Advanced architecture Embedded processor

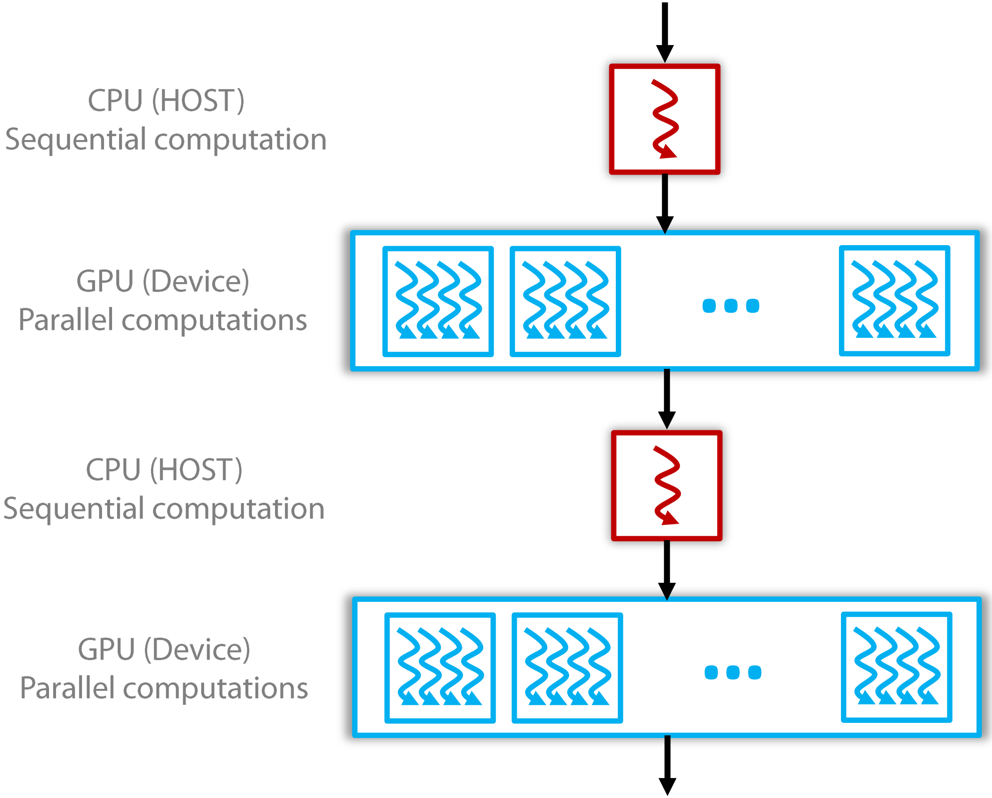
Introduction to CUDA

Bordeaux INP
ENSEIRB MATMECA

J. CRENNE – 2015/2016

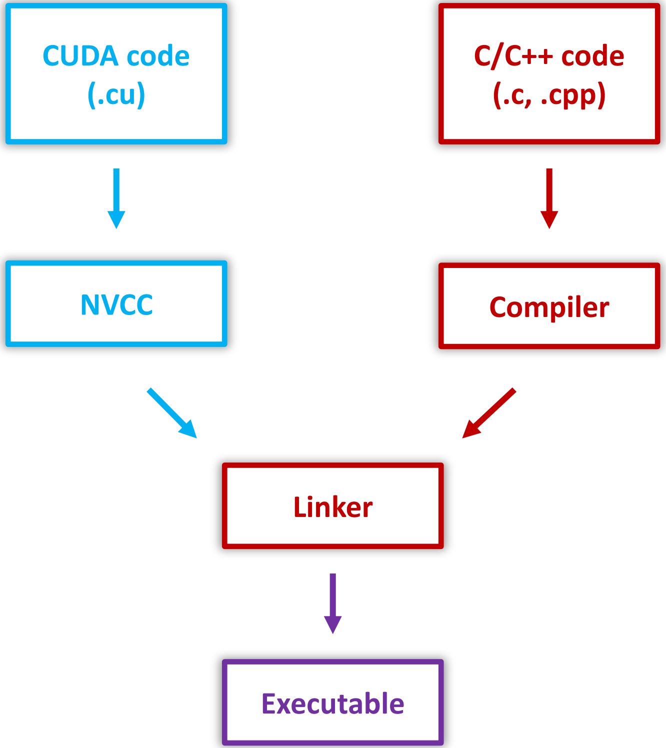
CUDA programming model

Computations execution



CUDA programming model

From code to executable



From GPU to thread

Some definitions

❑ The CPU (host) executes sequential functions and launches parallel computations on the GPU (device)

❑ Bridges (host/device) allow data sharing between the CPU and the GPU

❑ The GPU executes functions (kernels) using parallel instances (threads)

❑ These instances are organized in grids of blocks, and blocks of threads

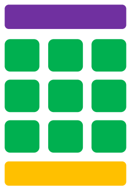
❑ Threads can only access to the GPU memory

From thread to GPU

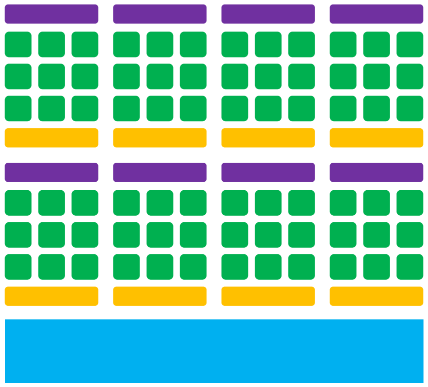
Hardware / Software mapping

Scalar Processor (SP)
Core

Streaming
Multiprocessor
(SM)

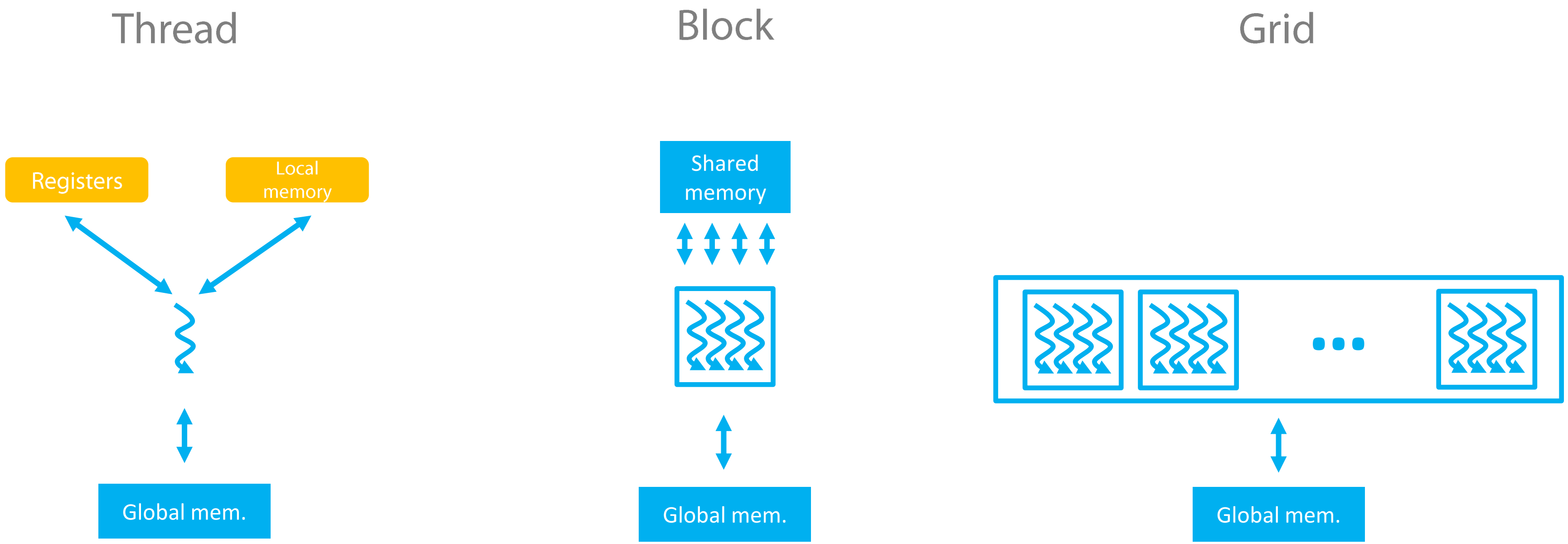


Graphics Processing Unit
(GPU)



From thread to GPU

Hardware / Software mapping



Thread

Registers

Local memory

Global mem.

Block

Shared memory

Global mem.

Grid

Global mem.

Threads (1/2)

Hints

- ❑ A thread is executed on a single SP
- ❑ A block is executed on a single SM
  - ❑ Threads are executed concurrently (i.e in parallel) in groups of 32 (warp)
  - ❑ One SM can execute several blocks
  - ❑ Threads from a block can share data using the global or the shared memory and can synchronize efficiently
- ❑ A block is executed on a single GPU
  - ❑ Blocks are executed in parallel or in serial in an undefined order
  - ❑ Threads from different blocks are not able to shared data if they're not alive simultaneously

Threads (2/2)

Hints

- Threads are associated to an ID in a block (1D, 2D or 3D tiling)

- Blocks are associated to an ID in a grid (1D or 2D tiling)

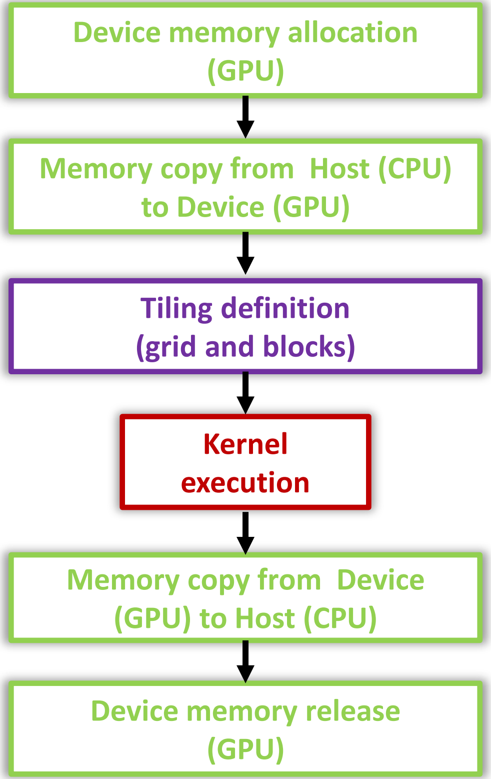
- Built-in C variables allow to access these IDs :

❑ threadIdx.x, threadIdx.y, threadIdx.z	thread identification number
❑ blockDim.x, blockDim.y, blockDim.z	number of threads
❑ blockIdx.x, blockIdx.y	block identification number
❑ dimGrid.x, dimGrid.y	number of blocks

- Choosing the right tiling depends on computation/data matching

GPU computation steps

How to execute a kernel ?


Device memory allocation (GPU)
Memory copy from Host (CPU) to Device (GPU)
Tiling definition (grid and blocks)
Kernel execution
Memory copy from Device (GPU) to Host (CPU)
Device memory release (GPU)

Going further

Fight the future

□ CUDA libraries

□ cuBLAS	(algebra computations)
□ cuFFT	(Fourier’s transforms)
□ cuRAND	(random numbers)
□ cuSPARSE	(sparse matrices)
□ NPP	(image, video and signal processing)

□ Development tools

□ Cuda-Memcheck	(access errors to GPU memory)
□ Cuda-GDB, Nvidia Nsight VS Eclipse editions	(debuggers)
□ Nvidia Visual Profiler	(profiling & optimization)
□ CUDA Occupancy Calculator	

Thanks !