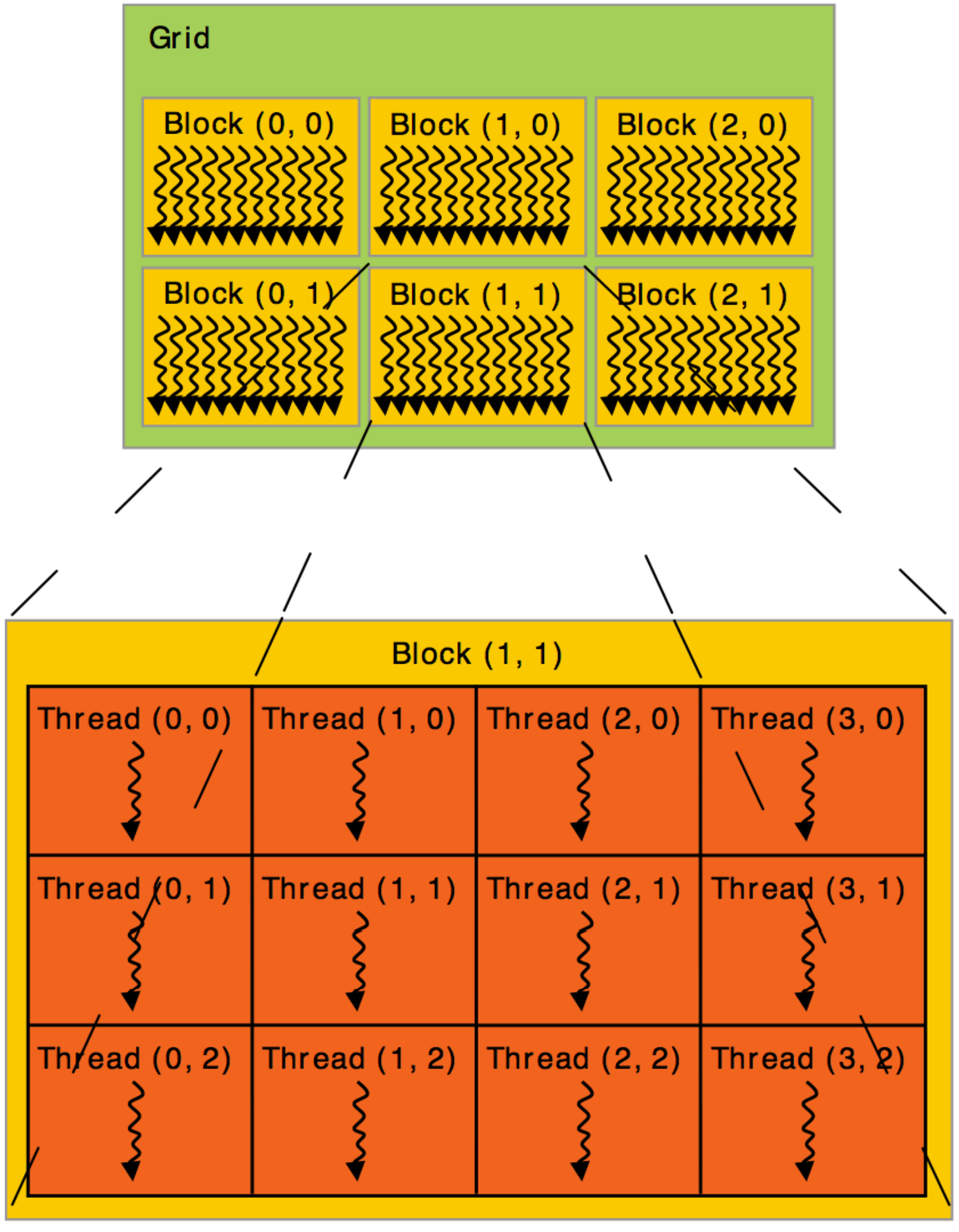
# Writing your own device functions

- Kinds of functions :
  - ✳ The one that you call from host (CPU) are called *kernels* and have prefix __global__
    - ✤ Always return void
  - ✳ The ones that you call form kernels : __device__
  - ✳ The general ones that you could call both form device and host : __host__ __device__
  - ✳ Except for dynamic parallelism, kernel cannot launch a kernel

- Example of a kernel
  - ✳ __global__ void myKernel (int arg1, float *arg2) (no ellipsis)
- Launch with
  - ✳ myKernel<<<nBlocks, nThreadsPerBlock>>>(arg1, arg2, …)
    - ✳ Use dim3 type for grid dimension and block dimension for 2D or 3D blocks
- Know the index of the working thread!
  - ✳ Use threadIdx.x, blockIdx.x, blockDim.x, gridDim.x (also y and z)

Example : two dimensional grid of two dimensional blocks



# Hello World

- KERNELS / hello_world.cu

# Computes operations on an array

- Allocate memory on device if not already done
  - ✳ use cudaMalloc(&pointer, memSize) on a *pointer to pointer* to alloc heap memory on device
- cudaMemSet(pointer, value, memSize)

- Array source :
  - ✳ CPU vector copied to device
    - ✤ with cudaMemcpy(destination_pointer, source_pointer, memSize, direction_of_the_copy)
    - ✤ Utilisation « easier » since 6.0
  - ✳ thrust raw_pointer

# Exercice 1

- Create two sequences on the CPU, S1 ranging from 1 to 100 with step 1, S2 ranging form 1 to 100 with step 2

- Transfer the arrays to device

- Compute the sum of  both arrays in a 3rd vector, using a blockSize of 32

- Don’t forget CUDA_CHECK and CUDA_CHECK_ERROR !