



Mineure CalHau2

# GPU architecture

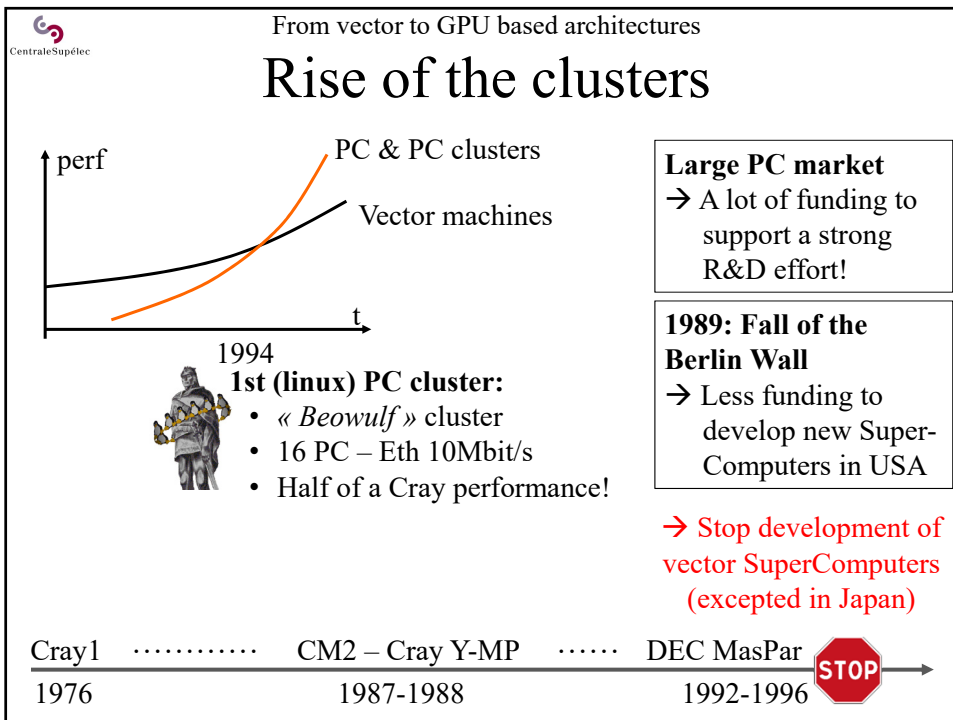
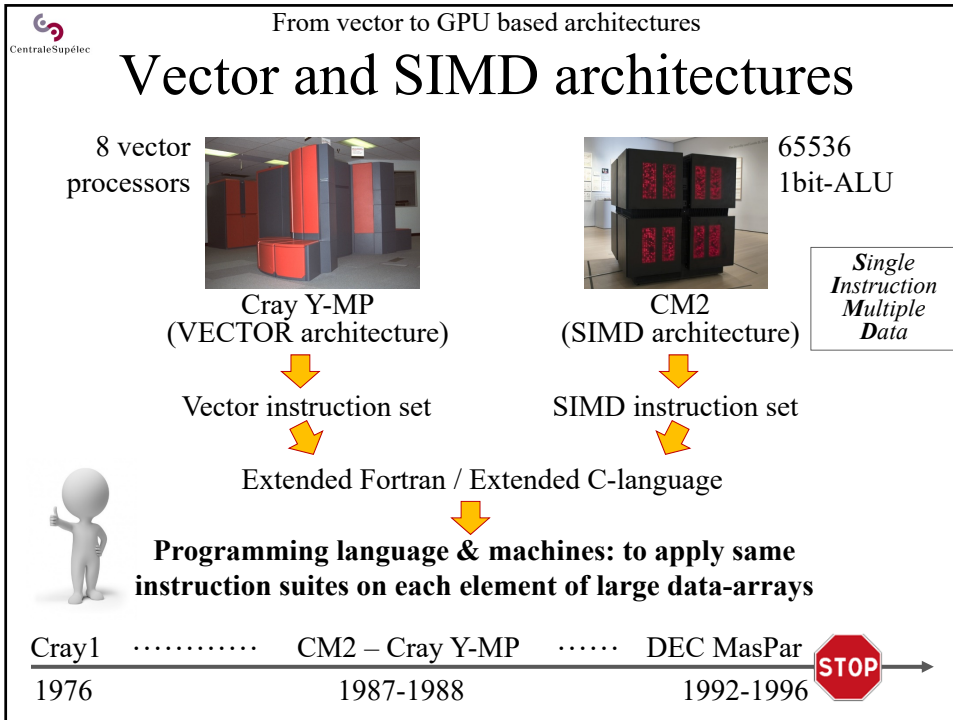
Stéphane Vialle

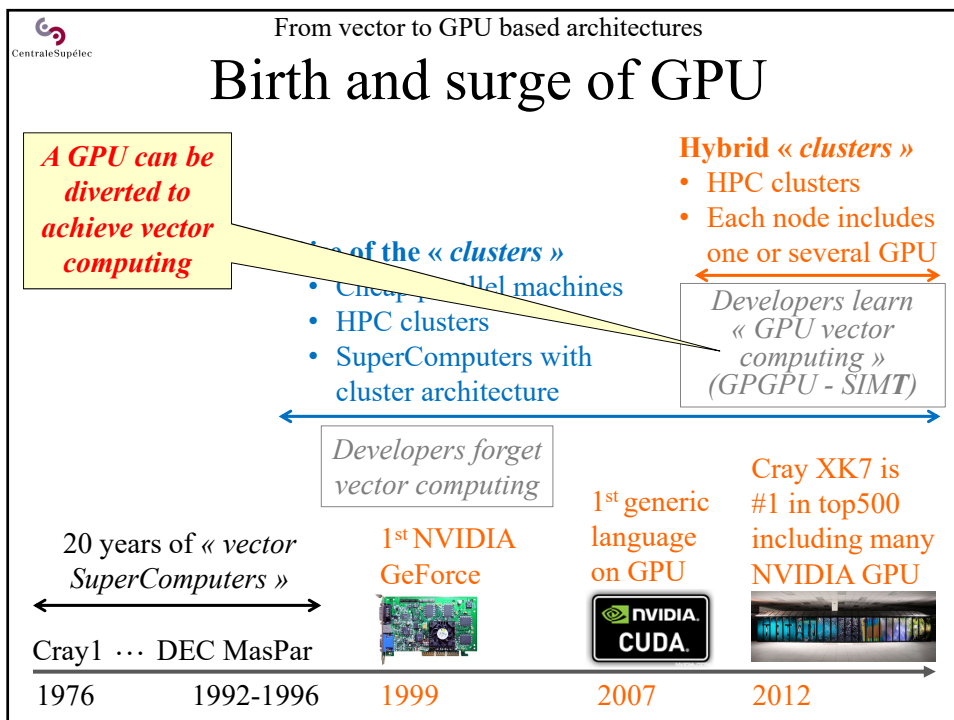
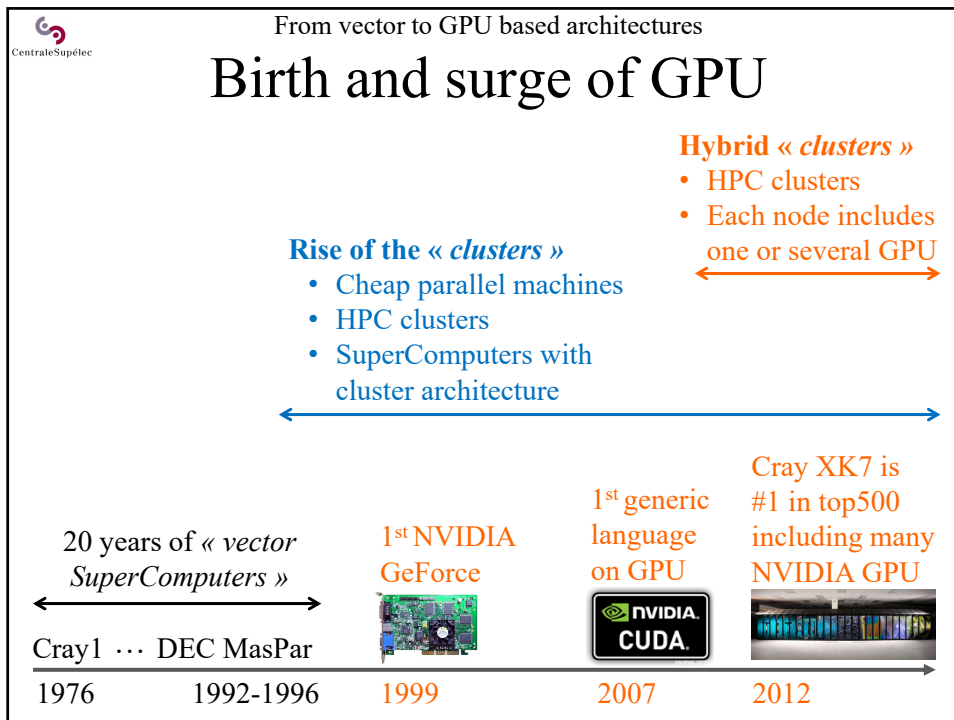


Stephane.Vialle@centralesupelec.fr  
<http://www.metz.supelec.fr/~vialle>

# GPU architecture

- 1 – From vector to GPU based architectures
- 2 – NVIDIA products
- 3 – CUDA cores architecture
- 4 – Recent architecture issues

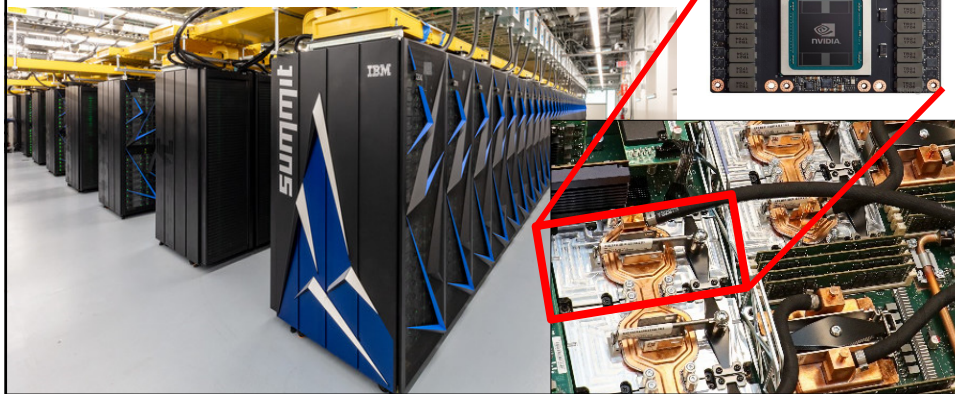




# Summit - USA: N°1 November 2019

143.5 Pflops

- 9 216 processors IBM POWER9 22C 3.07GHz
- **27 648 GPU Volta GV100: 6 GV100/node**  
→ 2 282 544 « cores » (CPU cores + CUDA cores)
- interconnect: Dual-rail Mellanox EDR Infiniband
- 10 MWatt



# GPU in the Top500

★ Group of TOP500 machines with GPU NVIDIA



# GPU architecture

- 1 – From vector to GPU based architectures
- 2 – NVIDIA products**
- 3 – CUDA cores architecture
- 4 – Recent architecture issues

From vector to GPU based architectures

## GeForce / Tesla families

2 NVIDIA product families, 2 strategies:



Available cores:

- Generic CUDA Cores
- Tensor Cores
- Ray Tracing Cores

Floating-point format:

- simple-precision:  
many units, high perf
- double-precision:  
**very few units, low perf**

Computing capabilities:

- **NOT certified**

Insertion into clusters

- **forbiden!**



Available cores:

- Generic CUDA Cores
- Tensor Cores
- ~~Ray Tracing Cores~~

Floating-point format:

- simple-precision:  
many units, high perf
- double-precision:  
many units, high perf

Computing capabilities:

- certified

Insertion into clusters

- authorized/recommended

From vector to GPU based architectures

# GeForce packaging



GeForce 256  
1999



Air cooled



Water cooled



GeForce RTX 2080 Ti  
2018

From vector to GPU based architectures

# Tesla V100 packaging



*GV100 has been a  
3 billion dollar  
R&D project*

# GPU architecture

- 1 – From vector to GPU based architectures
- 2 – NVIDIA products
- 3 – CUDA cores architecture**
- 4 – Recent architecture issues

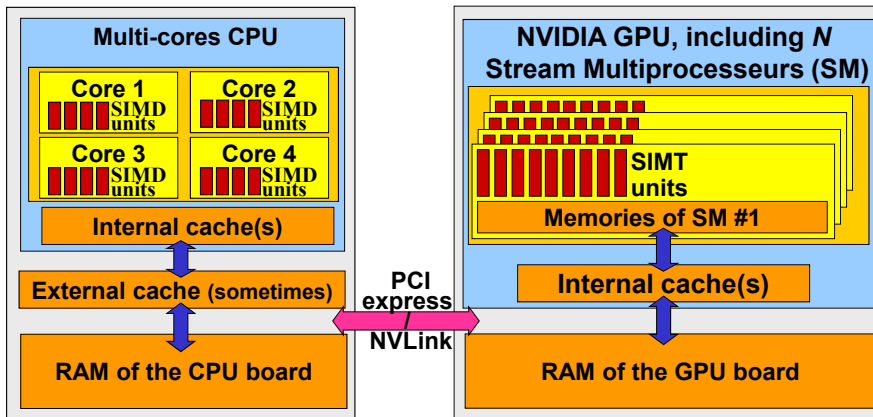
Architecture des GPU et principes de base de CUDA

## 1 – Architecture d'un GPU NVIDIA

- Vue d'ensemble CPU-GPU
- Principaux concepts d'architecture
- Cache générique des GPU
- Perception de l'architecture d'un GPU
- Nouveautés architecturales
- Quelques valeurs...

## CUDA cores architecture CPU-GPU overview

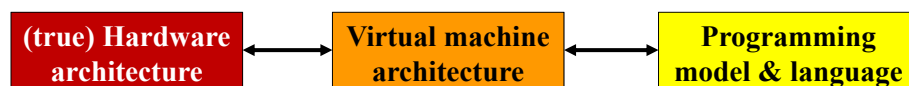
- Both CPU and GPU are « multi-cores » with « hierarchical memories »



- CPU and GPU boards communicate across a fast NVLink, or a std (and slow) PCI-eXpress

→ CPU uses GPU as a *scientific coprocessor* for vector/SIMD computing

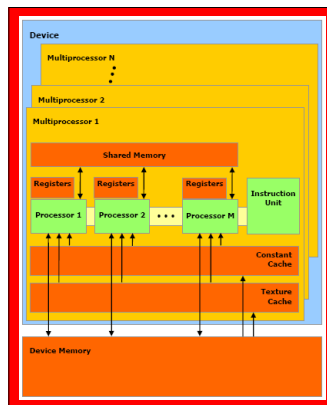
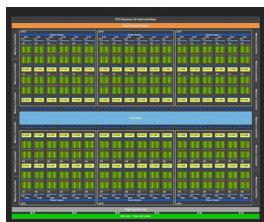
## CUDA cores architecture PTX virtual machine



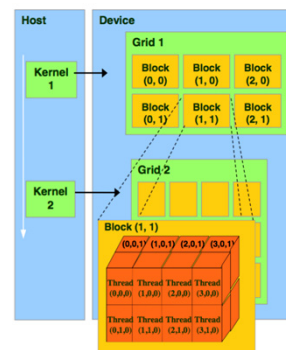
GPU chip

PTX

CUDA



In this chapter





# Stream Multiprocessors architecture

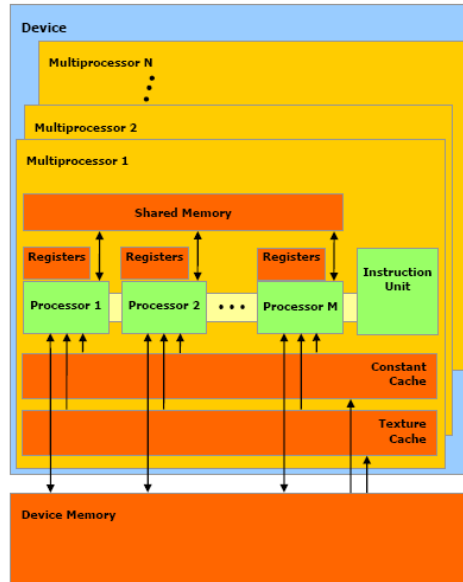
A GPU is a set of  $N$  Stream multiprocessors (SM):

- $N$  independent « SIMT » machines
- Sharing the GPU board memory

Before Volta & Turing architectures one SM included:

- 1 instruction decoder/unit
- 32 strongly synchronized *hardware threads*, running *warps* of 32 threads
- 32K-128K registers distributed among all *hardware threads* (and not shared)
- A fast memory shared between all running threads (of the SM)

+ A scheduler of warps of threads, among a larger block of threads



document nVIDIA

# Stream Multiprocessors architecture

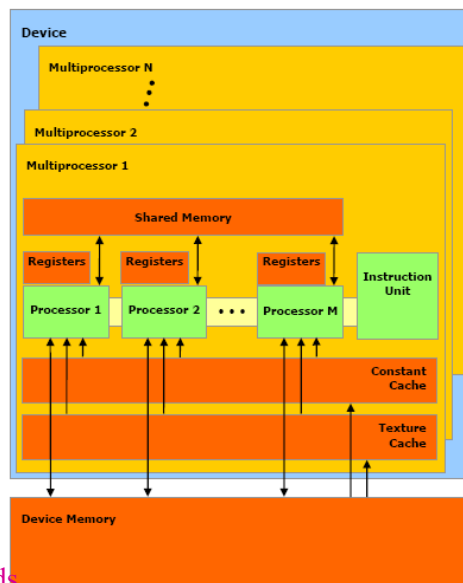
A GPU is a set of  $N$  Stream multiprocessors (SM):

- $N$  independent « SIMT » machines
- Sharing the GPU board memory

Since Volta & Turing architectures each SM is more complex:

- several instruction decoders/units
- 64 *hardware threads*
  - still running *warps* of 32 threads
  - less strongly synchronized
- 32K-128K registers distributed among all *hardware threads* (and not shared)
- A fast memory shared between all running threads (of the SM)

+ several schedulers of warps of threads



document nVIDIA

CentraleSupélec

CUDA cores architecture

# Stream Multiprocessors architecture

**What is a « CUDA core » ?**

→ It is a *hardware thread*:

- 1 ALU + access to SP and DP floating point units + some registers of the SM + access to the shared memory + access to the global memory of the GPU

**CUDA core**

**Comparison to CPU:**

- A CUDA core can be compared to one ALU of a CPU SIMD unit (AVX units)
- A CPU core can be compared to one Stream Multiprocessor of a GPU

document nVIDIA

CentraleSupélec

CUDA cores architecture

# GPU multiple memories

**Multiple memories are available:**

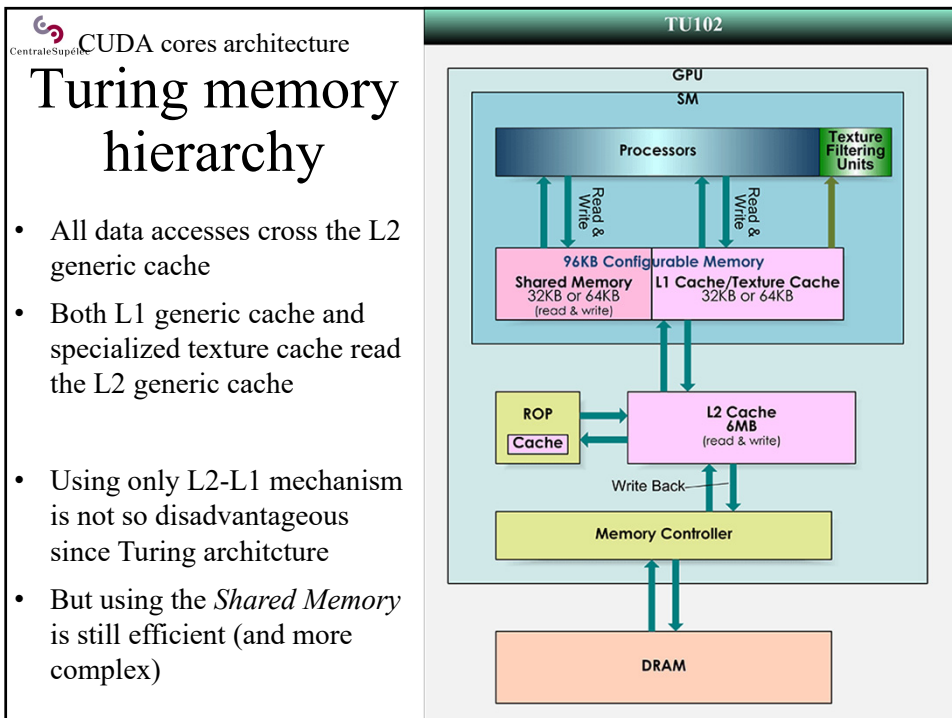
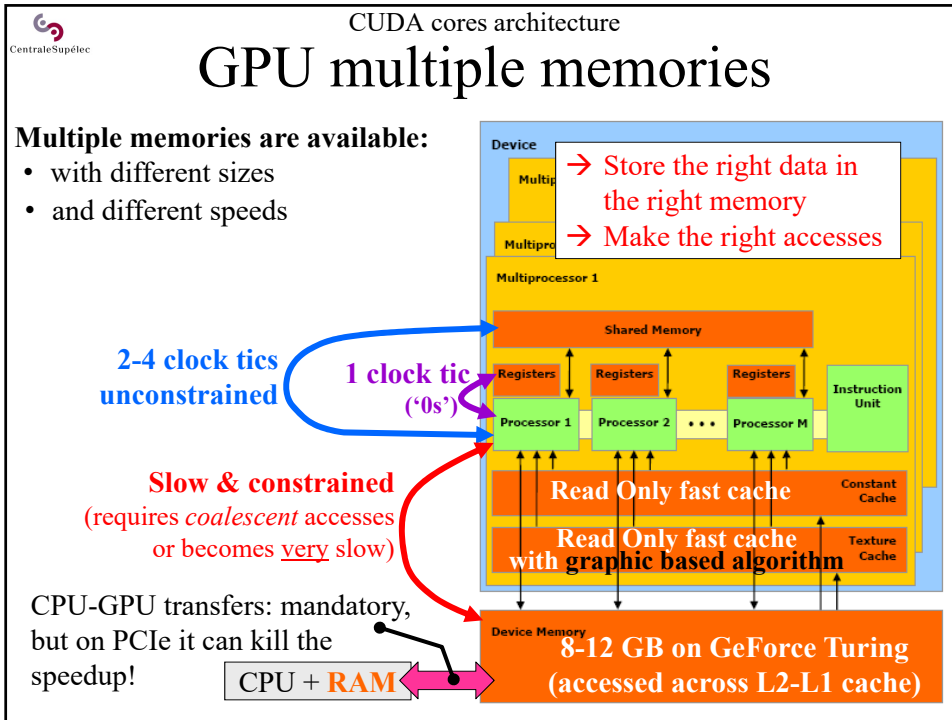
- with different sizes

Pascal { 96KB/SM but 48KB/block of th. →

Volta {

Turing { 64Kreg/SM (256KB/SM) →

8-12 GB on GeForce Turing (accessed across L2-L1 cache)



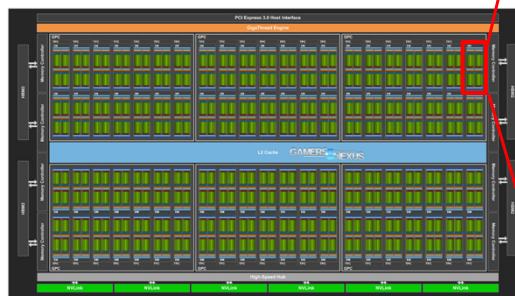
# GPU architecture

- 1 – From vector to GPU based architectures
- 2 – NVIDIA products
- 3 – CUDA cores architecture
- 4 – **Recent architecture issues**

## Recent architecture issues New Tesla GPU

### Volta architecture:

- ex : carte V100 (pro) pour le calcul
- unités de calcul entier
- unités de calcul simple précision
- unités de calcul double précision
- **Tensor cores**



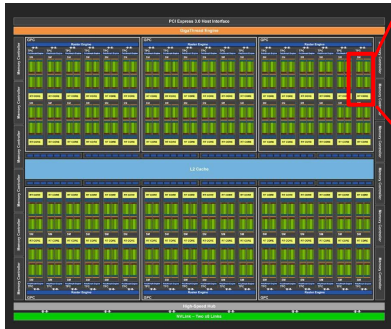
**1 Stream Multiprocessor**

# New GeForce GPU

## Turing architecture:

Ex : RTX 2080Ti GPU board

- unités de calcul entier
- unités de calcul simple précision
- **Tensor cores**
- **Ray Tracing cores**
- **(re-)Unified Cache L1 – Shared memory**



1 Stream Multiprocessor

# New GeForce GPU

## Turing architecture:

Ex : RTX 2080Ti GPU board

- 72 Stream Multiprocessor (SM)
- 64 CUDA cores / SM  
→ 4608 CUDA cores
- Double-precision computing is possible but slow
- 8 Tensor cores / SM  
→ 576 Tensor cores
- 1 Ray Tracing core / SM  
→ 72 RT cores
- **More efficient cache memory**



1 Stream Multiprocessor

# Motivation to design RT cores

## Ray Tracing cores:

- Final objective: « real time ray tracing for video »
- Currently: GPU not powerful enough  
→ Real Time RT on a subset of rays  
+ interpolation with Tensor Cores



**Video game remains the main market for NVIDIA**

→ GPU architecture evolutions must be useful for the video game market

*SOL MAN from NVIDIA SOL ray tracing demo running on a Turing TU102 GPU with NVIDIA RTX technology in real-time*

# Tensor Core features

## Tensor cores:

1 TC achieves a flow of product-add on a flow of 4x4 matrixes

- $D = A.B$  : produces a flow of D output matrixes
- $D = A.B + C$ , with accumulation of A.B product flow into C matrix

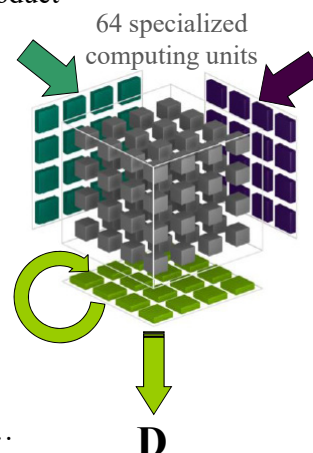
$$D = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

FP16 or FP32      FP16      FP16      FP16 or FP32

Possible mixed-precision:

- input matrixes encoded on 16 bits
- internal computing on 32 bits
- output matrix flow on 16 or 32 bits

**Useful for many kinds of applications:** image processing, Machine Learning, Linear Algebra...



## Tensor Core features

### Tensor cores:

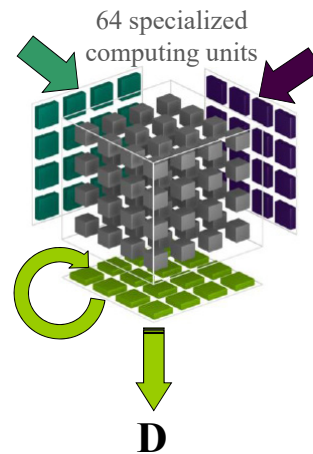
A Tensor Core achieves a flow of product-add on a flow of 4x4 matrixes

- $D = A.B$  : produces a flow of D output matrixes
- $D = A.B + C$ , with accumulation of A.B product flow into C matrix

### A Tensor core:

- is a hardware implementation of a matrix operator,
- is a very useful operator for modern applications,
- including graphic applications (main GPU market).

→ A math operator who deserves to occupy part of the chip!



## Cache memory improvement

### New fast memory:

- 96 KBytes per SM
- Used for both: L1 cache, shared memory, texture cache

*Rmk: The « shared memory » is an unmanaged L1 cache memory. The application developer has to design and implement a strategy adapted to its computations!*

- If the shared memory is unused, the 96 KBytes will be automatically used for L1 cache
- A new and more efficient cache management strategy has been implemented

### Objectives of this new fast memory architecture and management:

- To decrease the performance loss when not using shared memory...
- ... many users have refused to design and implement a new cache management strategy (too difficult).

# Evolution of the GPU features

(<https://en.wikipedia.org/wiki/CUDA>)

Feature support (unlisted features are supported for all compute abilities)	Compute capability (version)												
	1.0	1.1	1.2	1.3	2.x	3.0	3.2	3.5, 3.7, 5.0, 5.2	5.3	6.x	7.x	8.x	
Integer atomic functions operating on 32-bit words in global memory	No	Yes											
atomicExch() operating on 32-bit floating point values in global memory	No	Yes											
Integer atomic functions operating on 32-bit words in shared memory	No	Yes											
atomicExch() operating on 32-bit floating point values in shared memory	No	Yes											
Integer atomic functions operating on 64-bit words in global memory	No	Yes											
Warp vote functions	No	Yes											
Double-precision floating-point operations	No	Yes											
Atomic functions operating on 64-bit integer values in shared memory	No	Yes											
Floating-point atomic addition operating on 32-bit words in global and shared memory	No	Yes											
_ballot()	No	Yes											
_threadfence_system()	No	Yes											
_syncthreads_count(), _syncthreads_and(), _syncthreads_or()	No	Yes											
Surface functions	No	Yes											
3D grid of thread block	No	Yes											
Warp shuffle functions	No	Yes											
Funnel shift	No	Yes											
Dynamic parallelism	No	Yes											
Half-precision floating-point operations: addition, subtraction, multiplication, comparison, warp shuffle functions, conversion	No	No										Yes	
Atomic addition operating on 64-bit floating point values in global memory and shared memory	No	No										Yes	
Tensor core	No	No										Yes	

# Evolution of the GPU Tesla

Tesla Product	Tesla K40	Tesla M40	Tesla P100	Tesla V100
GPU	GK180 (Kepler)	GM200 (Maxwell)	GP100 (Pascal)	GV100 (Volta)
SMs	15	24	56	80
TPCs	15	24	28	40
FP32 Cores / SM	192	128	64	64
FP32 Cores / GPU	2880	3072	3584	5120
FP64 Cores / SM	64	4	32	32
FP64 Cores / GPU	960	96	1792	2560
Tensor Cores / SM	NA	NA	NA	8
Tensor Cores / GPU	NA	NA	NA	640
GPU Boost Clock	810/875 MHz	1114 MHz	1480 MHz	1530 MHz
Peak FP32 TFLOPS <sup>1</sup>	5	6.8	10.6	15.7
Peak FP64 TFLOPS <sup>1</sup>	1.7	.21	5.3	7.8
Peak Tensor TFLOPS <sup>1</sup>	NA	NA	NA	125

Some architecture features do not evolve monotonously



## GPU architecture

End