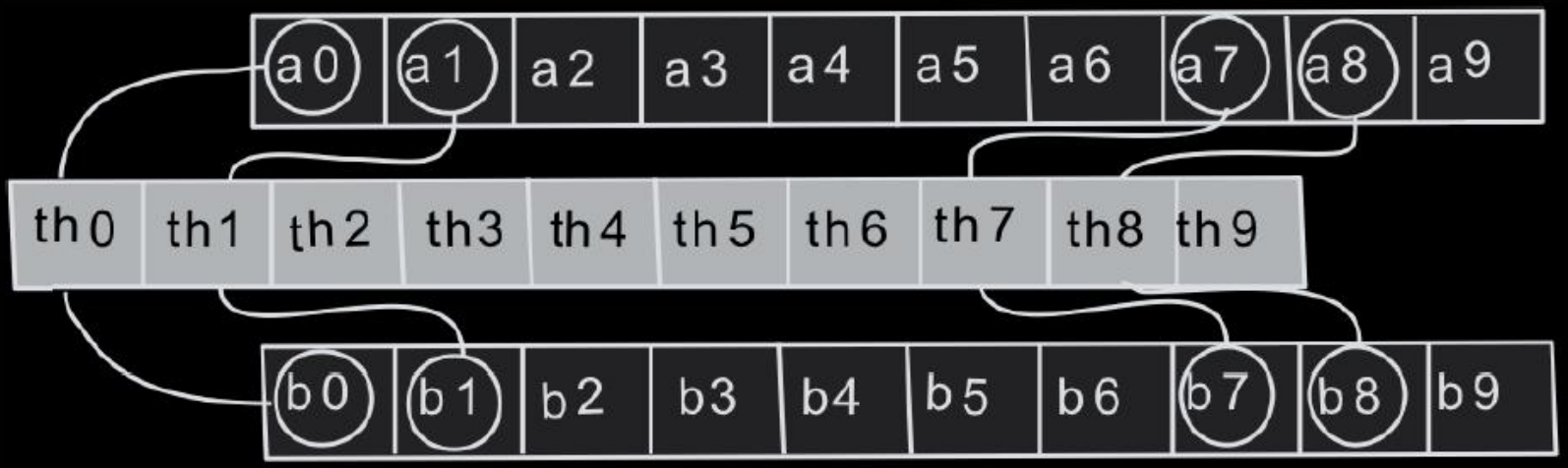
- Índice del hilo en x → threadIdx.x
- Índice del hilo en y → threadIdx.y
- Índice del hilo en z → threadIdx.z

- Índice del bloque en x → blockIdx.x
- Índice del bloque en y → blockIdx.y
- Índice del bloque en z → blockIdx.z

- Número de hilos por bloque en x → blockDim.x
- Número de bloques por grid en x → gridDim.x

Para calcular el ID de un hilo dentro de un bloque:

- Una dimensión: ID = threadIdx.x
- Dos dimensiones: ID = threadIdx.x + blockDim.x * threadIdx.y
- Tres dimensiones: ID = threadIdx.x + blockDim.x * threadIdx.y + blockDim.x * blockDim.y * threadIdx.z



02_sumaVectores_01.cu

Ejercicio:

- Sumar dos vectores de tamaño 100 en el GPU usando 10 bloques y 10 hilos por bloque.

Solución:

- tid = threadIdx.x + blockDim.x * blockIdx.x;

- Cuyo mapeo es:

BLOCK 5	BLOCK 9
tid= 0 + 5* 10 = 50	tid= 0 + 9* 10 = 90
tid= 1 + 5* 10 = 51	tid= 1 + 9* 10 = 91
tid= 2 + 5* 10 = 52	tid= 2 + 9* 10 = 92
tid= 3 + 5* 10 = 53	tid= 3 + 9* 10 = 93
tid= 4 + 5* 10 = 54	tid= 4 + 9* 10 = 94
tid= 5 + 5* 10 = 55	tid= 5 + 9* 10 = 95
tid= 6 + 5* 10 = 56	tid= 6 + 9* 10 = 96
tid= 7 + 5* 10 = 57	tid= 7 + 9* 10 = 97
tid= 8 + 5* 10 = 58	tid= 8 + 9* 10 = 98
tid= 9 + 5* 10 = 59	tid= 9 + 9* 10 = 99

03_sumaVectores_02.cu

Ejercicio:

- Sumar dos vectores de tamaño 100 en el GPU usando 10 bloques y 3 hilos por bloque.

04_sumaVectores_03.cu