Programació paral·lela

Tema 2: CUDA, Host code

De les GPU’s al GPGPU

GPU -> Graphics Processing Unit -> gràfics programables.

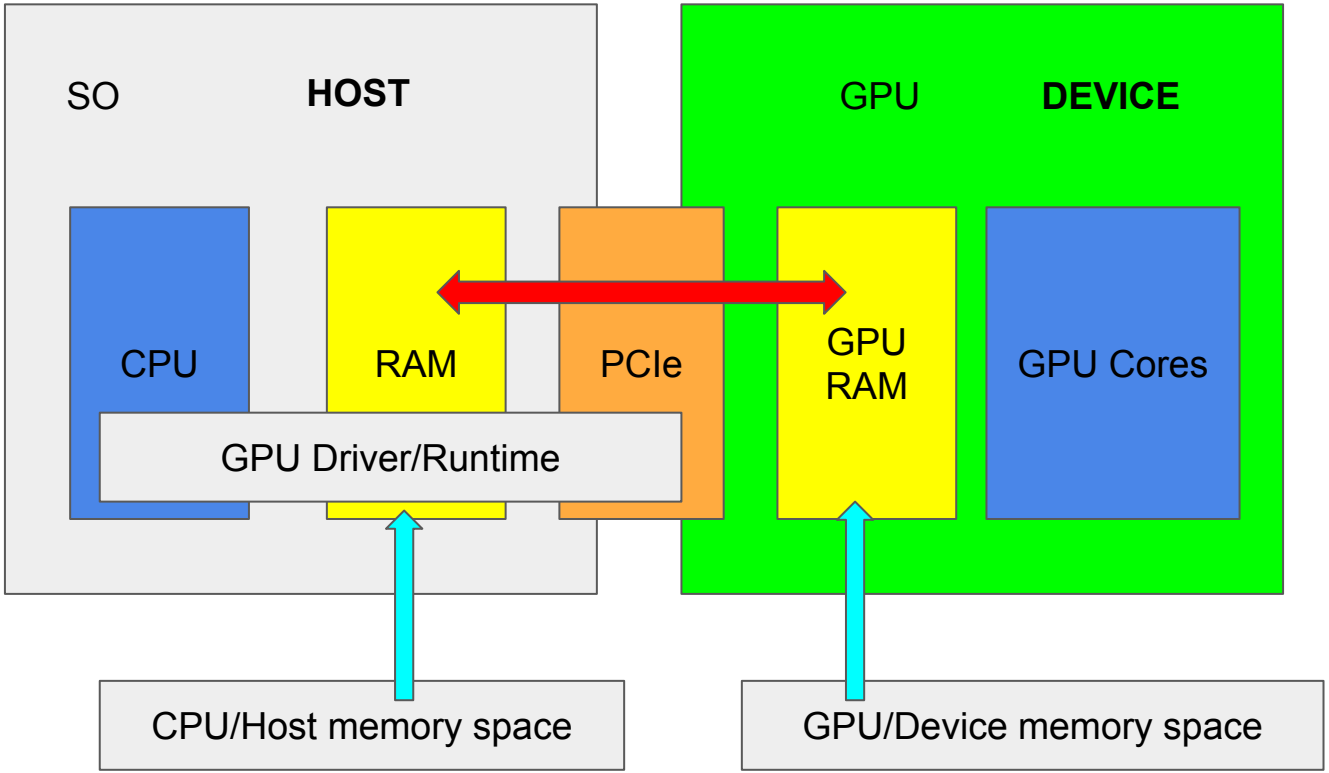
De les GPU’s al GPGPU

GPGPU -> General Purpose GPU -> operacions matricials i vectorials de propòsit general.

De les GPU’s al GPGPU

CUDA -> C Unified Device Architecture

Dos espais de memoria



Interacció CPU - GPU

- A la CPU o la GPU?
  - Com controlem a on s'executa el codi?
  - Com controlem a quina memoria està cada punter?
- Com copiem dades entre la CPU i la GPU?
- Com sincronitzem la CPU i la GPU?

Seqüència típica: codi de Host.

```
cudaMalloc(void **punter, int mida);
```

Dos espais de memoria: exemple

CPU memory space:

```
int* h_a = malloc(30);

h_a[0] = 341234; // Correct
```

GPU memory space:

```
int* d_a;

cudaError_t error;
error = cudaMalloc(&d_a, 30);

d_a[0] = 341234; // Error!!
```

Com utilitzar la memoria de GPU?

Only GPU functions can access GPU memory:

```
__global__ myGPUFunction(int* d_a) {
    d_a[0] = 341234; // Correct!!
}
```

Seqüència típica: codi de Host.

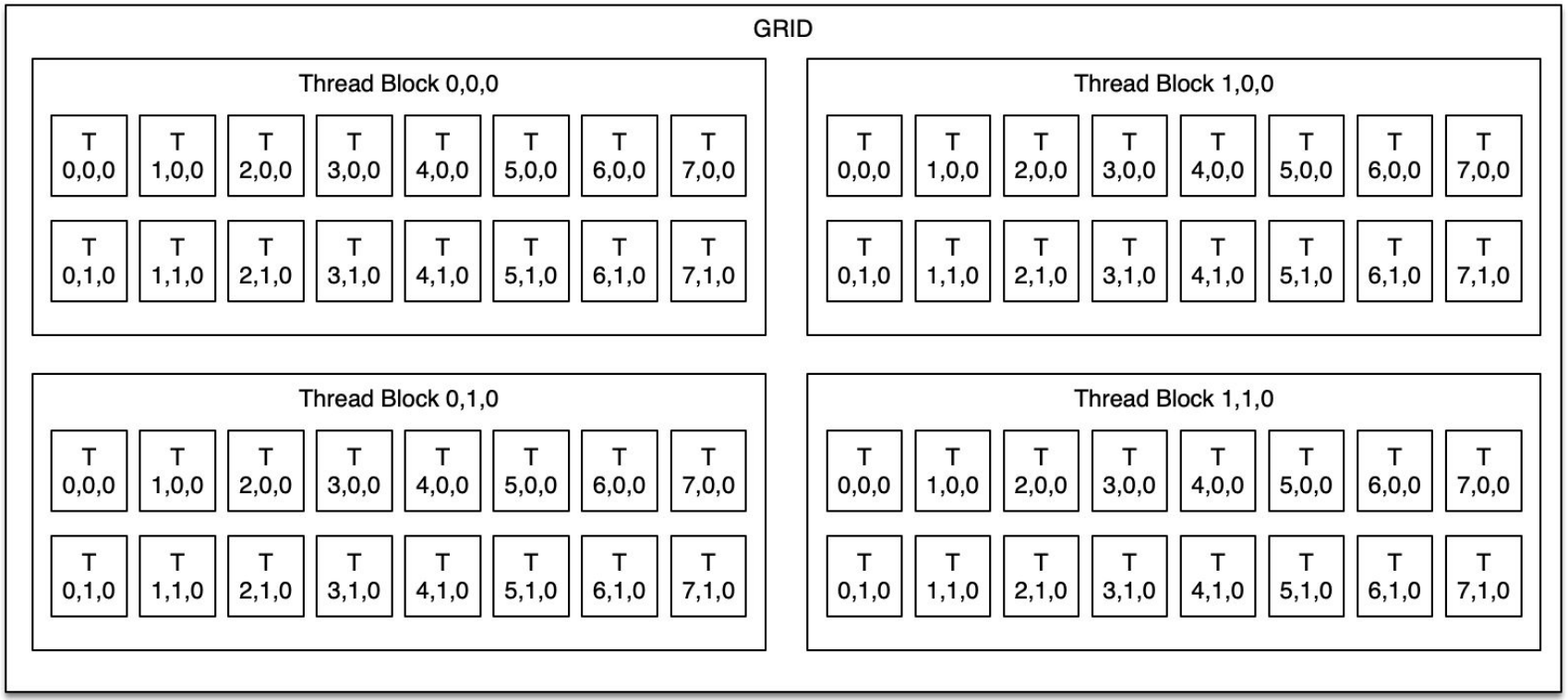
cudaMemcpy(void *punterDest, void* punterOrigen, size_t mida, enum cudaMemcpyKind tipus);

tipus = cudaMemcpyHostToDevice

Seqüència típica: codi de Host.

- *cudaMemcpyHostToHost* (Host -> Host)
- *cudaMemcpyHostToDevice* (Host -> Device)
- *cudaMemcpyDeviceToHost* (Device -> Host)
- *cudaMemcpyDeviceToDevice* (Device -> Device)
- *cudaMemcpyDefault* (Default based unified virtual address space)

CUDA Grid


GRID
Thread Block 0,0,0
T 0,0,0
T 1,0,0
T 2,0,0
T 3,0,0
T 4,0,0
T 5,0,0
T 6,0,0
T 7,0,0
T 0,1,0
T 1,1,0
T 2,1,0
T 3,1,0
T 4,1,0
T 5,1,0
T 6,1,0
T 7,1,0
Thread Block 1,0,0
T 0,0,0
T 1,0,0
T 2,0,0
T 3,0,0
T 4,0,0
T 5,0,0
T 6,0,0
T 7,0,0
T 0,1,0
T 1,1,0
T 2,1,0
T 3,1,0
T 4,1,0
T 5,1,0
T 6,1,0
T 7,1,0
Thread Block 0,1,0
T 0,0,0
T 1,0,0
T 2,0,0
T 3,0,0
T 4,0,0
T 5,0,0
T 6,0,0
T 7,0,0
T 0,1,0
T 1,1,0
T 2,1,0
T 3,1,0
T 4,1,0
T 5,1,0
T 6,1,0
T 7,1,0
Thread Block 1,1,0
T 0,0,0
T 1,0,0
T 2,0,0
T 3,0,0
T 4,0,0
T 5,0,0
T 6,0,0
T 7,0,0
T 0,1,0
T 1,1,0
T 2,1,0
T 3,1,0
T 4,1,0
T 5,1,0
T 6,1,0
T 7,1,0

Seqüència típica: codi de Host.

```
// Per desar la configuració del Grid i els Threadblocks

struct dim3 { int x; int y; int z; };

// Per exemple

dim3 grid(128, 1, 1); // 128 thread blocks en l'eix de les x

dim3 block(256, 2, 1); // 256 threads per block en l'eix de les x i 2 en el de les y
```

Seqüència típica: codi de Host.

```
my_kernel<<< grid, block, shared_mem_size, stream >>>(params...);
```

En aquest cas, stream = 0.

Seqüència típica: codi de Host.

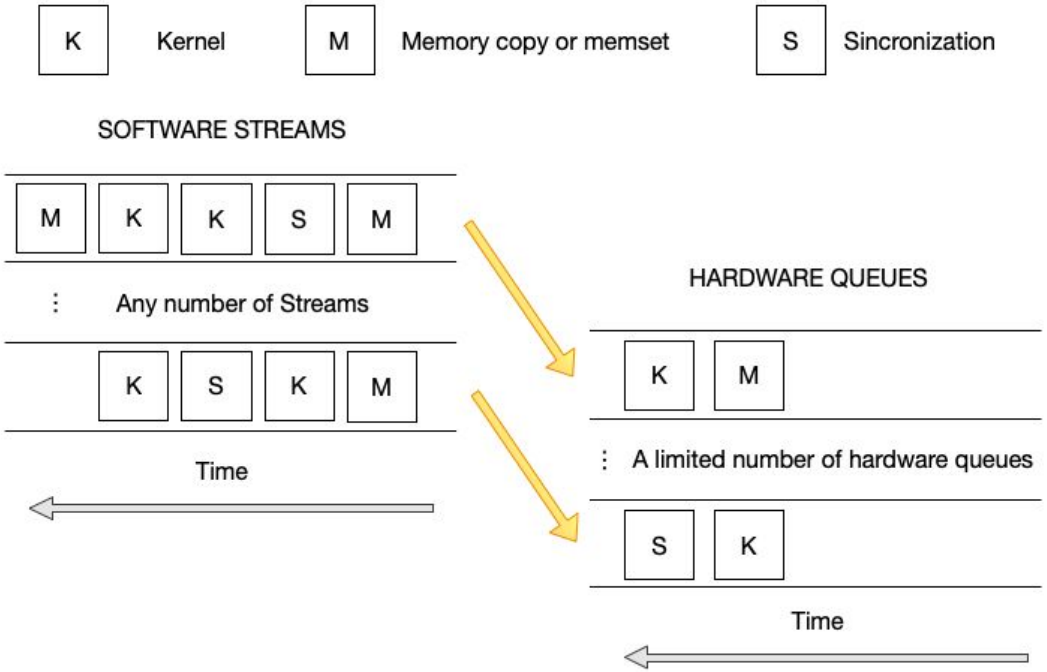
cudaMemcpy(void *punterDest, void* punterOrigen, size_t mida, enum cudaMemcpyKind tipus);

tipus = cudaMemcpyDeviceToHost

Asincronisme: Streams

```
cudaStream_t stream;
cudaStreamCreate(&stream);
cudaMemcpyAsync(... , cudaMemcpyHostToDevice, stream);
my_kernel<<< …, stream >>> (...);
cudaMemcpyAsync(... , cudaMemcpyDeviceToHost, stream);
cudaStreamSynchronize(stream);
```

Streams i cues de hardware



Asincronisme: sincronització

```
cudaDeviceSynchronize();
```

Asincronisme: sincronització

```
cudaStreamSynchronize(cudaStream_t stream);
```

Asincronisme: events

```
cudaEvent_t event;

cudaEventCreate(&event);
```

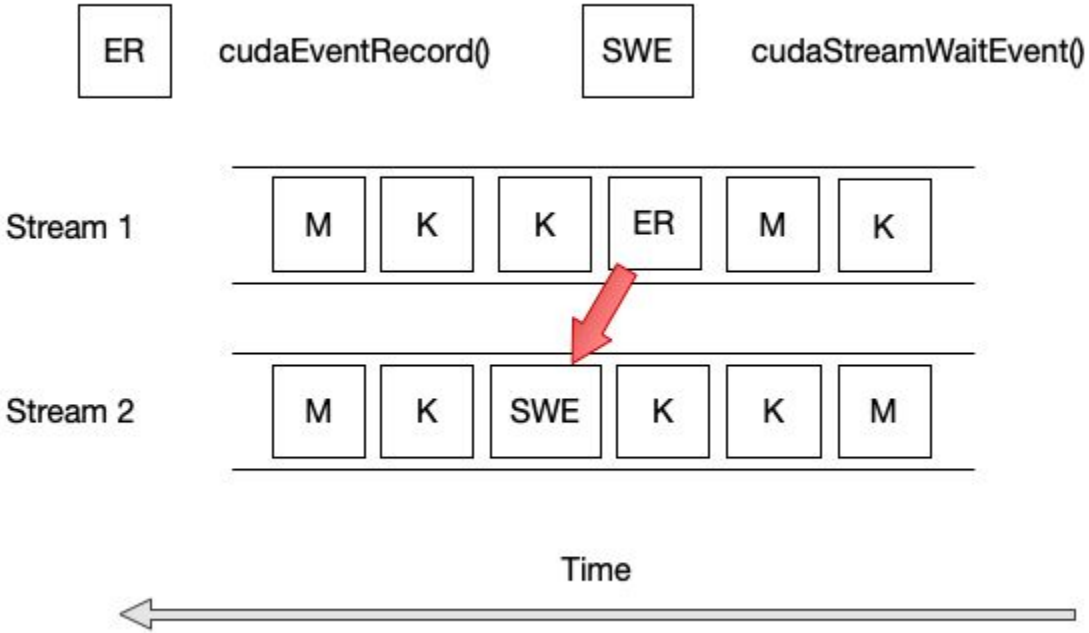
Asincronisme: sincronització amb events

```
cudaEventSynchronize(cudaEvent_t event);
```

Asincronisme: sincronització amb events

cudaStreamWaitEvent(cudaStream_t stream, cudaEvent_t event, unsigned int flags);

Asincronisme: sincronització amb events



Multithreading i CUDA

```
cudaDeviceSynchronize();
```

Multithreading i CUDA

```
cudaStreamSynchronize(cudaStream_t stream);
```

Multithreading i CUDA

```
cudaEventSynchronize(cudaEvent_t event);
```

Ecosistema de llibreries

https://developer.nvidia.com/gpu-accelerated-libraries