A brief introduction to high-performance computing:

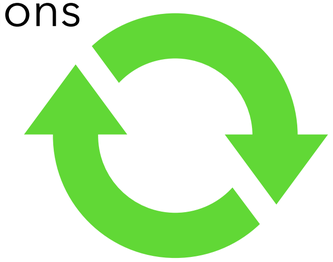
a computational physicist perspective

Antimo Marrazzo, Computational Physics Laboratory, UniTS, May 2023

Supercomputers for computational physics

- **Supercomputers** are essentially defined as the most powerful computer available at that time. These days a supercomputer is essentially "a purpose-built computer that embodies **millions of processors** or processor cores" (IBM, https://www.ibm.com/topics/hpc).
- Why do computational physicist need supercomputers?
  - Simulate a problem too **complex** to be even attempted on "regular" computers
  - Increase the **accuracy** of a simulations (e.g. denser grids, more spatial resolution)
  - Perform a **very large number** of simulations (high-throughput computing)
  - Exploit **high-performance hardware** (e.g. GPUs)
  - Scale economies (shared centralized resources are more cost efficient to buy and maintain)

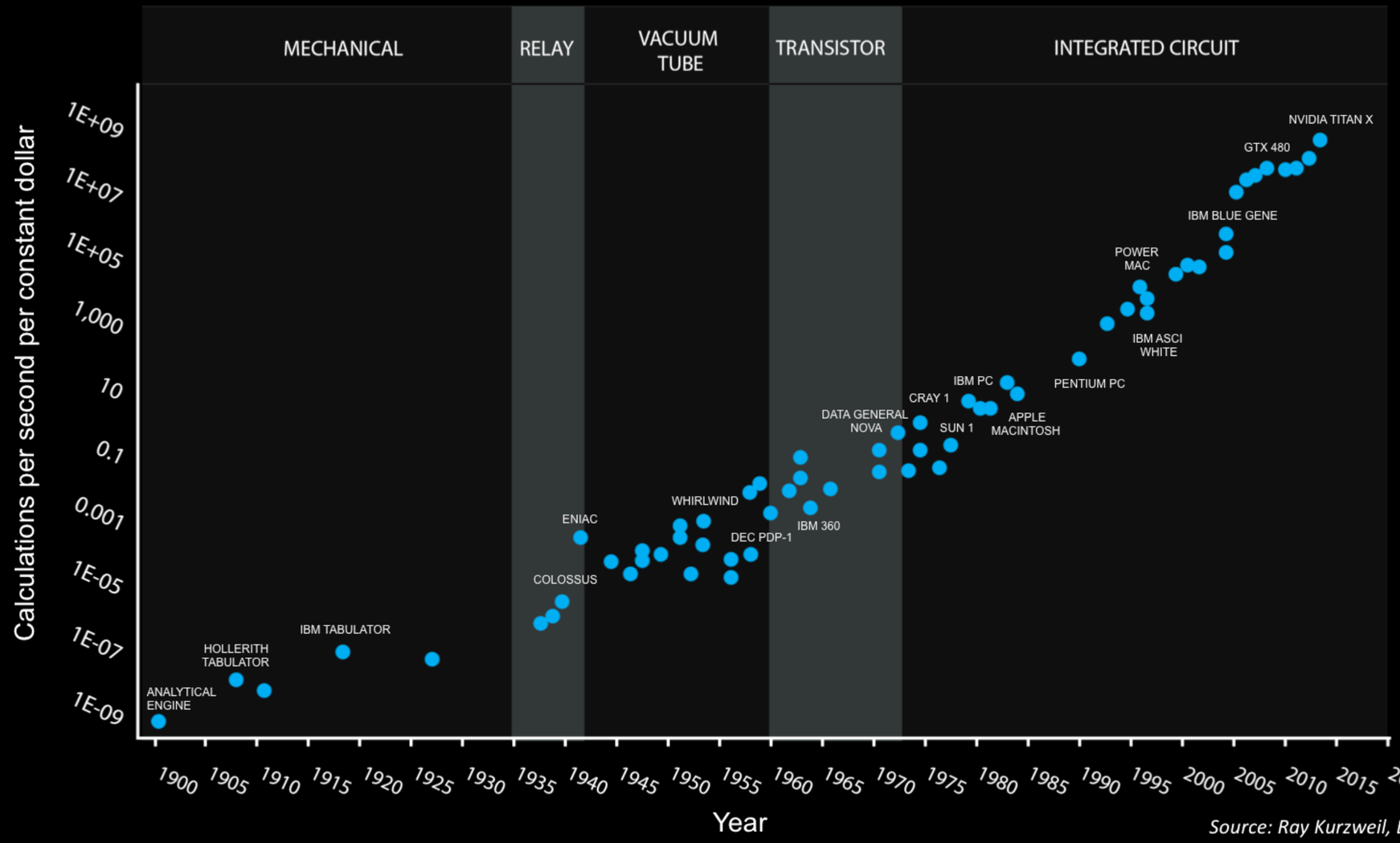
More complex simulations



More powerful computers

We simulate more complex systems with higher accuracy in a shorter time thanks to more powerful hardware and more sophisticated programming paradigms

120 years of Moore's law



Source: Ray Kurzweil, DFJ, Steve Jurvetson (https://www.flickr.com/photos/jurvetson/31409423572/)

"The complexity of devices (number of transistors per square inch in microprocessors) doubles every 18 months", Gordon Moore, INTEL co-founder, 1965

Exponentials in actions

1965

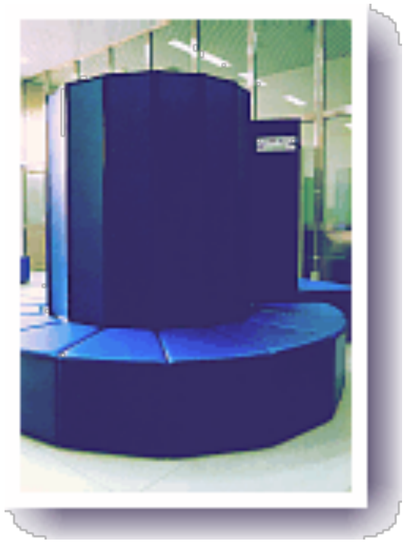
8Mb

2015

128Gb

STORAGE

1975



400 Mflops

2015

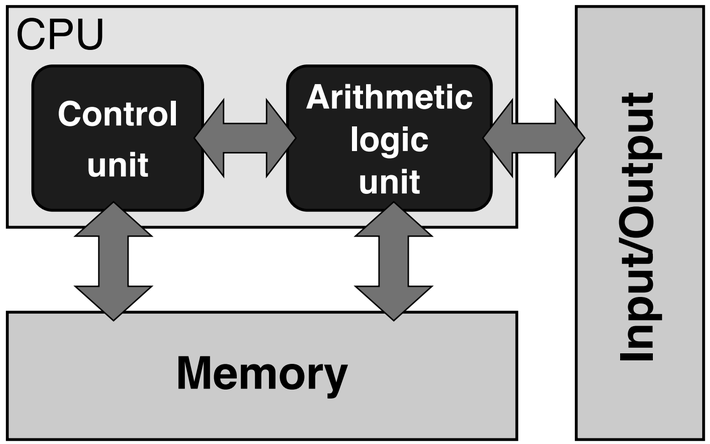
173 Gflops (GPU)

PERFORMANCE

Source: Andrew Emerson, Giovanni Erbacci, Introduction to HPC Architectures, CINECA

What performance means

- **Processor speed** → more operations per second
- **Memory size** → often crucial for Physics simulations!
- Bandwidth between processor and memory
- Bandwidth to the I/O system → Faster processors are useless if there are communication bottlenecks!
- Size and bandwidth of the cache
- Latency between processor, memory, and I/O system



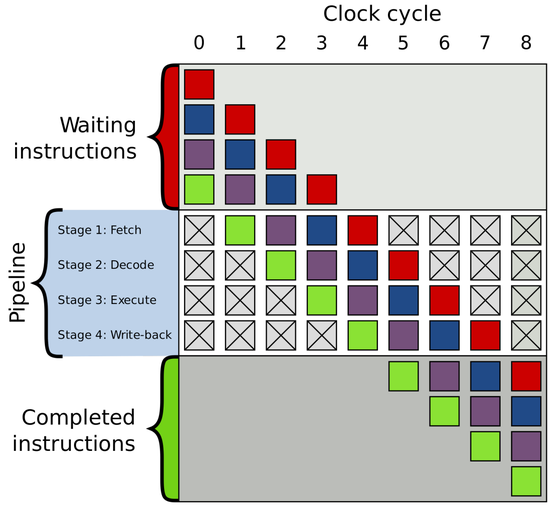
Stored-program computer architectural concept

(source: Introduction to HPC for Scientists and Engineers, Hager & Wellein, CRC press)

Beyond clock speed

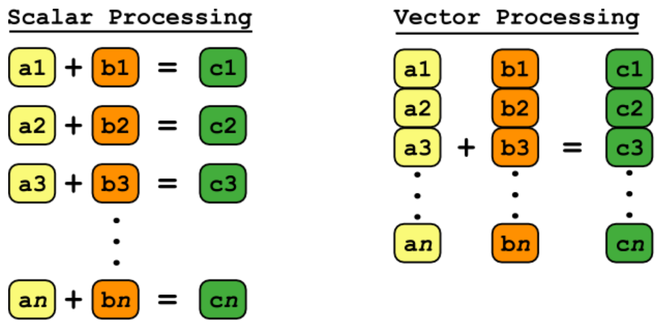
Modern CPUs are rather sophisticated devices (compared to the original von Neumann architecture), here some relevant features:

- **Pipelined functional units:** complex operations are split into simple components that can be executed using different functional units on the CPU, so the number of instructions executed per clock cycle increases.



Source: https://en.wikipedia.org/wiki/Instruction_pipelining#/media/File:Pipeline,_4_stage.svg

- **Vector CPU units** and data parallelism through **SIMD** (Single Instruction Multiple Data).

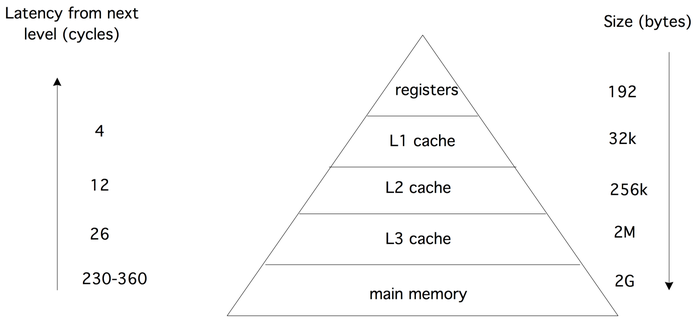



```
for i = 1 to n
    c[i] = a[i] + b[i]
end
```

```
c[1:n] = a[1:n] + b[1:n]
```

Source: Andrew Emerson, Giovanni Erbacci, Introduction to HPC Architectures, CINECA

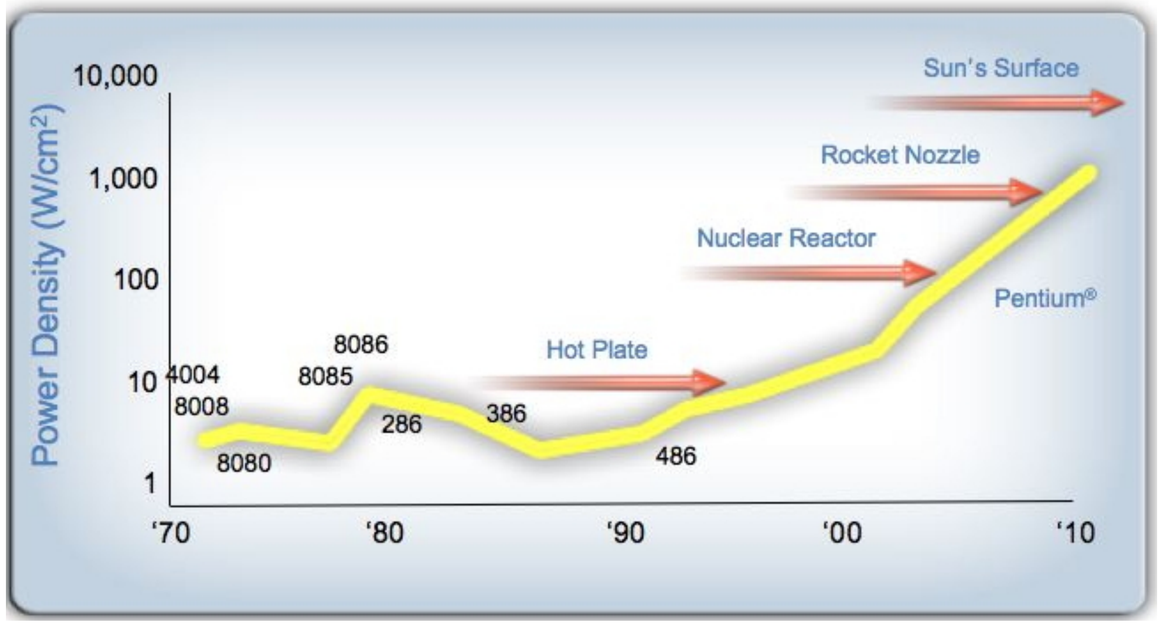
- **Caches**: low-capacity, high-speed memories that are commonly integrated on the CPU. NB: data transfer rates to main memory are dramatically slower compared to the CPU's arithmetics speeds (there has been increasing gap between processor and memory speeds).



Source: Introduction to High Performance Scientific Computing, Victor Eijkhout,

Go faster, go parallel

- Since more than ten years miniaturization of components has almost stopped (leakage current problem). In addition, the frequency can not be increase since this would raise the heat production of the chip too far.
- **Power wall**: sophistication of a single core can not be increased any further -> increase the amount of explicit parallalization -> multicore architectures & parallel programming

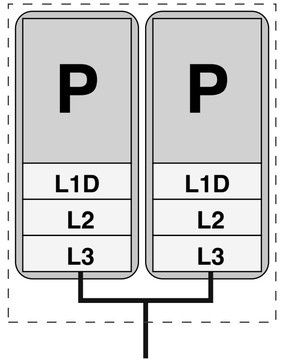


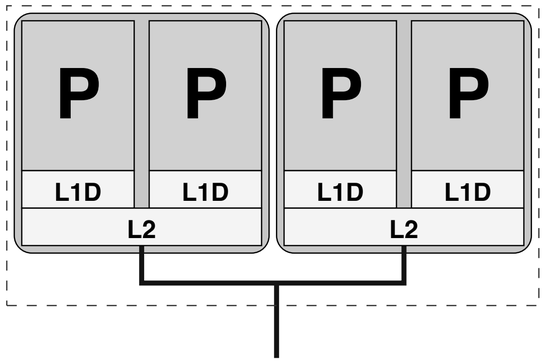
Projected heat dissipation of a CPU if trends had continued.

Source: Introduction to High Performance Scientific Computing, Victor Eijkhout, (original source Pat Helsinger)

Multi-core and multi-thread processors

- **Multi-core processors**. The chip is divided into multiple processing cores, with a mix of mix of shared and private caches.





- **Multi-thread processors**. Threads are streams of parallel instructions. Multiple threads can be executed on each processing unit.

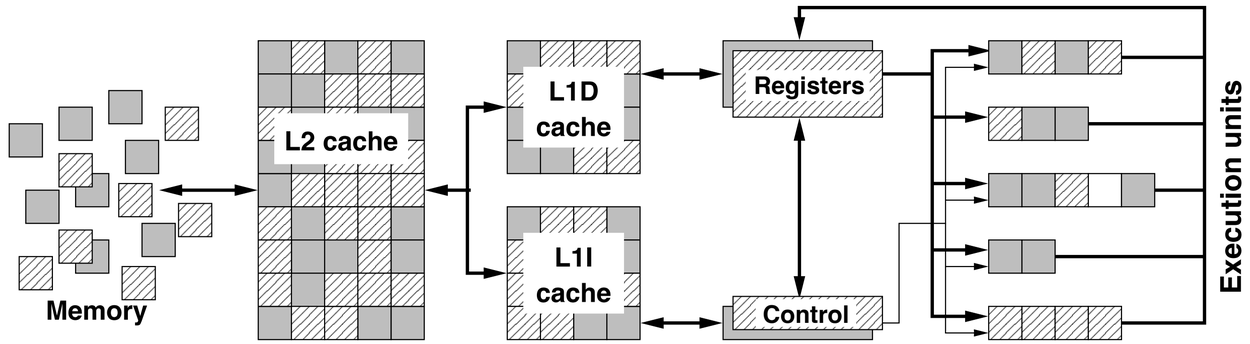


Figure 1.20: Simplified diagram of control/data flow in a (multi-)pipelined microprocessor with fine-grained two-way SMT. Two instruction streams (threads) share resources like caches and pipelines but retain their respective architectural state (registers, control units). Graphics by courtesy of Intel.

Sources: Introduction to HPC for Scientists and Engineers, Hager & Wellein, CRC press

High-performance computing (HPC)

Modern HPC solutions are based on these 3 pillars

- **[Software] Parallel & massively parallel computing**. Parallel computing runs multiple tasks simultaneously on multiple computer servers or processors. Massively parallel computing is parallel computing using tens of thousands to millions of processors or processor cores.

- **[Hardware] Computer clusters (also called HPC clusters)**. An HPC cluster consists of multiple high-speed computer servers networked together, with a centralized scheduler that manages the parallel computing workload. The computers, called nodes, use either high-performance multi-core CPUs or, more likely today, GPUs (graphical processing units), which are well suited for rigorous mathematical calculations, machine learning models and graphics-intensive tasks.

- **[Hardware] High-performance components.** All the other computing resources in an HPC cluster—networking, memory, storage and file systems—are high-speed, high-throughput and low-latency components that can keep pace with the nodes and **optimize the computing power and performance of the cluster.**

<u>These days, power consumption is becoming ever more critical than pure performance!</u>

adapted from https://www.ibm.com/topics/hpc, IBM

MPI & OpenMP

- **MPI** (**'Message Passing Interface'**)
  - The same program runs on all **processes, messages carry data** between processes. Those processes could be running on separate compute nodes, or different cores inside a node, or even on the same processor core, time-sharing its resources.
  - Implemented as a **library**
  - **Inter-node parallelization** and **distributed-memory**
  - Most HPC parallel applications use it
  - "assembly language of parallel programming", can also be seen as a programming model

- **OpenMP** is an **extension** to the programming languages **C and Fortran**.
  - Mainly parallel execution of **loops**
  - based on compiler **directives**, a preprocessor can schedule the parallel execution of the loop iterations.
  - based on **threads**, it features dynamic parallelism: the number of execution streams operating in parallel can vary from one part of the code to another
  - **Shared-memory** paradigm

```
PROGRAM hello_world_mpi
include 'mpif.h'

integer process_Rank, size_Of_Cluster, ierror

call MPI_INIT(ierror)
call MPI_COMM_SIZE(MPI_COMM_WORLD, size_Of_Cluster, ierror)
call MPI_COMM_RANK(MPI_COMM_WORLD, process_Rank, ierror)

DO i = 0, 3, 1
    IF(i == process_Rank) THEN
        print *, 'Hello World from process: ', process_Rank, 'of
', size_Of_Cluster
    END IF
    call MPI_BARRIER( MPI_COMM_WORLD, i_error)
END DO

call MPI_FINALIZE(ierror)
END PROGRAM
```

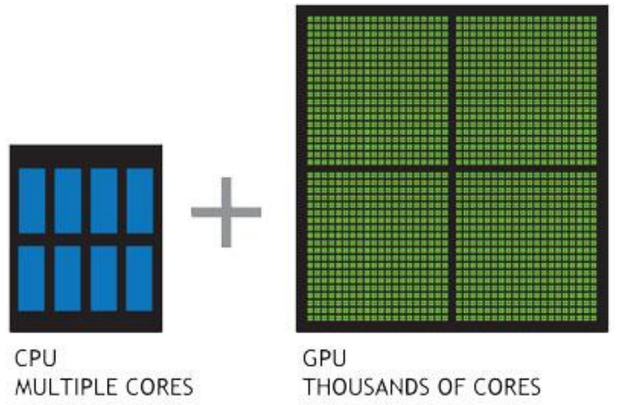
```
PROGRAM Parallel_Ordered_Hello
USE OMP_LIB

INTEGER :: thread_id

!$OMP PARALLEL PRIVATE(thread_id)
    thread_id = OMP_GET_THREAD_NUM()

    DO i=0,OMP_GET_MAX_THREADS()
        IF (i == thread_id) THEN
            PRINT *, "Hello from process: ", thread_id
        END IF
        !$OMP BARRIER
    END DO
!$OMP END PARALLEL

END
```

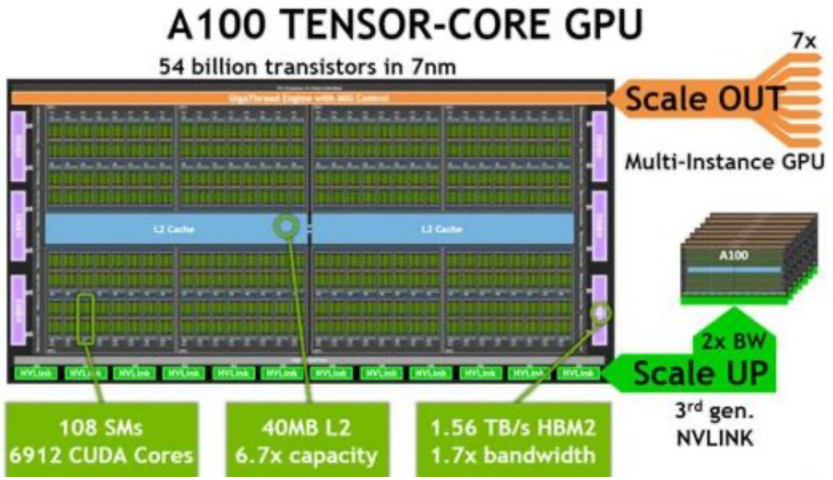
Sources: Introduction to High Performance Scientific Computing, Victor Eijkhout; Introduction to HPC for Scientists and Engineers, Hager & Wellein, CRC press

Source: https://curc.readthedocs.io/en/latest/index.html

GPUs

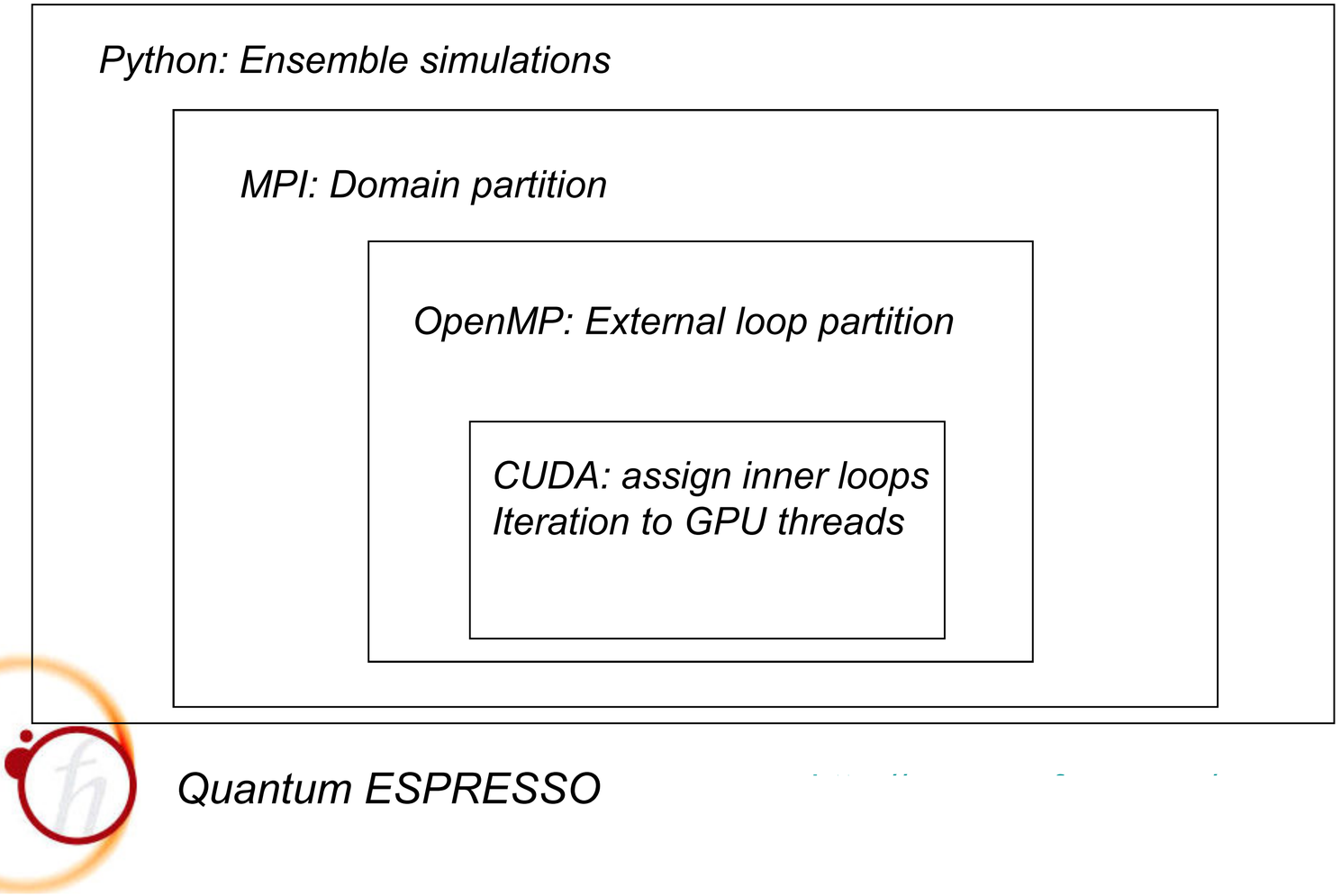
- While modern CPUs are designed to be moderately efficient for essentially all tasks, **co-processors** can be designed for specific tasks to improve performance and/or reduce power consumption.
- Very popular co-processors in HPC are **Graphics Processing Units (GPU)**
- GPUs are special purpose processors, designed for fast graphics processing and gradually evolved to be useful also for non-graphics computing.
- "Graphics pipeline": **identical operations are performed on many data elements (data parallelism)**, and a number of such blocks of data parallelism can be active at the same time.
- GPUs rely on a large amount of data parallelism and the ability to do a fast switch of context (data accessed by threads), optimal for graphics and scientific applications.
- Multiple programming strategies and languages (e.g. NVIDIA CUDA, OpenACC).





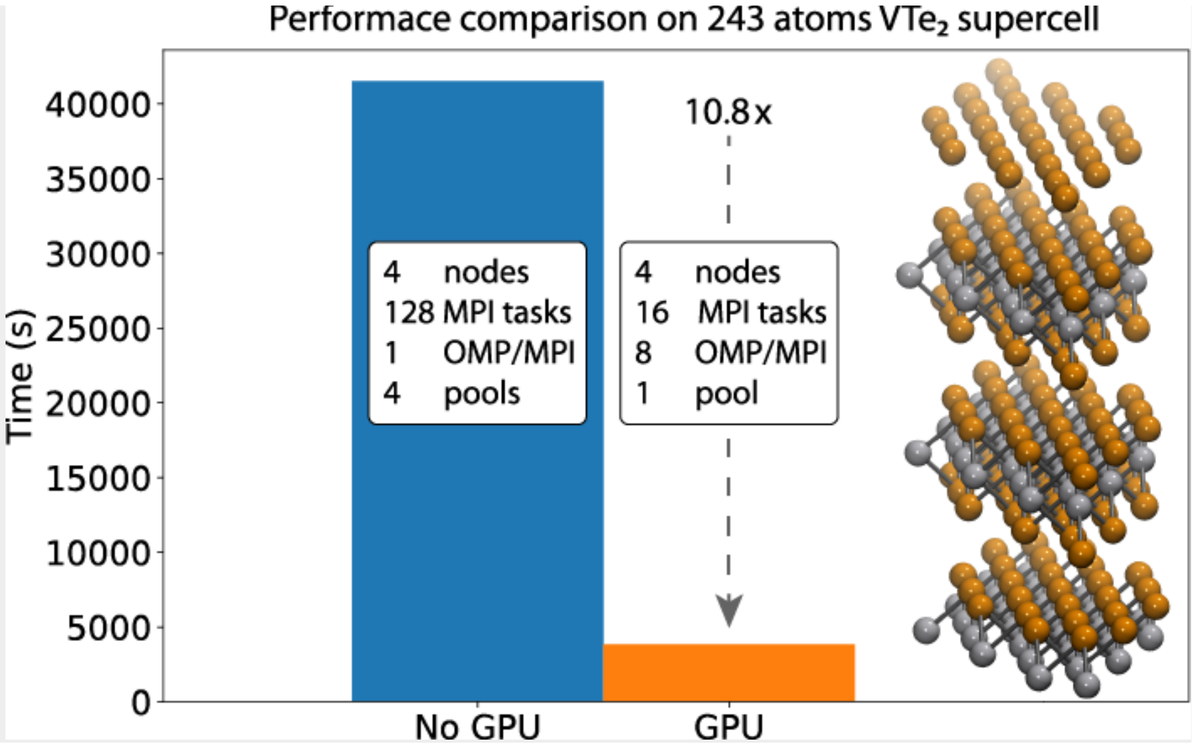
Source: NVIDIA

Hybrid parallelizations in QE

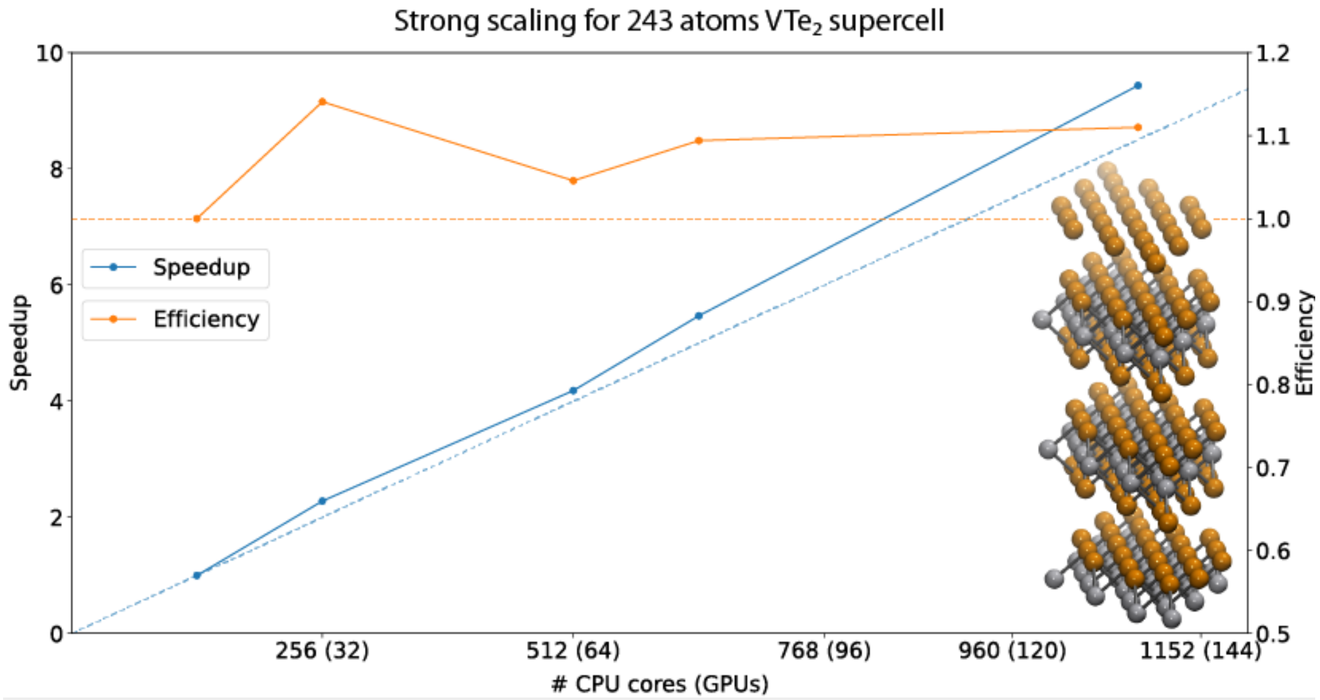


HPC software often uses multiple (hybrid) parallelization strategies, which are implemented with different programming languages.

Benchmarks and scaling tests



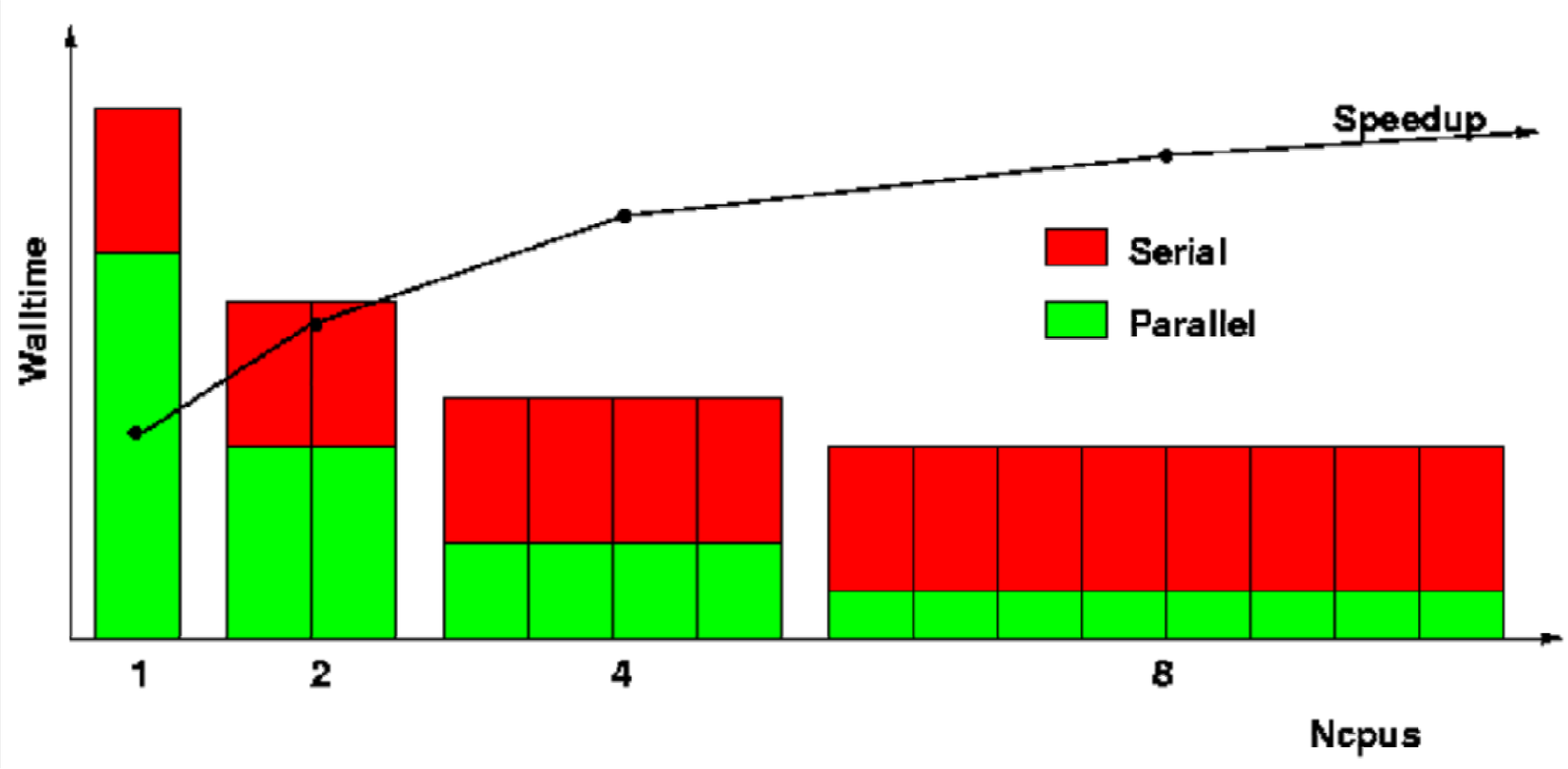
Performance comparison for CPU only and GPU accelerated computations. The SCF calculations were both performed on 4 nodes with 4 Nvidia V100 GPUs on CINECA's Marconi100 using a 3x3x3 supercell with 243 atoms for the VTe2 high temperature 1T structure. In the calculations 17 irreducible k-points were used distributed respectively on 1 and 4 pools in GPU accelerated and CPU only calculations.

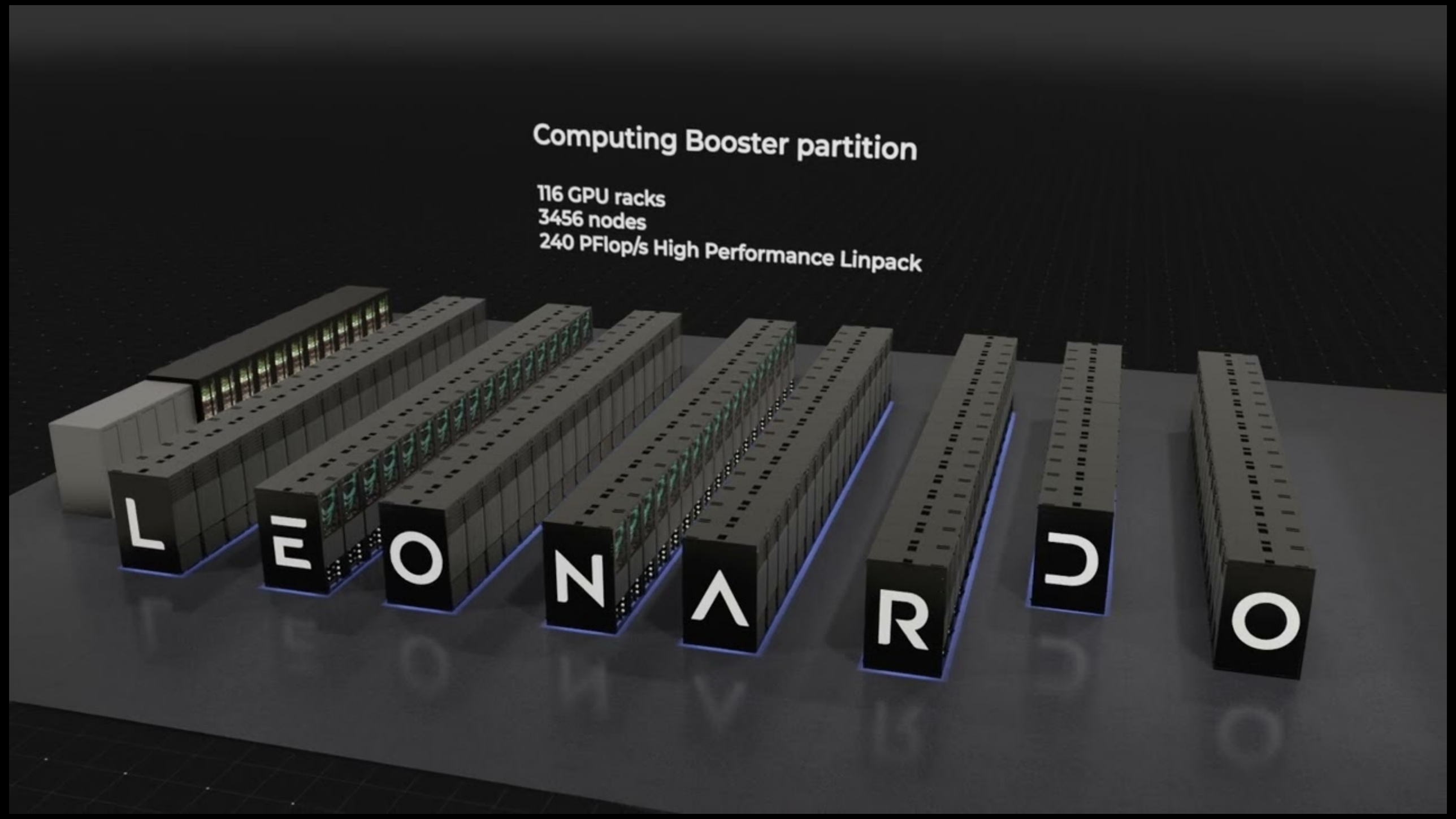


Strong scaling test for a QE PWscf calculation of a VTe2 supercell with 243 atoms performed on Marconi100 with 4 Nvidia V100 GPUs per node. Every calculation was performed using 1 MPI task and 8 OMP threads per GPU. The super-linear speed up is related to the better parallelization on reciprocal lattice k points which is allowed by the increase of available memory.

Amdahl's law

The **upper limit of scalability for parallel applications** is set by the fraction of overall execution time spent in the serial (non-scalable) part of the code




Computing Booster partition
116 GPU racks
3456 nodes
240 PFlop/s High Performance Linpack
LEONARDO

References and outlook

- **In order to write efficient scientific software, it is important to understand computer architectures.**
- **HPC applications often requires parallel programming or automation strategies.**
- This is relevant for 1) modern CPUs as you find them on your laptop, 2) even more crucial for HPC systems.
- Some references on HPC and parallel programming
  - ➡ Introduction to High Performance Scientific Computing, Victor Eijkhout
  - ➡ Introduction to HPC for Scientists and Engineers, Hager & Wellein, CRC press

The next generation of computational physicist and computational scientists will ***use and write massively parallel code*** *with* ***multiple programming languages*** *on* ***exascale supercomputers*** *(exaflops,* ***10^{18} 64-bit operations/second****), which will be mostly powered by accelerators (GPUs, FPGAs) and maybe by noisy intermediate-scale quantum (NISQ) computers.*