General Introduction to GPUs

NRIS

Norwegian research infrastructure services

Hicham Agueny, PhD
Scientific Computing Group
IT-department, UiB

One-day Program (10:00-15:45)

Slides and exercices

https://github.com/HichamAgueny/GPU-course.git

Heterogenous computing with OpenACC

I. General introduction to GPU
- **I-1. State-of-the-art supercomputers**
- **I-2. Introduction to GPU-hardware (CPU vs GPU)**
- **I-3. GPU-programming models and compiler supports**

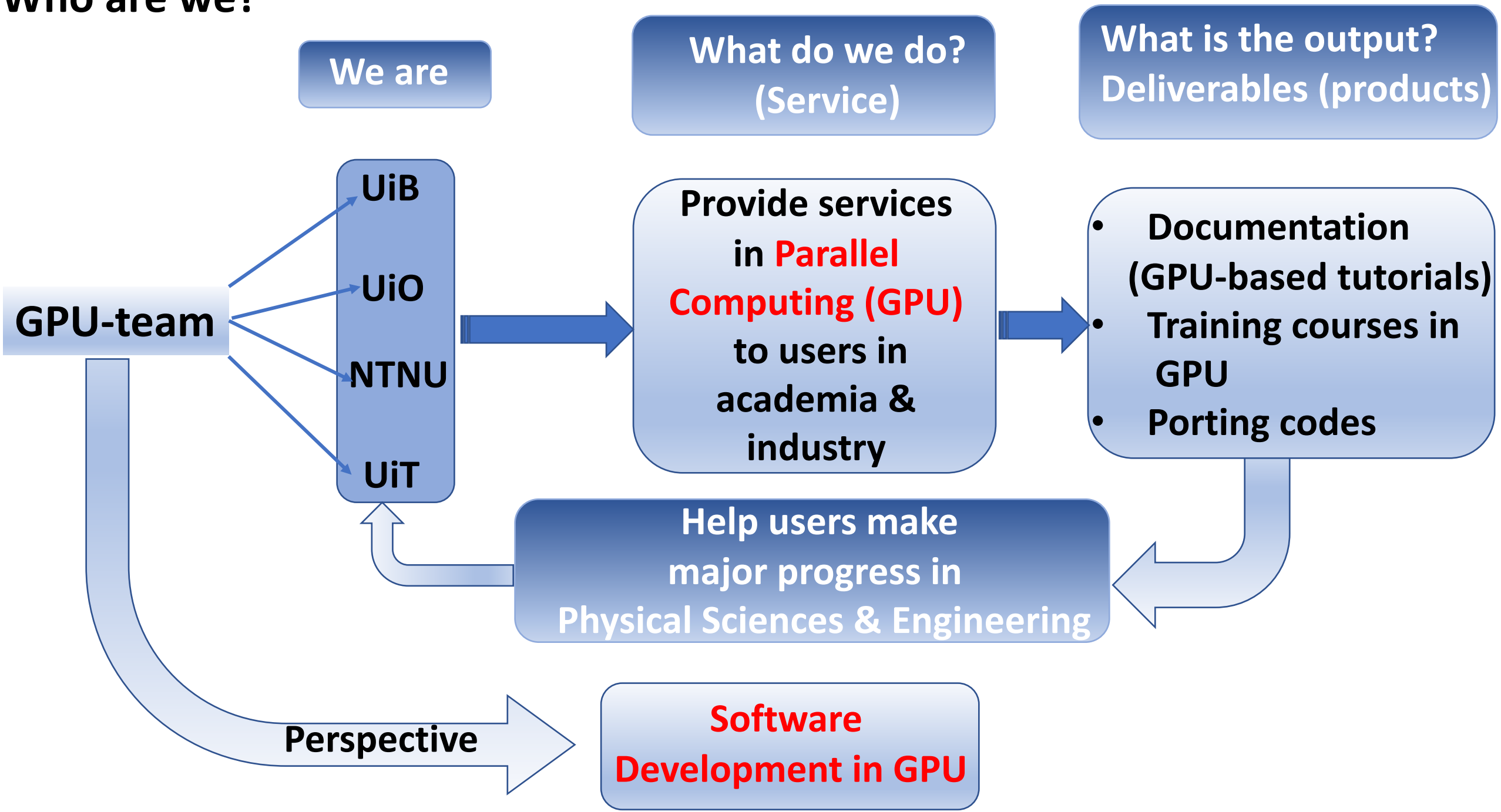
Break (11:00-11:15)

II. Showcase: OpenACC and functionality
- **II-1. Synchronous OpenACC**
- **II-2. Asynchronous OpenACC**
- **II-3. Code-profiling (Gprof+Nsight)**

Lunch (12:00-13:00)

III. Hands-on + discussion (2h)

Who are we?


We are
What do we do? (Service)
What is the output? Deliverables (products)
GPU-team
UiB
UiO
NTNU
UiT
Provide services in Parallel Computing (GPU) to users in academia & industry
Documentation (GPU-based tutorials)
Training courses in GPU
Porting codes
Help users make major progress in Physical Sciences & Engineering
Perspective
Software Development in GPU

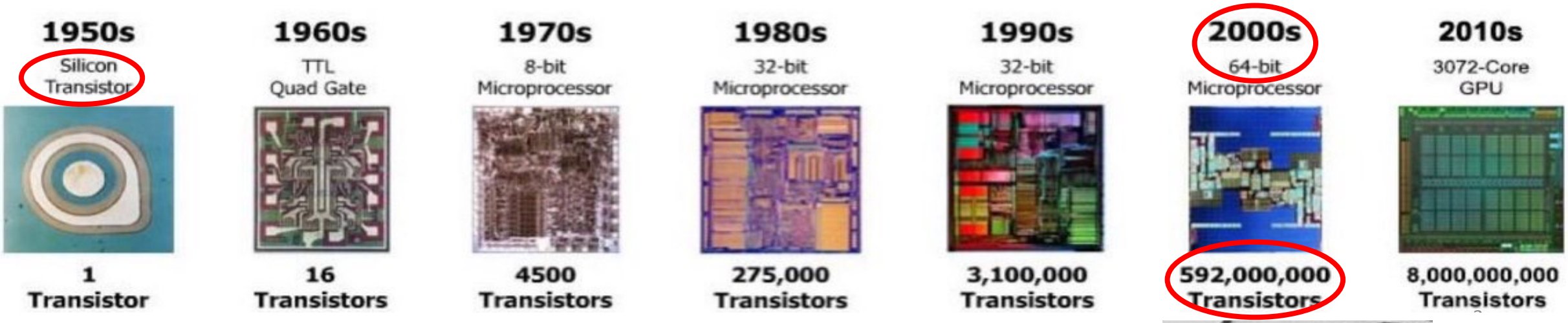
Overview

- **State-of-the-art supercomputers**
- **CPU & GPU architectures**
- **GPU-programming models and compiler supports**
- **Parallelism in heterogenous systems**

GPU: Graphics Processing Unit
CPU: Central Processing Unit

State-of-the-art supercomputers

Evolution of Processors: last 70 years



- Transistors are the **building block of a processor.**
- **Transistors** are fabricated using a concept based on **metal-oxide-silicon**.
- **Clock rate**: Transistors can switch on/off (one-cycle) from million times per second (MHz) up to billion times per second (GHz).
- The performance of a processor depends on how fast transistors can switch on/off.

64-bit in 2000s was a big step in performing precise calculations

Mohamed Attala

Invented in 1959 & In production Until 2018

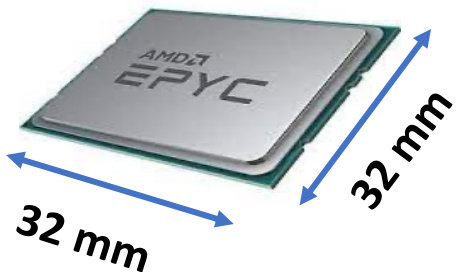
Dawon Kahng

Evolution of transistors in processor

AMD CPU

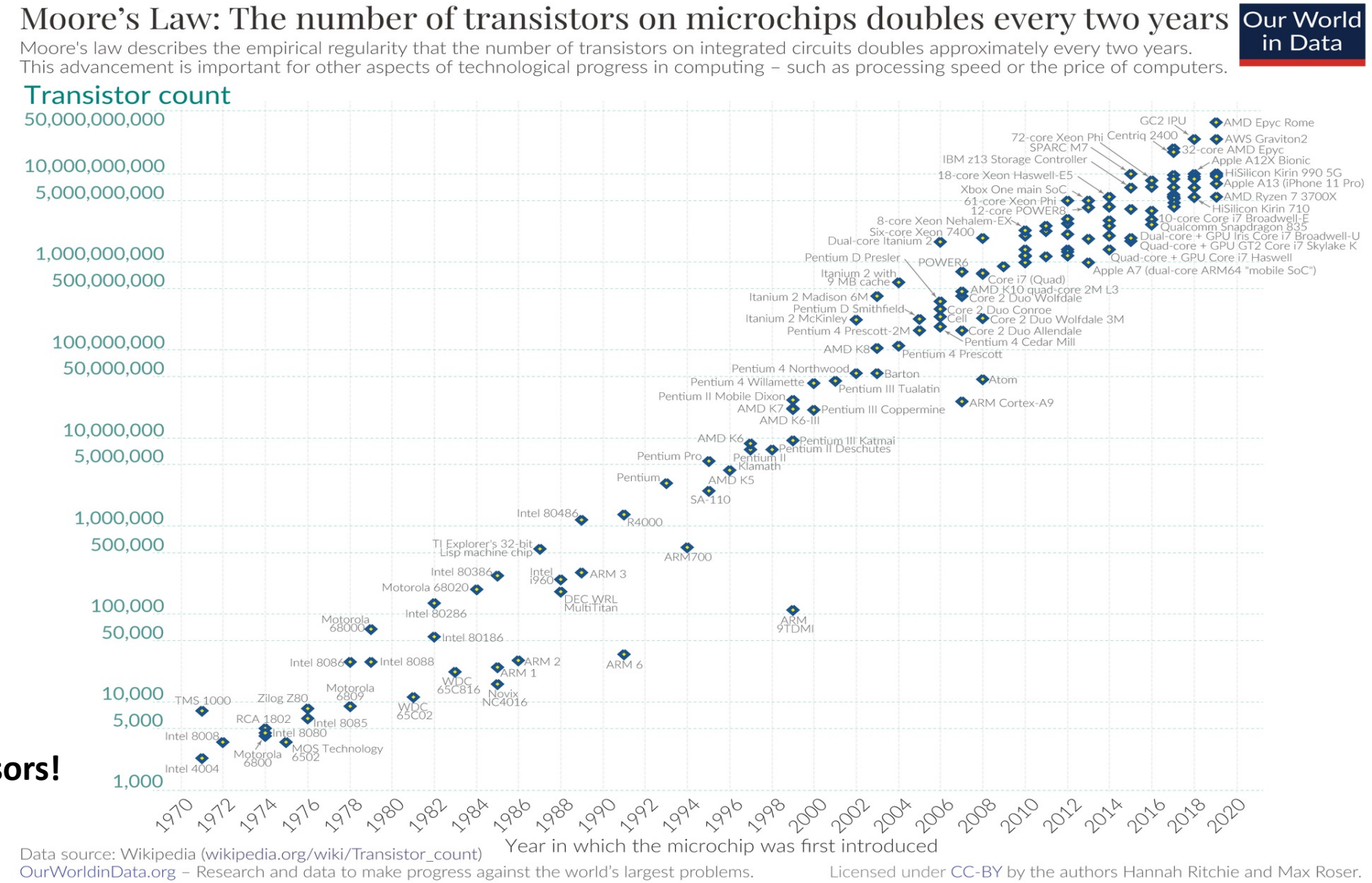
~40 billion transistors

1008 mm^2



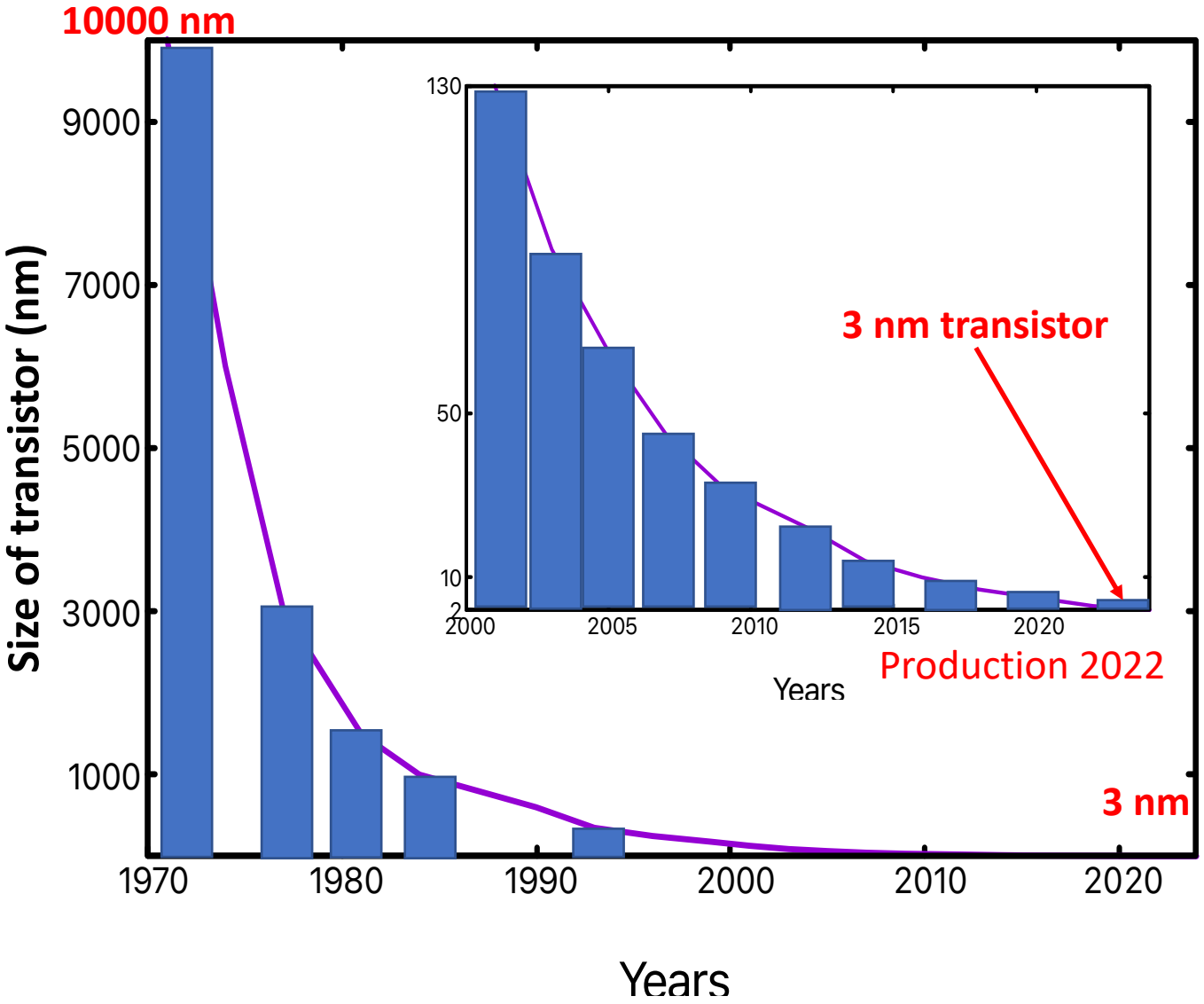
4-5 nm sized-transistor

This is a big step….in modern processors!



https://en.wikipedia.org/wiki/Moore%27s_law#cite_note-153

Evolution of transistors: last 50 years



About 3334 times smaller

BUT is it possible to go below nm….??

https://en.wikipedia.org/wiki/Moore%27s_law#cite_note-153

Perspective:

Towards PetaHz(10^6 GHz)-CPU with pico-m (10^-3 nm) sized-transistors

Atomic engineering

LETTERS
PUBLISHED ONLINE: 19 FEBRUARY 2012 | DOI: 10.1038/NNANO.2012.21
nature nanotechnology

A single-atom transistor

Martin Fuechsle[1], Jill A. Miwa[1], Suddhasatta Mahapatra[1], Hoon Ryu[2], Sunhee Lee[3], Oliver Warschkow[4], Lloyd C. L. Hollenberg[5], Gerhard Klimeck[3] and Michelle Y. Simmons[1]*

Fabrication of working devices such as transistors with extremely short gate lengths **requires** the ability to **position individual atoms** in materials with atomic precision. Ultimatly to make **atomic-scale logic circuits** that operate at the **picometer (10^-3 nm)-length scale**.

This is a breakthrough in physical sciences!

Perspective:

Towards PetaHz(10^6 GHz)-CPU with pico-m (10^-3 nm) sized-transistors

LETTER

RESEARCH LETTER

doi:10.1038/nature19821

Multi-petahertz electronic metrology

M. Garg[1], M. Zhan[1], T. T. Luu[1], H. Lakhotia[1], T. Klostermann[1], A. Guggenmos[1] & E. Goulielmakis[1]

The speed limit of electronics (signal processing) is determined by the frequency of electric current.
The use of light to drive electrons promises to access to vastly higher frequencies.
(increase the bandwidth of electronics).

Improving the performance by 6 order of magnitude.

From GHz (clock rate) towards PHz-based processors.

This is a breakthrough in physical sciences!

Atomic engineering

LETTERS

PUBLISHED ONLINE: 19 FEBRUARY 2012 | DOI: 10.1038/NNANO.2012.21

nature nanotechnology

A single-atom transistor

Martin Fuechsle[1], Jill A. Miwa[1], Suddhasatta Mahapatra[1], Hoon Ryu[2], Sunhee Lee[3], Oliver Warschkow[4], Lloyd C. L. Hollenberg[5], Gerhard Klimeck[3] and Michelle Y. Simmons[1]*

Fabrication of working devices such as transistors with extremely short gate lengths **requires** the ability to **position individual atoms** in materials with atomic precision.
Ultimatly to make **atomic-scale logic circuits** that operate at the **picometer (10^-3 nm)-length scale**.

Performance of a computer

Supercomputer

Cluster

Computer (or node/server)

Performance of a computer

- **The performance of a processor is measured by the quantity:**

FLOPS (**F**loiting-**P**oint of **O**pertaions **P**er **S**econd).

- It is a measure of the speed of a computer to perform arithmetic operations.

- **For a single processor**:

  FLOPS = (Clock speed)x(cores)x(FLOPs/cycle)=Peak performance

  FLOP is a way of encoding real numbers (i.e. DB 64bit or SP 32bit…)

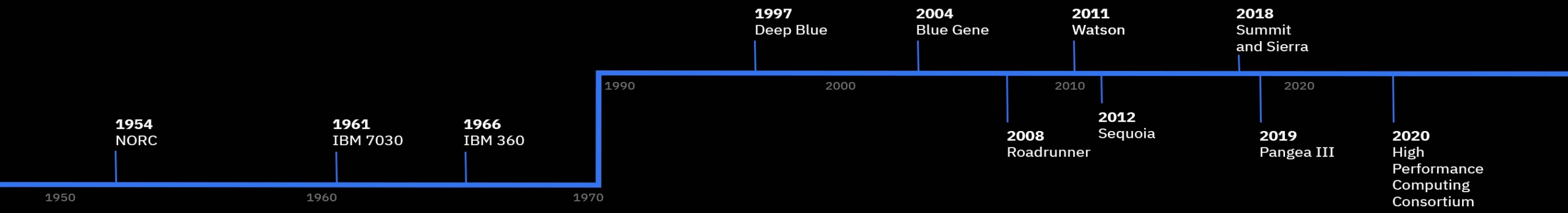
- 1 GigaFLOPS: processor can handel **billion floating-point (64 bit) operations every second**.
- **For matching: 1GigaFLOPS ~performing one calculation every second for 31.69 years**.
- 1 TeraFLOPS = 10^12 calculations per second.
- 1 PetaFLOPS = 10^15 calculations per second.

Supercomputer

Cluster

Computer (or node/server)

IBM Supercomputing timeline



1954
The Naval Ordnance Research Calculator helped forecast weather and performed other complex calculations.

1961
The **IBM 7030** was capable of 2 million operations per second.

1966
The **IBM 360** and its successors helped power NASA's Apollo program.

1997
Deep Blue wins its match with chess grandmaster Garry Kasparov.

2004
Blue Gene ushers in a new era of high-performance computing as it helps biologists explore gene development.

2008
Built for Los Alamos National Laboratory, **Roadrunner** is the first super-computer in the world to reach petaflop speed.

2011
Watson beats human competitors on Jeopardy!, earning a million-dollar jackpot for charity.

2012
Sequoia, the third-generation **Blue Gene** system, reaches speeds of 16.32 petaflops.

2018
Summit begins work at Oak Ridge National Laboratory; a sister machine, **Sierra,** launches at Lawrence Livermore National Laboratory.

2019
IBM builds **Pangea III,** the world's most powerful commercial super-computer, for Total to accurately locate new energy resources.

2020
IBM helps launch the COVID-19 **High Performance Computing Consortium** to research the COVID-19 virus and its potential cures.

MegaFLOPS
Million operation/s

1st supercomputer

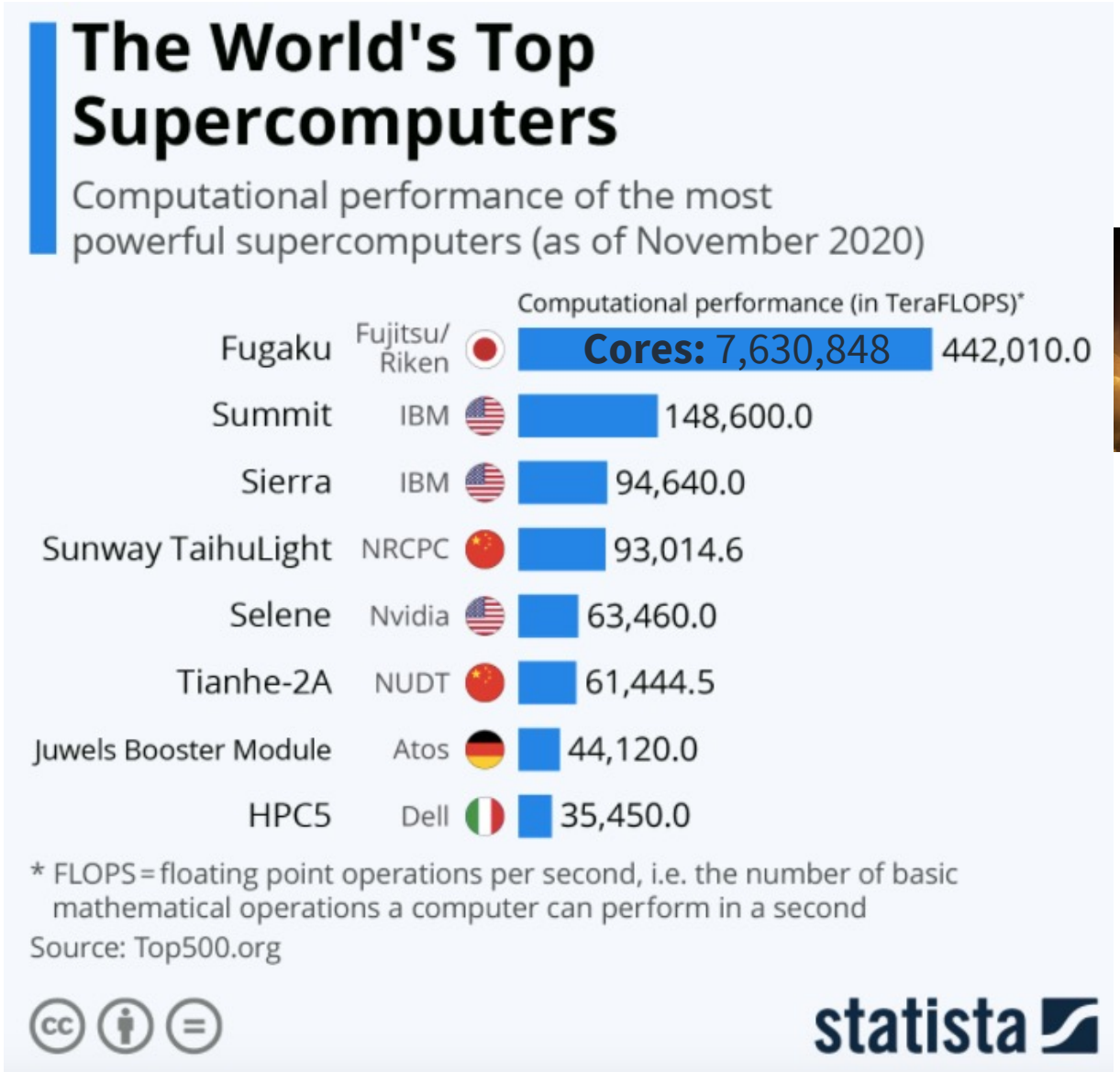
A new era of HPC

PetaFLOPS
Quadrillion

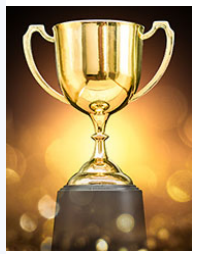
16 PetaFLOPS

1st P-FLOPS supercomputer

~500 PetaFLOPS



https://www.fujitsu.com/



The Fugaku (**158,976 nodes**) compute system was designed and built by Fujitsu and RIKEN. They take the first place worldwide in TOP500.

Peak performance
537 PetaFLOPS
=537 (quadrillion) 10^15 (64 bit) operations/s
Exceeding Summit by more 3x.

https://www.top500.org/

LUMI is one of the fastest supercomputers in the world

Taken from https://www.lumi-supercomputer.eu/

- **LUMI** (Large Unified Modern Infrastructure):
- **LUMI** is located in a data center in Kajaanni, **Finland**.
- Funded by the **EuroHPC JU** (50%) and a **consortium of 10 countries.**
- **LUMI consortium**: Finland, Belgium, The Czech republic, Denmark, Estonia, **Norway**, Poland, Sweden, Switzerland and Iceland.

Peak Performance

550 Petaflop/s
~550 quadrillion calculations per second

Computing power equivalent to

1 500 000

Modern laptop computers

About the size of a tenis court

Weight around 150 000 Kg

LUMI

Phase 1: CPU Partition (LUMI-C)

- **1536 compute nodes** with 2x AMD EPYC 7763 (Milan)
- 128 cores per node, **196 608 cores** total
- 256 GiB of memory per node
- Some nodes with 512 GiB and 1 TiB

Phase 2: GPU Partition (LUMI-G)

- Next generation AMD Instinct GPUs (**2560 nodes** each with **4 AMD MI250X GPUs**))
- 550 Pflops peak performance
- The fastest supercomputer in Europe
- Available summer (August-September) 2022

Norway's share (2 %: ~ 4000 cores on LUMI-C, ~ 200 GPUs on LUMI-G)

EuroHPC JU's share (50 %: ~ 100k cores on LUMI-C, ~ 5000 GPUs on LUMI-G)

Adapted from Orian Louant, LUMI User Support Team

What Supercomputers can be used for ?

To solve majore challenges in the world.

Supercomputer platfrom

High-Performance Computing (HPC)

Artificial Intelligence (AI)

To solve complicated problems in physical sciences engineering such that:

- Exploring the boundaries of quantum chemistry (first project LUMI).
- Predicting the structure of proteins using data-driven methodologies (ML, DL).
- Understanding the functionality **of COVID-19 virus** & potential cures.
- Desiging new molecues with unique functionality for modern technology.
- Delivering reliable weather and climate predictions.

Conclusion

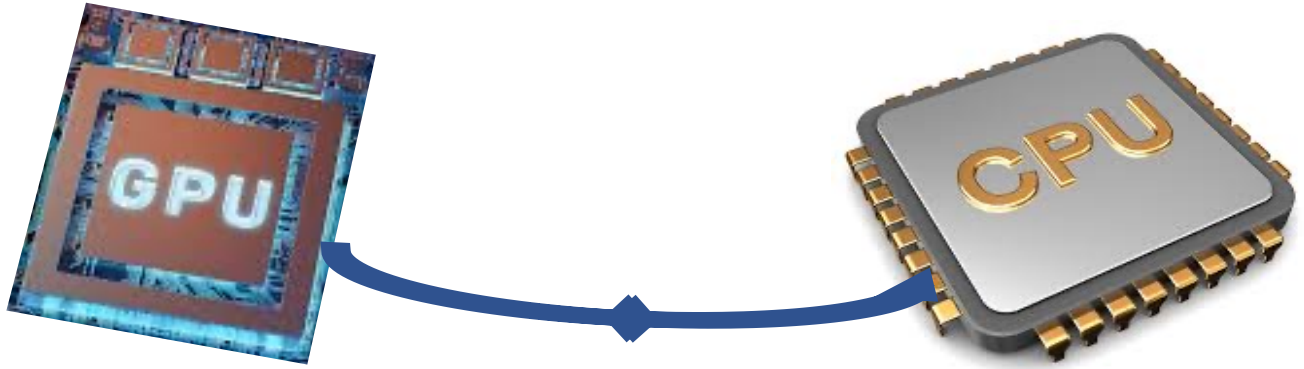
- The **lifetime** of high-performance **hardware** is less than **five years**, while **software can be used for decades**.
- **Software investments** should therefore provide more **flexibility to new and fast evolving technology**.

- **CPU & GPU –architectures**
- **Programming models**
- **Parallelism in OpenACC**

Outline

- CPU & GPU –architectures
- Programming models
- Parallelism in OpenACC

Heterogenous computing



GPU-Device

Thousands of cores

GPU: Graphics Processing Unit
CPU: Central Processing Unit

CPU-Host

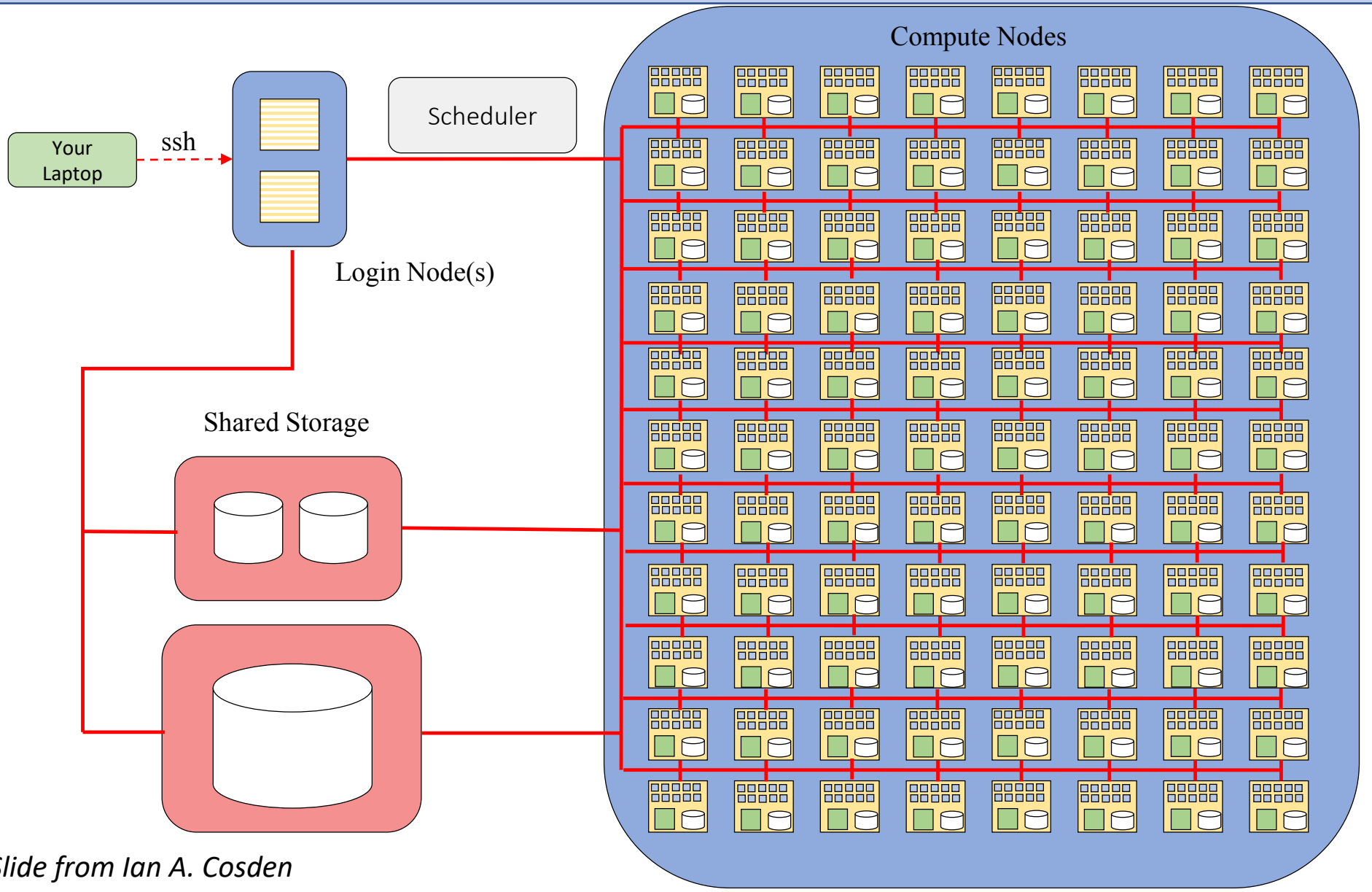
Multiple cores

Programming model
To offload a code region to a GPU-device

```
do i=1,n
   do j=1,m
  A(i,j) = B(i,j) + C(i,j)
   enddo
enddo
```

CPU-architecture

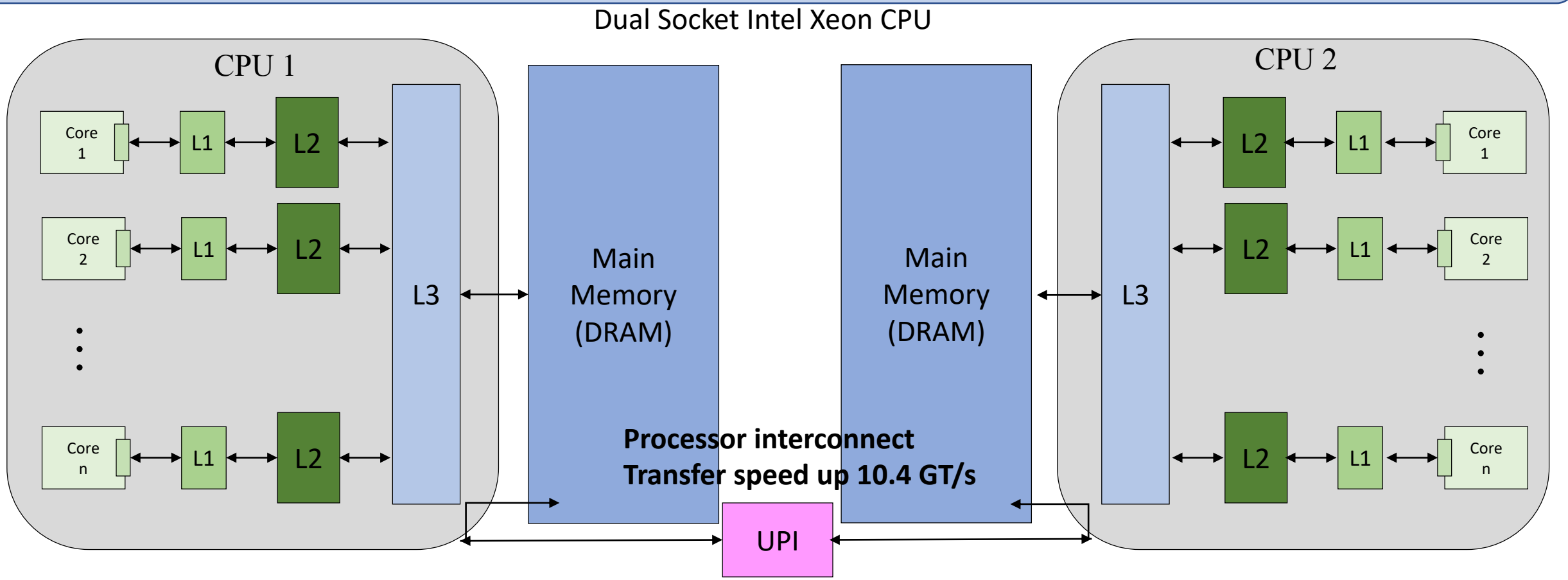
Architecture of Cluster



Slide from Ian A. Cosden

https://indico.cern.ch/event/814979/contributions/3401193/attachments/1831477/3105158/comp_arch_codas_2019.pdf

CPU-Architecture: ex. Betzy 128 cores



Access L1 Memory is Faster than L2

	Registers	L1 Cache	L2 Cache	L3 Cache	DRAM	Disk
Speed	1 cycle	~4 cycles	~10 cycles	~30 cycles	~200 cycles	10ms
Size	< KB per core	~32 KB per core	~256 KB per core	~35 MB per socket	~100 GB per socket	TB

https://indico.cern.ch/event/814979/contributions/3401193/attachments/1831477/3105158/comp_arch_codas_2019.pdf

GPU-architecture

Architecture of NVIDIA GPU devices

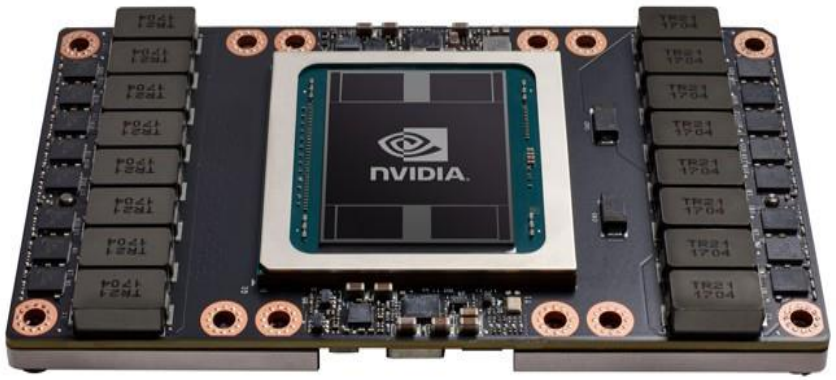


Figure 1. NVIDIA Tesla V100 SXM2 Module with Volta GV100 GPU

Figure 2. Volta GV100 full GPU with 84 SM Units

https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

Architecture of NVIDIA GPU devices

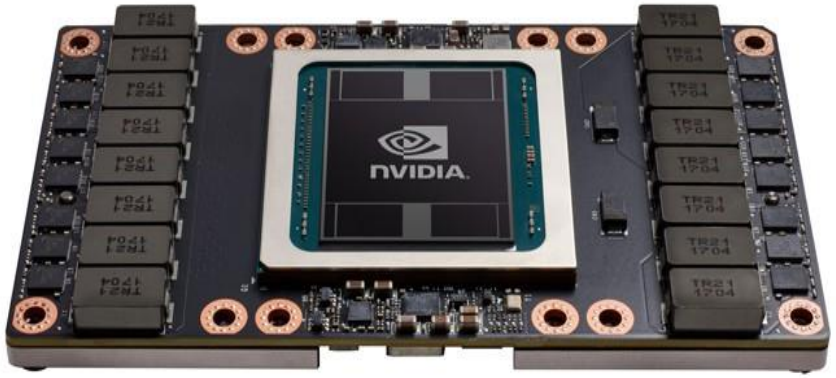


Figure 1. NVIDIA Tesla V100 SXM2 Module with Volta GV100 GPU

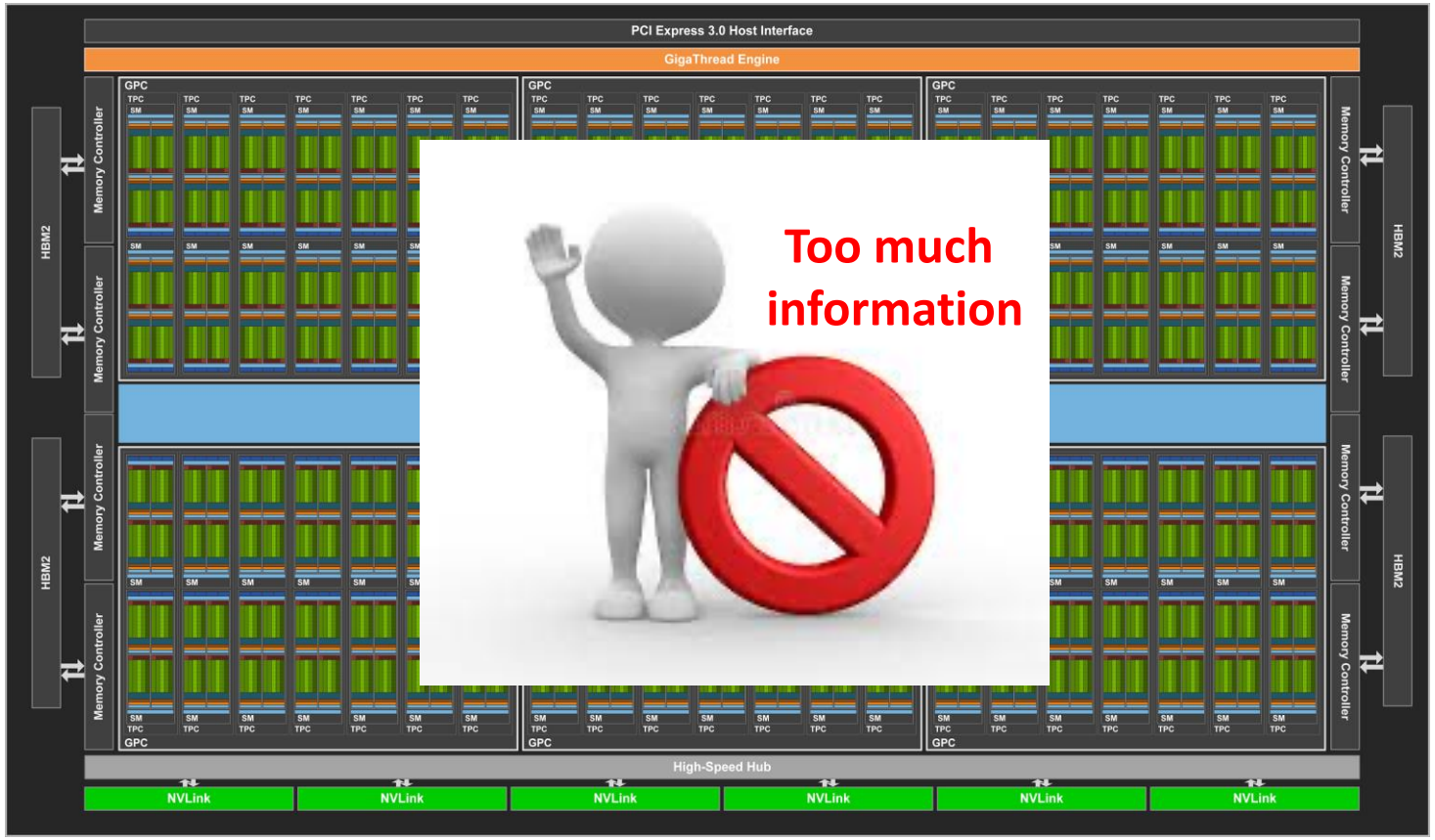
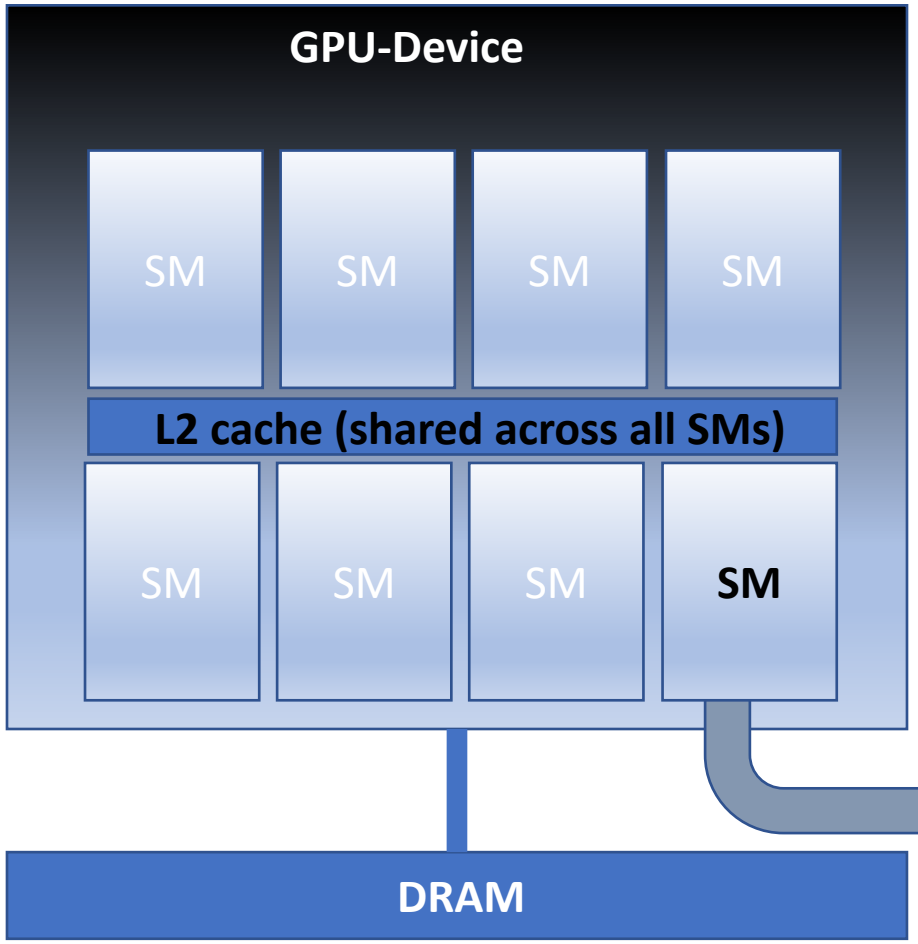


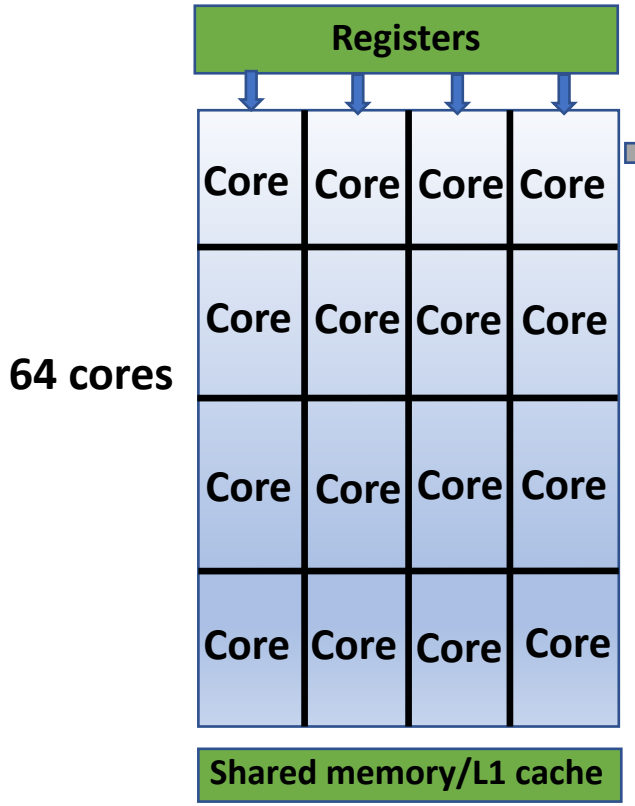
Figure 2. Volta GV100 full GPU with 84 SM Units

https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

Architecture of NVIDIA-GPU devices (simplified version)

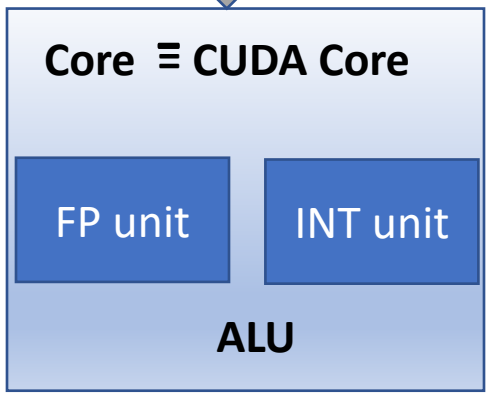


Streaming Multiprocessor (SM)



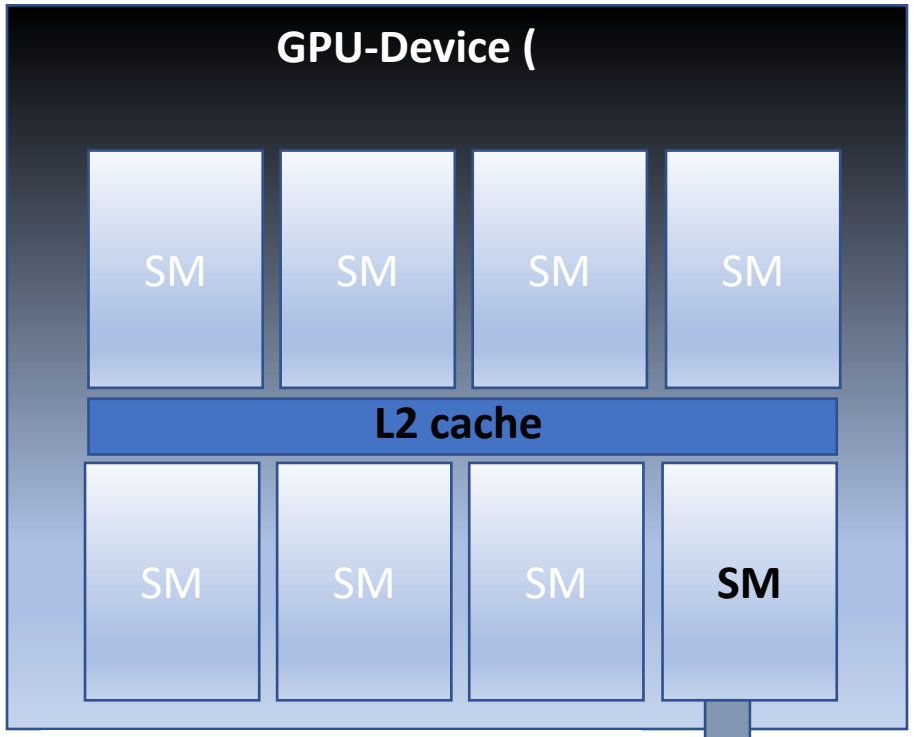
Data are stored temporarily. Registers are private to each thread.

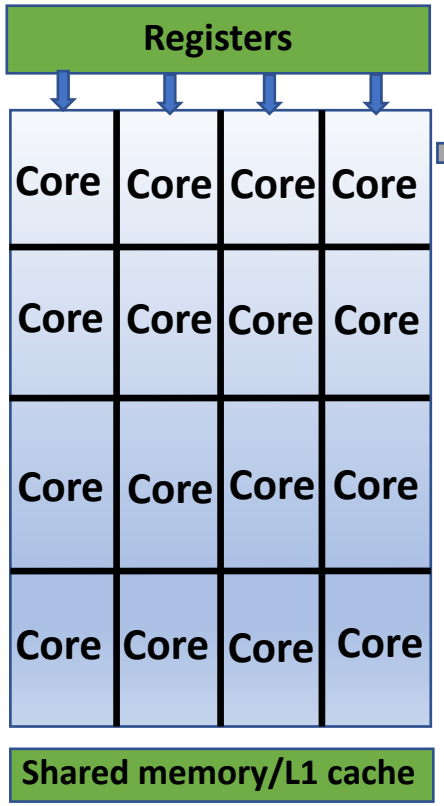
All threads share L1 cache.

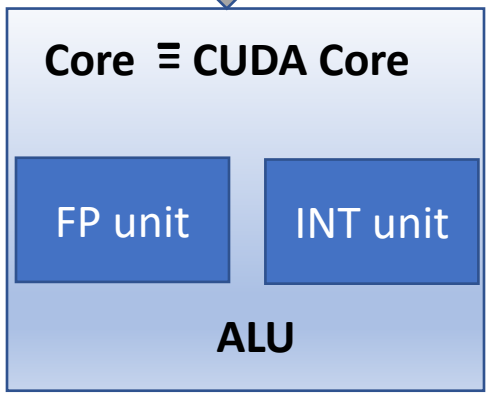


- Core includes ALU. Various logical operations for computations (FP32, FP64,) e.g. addition, multiplication and …
- ALU executes SIMD instructions. (processing multiple data with the same operation concurently (in parallel))

Architecture of NVIDIA-GPU devices (simplified version)







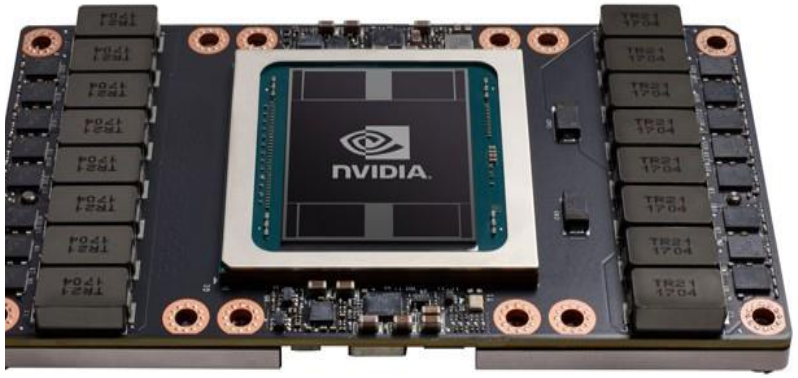
Streaming Multiprocessor (SM)

NVIDIA Tesla A100

For Tesla A100 GPU: 108 SMs, each SM has:

64 FP32 cores ===== A total of 6912 FP32 CUDA cores
64 INT32 cores ===== A total of 6912 INT32 CUDA cores.

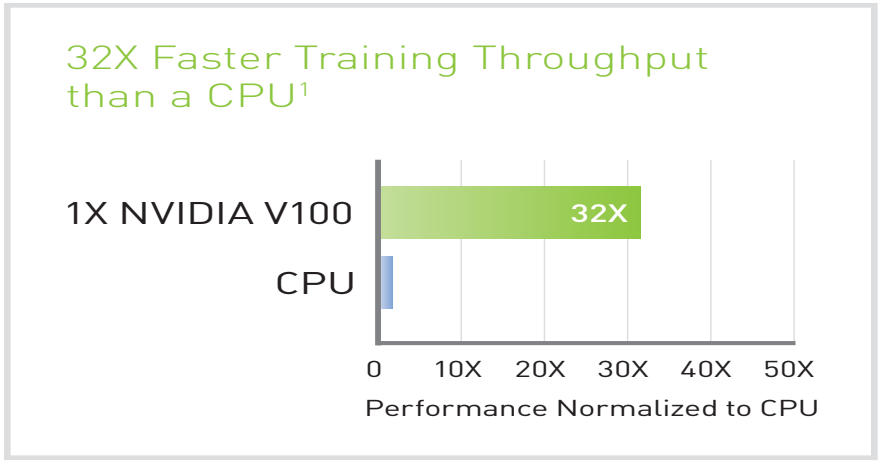
NVIDIA Tesla P100



NVIDIA Tesla V100

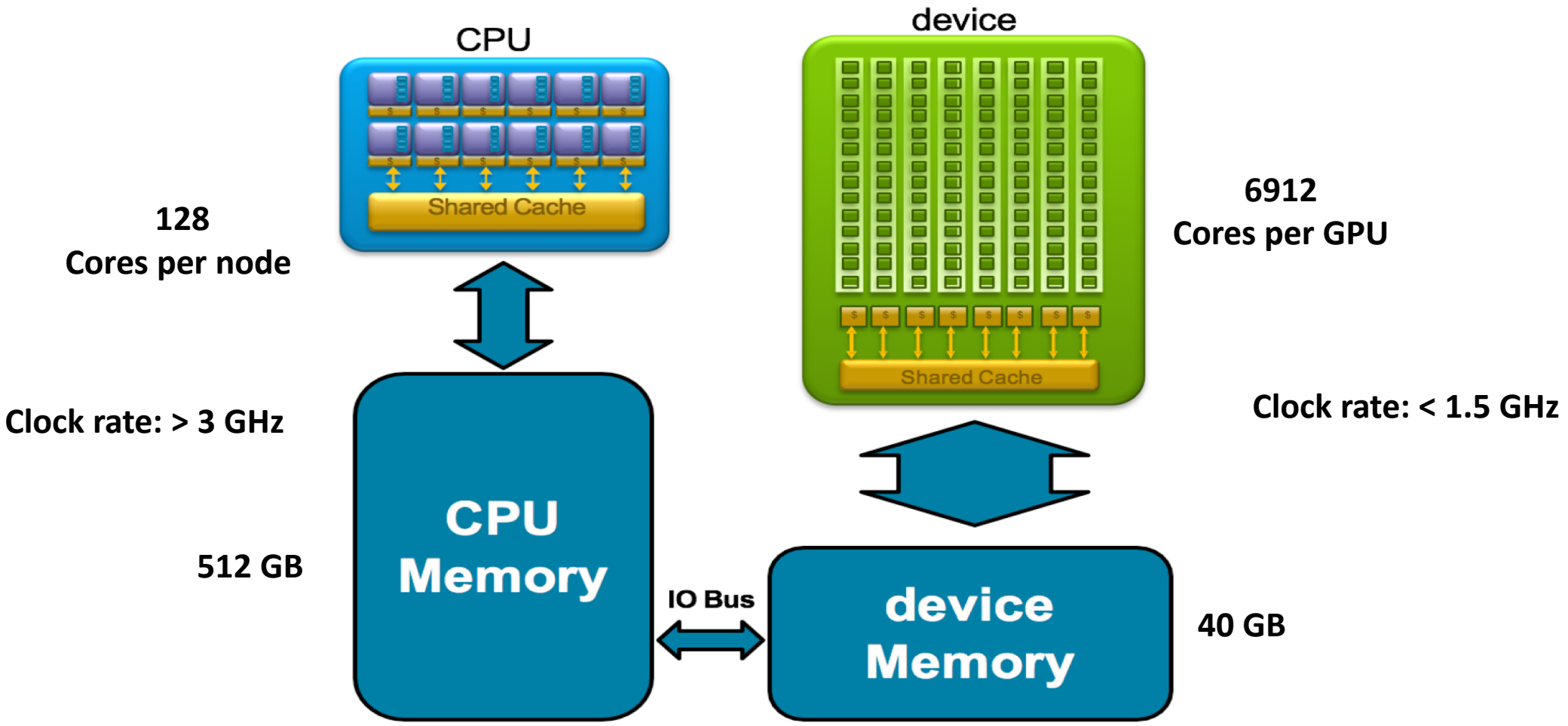
NVIDIA Tesla A100

Architecture	Tesla P100 (Pascal)	Tesla GV100 (Volta)	Tesla A100
SMs	56	84	108
NVIDIA CUDA cores	3584	5376	6912
Tensor cores/GPU	NA	672	432
Peak performance	9.3 TFLOPS	15.7 TFLOPS	39 TFLOPS
Transistors	15.3 billion	21.1 billion	54.2 billion
GPU die size	610 mm2	815 mm2	826 mm2



GPU-Dimension (cm)
26.7x11x4

Conclusion: CPU vs GPU



Taken from NVIDIA

Heterogenous programming models

Heterogenous programming models

Programming models can be hardware-dependent or hardware-independent

Directive based models:

High-level models

OpenACC

OpenMP

Compiler supports

NVHPC | GCC | Cray (only acc .2.0 F) | Clang | Cray | GCC | Icx (Intel)

Hardware

NVIDIA | NVIDIA | NVIDIA | AMD | AMD | AMD | Intel

AMD

Low level offload models:

CUDA

Only on NVIDIA

OpenCL

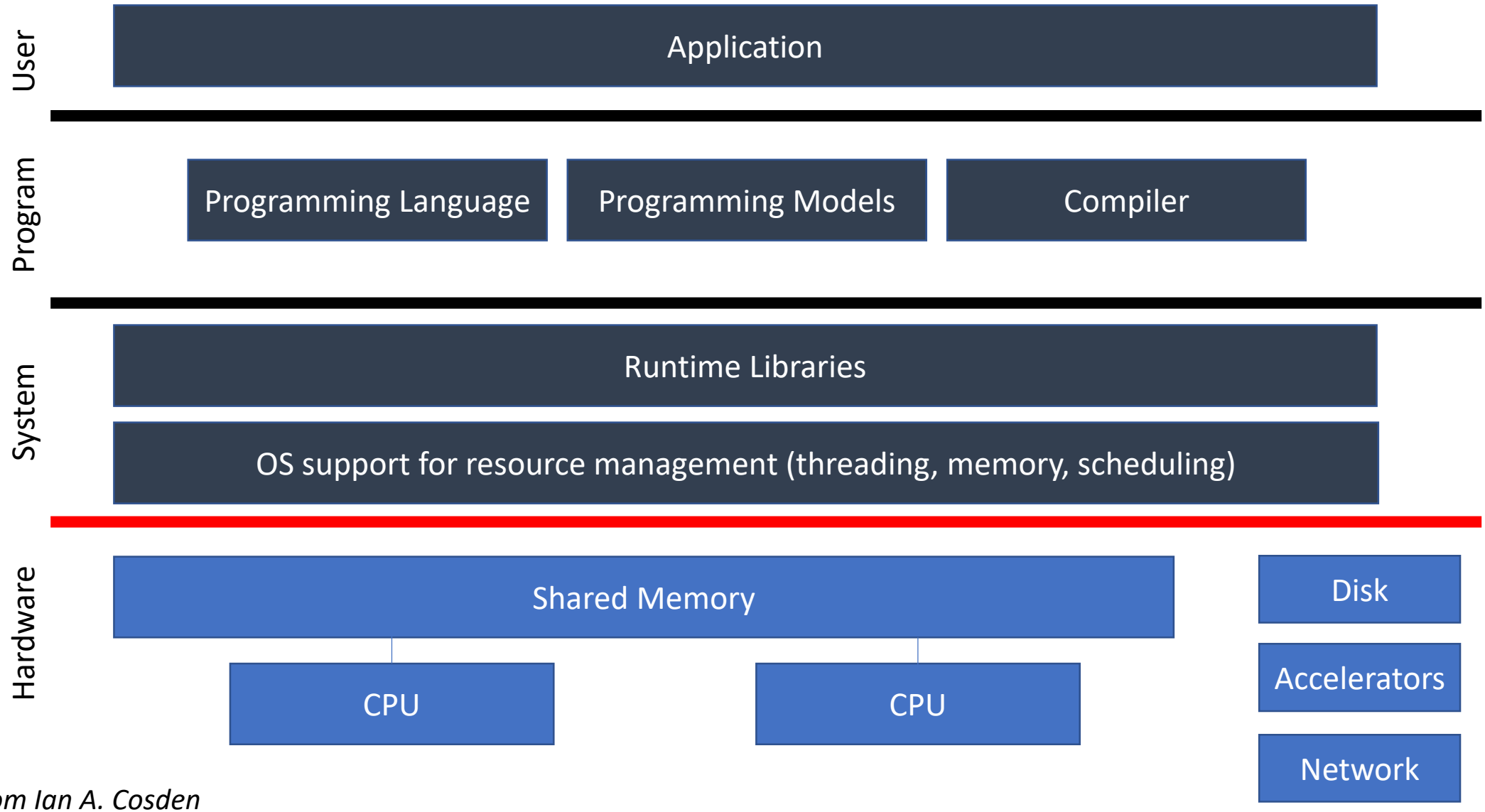
HIP

Only on AMD

Intel GPU:OpenCL can be migrated via SYCL API migrating from CUDA to DPC++. DPC++ tool is part of the Intel OneAPI Toolkit.

Hybrid multi-GPU programming: Combining MPI/OpenMP threading & OpenACC/OpenMP offloading

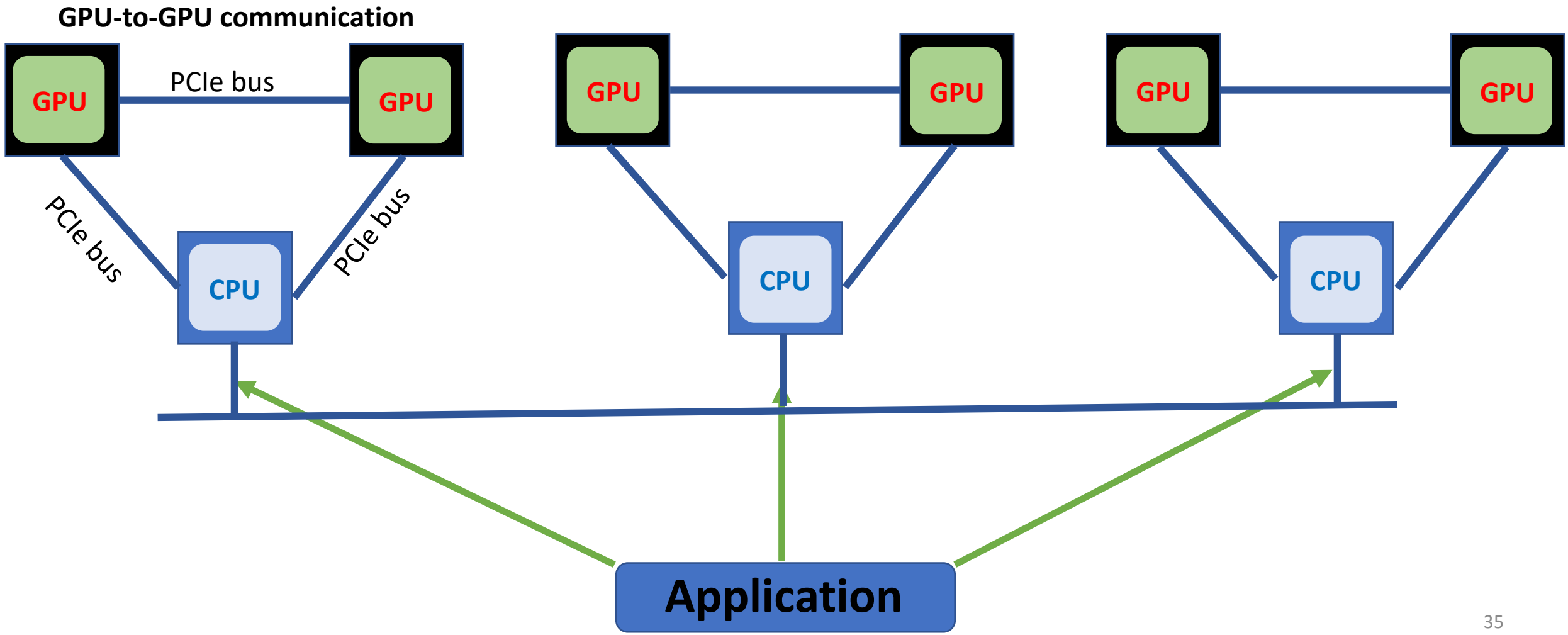
Hardware-architecture



Slide from Ian A. Cosden

Multi-GPU programming

MPI+GPU programming model



Parallelism in OpenACC/CUDA

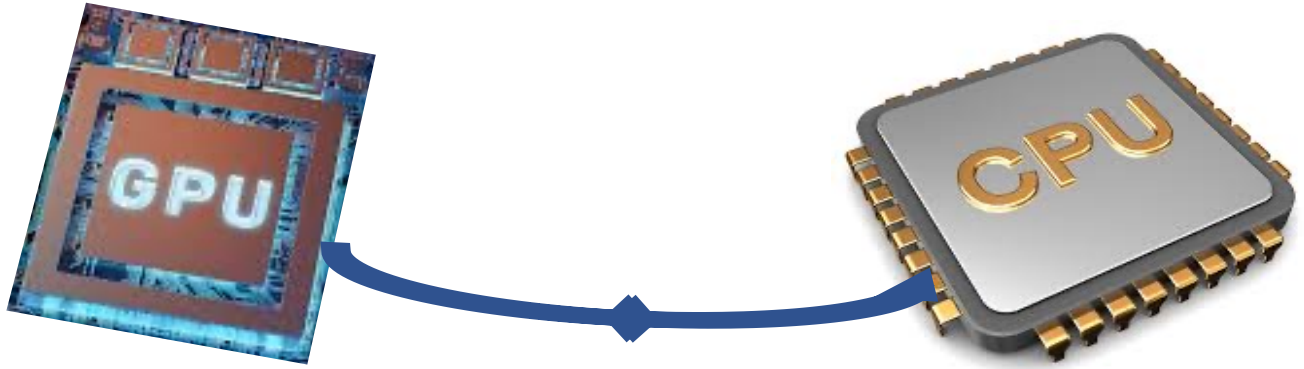
CUDA= Compute Unified Device Architecture

What is OpenACC?

OpenACC is a directive-based approach similar to OpenMP.
It is designed to simplify parallel programming between heterogenous systems CPU/GPU.
It is API (Application Programming Interface) to communicate between CPU and GPU.

Parallelism in OpenACC

Heterogenous computing



GPU-Device

Thousands of cores

Programming model
To offload a code region to a GPU-device

CPU-Host

Multiple cores

```
do i=1,n
   do j=1,m
  A(i,j) = B(i,j) + C(i,j)
   enddo
enddo
```

Parallelism in OpenACC

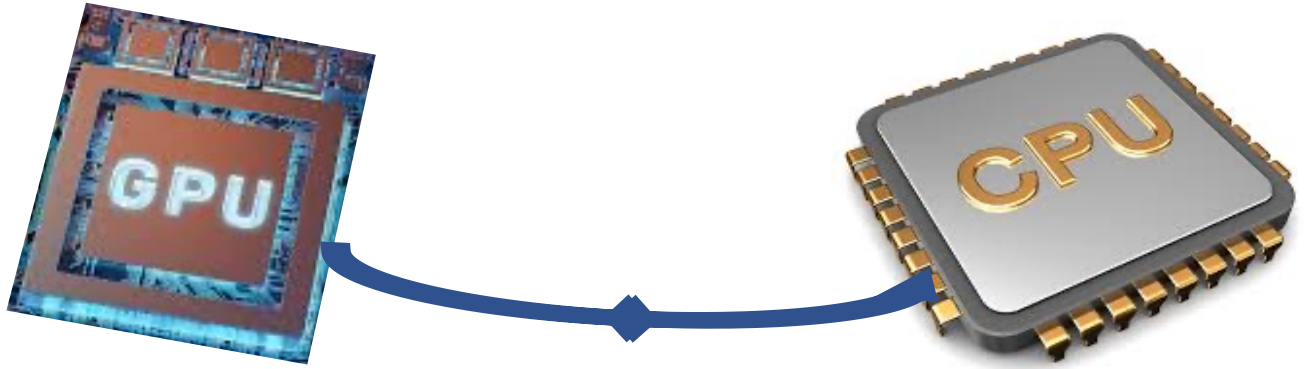
It uses CUDA to exploit parallelism.

Focus on how parallelism is done.

Message from the compiler

```
10, Generating Tesla code
11, !$acc loop gang, vector(32), worker(4)
! blockidx%x threadidx%x threadidx%y
12, !$acc loop seq
10, Generating implicit copyin(B(:,:),C(:,:))
[if not already present]
Generating implicit copyout(A(:,:))
[if not already present]
12, Loop is parallelizable
```

Heterogenous computing



GPU-Device

Thousands of cores

CPU-Host

Multiple cores

Incorporate directives that the compiler recognises. It tells the compiler to parallelize the loop.

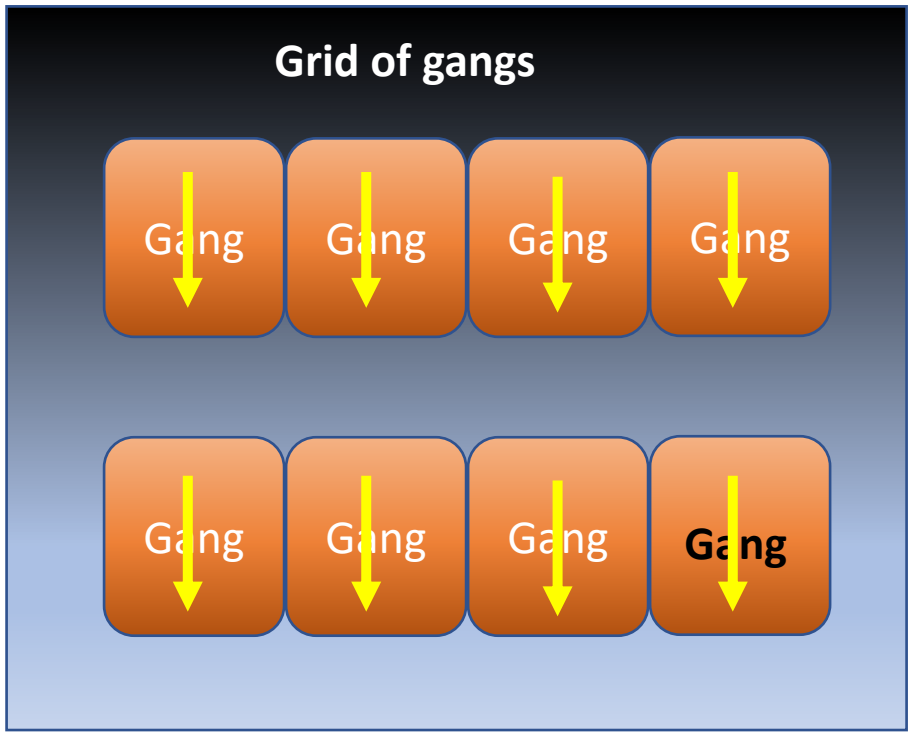
Programming model
To offload a code region to a GPU-device

```
!$acc parallel loop
do i=1,n
   do j=1,m
  A(i,j) = B(i,j) + C(i,j)
   enddo
Enddo
!$acc end parallel
```

Parallelism in OpenACC

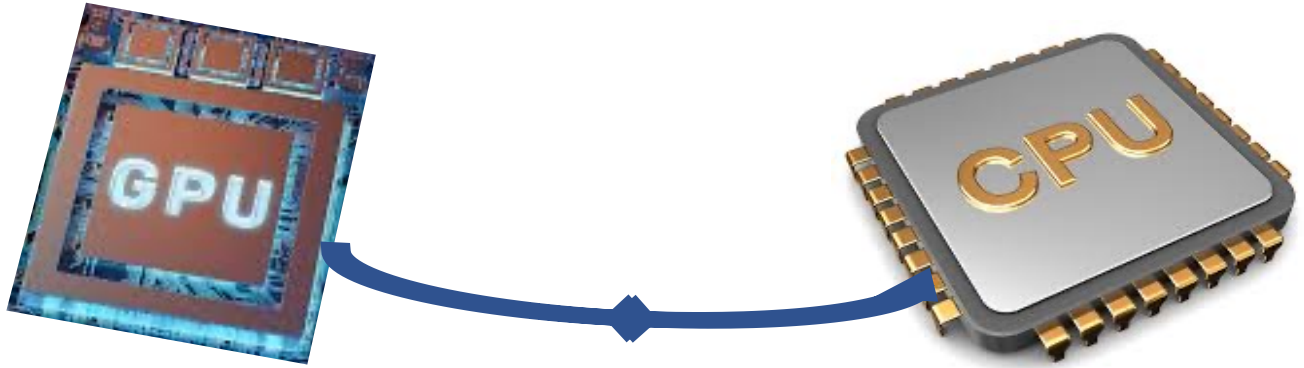
Software scheme

Only the "parallel" construct



Step 1: The compiler generates one or more gangs.
Step 2: The offloaded loop gets executed redundantly on each gang (get duplicated).

Heterogenous computing



GPU-Device

Thousands of cores

Incorporate directives that the compiler recognises. It tells the compiler to parallelize the loop.

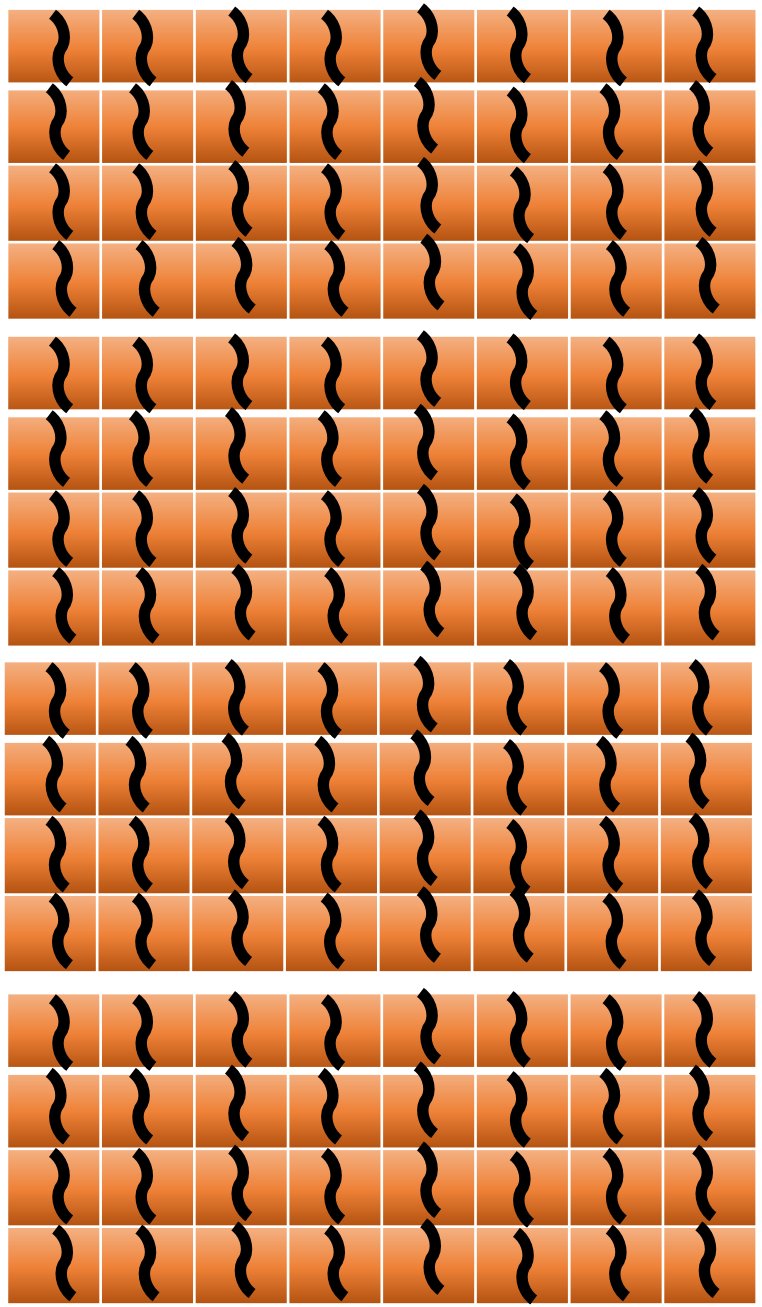
CPU-Host

Multiple cores

Programming model
To offload a code region to a GPU-device

```
!$acc parallel loop
do i=1,n
   do j=1,m
  A(i,j) = B(i,j) + C(i,j)
   enddo
Enddo
!$acc end parallel
```

Parallelism in OpenACC

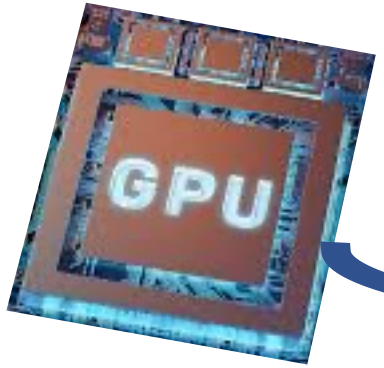


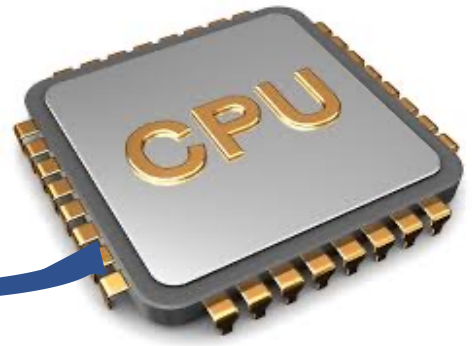
Workers
(Block of Threads)

Each piece of the loop runs by a thread at the same time.

Gang

Heterogenous computing





GPU-Device

Thousands of cores

CPU-Host

Multiple cores

Incorporate directives that the compiler recognises. It tells the compiler to parallelize the loop.

Programming model
To offload a code region to a GPU-device

```
!$acc parallel loop
do i=1,n
   do j=1,m
  A(i,j) = B(i,j) + C(i,j)
   enddo
Enddo
!$acc end parallel
```

Parallelism in OpenACC (it uses CUDA to exploit the parallelism)

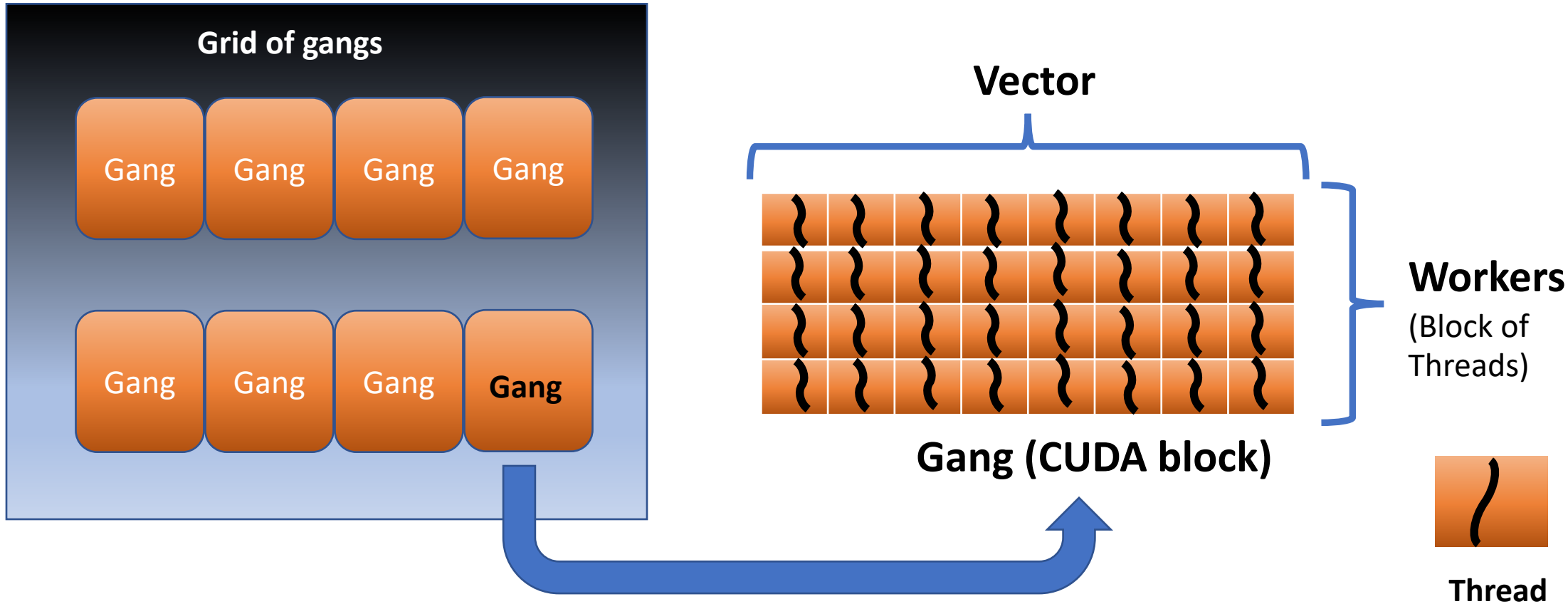
The parallelism in OpenACC is based on three concepts:

- **Gang :** Consists of a set of workers (block of thread)
- **Worker:** threads.
- **Vector: It** tells the OpenACC that it can exploit *vector parallelism*

Each thread has an index: it is used for calculating memory address locations.

vectorization naturally occurs in hardware across a **group of threads** referred to as a warp (**32 threads**) at the **same time**.

Software scheme



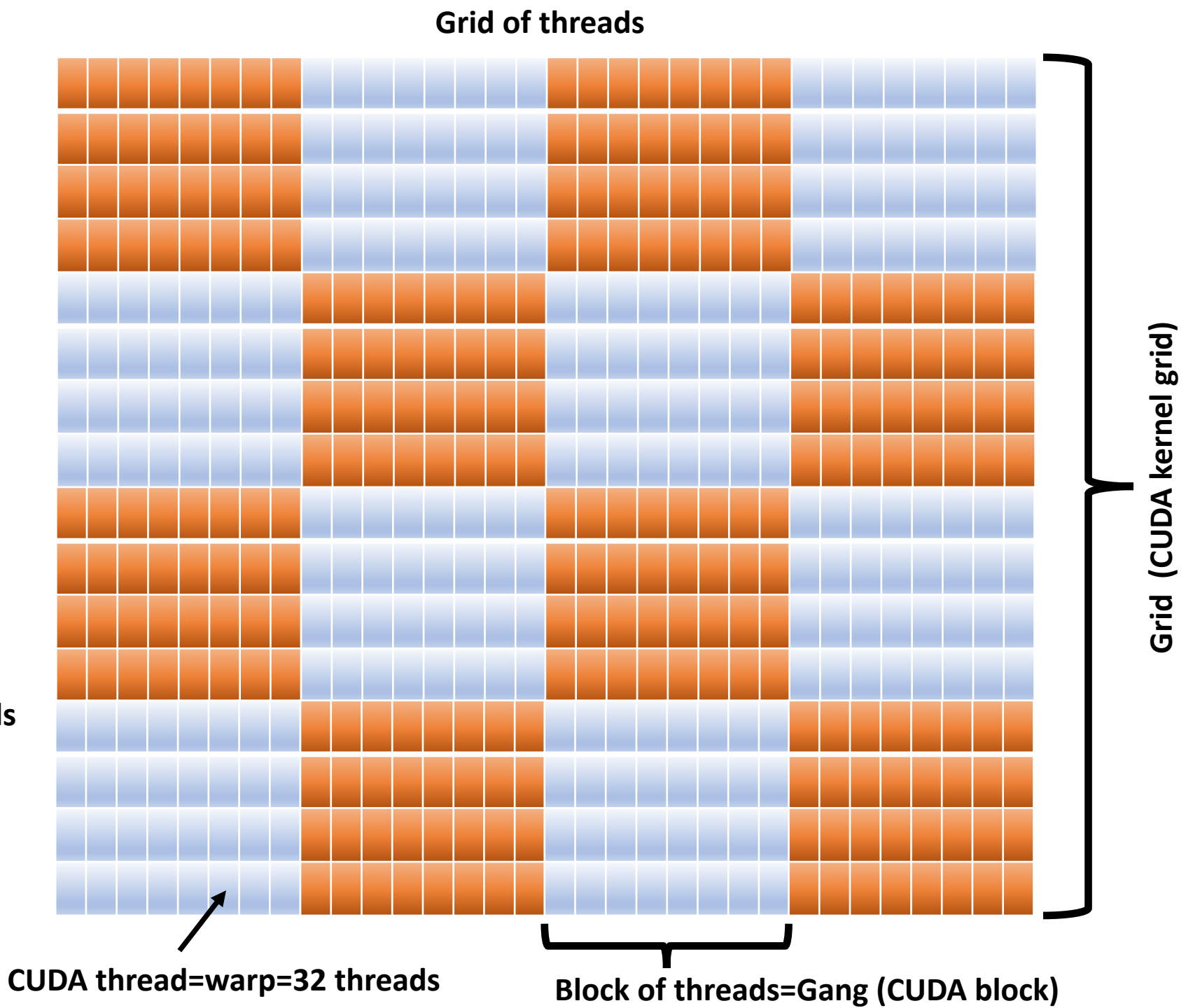
Mapping a matrix into a device

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Each thread executes a different piece
Of data item (data parallelism).

Each CUDA block has 1024 threads
(32 warps. A warp=collection of 32 threads
are executed simultaneously by a SM)

Execution on hardware
CUDA thread
CUDA block
CUDA grid



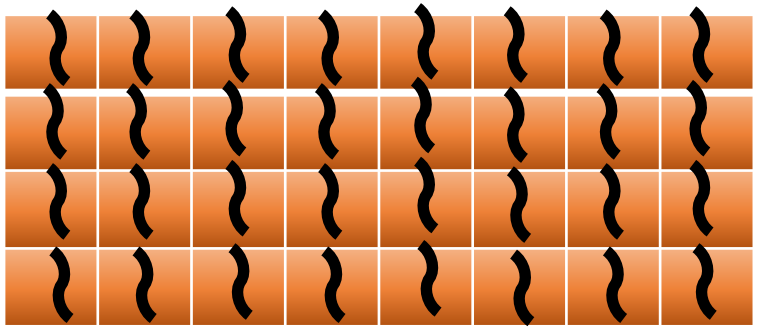
Execution on GPU

Software scheme

Hardware scheme

CUDA thread
(a full warp)

CUDA core



CUDA thread block

is executed on

core core core
core core core

Streaming Multiprocessor (SM)

Grid of gangs
(CUDA kernel grid)

Gang Gang Gang Gang

Gang Gang Gang Gang

GPU-Device

SM SM SM SM

SM SM SM SM

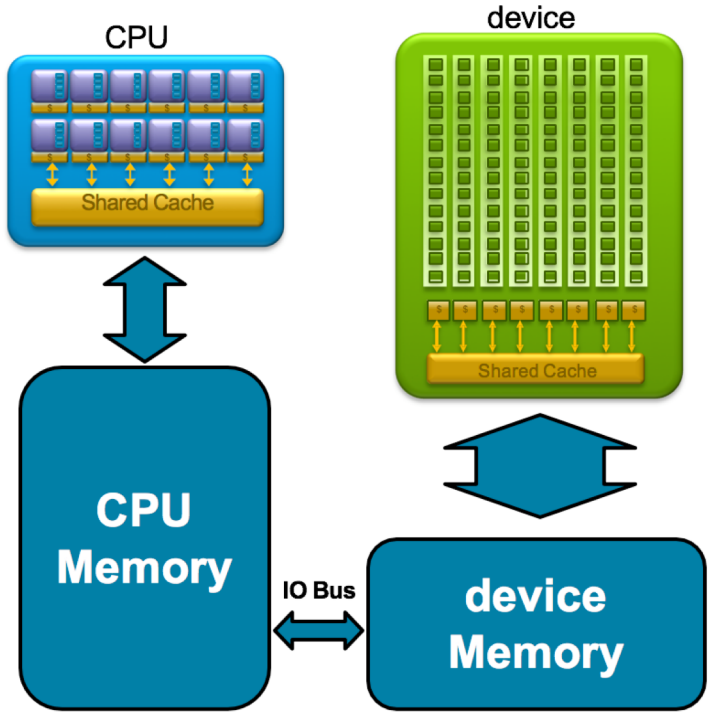
Conclusion

The parallelism in OpenACC is based on introducing three basic steps:

- **Step 1: The compiler generates one or more gangs.**
- **Step 2: The offloaded loop gets executed redundantly on each gang (get duplicated).**
- **Step 3: A block of thread will execute different pieces of the offloaded loop.**
- ***vector parallelism:* to perform multiple data-parallel operations at the same time.**
- vectorization naturally occurs in hardware across a group of threads referred to as a warp at the same time.

- **Data transfer between CPU and GPU is time consuming.**
- **The performance of GPU can be slower than CPU in some cases.**

- **Bottelneck in GPU-programming: Portability and hardware limitation.**
- **Converting codes affects the performance.**
- **Knowledge of GPU-architecture is important for GPU-programming.**



Taken from NVIDIA