Why are GPUs faster than CPUs for the matrix calculations of deep learning libraries?

Laércio Lima Pilla

laercio.lima-pilla@labri.fr

STORM
STATIC OPTIMIZATIONS – RUNTIME METHODS
LaBRI
cnrs
université de BORDEAUX
Bordeaux INP AQUITAINE
Inria

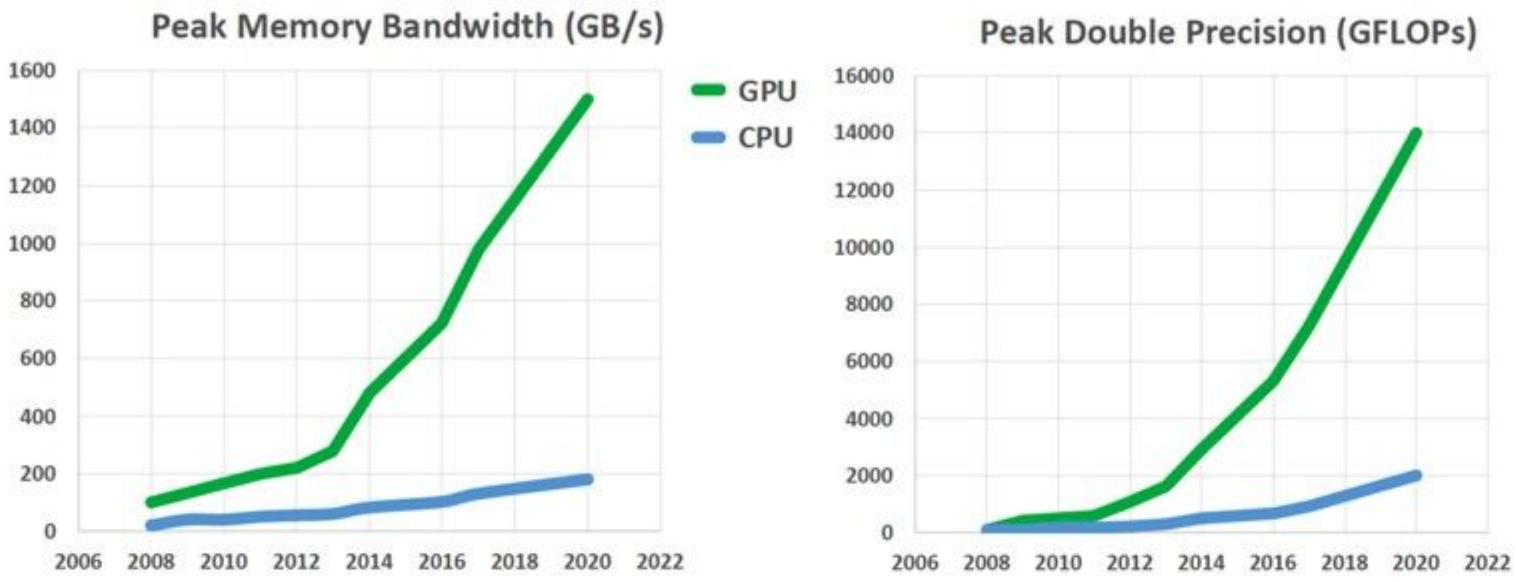
1. The quick answer

2. The longer explanation

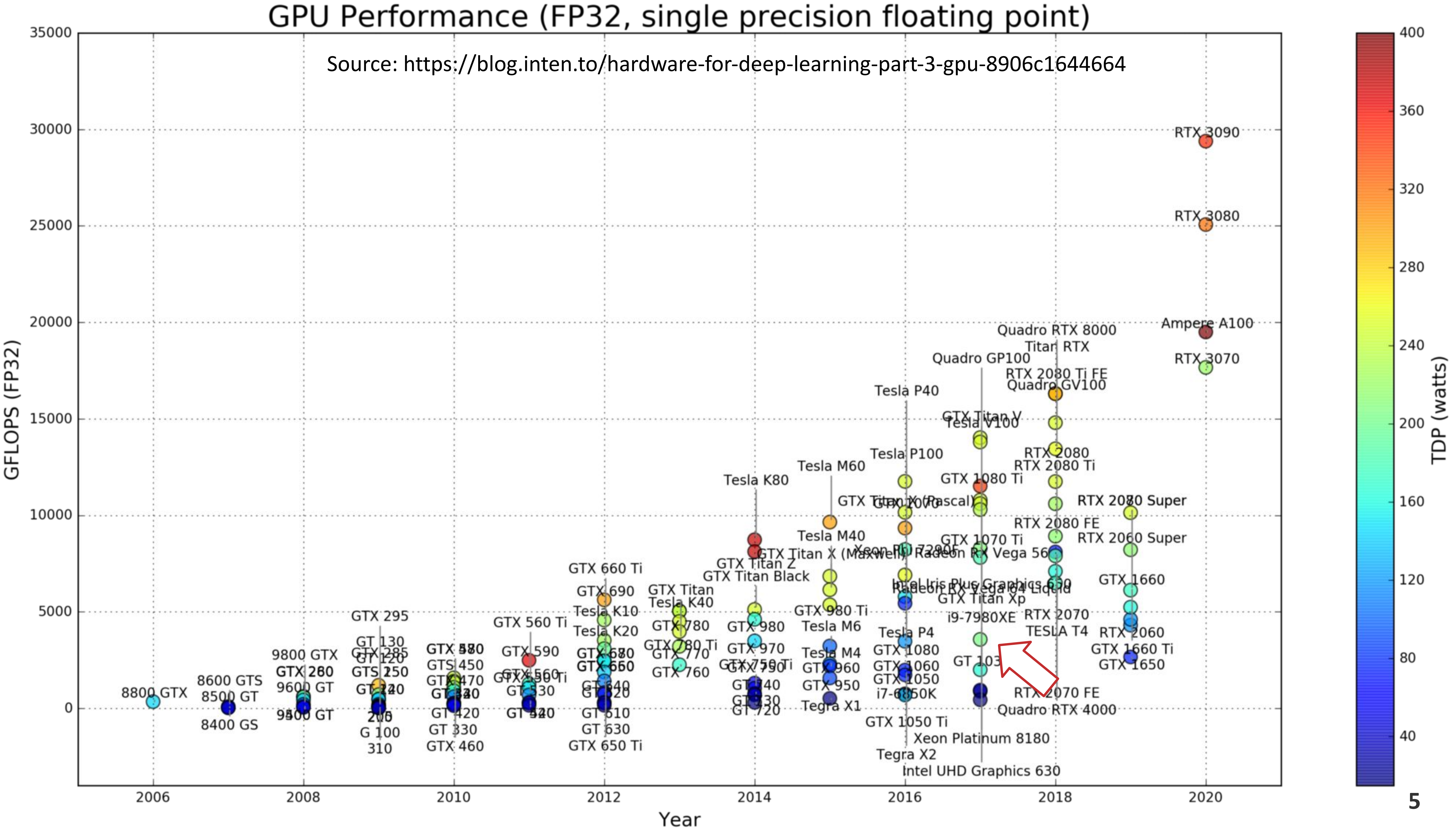
1. The quick answer

1. The quick answer

GPUs have a higher peak performance than CPUs and they are well-adapted for matrix operations.



Source:
https://www.nextplatform.com/2019/07/10/a-decade-of-accelerated-computing-augurs-well-for-gpus/


GPU Performance (FP32, single precision floating point)
Source: https://blog.inten.to/hardware-for-deep-learning-part-3-gpu-8906c1644664
GFLOPS (FP32)
Year
TDP (watts)
RTX 3090
RTX 3080
Ampere A100
RTX 3070
Quadro RTX 8000
Titan RTX
RTX 2080 Ti FE
Quadro GV100
Quadro GP100
Tesla P40
GTX Titan V
Tesla V100
Tesla P100
RTX 2080
RTX 2080 Ti
Tesla M60
Tesla K80
GTX 1080 Ti
RTX 2080 Super
RTX 2080 FE
RTX 2060 Super
Tesla M40
GTX 1070 Ti
GTX Titan Z
GTX Titan Black
GTX 660 Ti
GTX 1660
GTX 690
GTX Titan
Tesla K40
GTX Titan Xp
GTX 980 Ti
Tesla K10
GTX 295
GTX 560 Ti
i9-7980XE
RTX 2070
Tesla K20
GTX 780
GTX 980
Tesla M6
Tesla P4
TESLA T4
RTX 2060
GTX 590
GTX 1660 Ti
GTX 1650
9800 GTX
GT 1030
8600 GTS
8500 GT
8800 GTX
8400 GS
RTX 2070 FE
Quadro RTX 4000
Tegra X1
GTX 1050 Ti
Xeon Platinum 8180
Tegra X2
Intel UHD Graphics 630
G 100
310
GT 330
GTX 460
GTX 650 Ti
i7-6850K
GTX 760
GTX 770
GTX 950
GTX 960
GTX 970
GTX 1050
GTX 1060
GTX 1080

2. The longer explanation

The deal about parallelism

Example: finding an element in a sorted array

1	1	2	3	5	8	13	21	34	55

The simple way: O(n)

The binary search way: O(log n)

The parallel way: O(?)

The deal about parallelism

Example: finding an element in a sorted array

1	1	2	3	5	8	13	21	34	55

The simple way: O(n)

The binary search way: O(log n)

The parallel way: O(1)

How?

With n resources, each cell is checked at the same time, and anyone that finds the element writes it in the output

Different kinds of processing units

C is for **central**

Must be good for computing any kind of sequential task.

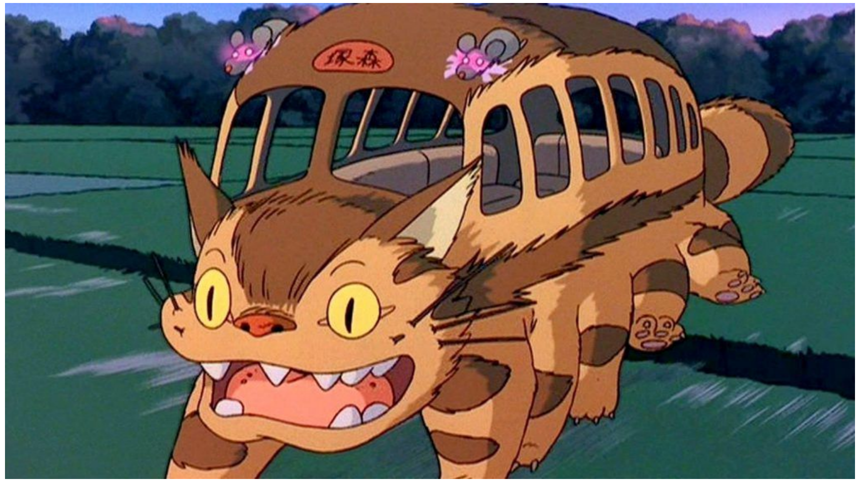
G is for **graphics**

Must be great for computing a bunch of pixels, triangles, etc.

Different kinds of processing units

C is for central

Must be good for computing any kind of sequential task.



G is for graphics

Must be great for computing a bunch of pixels, triangles, etc.

Different kinds of processing units

C is for **central**

Must be good for computing any kind of sequential task.

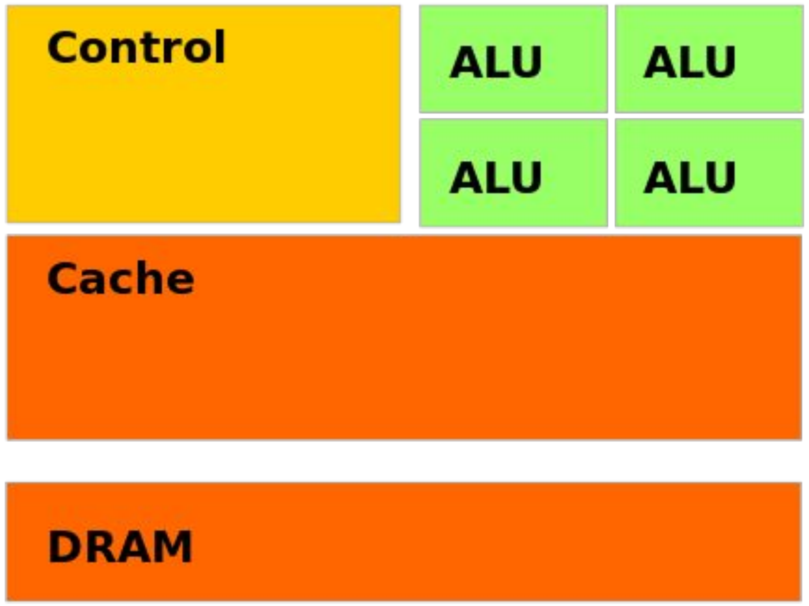
G is for **graphics**

Must be great for computing a bunch of pixels, triangles, etc.

Architectural differences

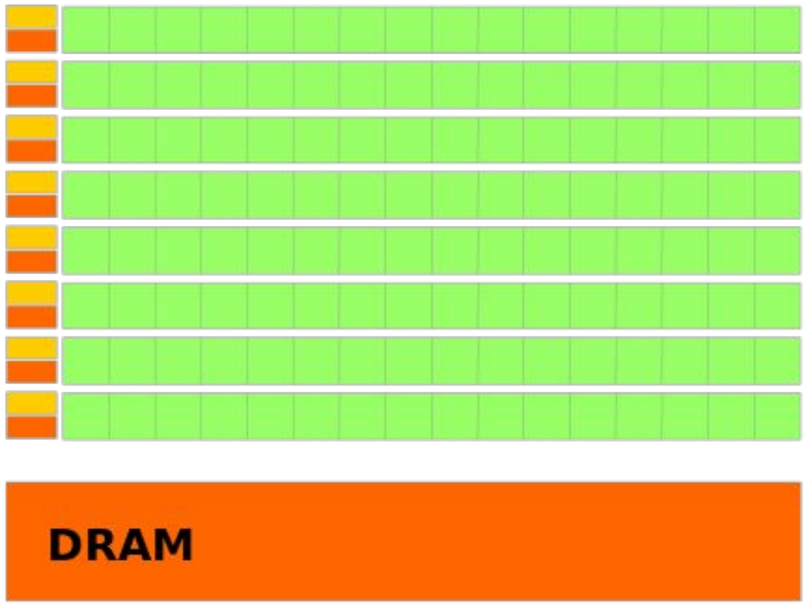
CPU

- Focused on latency
- A few cores
- Complex control
- Limited power



GPU

- Focused on throughput
- Several cores
- Simple control
- High power consumption



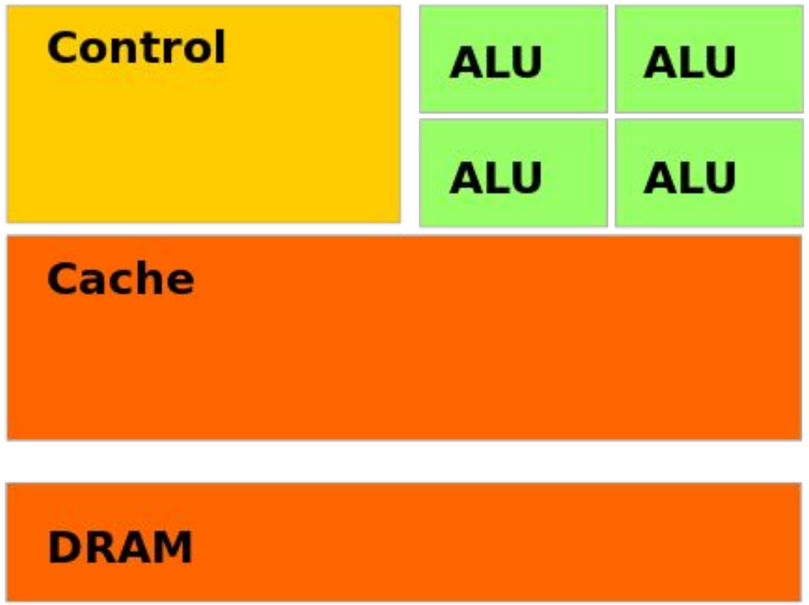
Architectural differences

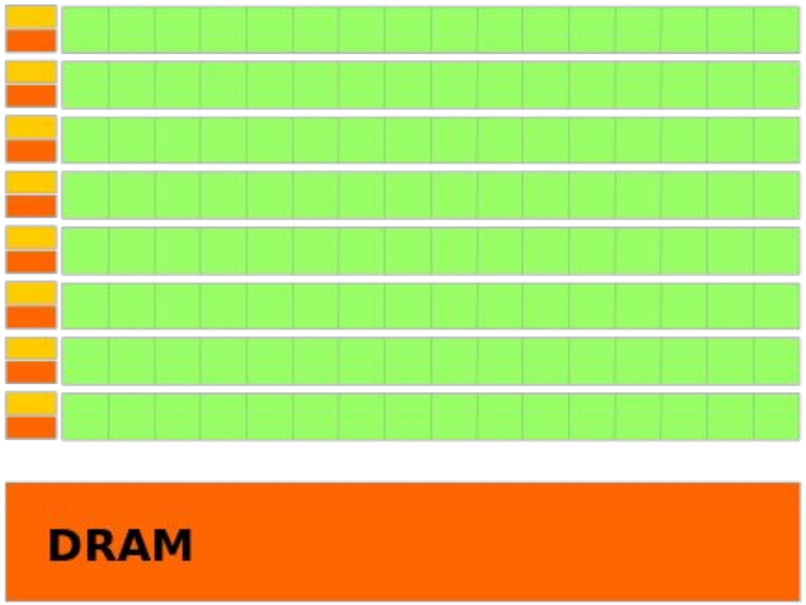
CPU

- Focused on latency
- Large capacity
- Large caches
- Coherent caches

GPU

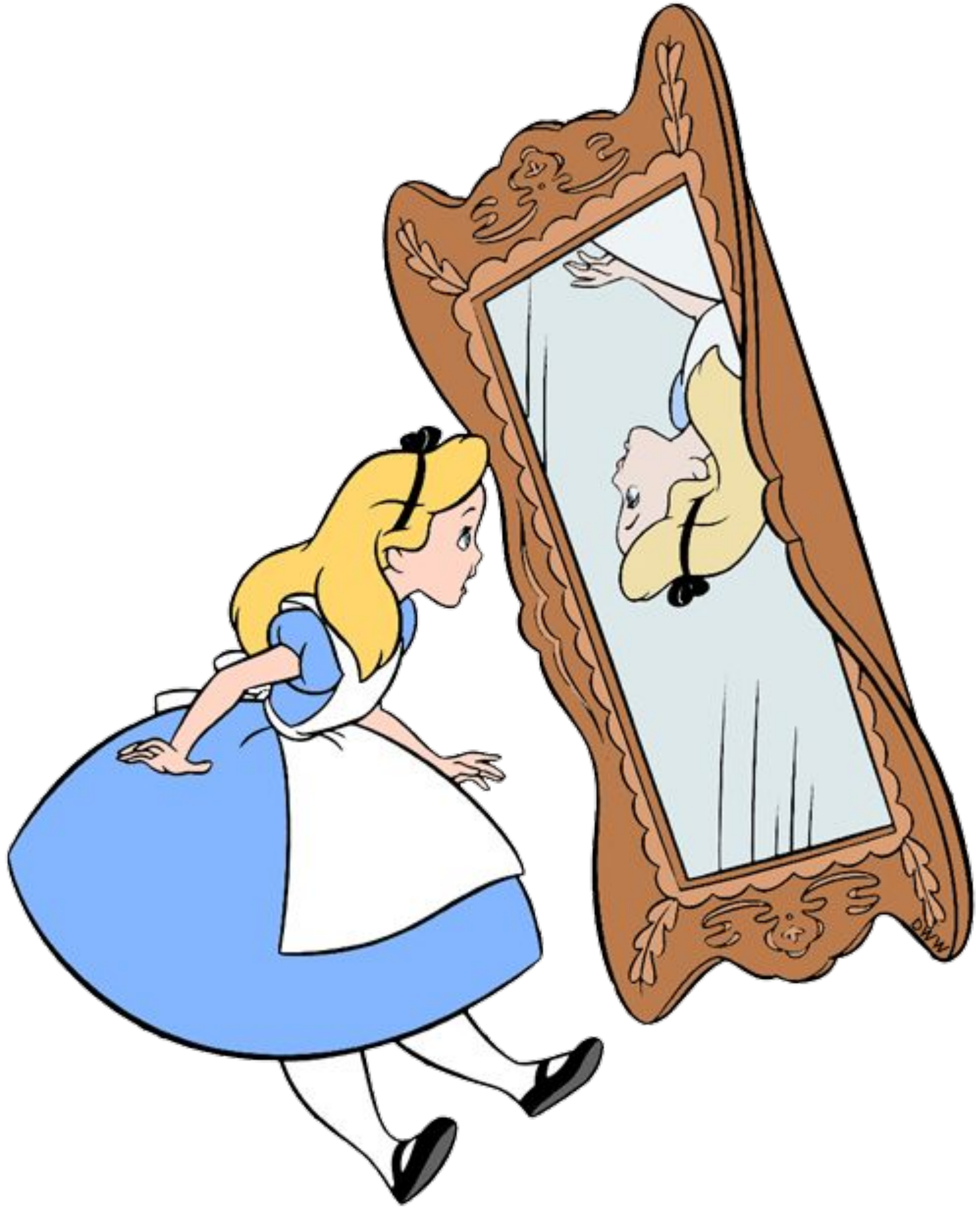
- Focused on bandwidth
- Small capacity
- Small caches
- Limited synchronization



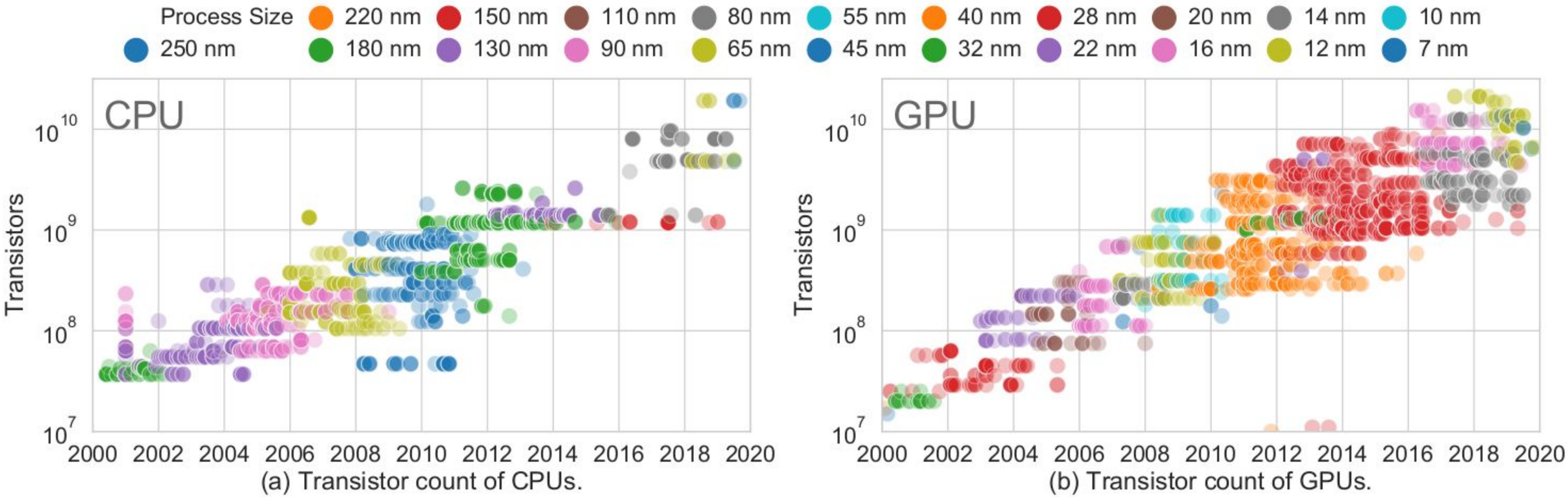


Parallelism and the Three Laws

Every design decision reflects how we handle parallelism



Moore’s Law

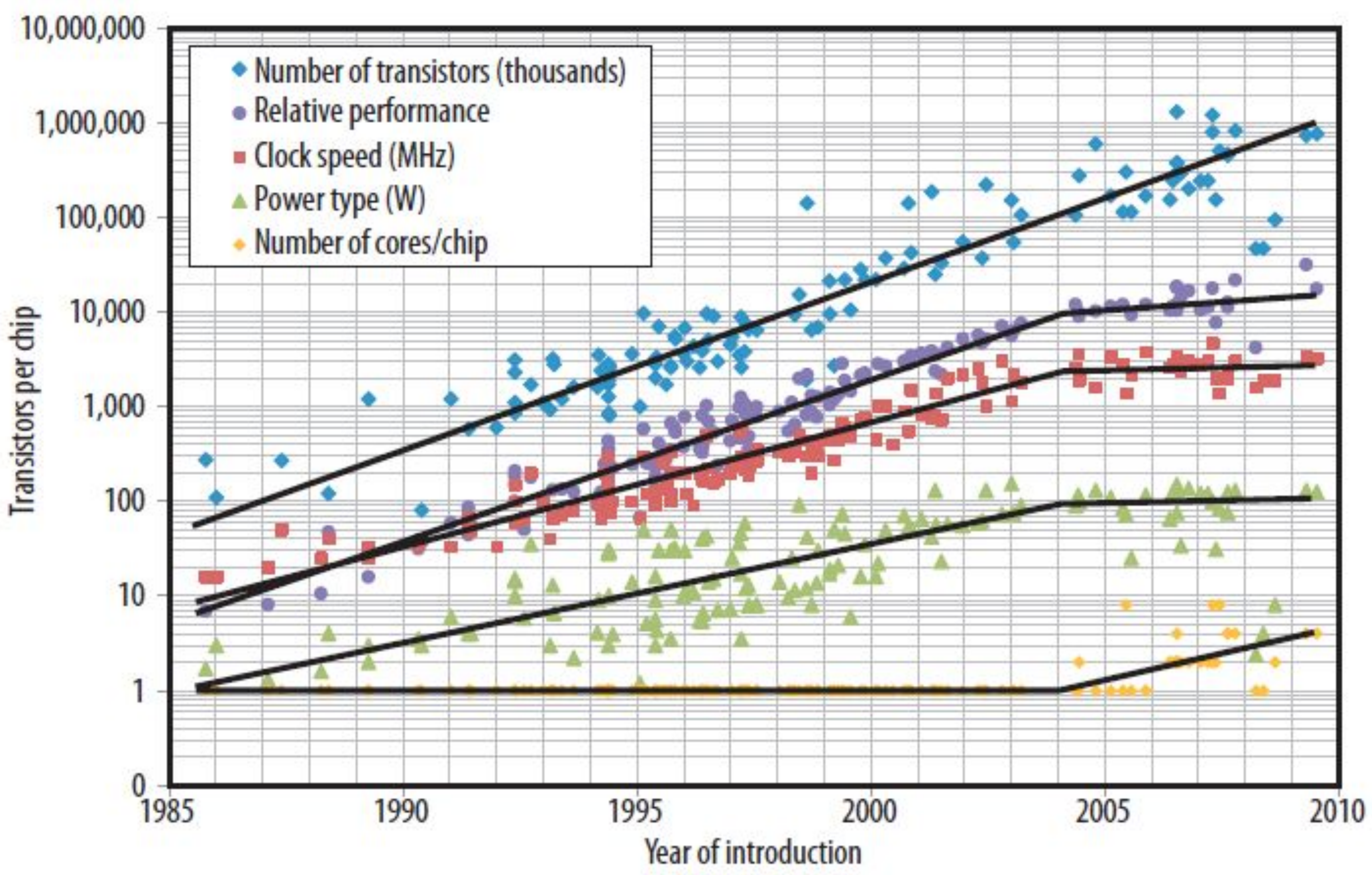


Source: https://arxiv.org/abs/1911.11313

“The number of transistors in chips doubles about every two years.”

This is still true.

Moore’s Law



Source: Computing Performance: Game Over or Next Level, IEEE Computer Magazine, January 2011, p. 33

“The number of transistors in chips doubles about every two years.”

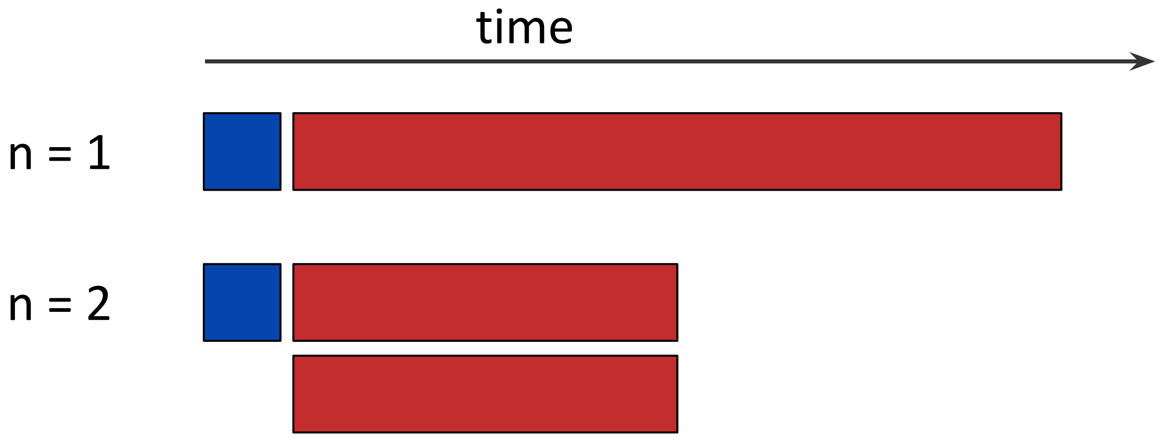
But the performance gains come mostly from parallelism now.

Amdahl’s Law

“The performance gains from the parallelization of a fixed workload are limited by its sequential portion.”

$$\text{Time}(n) = s*\text{Time}(1) + (1-s)*\text{Time}(1)/n,$$

for n: number of resources > 0, and s: sequential portion of the code in [0,1]



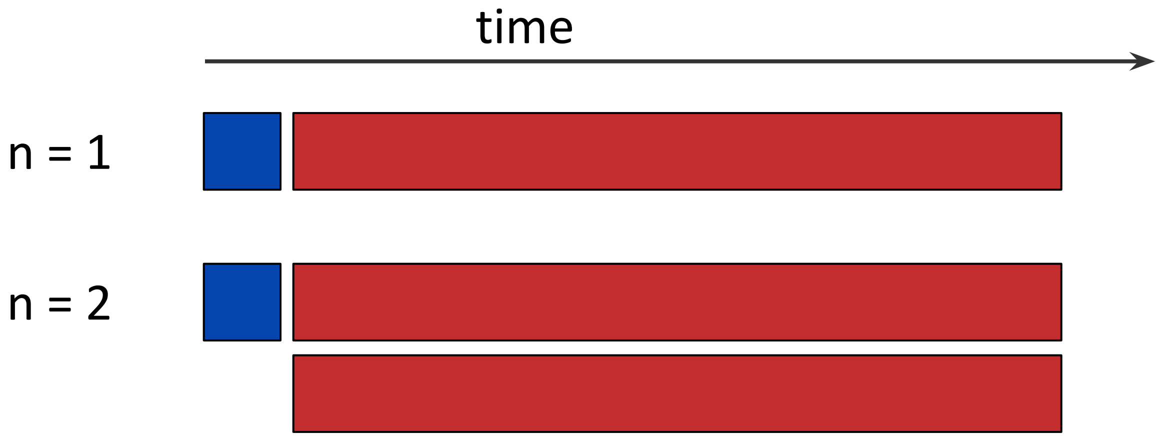
Only the most parallel codes can fully benefit from GPUs (strong scaling).

Gustafson's Law

"The size of a workload that can be computed in a fixed period of time is affected by its sequential portion."

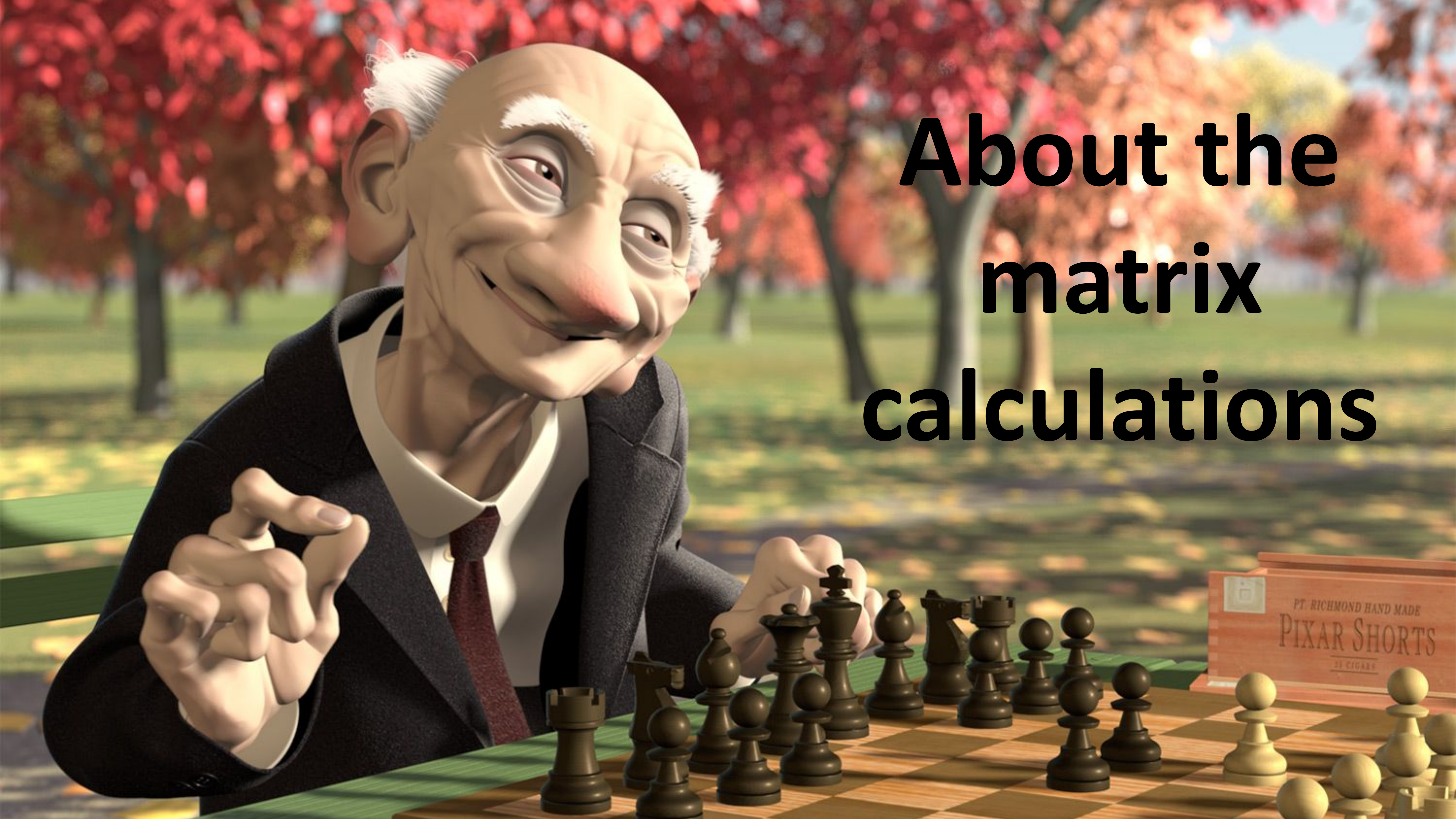
$$\text{Workload}(n) = s*\text{Workload}(1) + (1-s)*\text{Workload}(1)*n,$$

for n: number of resources > 0, and s: sequential portion of the code in [0,1]



More resources mean bigger problems can be treated (weak scaling).

About the matrix calculations

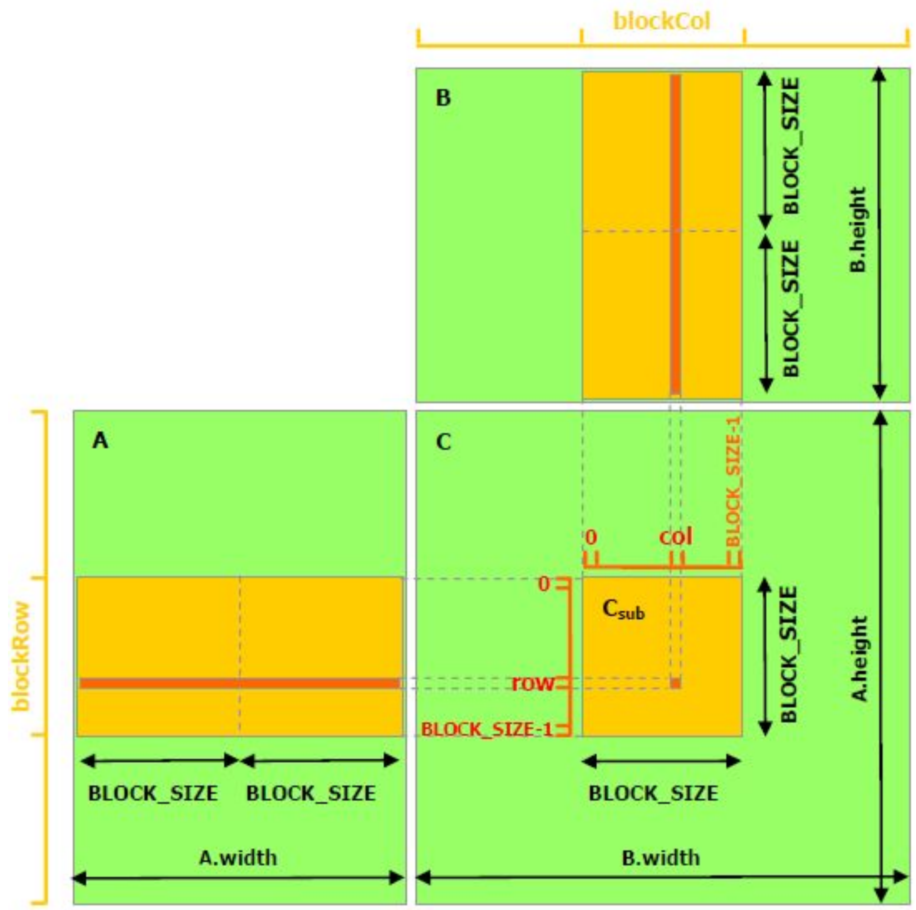


Matrix calculations

Features of 2D-matrix multiplications:

$$c_{ij} = \Sigma_{k=1}^{n} a_{ik} b_{kj}$$

- Very common, optimized kernel
- Operations: $O(n^3)$
- Data: $O(n^2)$
- Operations per cell in C: $O(n)$
- Each cell can be computed independently
- Memory accesses are regular and have both spatial and temporal locality



Matrix calculations

Features of convolutions:

- Very common operations in image processing (filtering)
- Similar to scalar products
- Each cell can be computed independently

1	2	3	4	5	6
7	8	9	1	2	3
4	5	6	7	8	9
1	2	3	4	5	6
7	8	9	1	2	3
4	5	6	7	8	9

$\otimes$

0	1	0
1	-4	1
0	1	0

=

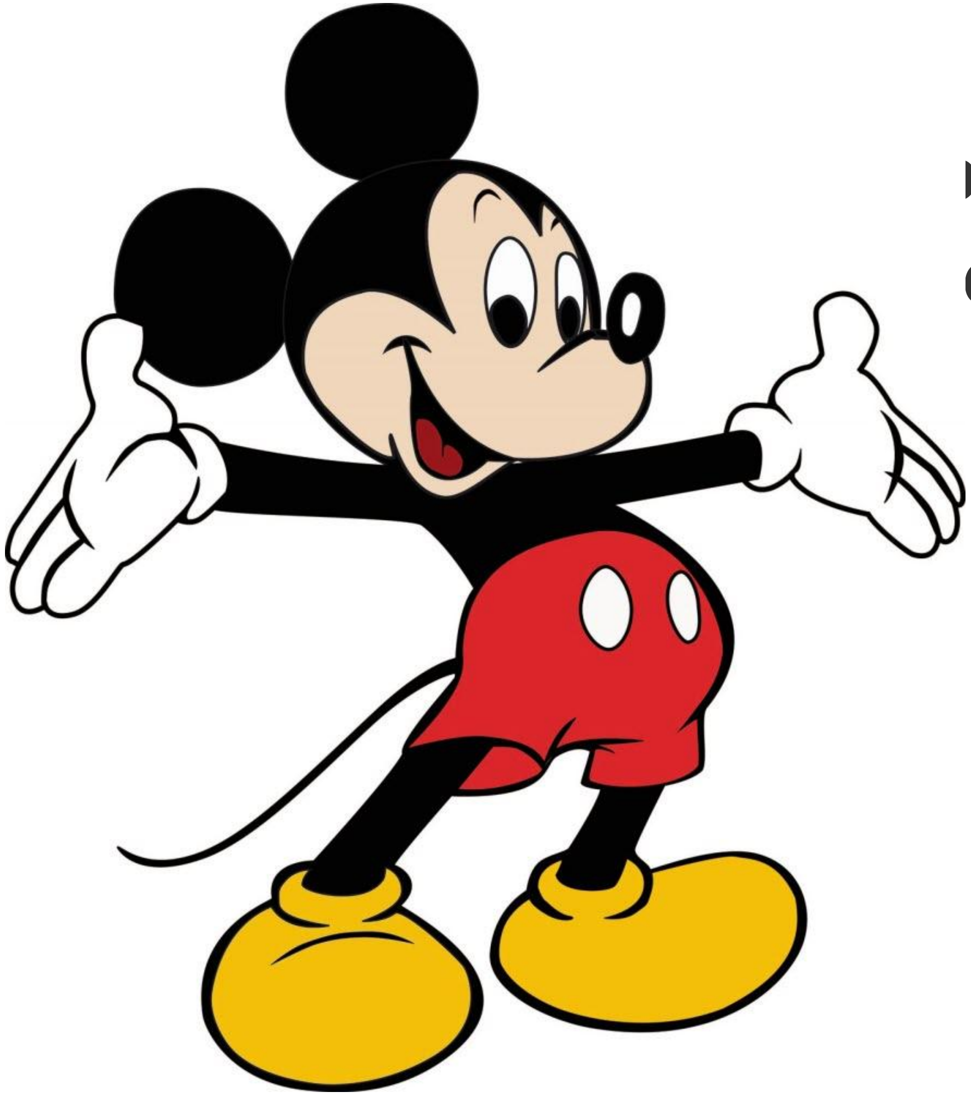
			-9		

9*0 + 1*1 + 2*0 +
6*1 + 7*-4 + 8*1 +
3*0 + 4*1 + 5*0 = -9

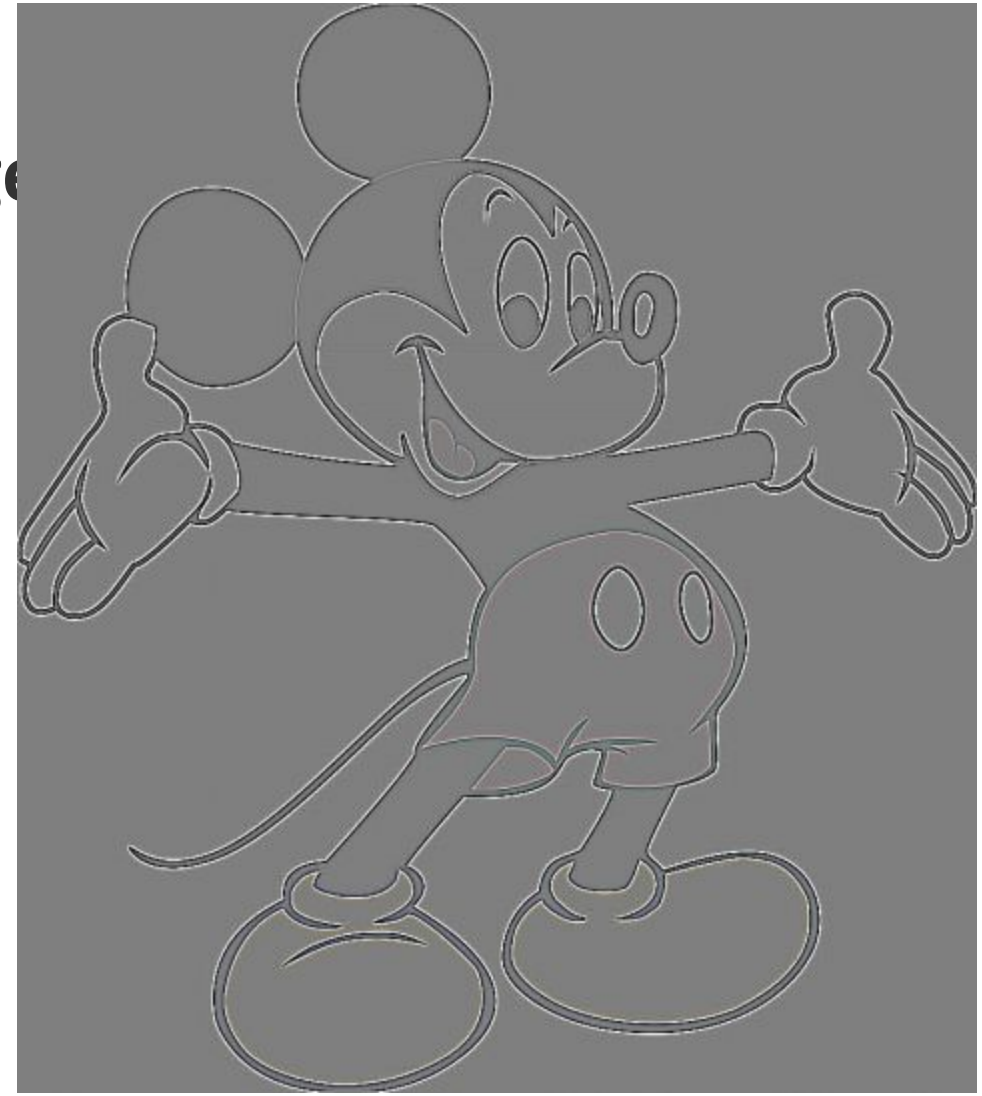
Matrix calculations

ons in image

cts



0	1	0
1	-4	1
0	1	0

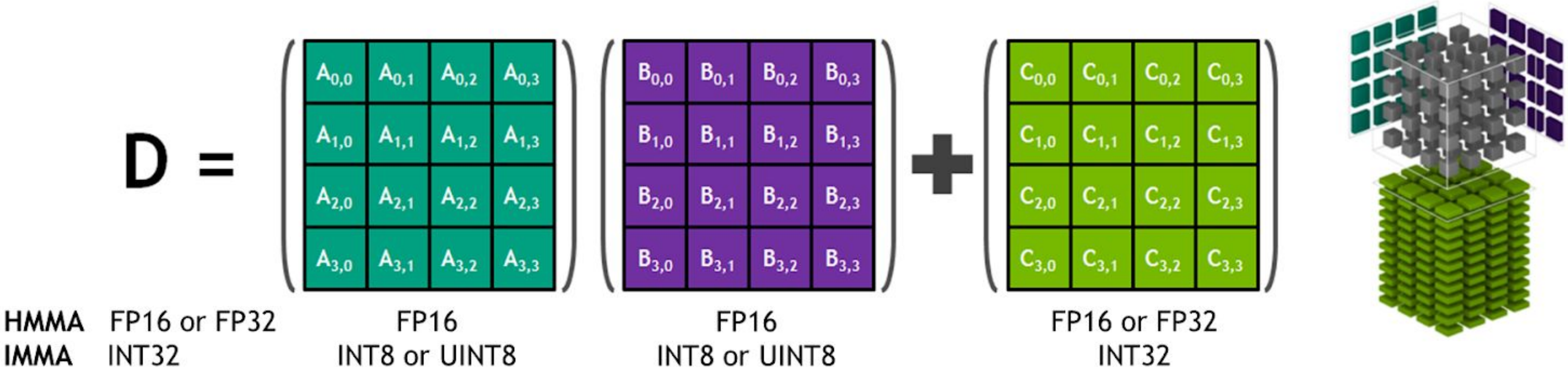


Matrix calculations

Features of tensor operations (and Tensor Cores):

$$D = AB + C$$

- **Small matrix products and accumulations**
- **Can work with mixed-precision data**
- **Tensor Cores as dedicated hardware for these operations**



Source: https://developer.nvidia.com/blog/nvidia-automatic-mixed-precision-tensorflow/

And remember:

GPUs have a higher peak performance than CPUs and they are well-adapted for matrix operations.

Laércio Lima Pilla
laercio.lima-pilla@labri.fr

STORM
STATIC OPTIMIZATIONS – RUNTIME METHODS